DocInsights 2026

Workshop on Document Intelligence and Understanding

Proceedings of the Workshop

October 24, 2026

©2026 Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-477-4

Preface

Welcome to the first Workshop on Document Intelligence and Understanding (DocInsights 2026), held in Budapest, Hungary, on October 24, 2026, co-located with EMNLP 2026.

Documents such as reports, forms, financial statements, scientific papers, and manuals are among the richest sources of information in science, industry, and society, combining text with tables, figures, and complex layouts. DocInsights brings together researchers and practitioners working on document understanding, from parsing and information extraction to multimodal retrieval, question answering, and reasoning over long and visually rich documents, with a focus on reliable, evidence-grounded systems.

This year, the main workshop track received 82 direct submissions (excluding one withdrawn submission) and 18 papers committed through ACL Rolling Review (ARR). We accepted 39 of these 100 papers (39%): 29 appear in these proceedings, and 10 non-archival papers are presented at the workshop only. The accepted papers cover document parsing and OCR robustness, table understanding and extraction, multimodal and long-document retrieval-augmented generation, visual document retrieval, evaluation methodology for document AI, agents for document and table processing, and applications to financial, biomedical, policy, and scientific documents.

The workshop also hosted two shared tasks: DocSem, on evidence-grounded quantitative reasoning over documents, and Dr.DocBench, on expert-level document parsing. We accepted 12 system description papers, 5 of which appear in these proceedings under Shared Task Papers.

We are grateful to the 147 members of our program committee for their careful and timely reviews, and to all the authors and shared task participants. We especially thank our shared task organizers, Jyotika Singh, Karan Dua, Hitesh Patel, Rahul Suresh, Xiaolong Luo, and Alexandra Bezea-Tudor, for designing and running the DocSem and Dr.DocBench challenges.

We hope you enjoy the workshop!

The DocInsights 2026 Organizers
Santosh T.Y.S.S., Amit Agarwal, Vivek Gupta, Yova Kementchedjhieva, Aparna Garimella, and Yufang Hou

Organizing Committee

Organizers

Santosh T.Y.S.S., Amazon, USA
Amit Agarwal, Oracle, USA
Vivek Gupta, Arizona State University, USA
Yova Kementchedjhieva, MBZUAI, UAE
Aparna Garimella, Adobe Inc, India
Yufang Hou, IT:U Austria / IBM Research Europe

Shared Task Organizers

Jyotika Singh, Oracle AI
Karan Dua, Oracle AI
Hitesh Patel, Oracle AI
Rahul Suresh, Abaka AI
Xiaolong Luo, Harvard University
Alexandra Bezea-Tudor, Abaka AI

Program Committee

Program Committee

Amirhossein Abaskohi, University of British Columbia
Mahmud Elahi Akhter, Queen Mary, University of London
Tejas Anvekar, Arizona State University
Longju Bai, University of Michigan - Ann Arbor
George Arthur Baker, University of Chicago
Bal Krishna Bal, Kathmandu University
Nikita Bhutani, Megagon Labs, Inc
Zahra Bokaei, University of Edinburgh
Shubin Cai, Shenzhen University
Xinyu Cai, Shanghai Artificial Intelligence Laboratory
Lele Cao, Microsoft
Yixuan Cao, Institute of Computing Technology, Chinese Academy of Sciences
Abhijit Chakraborty, Arizona State University and MongoDB
Jiayi Chen, Fudan University
Ming-Bin Chen, University of Melbourne
Xinchi Chen, Fudan University
Zesen Cheng, Peking University
Purva Chiniya, Amazon
Jeonghwan Choi, Korea Advanced Institute of Science & Technology
Jihun Choi, Knowledge Work Inc.
Anna Corazza, Università Federico II di Napoli, Italy
Xiang Dai, CSIRO
Preetam Prabhu Srikar Dammu, University of Washington
Yufan Dang, Tsinghua University
Pavel Denisov, Fraunhofer IAIS
Hang Ding, Shanghai Jiaotong University
Wentao Ding, Beijing Institute for General Artificial Intelligence
Hervé Déjean, Naver Labs Europe
Xiran Fan, VISA
Ian T. Foster, University of Chicago and Argonne National Laboratory
Sheela Ramprakash Goenka, Google Deepmind
Boyu Gou, Ohio State University, Columbus
Ray Groshan, University of Maryland, Baltimore County
Sonam Gupta, International Business Machines
Masoud Hashemi, ServiceNow Inc
Hilaf Hasson, Cohesity
Qijia He, University of Washington
Pingjun Hong, Universität Vienna
Arya Honraopatil, University of Maryland, Baltimore County
Jun Hou, Virginia Polytechnic Institute and State University
Bodun Hu, University of Texas at Austin
Wei-Chieh Huang, University of Illinois at Chicago
Jena D. Hwang, Allen Institute for Artificial Intelligence
Pooja Jhunjhunwala, Google
Yuelyu Ji, University of Pittsburgh
Ziyou Jiang, Institute of Software Chinese Academy of Sciences

Ruochun Jin, National University of Defense Technology
Gyuwan Kim, University of California, Santa Barbara
Taehyeong Kim, Seoul National University
Dongyuan Li, The University of Tokyo and Institute of Science Tokyo
Rongchun Li, National University of Defense Technology
Ruosen Li, University of Texas at Dallas
Ryan Li, Tesla AI
Yantao Li, nanjing university
Yinghao Li, Shanxi University
Zhuoqun Li, University of the Chinese Academy of Sciences
Ruixue Lian, Amazon
I-Bin Liao, Hon Hai Technology Group (Foxconn®)
Xinting Liao, University of British Columbia and Vector Institute
Shanshan Liu, National Institute for Materials Science and University of Tsukuba, Tsukuba University
Yunfei Long, Queen Mary, University of London
Shangyu Lou, San Diego State University and University of California, Santa Barbara
Jianfeng Lu, Nanjing university of Science & Technology
Guoqing Luo, University of Alberta
Haodi Ma, University of Florida
Khyati Mahajan, ServiceNow Inc
Harsh Maheshwari, Amazon
Ziqing Man, UCLA Computer Science Department, University of California, Los Angeles
Qianren Mao, Zhongguancun Laboratory, Beijing, P.R.China. and Beihang University
Chunyu Miao, University of Illinois at Chicago
Zhongtao Miao, The University of Tokyo
Ishani Mondal, University of Maryland, College Park
Noriki Nishida, AIST, National Institute of Advanced Industrial Science and Technology
Yusuke Oda, National Institute of Informatics and Nara Institute of Science and Technology
Hamidah Oderinwale, McGill University
Naoki Otani, Megagon Labs
Mingyu Ouyang, National University of Singapore
Matteo Paganelli, University of Modena and Reggio Emilia
Jeongsik Park, GE HealthCare
Dhaval C Patel, International Business Machines
Xinyu Pi, University of California, San Diego
Xin Quan, University of Manchester
Saman Rahbar, Dialpad AI , Nurty Technologies Inc, University of Calgary and University of British Columbia
Sourajit Saha, University of Maryland, Baltimore County
Tobias Schimanski, University of Zurich
Nisarg Shah, Uber
Arun Sharma, University of Minnesota - Twin Cities
Neelesh Kumar Shukla, Oracle
Vaibhavi Singh, New York University
Sebastian Stüker, Zoom Video Communications
Xiangguo Sun, Southeast University
Zexu Sun, Baidu
Zhouhao Sun, Harbin Institute of Technology
Jing Tang, Huazhong University of Science and Technology
Wei Tao, LIGHTSPEED

Jinlong Tian, National University of Defense Technology
Jannis Vamvas, University of Zurich
Benlu Wang, Department of Computer Science, University of Massachusetts at Amherst
Chenan Wang, College of William and Mary
Danqing Wang, CMU, Carnegie Mellon University
Haocheng Wang, The Hong Kong University of Science and Technology and ETHZ - ETH Zurich
Haoran Wang, Emory University
Juanyan Wang, Netradyne
Shanyong Wang, Xiaohongshu
Wei Wang, Nankai University
Xiaobo Wang, Beijing Institute for General Artificial Intelligence and University of Science and Technology of China
Yichen Wang, Georgia Institute of Technology
Zixuan Wang, Zhejiang University and Baidu
Xing Wei, Xi'an Jiaotong University
Tianhao Wu, Tianjin University
Yinjun Wu, Peking University
Yuankai Wu, Sichuan University
Zengqing Wu, Osaka University
Zhiheng Wu, Baidu
Meng Xiao, National University of Singapore and Chinese Academy of Sciences
Ke Xu, University of Liverpool
Pei Xu, Institute of automation, Chinese academy of science, Chinese Academy of Sciences
Zhen Xu, PCG Tencent
Jiangpeng Yan, E Fund Management Co., Ltd.
Ming Yan, Xinjiang University
Jinfa Yang, Fuzhou University
Tianyu Yang, Georg-August Universität Göttingen
Xin Yang, Southwestern University of Finance and Economics
Zhiyu Yao, Tsinghua University
Yangfan Ye, Harbin Institute of Technology
Ryota Yoshihashi, Institute of Science Tokyo and LY Corporation
Hang Yu, Shanghai University
Ruihong Zeng, Mohamed bin Zayed University of Artificial Intelligence
Chi Zhang, Beijing Institute of Technology
Han Zhang, Renmin University of China
Jinghui Zhang, Mohamed bin Zayed University of Artificial Intelligence
Liang Zhang, SUN YAT-SEN UNIVERSITY
Pingping Zhang, Dalian University of Technology
Shuyi Zhang, Wuhan University
Xitong Zhang, Qualcomm Inc, QualComm
Yunqi Zhang, Zhongguancun Laboratory
Zhong Zhang, University of Electronic Science and Technology of China
Ziqi Zhang, Institute of automation, Chinese academy of science, Chinese Academy of Sciences
Kaiyan Zhao, The University of Tokyo
Yang Zhao, Tsinghua University, Tsinghua University
Yi Zhao, Harbin Institute of Technology
Zhenjie Zhao, Nankai University
Zijian Zhao, The Hong Kong University of Science and Technology
Juepeng Zheng, SUN YAT-SEN UNIVERSITY
Guochen Zhou, City University of Hong Kong

Jiawei Zhou, Fudan University
Zirui Zhuang, Beijing University of Posts and Telecommunications

Table of Contents

A Human-Annotated Reference Corpus for Medication Information Extraction from MIMIC-IV Discharge Summaries
Provia Kadusabe, Abhishek Kaushik and Fiona Lawless 1

The Right Information Extraction Pipeline Depends on the Document: Accuracy-Energy Trade-offs for Small, Local Models
Christoph Marjan Markus Walser, Mauricio Fadel Argerich and Jonathan Fürst 12

Splitting Documents at Lower Cost: Multi-Split Boundary Decisions for LLM-Based Page Stream Segmentation
Nikhil Reddy Pottanigari, Sepideh Kharaghani, Saverio Vadacchino, Alejandro Posada and Ying Zhang 30

Text, Pixels, or Both? Evaluating Input Representations for Multimodal Document QA
Nikhil Reddy Pottanigari, Sepideh Kharaghani, Saverio Vadacchino, Alejandro Posada, Kurt MacDonald and Ying Zhang 46

AdaptiveDocRAG: Efficient Document Understanding with Multimodal Large Language Models
Surya Selvam and Anand Raghunathan 63

Most of a Document-VQA Score Is Answer Formatting
Srinivas Harish 73

Graph-Guided Passage Retrieval for Author-Centric Structured Feedback
Maitreya Prafulla Chitale, Ketaki Mangesh Shetye, Harshit Gupta, Manav Chaudhary, Manish Shrivastava and Vasudeva Varma 81

GraphER: An Efficient Graph-Based Enrichment and Reranking Method for Retrieval-Augmented Generation
Ruizhong Miao, Yuying Wang, Rongguang Wang, Chenyang Li, Tao Sheng, Sujith Ravi and Dan Roth 100

LongDA: Benchmarking LLM Agents for Long-Document Data Analysis
Yiyang Li, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Keerthiram Murugesan, Chuxu Zhang and Yanfang Ye 112

Reflect and Retrieve: Multimodal Agentic Retrieval with Reflection and Query Agents for Document Search
Tryambak Gangopadhyay, Md Sirajus Salekin, Yiyue Qian, Nkechinyere Nneka Agu, Shinan Zhang and Rinne Han 138

LLMs and Reasoning in Health Insurance Policy Documents: A Dataset and Argument Graph Extraction Study
Govind Arun, Sidharth N Krishna, Susie Xi Rao and Ajo Babu George 155

NutriChat: Evaluating Safety-Aware Retrieval-Augmented Generation for Nutrition Textbook Question Answering
Hadis Ghafouri 163

Measuring Artifact Reliance in Machine-Generated Text Detectors
Anastasia Voznyuk and Andrey Grabovoy 174

Graph Retrieval in Multimodal RAG for Biomedical Documents: A Pharmacokinetics and Pharmacodynamics Case Study
Primoz Kocbek, Katarina Meštrović, Nguyen Thu Huong, Ales Fajmut and Gregor Stiglic . . . 186

Q-STR: Diagnosing Query-Driven Text-to-Table Reasoning in Cricket Narratives
Ritam Upadhyay, Naman Ahuja, Rishabh Baral, Aparna Garimella and Vivek Gupta 196

Ontology-Guided Procedural Knowledge Extraction and Validation Case Study on the Lockout-Tagout Documents
Cennet Oguz, Noon Pokaratsiri Goldstein, Jörg Steffen, Reetam Biswas and Günter Neumann217

Perception, Layout, and Validation: Calibrated Confidence for Reliable Straight-Through Processing of Financial Documents
Yichao Jin, Yushuo Wang, Han Yuxuan, Ching Yee Sonia Kwan, Weiyang Song, Kent Chiu Jin Chun, Chong Hwee Wong, Tiong Kiat Wong, Kenneth Zhu Ke and Jingyuan Zhao 232

Tabstructor: Reconstructing Latent Borders in Borderless and Partially Ruled Tables
Alvin Sarraga Alon, Daniele De Rosa, Johannes Friedl and Riza Batista-Navarro 244

Silent Structured-Content Corruption in Document Parsing: A Span-Typed Diagnostic and a Released Cross-Parser Resource
Ruiyang Zhang 273

Sorry Robot, Happy Human: Vision-Language Models Read Only One of Two Legible Typographic Layers
Mert İncidelen, Yamen Kashkash, Asya Berker and Murat Aydogan 282

What Should Occupy the Context Budget? Latent Compression versus Budget-Matched Truncation in Document QA
Artem Cherepanov and Albina Burlova 287

RAG Components Are Not Independently Rankable
Vishnu Prasad Vijaya Kumar, Iaroslav Neverov, Pavel Chizhov, Anton Changalidis, Vaiva Kazanaviciute, Pierre-Carl Langlais, Anastasia Stasenko and Ivan P. Yamshchikov 303

LaTeXTrans Structure-Aware Translation of Complex LaTeX Documents
Ziming Zhu, Chenglong Wang, Haosong Xu, Shunjie Xing, Yifu Huo, Fengning Tian, Quan Du, Di Yang, Chunliang Zhang, Tong Xiao and JingBo Zhu 324

Did You CoSeCh?: Faithfulness- and Coherence-Aware Auditing of Content Selection in Poster Generation
Aaditya Deshpande, Savita Bhat and Shirish Karande 344

Look Where It Matters: High-Resolution Crops Retrieval for Efficient VLMs
Nimrod Shabtay, Moshe Kimhi, Sivan Harary, Artem Spector, Ehud Rivlin, Chaim Baskin, Raja Giryes and Eli Schwartz 357

DistilVDR: A Compact End-to-End Visual Document Retriever via Dual-Student Distillation
Zhuchenyang Liu, Ziyi Wang, Yao Zhang and Yu Xiao 385

Comparative Study of AI Agent Architecture Efficiency in Table Processing Tasks
Timur Gabbasov, Aleksandr Sandalov, Aleksandr Maksimov, Maria Khodorchenko and Nikolay Alekseevich Butakov 400

MaDOVE: a Dataset for Multilingual Document VQA Evaluation
Elio Musacchio, Lucia Siciliani, Pierpaolo Basile, Giovanni Semeraro, Mark Federico Zampedroni and Asia Beatrice Uboldi 409

Generating Edit-Inducing Questions for AI Research Manuscripts
Sebastian Antony Joseph, Zichao Wang, Jennifer Healey, Alexa Siu, Junyi Jessy Li and Ani Nenkova 428

Shared Task Papers

When the Evidence Is Right, the Math Is Easy: Understanding the Challenges of Document-Grounded Reasoning
Aftab Hossain Mikat and Tamjid Hasan Fahim 440

Acessilia at Dr.DocBench 2026: Agentic Document Parsing through Canonical Tree Adjudication
Marcelo Akira Inuzuka, Marcos Paulo C M Queiroz, Jhonata Fernandes Cordeiro and Eduardo Fraga Pereira 448

Evidence First, Arithmetic Second: A System Report and Failure Analysis for DOCSEM
Divya Godara and Sachin Gupta 460

Retrieve, Reason, Verify: A Modular Multi-Model System for DocSem
Pranavi Tadivalasa 465

A Cost-Efficient Vision-Text Cascade for Document QA: What Resolution Buys and What It Doesn't
Urjita Bedekar 477

A Human-Annotated Reference Corpus for Medication Information Extraction from MIMIC-IV Discharge Summaries

Provia Kadusabe[1], Abhishek Kaushik[2], Fiona Lawless[1]

[1]Dundalk Institute of Technology, Ireland
[2]Maynooth University, Ireland

{provia.kadusabe, fiona.lawless}@dkit.ie
abhishek.kaushik@mu.ie

Abstract

Medication information extraction from clinical notes supports medication reconciliation, pharmacovigilance, and clinical decision support, helping reduce medication errors and preventable patient harm. However, the linguistic variability and contextual dependence make medication mentions and their associated attributes difficult to identify consistently. Human-annotated reference corpora are therefore important for system development and rigorous evaluation, yet remain scarce. MIMIC-IV-Note provides a large-scale collection of discharge summaries but lacks a human-annotated, medication-specific reference corpus. We present a corpus of 305 MIMIC-IV-Note discharge summaries annotated by four annotators using a written guideline and adjudicated into a single reference version. The corpus covers seven entity types, a three-valued medication-status attribute, and five medication–attribute relation types. Pairwise inter-annotator agreement reached an exact-span F-measure of 0.896 and a character-level F-measure of 0.927. Medication mentions are split between medication-list sections (55.4%) and narrative sections (44.6%), with attribute information concentrated in the list sections (2.86 versus 0.59 attribute types per mention). A supervised entity-recognition baseline achieved a micro-averaged exact-span F-measure of 0.945 against the adjudicated reference on the held-out test set. Error analysis showed that most remaining errors involved span-boundary disagreements, particularly for the reason and duration attributes. We plan to release the corpus through PhysioNet to support reproducible development and evaluation of medication information extraction systems on MIMIC-IV-Note and to provide a curated human reference for assessing automatically generated medication labels.

Keywords: medication information extraction; clinical NLP; MIMIC-IV-Note; named entity recognition; relation extraction

1 Introduction

Medication information is central to clinical documentation. It is distributed across structured electronic health record fields and unstructured clinical notes. Structured medication lists and orders provide an accessible record of prescribed medications, but they do not necessarily capture the complete medication history. Clinical notes may contain more information such as medication changes that are not represented in structured fields, including medication initiation, discontinuation, and dose modification, together with contextual information such as temporality, certainty, and reason for the change (Mahajan et al., 2022, 2023).

Reliable extraction from unstructured clinical notes can support medication reconciliation and reduce medication-related harm (Killin et al., 2021), support pharmacovigilance and adverse drug event detection (Jagannatha et al., 2019; Henry et al., 2020), and provide structured inputs for clinical decision-support applications (Sutton et al., 2020).

However, medication information extraction from clinical text remains difficult because of abbreviations, shorthand, spelling variants, and domain-specific terminology. Determining medication status and temporality or linking attributes to the correct medication may also require surrounding context (Guo et al., 2024). Human-annotated reference corpora are therefore important for system development and rigorous evaluation.

Despite their importance, human-annotated clinical corpora remain scarce because privacy, ethical, and de-identification constraints restrict sharing beyond the institutions where the notes were produced (Wu et al., 2024). This limits reproducibility, direct system comparison, and assessment of generalisability across institutions.

MIMIC-IV-Note partially addresses the limited availability of clinical text by providing credentialed researchers with 331,794 de-identified dis-

The Proceedings of Workshop on Document Intelligence and Understanding, pages 1–11
October 24, 2026 ©2026 Association for Computational Linguistics

charge summaries linked to the underlying MIMIC-IV electronic health record (Johnson et al., 2023). The core release, however, does not include human medication annotations. A separate resource provides medication labels for 600 discharge summaries, but they were generated using a large language model and their reported evaluation was conducted against the 2009 i2b2 corpus rather than against human annotations of the released MIMIC-IV notes (Goel et al., 2023b,a). Thus, no publicly released, human-annotated, medication-specific reference corpus is currently available for developing and evaluating medication information extraction systems on MIMIC-IV-Note.

To address this gap, we construct a human-annotated reference corpus of 305 MIMIC-IV-Note discharge summaries. The corpus captures medication mentions, their associated attributes and status, and the relations linking attributes to medications. Our contributions are:

- A fully adjudicated, human-annotated corpus of 305 MIMIC-IV-Note discharge summaries, supporting medication and attribute named-entity recognition, medication–attribute relation extraction, and medication-status classification across medication-list and narrative sections.
- A transparent annotation protocol and reproducible release, including a written annotation guideline, inter-annotator agreement analyses for entities, medication status, relations, and medication grouping, patient-disjoint data splits, and code for reproducing the corpus analyses.
- An empirical analysis of the distribution of medication mentions and attributes across discharge summaries, showing that medication-list sections and narrative sections contain comparable numbers of medication mentions but substantially different levels of attribute information.
- A reproducible supervised baseline for medication and attribute named-entity recognition that combines transformer encoders with a conditional random field, providing a reference point for future work.

These contributions provide, to our knowledge, the first human-annotated, medication-specific reference corpus for MIMIC-IV-Note, and a reproducible benchmark for developing and evaluating medication information extraction systems on it.

2 Related Work

Medication information extraction has developed through a sequence of shared tasks and corpora that progressively expanded the representation of medication information in clinical text.

The 2009 i2b2 medication challenge established a major benchmark for medication information extraction from Partners HealthCare discharge summaries (Uzuner et al., 2010a). Its reference test set contains 251 summaries annotated for medication mentions, dose, route, frequency, duration, reason, and whether the information occurs in list or narrative text. The annotations were produced through a community process and achieved an F-measure above 0.90 against expert annotations (Uzuner et al., 2010b). The corpus established the core medication attributes adopted by many subsequent extraction tasks.

MADE 1.0 extended medication information extraction to pharmacovigilance using 1,089 longitudinal notes from 21 patients with cancer at the University of Massachusetts Memorial Hospital. The corpus annotates nine entity types and seven relation types, covering medications, indications, adverse drug events, and their associated attributes (Jagannatha et al., 2019).

The 2018 n2c2 adverse-drug-event and medication challenge introduced a corpus of 505 MIMIC-III discharge summaries annotated for nine concept types: drug, strength, form, dosage, frequency, route, duration, reason, and adverse drug event. It also defines eight relation types linking non-drug concepts to their associated medications (Henry et al., 2020). The corpus extends the i2b2 medication schema by integrating adverse-drug-event extraction.

The Contextualized Medication Event Dataset (CMED), used in the 2022 n2c2 shared task, extended medication information extraction to the contextual characterisation of medication changes (Mahajan et al., 2022, 2023). It uses notes from the 2014 i2b2/UTHealth corpus and annotates medication-change events by action, negation, temporality, certainty, and actor. CMED supports medication-timeline reconstruction but does not annotate medication attributes or link them to individual medication mentions.

Together, these corpora established the annota-

tion schemas and evaluation benchmarks that have supported medication information extraction. The i2b2, 2018 n2c2, and MADE corpora are based on source collections that predate the MIMIC-IV release. They therefore do not provide a human-annotated reference for evaluating medication information extraction on MIMIC-IV-Note.

MIMIC-IV-Note provides a more consistently maintained source of clinical text and remains available to credentialed researchers through PhysioNet (Johnson et al., 2023). For this collection, Goel et al. released i2b2-style medication labels for 600 discharge summaries, generated using a pipeline based on a large language model (Goel et al., 2023b,a). Their work demonstrates the potential of language models to reduce the human effort required for clinical information-extraction annotation. However, the reported medication evaluation uses the 2009 i2b2 reference, and no human evaluation of the released labels on MIMIC-IV-Note is reported.

To our knowledge, no publicly released human-annotated reference corpus has previously covered medication attributes and their relations in MIMIC-IV-Note discharge summaries. To address this gap, we develop a human-annotated corpus whose annotation schema builds on earlier medication information extraction corpora. It retains the core medication, dose, route, frequency, duration, and reason categories while extending the schema with allergy and medication status. Relative to i2b2, it analyses the document sections in which medications occur (medication-list versus narrative sections) rather than attaching a list-or-narrative field to each medication record, and uses explicit relations to link attributes to the medications they describe. Relative to the 2018 n2c2 schema, it adds an allergy entity and a medication-status attribute, and targets medication information rather than adverse drug events. Relative to CMED, it uses a three-valued medication-status attribute and adds the detailed medication attributes and medication–attribute relations that CMED does not annotate.

3 Corpus Construction

We constructed the corpus in five stages (Figure 1). We first selected documents by sampling from MIMIC-IV-Note, then developed the annotation schema and guideline. Annotation began with annotator training and proceeded to independent annotation. We then measured inter-annotator agreement on the independent annotations, and finally adjudicated and curated them to produce the final corpus.

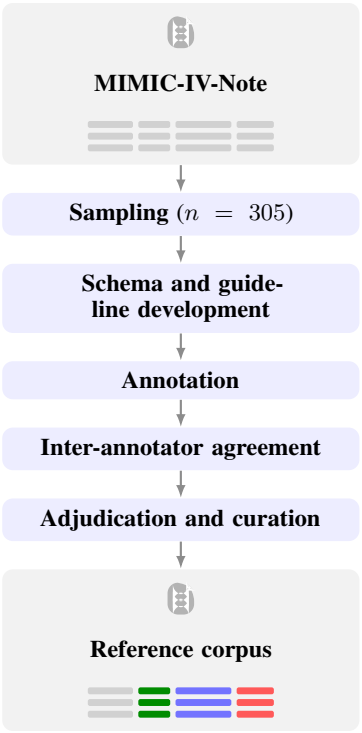


Figure 1: The corpus-construction pipeline.

3.1 Data Source and Sampling

The corpus is drawn from MIMIC-IV-Note v2.2 (Johnson et al., 2023), which contains 331,794 de-identified discharge summaries from the Beth Israel Deaconess Medical Center in Boston.

We sampled from the full set of discharge summaries. Notes shorter than 500 characters were removed. We stratified the remaining notes on two axes, document length and medication density, each split into three bands at the tertiles of the eligible notes, to give a nine-cell grid, and drew a roughly equal number of notes from each cell so that short documents with many medications and long documents with few are not underrepresented. The discharge-medication section served as a proxy for medication density, since it could be counted before any annotation existed. The tertile cuts fell at 8,271 and 11,678 characters for length and at 7 and 13 entries for density. Notes with zero entries in the discharge-medication section were retained, since medications may appear elsewhere in the note. Sampling drew 33 notes per cell with a fixed seed of 42, the remaining 8 notes going to the largest cell, to form a final corpus of 305.

3.2 Annotation Schema and Guideline

Our schema builds on the 2009 i2b2 medication challenge and extends it.

Entity types. Seven entity types are annotated:

- Medication: the name of a drug, including brand names, generic names, and active ingredients.
- Dose: the amount of a medication given in a single administration.
- Route: how the medication is administered.
- Frequency: how often the medication is taken.
- Duration: how long the medication is taken.
- Reason: the clinical indication for the medication.
- Allergy: the drug or substance a documented allergy is to.

Medication status. Each medication carries a status attribute:

- active: the default, left blank;
- discontinued: the text states the medication was stopped or held;
- neither: the medication is named as past history rather than a current one, was considered but not prescribed or was explicitly rejected, or belongs to someone other than the patient.

Attribute relations. A medication's dose, route, frequency, duration and reason are each linked to it by a relation: *has_dose*, *has_route*, *has_frequency*, *has_duration*, and *has_reason*. An attribute that applies to several medications is linked to each of them, so a single attribute span may carry more than one relation. An allergy is not a medication attribute, and therefore it carries no relation. Figure 2 shows a synthetic annotated example.

Guideline development. The annotation guideline was developed iteratively, beginning with the seven-field schema used in the 2009 i2b2 medication challenge. The lead annotator first annotated a small set of notes and revised the guideline to reflect the phenomena observed in the corpus. During annotator training, clarifying questions revealed cases that the guideline did not yet address. These cases were incorporated into a final version that was fixed before the main annotation began.

The guideline defines each entity type through inclusion and exclusion criteria and specifies standard abbreviations for route and frequency, permitted status values, and relation types. Attributes are annotated only when they are explicitly stated in the text; annotators do not infer routes, frequencies, or other information that is not documented. Medication mentions are annotated in both medication-list and narrative sections. Each medication mention, together with its linked attributes, constitutes a medication group.

3.3 Annotation Workflow

Annotators. The corpus was annotated by four annotators with backgrounds in computer science. Because the task involved extracting information explicitly stated in the clinical notes, the annotation guideline assumed no prior medical or linguistic expertise. We adopted the schema used in the i2b2 medication challenge, which conducted a community annotation experiment and found that non-expert annotators achieved agreement comparable to that of experts. In that experiment, the community annotations agreed with expert annotations at F above 0.90 (Uzuner et al., 2010b).

Annotator training. Before independent annotation, the four annotators were trained on the guideline using a shared set of five notes. These notes were retained in the final corpus but excluded from the independent annotation and agreement analyses. The remaining 300 notes were then distributed among the annotators, of which 41 were independently annotated by more than one annotator (Section 3.4).

Annotation platform and recommender. Annotation was carried out in INCEpTION (Klie et al., 2018), a web-based annotation platform. For notes assigned to a single annotator, a string-matching recommender was enabled to accelerate annotation. The recommender proposed spans whose surface forms matched text that the same annotator had previously annotated. These suggestions were not applied automatically; each was accepted or rejected by the annotator.

The recommender was disabled for notes annotated independently by multiple annotators. These notes were used to measure inter-annotator agreement.

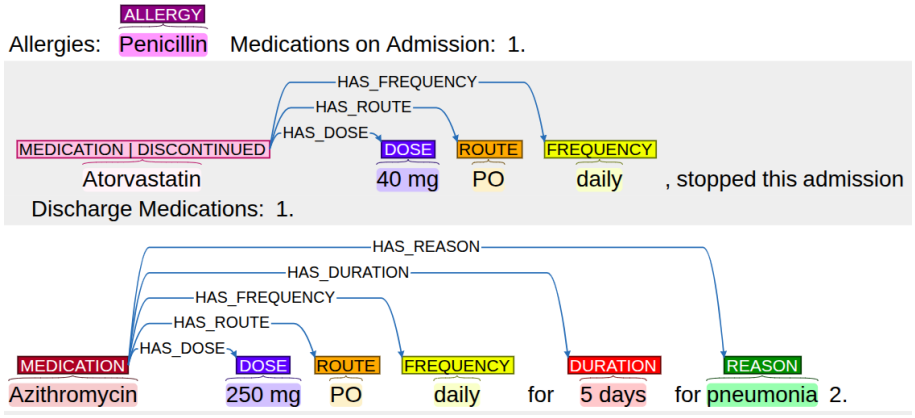


Figure 2: An example of an annotated note. The example is synthetic and contains no content from MIMIC.

3.4 Inter-Annotator Agreement

Agreement was measured from the independent annotations, before curation.

Metrics. Span agreement is reported as pairwise F-measure, since span-detection tasks provide no predefined set of negative units for estimating chance agreement (Hripcsak and Rothschild, 2005). For an entity type, the two annotators' spans form sets A and B, with duplicates removed, and

$$F(A, B) = \frac{2\,|A \cap B|}{|A| + |B|}. \qquad (1)$$

Under exact agreement, spans match only at identical character offsets. For character-level agreement, each set is replaced by the character positions its spans cover,

$$\mathrm{ch}(A) = \bigcup_{(b,e)\in A} \{\, b, b+1, \ldots, e-1 \,\}, \qquad (2)$$

and F is computed over $\mathrm{ch}(A)$ and $\mathrm{ch}(B)$, giving partial credit when annotators mark the same mention but disagree on its boundaries.

F is computed per entity type for each annotator pair on each shared note. A type absent from both annotations is excluded; a type marked by only one scores zero. The overall figure pools counts across types within a comparison (an annotator pair on a shared note) before computing F, and comparisons are averaged with equal weight. Confidence intervals for exact agreement use 2,000 note-level bootstrap samples.

Status agreement is computed over medication spans both annotators marked at identical offsets; as these fix the units and the label set is closed, Cohen's κ (Cohen, 1960) is reported with observed agreement. Group agreement is F over (span, medication) pairs, each medication paired with itself and each attribute with its parent, so an attribute matches only when both annotators mark it identically and link it to the same medication. Attachment agreement is the proportion of attribute spans both marked and linked that they assign to the same medication, conditional on both having marked the span.

Results. Forty-one notes were annotated independently by more than one annotator: 15 by two, 10 by three, and 16 by all four. Each pair of annotators sharing a note yielded one comparison, for a total of 141 comparisons. Overall agreement was 0.896 under exact matching and 0.927 at the character level (Table 1). Exact agreement exceeded 0.87 for medication, dose, route, and frequency. Duration, reason, and allergy had lower estimates and wider confidence intervals. The character-level gains for duration and reason show that boundary differences account for some, but not all, of their disagreement.

Route was the only entity type for which character-level agreement was lower than exact agreement. This reflects the length weighting of the character-level measure: unmatched route spans were typically longer written-out forms, whereas matched spans were often short abbreviations.

Across the pairwise comparisons, 4,603 matched medication-span instances were available for status agreement: observed agreement was 0.977 and Cohen's κ was 0.699. The lower κ reflects the strong class imbalance, with 95.9% of these instances labelled active.

Medication-group agreement was 0.872. Among attribute spans marked and linked by both annotators, 95.7% were assigned to the same medication. Thus, most group-level disagreement arose from

differences in span identification.

Entity type	Exact	95% CI	Char.	n
Medication	0.874	[0.844, 0.901]	0.916	41
Dose	0.946	[0.930, 0.961]	0.972	41
Route	0.925	[0.894, 0.952]	0.894	41
Frequency	0.892	[0.856, 0.925]	0.963	41
Duration	0.583	[0.441, 0.703]	0.657	21
Reason	0.566	[0.459, 0.674]	0.672	37
Allergy	0.653	[0.485, 0.839]	0.678	19
Overall	0.896	[0.876, 0.915]	0.927	41

Table 1: Pairwise inter-annotator agreement by entity type under exact and character-level matching; 95% bootstrap CIs.

3.5 Adjudication and Validation

Curation was performed using the INCEpTION curation view. For notes annotated by more than one annotator, annotations that matched exactly were treated as agreements, while discrepancies were highlighted for review. The lead annotator resolved these discrepancies according to the annotation guideline. Notes annotated by a single annotator were also reviewed against the guideline. This process produced one curated annotation layer for each note.

After curation, the corpus was checked using rule-based validation procedures derived from the guideline, and all flagged cases were reviewed manually and corrected where necessary. This produced the final corpus.

4 Corpus Analysis

4.1 Corpus Composition

The 305 notes contain 484,665 whitespace-delimited tokens, 35,097 entity spans, and 23,242 relations. Table 2 gives the distribution by entity type. Thus, although the corpus contains a relatively small number of documents, each note is annotation-dense, with a mean of 115.1 entity spans and 76.2 relations per note.

Medication mentions constitute 32.7% of all spans, with a mean of 37.6 per note; one note contains no medication mention. Duration and allergy together account for less than 2% of all spans. Of the 11,472 medication mentions, 370 are labelled discontinued and 290 neither; the remaining 94.2% have the default active status. Medication spans have a median of 1 token and dose spans 2, and no entity type has a median above 2.

Entity type	Spans	Per note
Medication	11,472	37.6
Dose	8,740	28.7
Frequency	6,764	22.2
Route	6,009	19.7
Reason	1,416	4.6
Allergy	420	1.4
Duration	276	0.9
All spans	35,097	115.1
Relations	23,242	76.2

Table 2: Corpus composition.

4.2 Medication Information by Document Section

Document sections were identified using corpus-derived header patterns. Candidate headers were strings ending in a colon at the beginning of a line or following two spaces. We distinguished section headers from internal subheadings by the presence of a preceding blank line: candidates were preceded by a blank line in over 62% of their occurrences or in under 36%, with none in between, so this cutoff was set by the corpus itself. This procedure yielded 25 recurring section headers, including explicit variants for headers partially affected by de-identification. Variant headers were mapped to canonical section labels, and every medication mention was assigned to one of the resulting sections; none occurred before the first recognised header.

Of all medication mentions, 55.4% occur in the two medication-list sections, medications on admission and discharge medications, and 44.6% in the narrative sections. The largest concentrations outside the lists are in the brief hospital course (18.8%), history of present illness (7.8%), pertinent results (6.9%), discharge instructions (6.8%), and past medical history (3.8%).

The number and type of attributes linked to a medication differ substantially between medication-list and narrative sections. Overall, 31.3% of medication mentions have no linked attributes, and this proportion differs between the two (Table 3).

In the medication-list sections, a medication mention carries a mean of 2.86 distinct attribute types, usually dose, route, and frequency; in the narrative sections the mean is 0.59, and 65.5% of mentions are bare medication names. Detailed medication information is thus concentrated in the medication-list sections, even though the two contain comparable numbers of mentions.

Attribute types on a medication	List	Narrative
0 (bare medication name)	3.8%	65.5%
1	3.7%	17.2%
2	10.7%	11.2%
3	66.9%	5.1%
4 or 5	14.9%	0.9%
Mean	2.86	0.59
Medications	6,361	5,111

Table 3: Attribute types per medication mention in list and narrative sections.

5 Baseline Experiments

5.1 Data

We evaluated on a fixed training, development, and test partition, summarised in Table 4, which we plan to release with the corpus. The partition is stratified using the length and medication-density grid used to sample the corpus, preserving a comparable mix of document types across splits. It is also patient-disjoint: all notes from a patient are assigned to the same split, preventing evaluation on patients represented in the training data. The five shared annotator-training notes are included in the training set.

Split	Notes	Tokens	Medications	Spans
Train	185	292,000	6,794	20,860
Dev	62	99,163	2,534	7,526
Test	58	93,502	2,144	6,711

Table 4: Fixed splits, stratified and patient-disjoint.

5.2 Setup

We established a supervised baseline for medication and attribute named-entity recognition using a transformer encoder with a conditional random field over BIO tags. Notes exceeding the encoder limit are divided into 512-token windows that overlap by 128 tokens. For tokens appearing in multiple windows, predictions are retained from the window in which the token is furthest from a boundary, and character-level spans are reconstructed once per note.

We fine-tuned Bio_ClinicalBERT (Alsentzer et al., 2019) and PubMedBERT (Gu et al., 2021) using AdamW at a learning rate of 3×10^{-5} and an effective batch size of 32 (batch size 8 with gradient accumulation of 4), for up to 15 epochs; checkpoint selection and early stopping (patience 3) used the development-set exact-span F-measure. Performance was evaluated on the held-out test set using the exact and character-level criteria defined in Section 3.4, with counts pooled across notes. Each configuration was run with three random seeds; we report the mean across seeds, with the standard deviation for the overall figures.

5.3 Results

The two encoders were comparable, with PubMedBERT marginally higher overall (Table 5): mean exact-span and character-level F-measures of 0.945 and 0.967, compared with 0.941 and 0.966 for Bio_ClinicalBERT. Exact-span F-measure exceeded 0.93 for medication, dose, route, and frequency with both encoders. Performance was lower for duration and reason, and PubMedBERT was higher on reason and allergy.

The baseline exact-span F-measure was higher than the pairwise inter-annotator agreement reported in Section 3.4, but the values are not directly comparable. The model was evaluated against a single adjudicated reference with counts pooled across the test set, whereas agreement was averaged over pairwise note-level comparisons between independent pre-adjudication annotations.

5.4 Data-Split Sensitivity and Learning-Curve Analysis

To assess sensitivity to the fixed partition, we regenerated the patient-disjoint, stratified 60/20/20 split with ten random seeds, including the released split, and retrained both encoders on each. The training seed was held fixed, so variation reflects split choice rather than the training-seed variation reported in Table 5. Across splits (Figure 3), mean exact-span F-measure was 0.936 (95% CI [0.931, 0.941]) for Bio_ClinicalBERT and 0.940 ([0.938, 0.943]) for PubMedBERT, with ranges of 0.922–0.942 and 0.931–0.944, respectively. PubMedBERT was marginally but significantly higher (mean difference 0.004, 95% CI [0.001, 0.007]; two-sided paired t-test, $t(9)$, $p = 0.010$).

We also examined whether the corpus is large enough to train a model by retraining on 25, 50, 75, and 100% of the training set with development and test fixed (Figure 4). Performance improved most at smaller training sizes and then flattened. From 139 to 185 notes, exact-span F-measure increased from 0.936 to 0.941 for Bio_ClinicalBERT and from 0.938 to 0.944 for PubMedBERT. Although the curve has not fully saturated, the corpus is already in a region of diminishing returns.

	Bio_ClinicalBERT		PubMedBERT	
Entity type	**Exact**	**Char.**	**Exact**	**Char.**
Medication	0.938	0.975	0.939	0.976
Dose	0.949	0.978	0.952	0.981
Route	0.970	0.961	0.972	0.962
Frequency	0.963	0.980	0.964	0.979
Duration	0.719	0.791	0.701	0.707
Reason	0.756	0.829	0.796	0.838
Allergy	0.916	0.956	0.951	0.972
Overall	0.941±0.001	0.966±0.001	0.945±0.003	0.967±<0.001

Table 5: Baseline entity-recognition F-measure on the 58-note test set, averaged over three seeds.

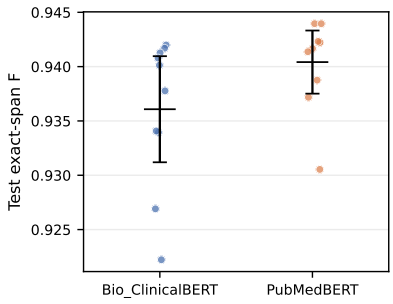


Figure 3: Exact-span F-measure across ten patient-disjoint stratified splits; bars show 95% CIs.

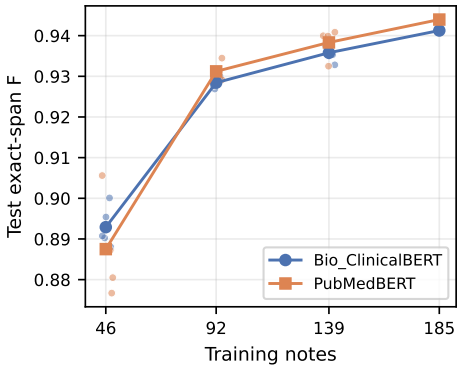


Figure 4: Exact-span F-measure by training-set size, with development and test sets fixed.

5.5 Error Analysis

We compared predicted and reference spans on the test set, categorising non-exact matches as boundary, type, missed, or spurious errors (Table 6). Type errors were rare (7 for each encoder), while most errors involved span detection or boundary selection. Boundary errors accounted for 168 of 306 non-exact reference spans for Bio_ClinicalBERT and 173 of 364 for PubMedBERT, which is consistent with the higher character-level than exact-span agreement observed between annotators.

The encoders had similar overall F-measures but different error patterns. Bio_ClinicalBERT produced more spurious and fewer missed spans, whereas PubMedBERT showed the reverse pattern. Errors were relatively uncommon for medication, dose, route, and frequency, but reason and duration remained more difficult. Thus, the high aggregate F-measure is driven mainly by frequent entity types, while performance on the rarer reason and duration categories remains less reliable, indicating that extraction of these more context-dependent attributes is not yet solved.

6 Discussion

The corpus demonstrates that medication information in discharge summaries is not confined to dedicated medication-list sections. The two medication-list sections and the narrative sections contain comparable numbers of medication mentions, but the information associated with those mentions differs substantially. Mentions in the lists contain a mean of 2.86 attribute types, compared with 0.59 in the narrative. A system restricted to the medication-list sections would therefore recover only about half of all medication mentions. Although it would capture most mentions documented with a complete dose, route, and frequency, it would miss those recorded elsewhere, particularly in the brief hospital course, history of present illness, and discharge instructions sections. Because medication reconciliation and related safety tasks require a complete medication record, the lists alone are insufficient. By annotating medication mentions wherever they occur, the corpus supports the development and evaluation of systems that process the entire note.

The agreement and baseline results together clarify which aspects of the task are well defined and which remain challenging. Agreement is high for medication, dose, route, and frequency, which are

		Bio_ClinicalBERT				PubMedBERT			
Entity type	**Ref.**	**Bound.**	**Type**	**Miss**	**Spur.**	**Bound.**	**Type**	**Miss**	**Spur.**
Medication	2,144	54	1	48	90	60	0	57	93
Dose	1,685	52	1	6	62	55	1	18	54
Route	1,163	11	1	19	32	7	1	30	19
Frequency	1,309	20	3	13	51	16	4	23	46
Duration	56	4	0	7	16	2	1	17	7
Reason	295	26	1	38	46	32	0	39	35
Allergy	59	1	0	0	10	1	0	0	3
Overall	6,711	168	7	131	307	173	7	184	257

Table 6: Test-set error counts by entity type. Ref.: reference; Bound.: boundary; Type: type; Miss: missed; Spur.: spurious.

frequent and have short, clearly bounded surface forms. It is lower for duration, reason, and allergy, which occur less often and have less clearly defined span boundaries. The improvements observed under character-level evaluation for duration and reason suggest that some disagreement concerns span extent rather than the underlying entity identity. The supervised baseline follows the same pattern: it achieves exact-span F-measures above 0.93 for the frequent entity types but performs substantially worse on duration and reason. These categories therefore appear to be intrinsically difficult for both annotation and modelling.

Our baseline covers entity recognition; the relation- and status-classification tasks the corpus defines, together with a direct evaluation of large language model annotations against the human reference, remain for future work.

A cross-dataset evaluation against an established medication corpus such as the 2018 n2c2 dataset would provide additional evidence about how well models trained on this corpus generalise across clinical-note collections and remains an important direction for future work.

7 Conclusion

We introduced the first human-annotated corpus for medication information extraction from MIMIC-IV discharge summaries. We reported its inter-annotator agreement and a supervised baseline that achieved an exact-span F-measure of 0.945 on the held-out test set, and we plan to release the corpus through PhysioNet together with its annotation guideline, patient-disjoint data splits, and analysis code. Unlike resources restricted to dedicated medication-list sections, it annotates medication mentions and their attributes throughout the entire note. By providing an in-domain human reference where previous resources relied on automatically generated labels, the corpus will enable the development, evaluation, and validation of medication information extraction systems for MIMIC-IV-Note. In turn, these systems can support medication reconciliation and other safety-critical applications aimed at reducing medication errors and preventable patient harm.

Limitations

The corpus is derived from a single hospital system, is limited to English-language discharge summaries, and may therefore not generalise to other institutions, note types, or languages. In addition, the sample was stratified rather than designed to be representative of MIMIC-IV-Note as a whole. Consequently, the observed entity frequencies and the proportion of medication mentions occurring in medication lists (55.4%) should be interpreted as properties of this sample rather than population estimates.

We intend the corpus, together with its annotation guideline, data splits, and analysis code, to serve as a reproducible basis for extension to further institutions, note types, and languages.

Access and Licensing

We plan to release the corpus through PhysioNet under the PhysioNet Credentialed Health Data License, following the access conditions for MIMIC-IV-Note.

Ethics Statement

The corpus is a secondary annotation layer over MIMIC-IV-Note, a de-identified clinical database whose collection and release received institutional review board approval and whose distribution is

managed by PhysioNet. Access requires PhysioNet credentialing and acceptance of the Credentialed Health Data Use Agreement; any release of the annotations would inherit these conditions and go only through the same repository. The annotation work used the de-identified text as distributed, made no attempt to re-identify patients, and reproduces no note text in this paper, where any examples used are synthetic.

Acknowledgments

The authors would like to thank the data annotators for their time, and valuable contributions to the annotation process. The authors also acknowledge the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this manuscript.

This research was funded through the CREATE-DkIT project, supported by the HEA TU-Rise program and co-financed by the Government of Ireland and the European Union through the Southern, Eastern & Midland Regional Program of the ERDF 2021-27 and the Northern & Western Regional Programme 2021–27.



References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Akshay Goel, Almog Gueta, Omry Gilon, Sofia Erell, and Amir Feder. 2023a. Medication Extraction Labels for MIMIC-IV-Note Clinical Database. *PhysioNet*. Version 1.0.0.

Akshay Goel, Almog Gueta, Omry Gilon, Chang Liu, Sofia Erell, Lan Huong Nguyen, Xiaohong Hao, Bolous Jaber, Shashir Reddy, Rupesh Kartha, Jean Steiner, Itay Laish, and Amir Feder. 2023b. Llms accelerate annotation for medical information extraction. In *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 82–100. PMLR.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

Yuting Guo, Yao Ge, and Abeed Sarker. 2024. Detection of medication mentions and medication change events in clinical notes using transformer-based models. In *Studies in health technology and informatics*, volume 310, page 685.

Sam Henry, Kevin Buchan, Michele Filannino, Amber Stubbs, and Ozlem Uzuner. 2020. 2018 n2c2 shared task on adverse drug events and medication extraction in electronic health records. *Journal of the American Medical Informatics Association*, 27(1):3–12.

George Hripcsak and Adam S Rothschild. 2005. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American medical informatics association*, 12(3):296–298.

Abhyuday Jagannatha, Feifan Liu, Weisong Liu, and Hong Yu. 2019. Overview of the first natural language processing challenge for extracting medication, indication, and adverse drug events from electronic health record notes (made 1.0). *Drug safety*, 42(1):99–111.

Alistair Johnson, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. 2023. MIMIC-IV-Note: Deidentified free-text clinical notes. *PhysioNet*. Version 2.2.

Lauren Killin, Areej Hezam, Kelly K Anderson, and Blayne Welk. 2021. Advanced medication reconciliation: a systematic review of the impact on medication errors and adverse drug events associated with transitions of care. *The Joint Commission Journal on Quality and Patient Safety*, 47(7):438–451.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

Diwakar Mahajan, Jennifer J Liang, and Ching-Huei Tsou. 2022. Toward understanding clinical context of medication change events in clinical narratives. In *AMIA Annual Symposium Proceedings*, volume 2021, page 833.

Diwakar Mahajan, Jennifer J Liang, Ching-Huei Tsou, and Özlem Uzuner. 2023. Overview of the 2022 n2c2 shared task on contextualized medication event extraction in clinical notes. *Journal of biomedical informatics*, 144:104432.

Reed T Sutton, David Pincock, Daniel C Baumgart, Daniel C Sadowski, Richard N Fedorak, and Karen I Kroeker. 2020. An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ digital medicine*, 3(1):17.

Özlem Uzuner, Imre Solti, and Eithon Cadag. 2010a. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518.

Özlem Uzuner, Imre Solti, Fei Xia, and Eithon Cadag. 2010b. Community annotation experiment for ground truth generation for the i2b2 medication challenge. *Journal of the American Medical Informatics Association*, 17(5):519–523.

Jiageng Wu, Xiaocong Liu, Minghui Li, Wanxin Li, Zichang Su, Shixu Lin, Lucas Garay, Zhiyun Zhang, Yujie Zhang, Qingcheng Zeng, Jie Shen, Changzheng Yuan, and Jie Yang. 2024. Clinical text datasets for medical artificial intelligence and large language models: A systematic review. *NEJM AI*, 1(6).

The Right Information Extraction Pipeline Depends on the Document: Accuracy–Energy Trade-offs for Small, Local Models

Christoph Walser
Zurich University of Applied Sciences, Switzerland
walchr01@students.zhaw.ch

Mauricio Fadel Argerich
Universidad Politécnica de Madrid, Spain
mauricio.fadel@alumnos.upm.es

Jonathan Fürst
Zurich University of Applied Sciences, Switzerland
jonathan.fuerst@zhaw.ch

Abstract

Whether an information extraction pipeline should process page images or parsed text depends on the document, and the answer flips across the layout spectrum. We study this trade-off under a constraint that rules out (closed) cloud services: privacy-sensitive documents processed on-premise by small ($\leq$ 8B parameter) text-only and vision–language models, evaluated on both accuracy and energy over a design space spanning input representation, model family, and inference configuration. Benchmarking on the near-plain-text Kleister-NDA contracts and the layout-rich VRDU forms, we find that batching is the dominant energy lever, cutting energy per page by 38–85% at no cost in accuracy, while FP8 quantization saves 27–32% when requests are served one at a time but less than 1 mWh per page (9–19%) once batching is applied. Preprocessing dominates what remains: neural OCR costs 17$\times$ more energy per page than classical OCR and never reaches the Pareto frontier. Which representation wins flips with the type of document: vision–language models on layout-rich documents and small text-only models with a cheap parser on near-plain text, where they are both more accurate and cheaper than any vision–language configuration. Our work yields concrete guidelines for energy-efficient, privacy-compliant local information extraction.

1 Introduction

Documents in practice span a wide layout spectrum. At one end sit layout-rich documents (e.g., registration forms, invoices), where the meaning of a value depends on its position in a two-dimensional arrangement of boxes and tables. At the other sit near-plain text documents, such as contracts and reports, that encode almost nothing in their visual structure (Ding et al., 2025; Watanabe et al., 1995). Extraction pipelines are not indifferent to this: a model that reads page images directly has an advantage where layout carries meaning, and pays a cost for a modality it does not need when processing near-plain text documents.

Large Language Models (LLMs) and Vision Language Models (VLMs) have transformed key information extraction (KIE), replacing rigid rule-based heuristics (Luo et al., 2024; Colakoglu et al., 2025). However, their deployment remains constrained: Large transformer architectures demand computational resources that resource-constrained settings cannot supply (Caravaca, 2026), and routing sensitive documents to third-party providers risks data leakage and frequently violates data protection regulations in domains like finance and healthcare (Ding et al., 2025). Both pressures push toward small, locally deployable models running on-premise under tighter data governance. We therefore propose KIE should be considered *not as a prediction problem alone but as a multi-objective one, balancing extraction quality against end-to-end energy*, across small ($\leq$ 8B parameter) general-purpose LLMs, VLMs, and specialized architectures.

Following this proposal, we benchmark pipelines at both ends of the layout spectrum: the near-plain-text Kleister-NDA contracts (Stanisławek et al., 2021) and the densely structured VRDU registration forms (Wang et al., 2023). The design space covers visual-first processing on raw page images and text-first processing via OCR, across model families and inference configurations. Holding the model, parser and serving configuration fixed across two corpora that differ in both layout and length is what lets us show that the preferred pipeline inverts between them. The main contributions of our work are:

- A systematic comparison of local document IE pipelines from an accuracy–energy perspective.
- An analysis of how preprocessing choices (na-

The Proceedings of Workshop on Document Intelligence and Understanding, pages 12–29
October 24, 2026 ©2026 Association for Computational Linguistics

tive vision vs. OCR vs. parser) impact both total pipeline energy and downstream extraction accuracy.

- An evaluation of deployment optimizations, showing that request batching reduces energy per page by 38–85% without harming extraction quality, and that the benefit of FP8 quantization shrinks roughly tenfold once batching is in place.
- Practical deployment recommendations for building accurate, energy-efficient, and privacy-compliant IE pipelines.

All configurations and code to reproduce our results are available at `https://github.com/chrewbroccoli/local-ie-energy`.

2 Related Work

A growing body of work measures the energy of large language model (LLM) inference and proposes strategies for sustainable deployment (Luccioni et al., 2024; Fernandez et al., 2025; Poddar et al., 2025; Argerich and Patiño-Martínez, 2024). Luccioni et al. (2024) show that inference, not training, dominates the lifetime energy cost of widely deployed models, and that cost varies by orders of magnitude across task types. Fernandez et al. (2025) show that optimizations such as batch size, decoding strategy, and serving stack (e.g., PyTorch, CUDA Graphs, vLLM) are highly sensitive to workload geometry and hardware, and that appropriate configurations can cut total energy use by up to 73%. Complementary benchmarks measure energy per request across models and concurrency levels, finding that cost per request falls as batch size and concurrency increase (Poddar et al., 2025; Pronk and Zhao, 2025). Watt Counts (Argerich et al., 2026a) extends such measurements to 50 LLMs on 10 GPU architectures and shows that the most energy-efficient GPU differs across models and deployment scenarios, while WattGPU (Argerich et al., 2026b) predicts inference power and latency on unseen GPUs and LLMs from public specifications alone. Bench360 (Stuhlmann et al., 2026) provides a framework for profiling local LLM inference with custom tasks. Argerich and Patiño-Martínez (2024) advocate task-agnostic guidelines: select efficient models, use quantized variants where available, and increase batch size until GPU utilization saturates. Delavande et al. (2026) qualify this, showing that lower-precision formats gain most in compute-bound prefill phases while dequantization overheads can negate the savings, and that batching improves efficiency across all inference phases by reducing GPU idle time. These findings motivate our exploration of small, locally deployable models, FP8 quantization, and vLLM-backed batch inference.

LLM-based key information extraction (KIE) has advanced from classic text-only IE to layout-aware and multimodal document understanding (Xu et al., 2024; Ding et al., 2025). Surveys of generative IE summarize prompting and instruction-tuning strategies and show that general-purpose LLMs can match specialized architectures on many IE tasks (Xu et al., 2024). LayoutLLM introduces layout instruction tuning so that LLMs can process serialized two-dimensional document structures (Luo et al., 2024), building on earlier work that combines text, layout, and visual cues for extraction from complex business documents (Katti et al., 2018; Denk and Reisswig, 2019; Xu et al., 2020; Borchmann et al., 2021, 2025; Stanisławek et al., 2021; Wang et al., 2023). A parallel line removes the text-extraction stage entirely: OCR-free architectures such as Donut (Kim et al., 2022) and Pix2Struct (Lee et al., 2023) read page images directly, a strategy today's general-purpose VLMs inherit and that we evaluate here from an energy perspective. At the complex end of the spectrum, agentic pipelines built on large cloud models plan and decompose extraction from dense, multi-page, multilingual regulatory documents (Colakoglu et al., 2026). We bridge these lines by measuring accuracy and end-to-end energy across the layout spectrum rather than at a single point on it, spanning OCR technologies, small text-only and multimodal LLMs, specialized layout models, and batch- and quantization-aware vLLM inference.

3 Design Space of Resource-Aware IE with Small Local LLMs

Figure 1 summarizes the design space: for a given document and schema we need to decide the input representation, the model and the serving configuration under energy and accuracy constraints.

3.1 Task Definition

We define the resource-aware KIE task as a mapping from a document and a system configuration to a joint output space of structured data and com-

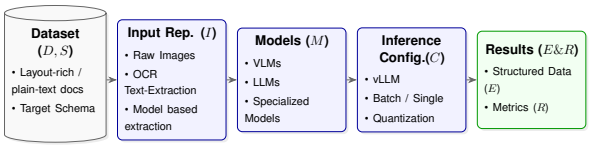


Figure 1: Overview of the benchmarking design space. We investigate the dimensions (D, S, I, M, C) to evaluate the trade-offs between extracted structured information (E) and resource consumption (R).

putational cost:

$$\text{KIE}_{\text{res}} : (D, S, I, M, C) \rightarrow (E, R)$$

On the input side, D is the document set and S the target schema of fields to populate; I is the input representation, either raw page images or the output of a text-extraction pipeline; M is the model, whether text-only LLM, VLM, or specialized layout model; and C is the inference configuration, here batch size and numeric precision. On the output side, E is the extracted structured data, validated against S, and R is the measured resource cost, here energy per page.

3.2 Dimensions of the Design Space

The three dimensions interact. **Input representation** (I) determines how a document is exposed to the model: layout is preserved implicitly through pixels (visual-first) or explicitly through serialized, layout-aware text from a parser or OCR engine (text-first). **Model architecture** (M) determines which information pathways are available: a VLM consumes page images directly, a text-only LLM depends entirely on the upstream parser, and a specialized layout model additionally requires word-level bounding boxes. **Inference configuration** (C) determines how the serving engine allocates memory and schedules requests, fixing the achievable hardware utilization. These choices are not independent: bypassing the parser removes an energy term but shifts cost into the model, and the batch size that saturates a 2B VLM differs from the one that saturates an 8B text model.

Each configuration is scored on a dual objective: extracted structures are validated against the target schema for accuracy (E), and the full execution is profiled for energy (R). We measure the entire pipeline, so parsing energy is charged to the configurations that incur it.

4 Methodology & Experimental Setup

The evaluation proceeds in three phases, one per dimension of the design space, each addressing one research question.

RQ1 Baselines. *How do small VLMs, text-only LLMs with basic OCR, and specialized models differ in their accuracy–energy profiles on text-heavy and layout-rich documents under a simple, unoptimized serving configuration?* We evaluate the three model categories using single-request inference: vision-language models processing raw images directly, text-only LLMs using Tesseract OCR as the baseline text input, and specialized layout models (Arctic-TILT and NuExtract).

RQ2 Input Representation. *How do different document parsing strategies — embedded-text parsers, classical OCR, and neural OCR — affect end-to-end energy and extraction accuracy, and how does this depend on whether documents are born-digital or scanned and layout-rich?* For the text-only models we substitute the Tesseract baseline with Docling and DeepSeek-OCR 2, measuring the trade-off between text-extraction quality and the energy overhead of each mechanism.

RQ3 Inference Optimization. *How do batch size and FP8 quantization shape the accuracy–energy trade-off?* We sweep batch size and precision under vLLM across all pipelines and measure the effect on energy per page and on extraction quality.

4.1 Models

We evaluate three model categories: vision language models (VLMs), text-only LLMs, and specialized layout models. Table 1 lists them with their modality and evaluation mode.

4.2 Datasets

We evaluate on two datasets: the text-heavy Kleister-NDA (Stanisławek et al., 2021) and the visually complex VRDU Registration (Wang et al., 2023), with an example of each in Figure 2. Information in VRDU is carried by 2D spatial arrangements and visual elements such as tables, whereas Kleister-NDA documents have minimal visual structure and follow a one-dimensional textual flow. VRDU ships its own baseline OCR text, which we use as an additional reference point when ranking parsers.

Extraction is evaluated on 500 VRDU Registration documents (up to six schema fields per document) and on 337 Kleister-NDA documents (up

Table 1: Evaluated models. Only Arctic-TILT saw a target dataset during training; all others are zero-shot.

Model	Modality	Eval. Mode
Specialized Models		
Arctic-TILT (Borchmann et al., 2025)	OCR+Layout	In-domain
NuExtract-2.0-4B (NuMind, 2026)	VL	Zero-shot
General-Purpose Models		
Qwen3-VL-Instruct (2B, 4B, 8B) (Qwen Team, 2025b)	VL	Zero-shot
Llama-3.2-Instruct (1B, 3B) (Grattafiori et al., 2024)	Text	Zero-shot
Ministral-3-3B (Mistral AI, 2025)		
Mistral-7B (Jiang et al., 2023)		
Qwen3 (0.6B, 1.7B, 4B, 8B) (Qwen Team, 2025a)		

to four fields: `effective_date`, `jurisdiction`, `party`, `term`). In both cases only fields present in the ground truth of a given document are requested. Parsing energy (Section 4.3) was profiled over 500 documents per dataset and combined with inference energy before normalization.

The two corpora also differ in length, which determines how we report energy. Over the same 500 profiled documents, Kleister-NDA runs to 2934 pages against 915 for VRDU, i.e. 5.87 against 1.83 pages per document. Energy per document would therefore conflate the cost of processing a page with the number of pages a document happens to have, and the two datasets could not be compared on it. We report energy *per page* throughout.

Length is also a confound in its own right: the low-layout corpus is the long one, so layout complexity and document length co-vary between our two datasets and cannot be separated by comparing them. Normalizing by page removes length from the energy axis but not from the experiment, and vision runs additionally see at most the first 10 (Kleister-NDA) or 5 (VRDU) pages of a document. We therefore read the results as a contrast between two *document types* — short, layout-rich forms and long, near-plain-text contracts — rather than as an isolated effect of layout; Section 5.4 examines what the per-page energy data can and cannot say about length.

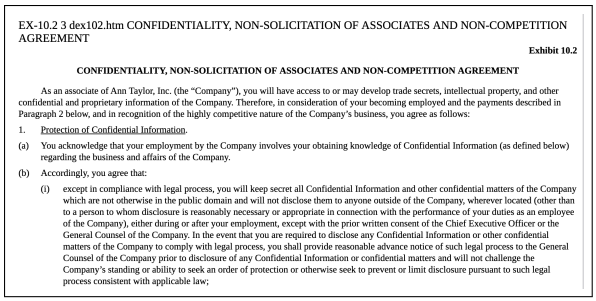
One asymmetry needs to be stated up front. Arctic-TILT is released already fine-tuned on Kleister-NDA (Borchmann et al., 2025), so on that dataset it is the only model in our study that has seen the target corpus during supervised training, and part of our evaluation set is likely to overlap with its training data. This is a property of the released checkpoint rather than of our protocol, and we do not attempt to correct for it. We report Arctic-TILT's Kleister-NDA results for reference, mark them as in-domain in every table and figure, and exclude them whenever we identify a best configuration or compute a Pareto frontier on that dataset. On VRDU, which Arctic-TILT has not been trained on, it is treated like every other model.

4.3 Document Parsing

We compare three parsers: Tesseract (Smith, 2007) (CPU), Docling (Livathinos et al., 2025) (CPU + GPU acceleration), and DeepSeek-OCR 2 (DeepSeek-AI, 2026) (GPU). We profile the PDF-to-Markdown conversions with CodeCarbon (Courty et al., 2024), using multi-processing for the conversion scripts and batched inference for the GPU-accelerated parser. Each parser was run over 500 documents per dataset. So that the idle power of the L4 is not charged to a CPU-bound pipeline, GPU tracking is disabled in CodeCarbon for Tesseract (`gpu_ids=[]`), leaving only its CPU and RAM energy; Docling and DeepSeek-OCR 2 run on the GPU and are charged for it. Arctic-TILT additionally requires word-level bounding boxes, which DeepSeek-OCR 2 does not provide; we interpolate them from its paragraph-level output (Appendix A.1). The parsers do not agree on page segmentation, so per-page figures are computed against each tool's own page count (Table 2).

4.4 Inference

Inference for both text-only and vision-language models ran on a single NVIDIA L4 GPU using vLLM (Kwon et al., 2023). Because the Chat Completions API is request-oriented rather than a native offline batching interface, we issue concurrent asynchronous requests and let vLLM batch them internally. Quantization is common in resource-constrained deployments, so we evaluate each model at FP16 and FP8. Since batch size improves resource utilization and reduces energy (Argerich and Patiño-Martínez, 2024), we evaluate batch sizes of 1, 5, 10, 20, 40 and 60, subject to memory availability. Decoding follows each model family's recommended sampling defaults, with reasoning traces disabled for the Qwen3 models; the full parameters, and evidence that no run was limited by



EX-10.2 3 dex102.htm CONFIDENTIALITY, NON-SOLICITATION OF ASSOCIATES AND NON-COMPETITION AGREEMENT

Exhibit 10.2

CONFIDENTIALITY, NON-SOLICITATION OF ASSOCIATES AND NON-COMPETITION AGREEMENT

As an associate of Ann Taylor, Inc. (the "Company"), you will have access to or may develop trade secrets, intellectual property, and other confidential and proprietary information of the Company. Therefore, in consideration of your becoming employed and the payments described in Paragraph 2 below, and in recognition of the highly competitive nature of the Company's business, you agree as follows:

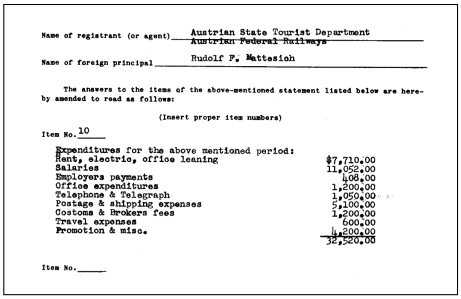
1. Protection of Confidential Information.

(a) You acknowledge that your employment by the Company involves your obtaining knowledge of Confidential Information (as defined below) regarding the business and affairs of the Company.

(b) Accordingly, you agree that:

(i) except in compliance with legal process, you will keep secret all Confidential Information and other confidential matters of the Company which are not otherwise in the public domain and will not disclose them to anyone outside of the Company, wherever located (other than to a person to whom disclosure is reasonably necessary or appropriate in connection with the performance of your duties as an employee of the Company), either during or after your employment, except with the prior written consent of the Chief Executive Officer or the General Counsel of the Company. In the event that you are required to disclose any Confidential Information or other confidential matters of the Company to comply with legal process, you shall provide reasonable advance notice of such legal process to the General Counsel of the Company prior to disclosure of any Confidential Information or confidential matters and will not challenge the Company's standing or ability to seek an order of protection or otherwise seek to prevent or limit disclosure pursuant to such legal process consistent with applicable law;

(a) Example of Document from Kleister-NDA Dataset



Name of registrant (or agent) Austrian State Tourist Department ~~Austrian Federal Railways~~

Name of foreign principal Rudolf F. Mattesich

The answers to the items of the above-mentioned statement listed below are hereby amended to read as follows:

(Insert proper item numbers)

Item No. 10

Expenditures for the above mentioned period:

Rent, electric, office leaning	$7,710.00
Salaries	11,052.00
Employers payments	408.00
Office expenditures	200.00
Telephone & Telegraph	1,200.00
Postage & shipping expenses	1,050.00
Customs & Brokers fees	5,100.00
Travel expenses	1,200.00
Promotion & misc.	600.00
	4,200.00
	32,520.00

Item No.

(b) Document from the VRDU dataset

Figure 2: A visual comparison between a low-layout document (from the Kleister-NDA dataset) and a highly structured document (from the VRDU dataset). Models must leverage 2D spatial features to successfully extract information from the document on the right.

its token budget, are given in Appendix A. Because the text-only and vision arms therefore use different sampling settings, we bounded the effect by re-running the eight text models on VRDU under the vision arm's settings with everything else fixed: at batch size 10, exact match moved by -1.6 to $+3.2$ points (mean $+2.0$), well below the gaps of 7 points or more between size-matched text and vision models on which the modality comparison rests. Prompt templates for both datasets are listed in Appendix B. We benchmark the general-purpose LLMs and VLMs with Bench360 (Stuhlmann et al., 2026), and the specialized models (NuExtract and Arctic-TILT) with custom inference scripts profiled by CodeCarbon.

4.5 Metrics

We report average field exact match and energy per page; F1 and fuzzy match are also computed and reported in Appendix C.

Average Field Exact Match (EM): For each document we compare the predicted and gold value sets of every requested field and report the proportion of answered fields whose set matches the gold set exactly. Values are lowercased and standardized (e.g., converting text-based numbers to digits, normalizing dates) prior to comparison. Fields that a model declines to answer are not counted in the denominator, so EM behaves as a field-level precision; the value-level F1 reported in Appendix C penalizes omissions and should be consulted alongside it.

Energy Consumption: Total energy of the execution pipeline, covering both parsing (CodeCarbon (Courty et al., 2024)) and model inference (Bench360), in milliwatt-hours per page (mWh/pg), dividing by the PDF page count of each corpus (5.87 pages per document on Kleister-NDA, 1.83 on VRDU). Table 2 is the one exception, for the reason given in Section 4.3. Both tools sample GPU power through NVML at 10 Hz and model the CPU and RAM contributions rather than measuring them; the Limitations discuss what this implies for CPU-only pipelines. Model loading is excluded, so the figures describe steady-state serving rather than cold start.

5 Results

5.1 RQ1 Baselines

Within each model family, larger models generally have higher accuracy but also higher energy per page (Figure 3). Which architecture leads depends on the dataset: among the zero-shot models, Qwen3-VL and NuExtract are ahead on the layout-rich VRDU forms, while small text-only models with OCR are ahead on the text-heavy Kleister-NDA.

Arctic-TILT reports 92% exact match on Kleister-NDA, but that figure is in-domain (Section 4.2) and not comparable to the zero-shot rows; we report it for reference and exclude it from best-configuration and Pareto claims on that dataset. Its VRDU behaviour is the informative comparison: 65% exact match, below both Qwen3-VL-8B and NuExtract.

5.2 RQ2 Input Representation

Table 2 shows that whether a document is born-digital or scanned dictates the energy profile of the PDF-to-Markdown conversion. For born-digital documents (Kleister-NDA), Docling is the most efficient choice because it parses embedded text and bypasses visual OCR, which Tesseract ignores at a higher computational cost. For scanned documents (VRDU), Docling is forced to perform visual OCR, sharply increasing its energy footprint and making Tesseract the more efficient option. In both cases

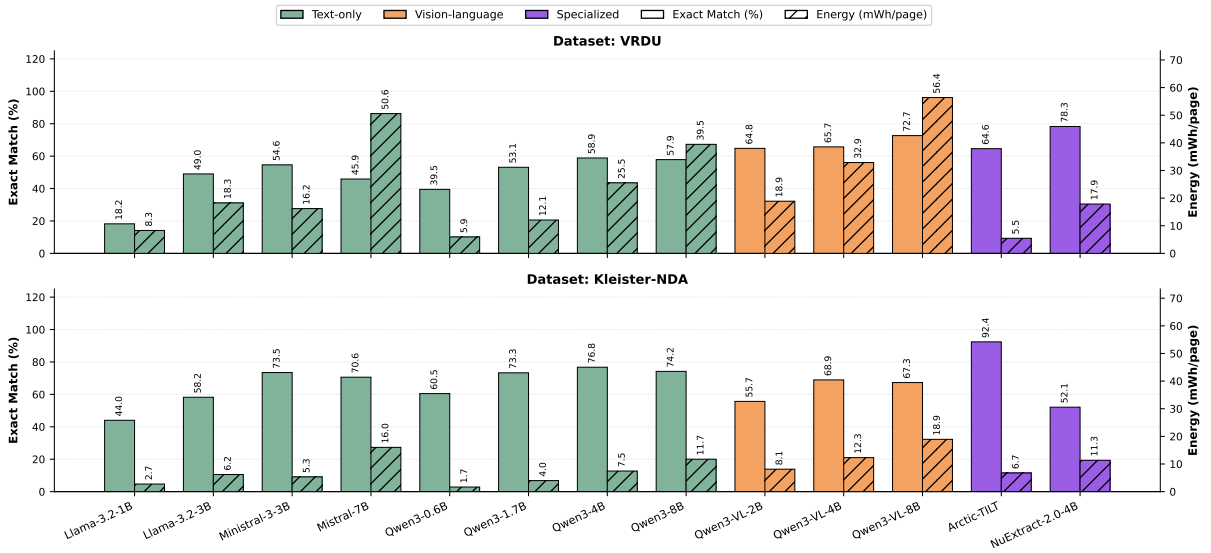


Figure 3: Baseline extraction accuracy and energy consumption per page on VRDU (top) and Kleister-NDA (bottom), at batch size 1 without quantization, using Tesseract-based PDF-to-Markdown conversion for all text-only models. Both panels share the model axis and the energy scale, so the two datasets can be read against each other. Energy covers model inference only; parsing energy is added in the end-to-end analysis of Section 5.4. Arctic-TILT's Kleister-NDA result is in-domain (fine-tuned on that corpus), shown for reference only and not part of any zero-shot comparison.

DeepSeek-OCR 2 consumes by far the most energy: 17× Tesseract per page on Kleister-NDA and 18× on VRDU.

These comparisons rest on whole-machine attribution, which does not treat every pipeline alike. Tesseract is measured without the GPU term (Section 4.3), so it is not charged for the idle L4; if anything this favors it. The GPU-accelerated parsers and all model inference include the GPU. CPU and RAM power are modeled rather than metered (RAM at a flat 20 W), so the absolute energy gap between parser-heavy and vision-language pipelines carries some estimation error. The effects we draw conclusions from — the ordering of the parsers within each dataset and the order-of-magnitude cost of neural OCR — are far larger than that error.

Table 2: Total and per-page energy consumption (Wh) for PDF to Markdown conversion, over 500 documents per dataset. Per-page figures use each tool's own page count, which differs because Tesseract rasterizes the PDFs (3016 pages on Kleister-NDA, 1025 on VRDU) while Docling and DeepSeek-OCR 2 follow the PDF page structure (2934 and 915).

	Kleister-NDA		VRDU	
Tool	**Total**	**/Pg.**	**Total**	**/Pg.**
Tesseract	12.91	0.0043	4.87	0.0047
Docling	4.38	0.0015	12.95	0.0141
DeepSeek-OCR 2	220.49	0.0751	76.41	0.0835

Extraction quality follows the same split (Figure 4). On the scanned, layout-rich VRDU forms, neural OCR (DeepSeek-OCR 2) is the most accurate representation for every general-purpose LLM except one, beating the better of Tesseract and Docling by 7–15 exact-match points. Two configurations run counter. Llama-3.2-1B, the smallest model, degrades sharply with DeepSeek-OCR 2 (16.5 EM with Tesseract to 11.2), suggesting its limited capacity is more sensitive to the longer, denser neural OCR output than to transcription quality. Arctic-TILT also performs worse, likely from spatial fidelity lost when approximating word-level bounding boxes from paragraph-level outputs (Appendix A.1). In contrast, for digital, text-heavy documents (Kleister-NDA), extraction accuracy varies by at most 6 points across the three parsing strategies for the general-purpose LLMs, with no parser consistently ahead. The exception is again Arctic-TILT, which spans 81.5–92.4 EM because it consumes bounding boxes rather than plain text and is therefore directly exposed to the layout fidelity of the parser. For text-only models on low-layout documents, then, a cheap parser or classical OCR suffices.

5.3 RQ3 Inference Optimization

Batching is the single most effective energy lever we measured (Figure 5). Moving from single-

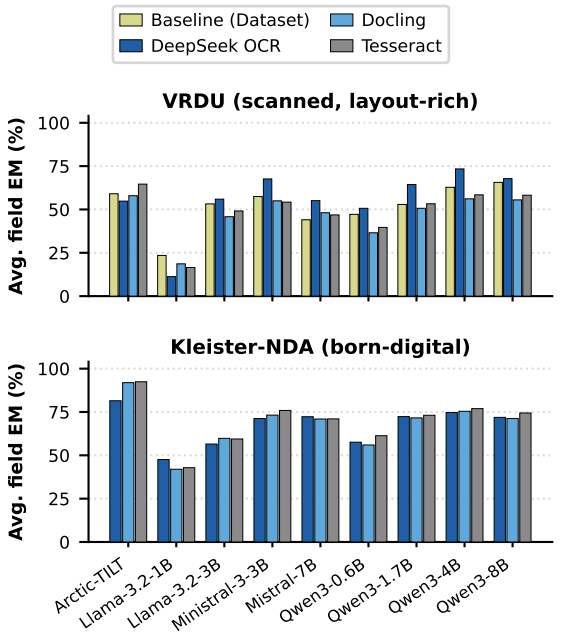


Figure 4: Average field exact match for each model under the four input representations (dataset-provided OCR, DeepSeek-OCR 2, Docling, Tesseract), at batch size 10 without quantization. Kleister-NDA ships no baseline OCR. Arctic-TILT's Kleister-NDA bars are in-domain (fine-tuned on that corpus). Underlying values are in Tables 3 and 4.

request serving to the best batch size reduces energy per page by 38–85%, in this single-run sweep monotonically to batch size 60 for ten of the eleven models. Returns diminish steeply: most of the benefit is realized by batch size 10, and for Qwen3-8B energy drops from 39.5 to 9.8 mWh/pg by batch size 10 but only to 6.9 mWh/pg at batch size 60, so the last sixfold increase buys 8 of the 83 percentage points of savings.

The exception is the largest vision-language model. Qwen3-VL-8B reaches its minimum of 34.7 mWh/pg already at batch size 10 and then flattens, holding between 35.3 and 36.4 mWh/pg through batch size 60, because it sits close to the VRAM budget of the L4. Qwen3-VL-4B levels off in the same way once measured repeatedly: its single-run sweep keeps falling to batch size 60, but over three runs (below) its minimum is at batch size 20 (11.2 mWh/pg), with a slight rise to 11.7 at batch size 60. The practical saturation point is therefore set by a model's memory headroom, not by a batch size that transfers across models.

Figure 6 compares unquantized (FP16) and FP8-quantized variants of Qwen3-4B and Qwen3-VL-4B across batch sizes on VRDU, each configuration measured three times; we report mean±standard deviation. At batch size 1, FP8 delivers substantial savings: 32.9±0.3 to 24.0±0.5 mWh/pg for Qwen3-VL-4B (-27%) and 25.2±0.1 to 17.1±0.1 mWh/pg for Qwen3-4B (-32%). Batching removes most of that advantage. At each model's best batch size the two precisions are 11.22±0.01 (FP16) against 10.26±0.02 mWh/pg (FP8) for Qwen3-VL-4B, and 3.81±0.04 against 3.08±0.01 mWh/pg for Qwen3-4B. FP8 therefore still saves 0.96 and 0.73 mWh/pg (-9% and -19%), a difference that is consistent across runs but roughly ten times smaller in absolute terms than at batch size 1. Accuracy is essentially unaffected: FP8 costs the vision model 0.6 exact-match points and gains the text model 0.3.

Batching and FP8 therefore act largely as substitutes rather than complements in this regime: both raise arithmetic intensity, and once batching has done so, low-precision arithmetic has much less idle capacity left to recover, consistent with the dequantization overheads reported by Delavande et al. (2026). In a batched deployment FP8 is still worth a further 9–19% of a much smaller energy budget, but its stronger argument is memory headroom.

5.4 Pareto Frontier

Figure 7 summarizes the end-to-end accuracy–energy trade-offs, making the impact of parsing energy clear. Text-only models paired with DeepSeek-OCR 2 are the most accurate text-only configurations on VRDU, but their parsing cost is an order of magnitude larger than the inference it feeds, and no DeepSeek-OCR 2 configuration reaches the frontier on either dataset. Every frontier and every comparison we draw on Kleister-NDA is strictly zero-shot: Arctic-TILT, fine-tuned on that corpus, is plotted on its panel only as an in-domain reference point and enters no frontier or zero-shot comparison (Section 5.1).

The composition of the frontier inverts between the two datasets. On VRDU it consists almost entirely of models that read page images directly: Qwen3-VL-2B (65.1% EM at 7.9 mWh/pg), Qwen3-VL-4B (65.6% at 12.2 mWh/pg) and NuExtract-2.0-4B (78.3% at 17.8 mWh/pg), with only the cheapest text-only configuration surviving at the low-accuracy end. On Kleister-NDA no vision-language model reaches the frontier at all; it is held end to end by text-only models paired with a cheap parser, from Qwen3-0.6B + Docling (56.0% at 2.4 mWh/pg) up to Qwen3-4B + Tesser-

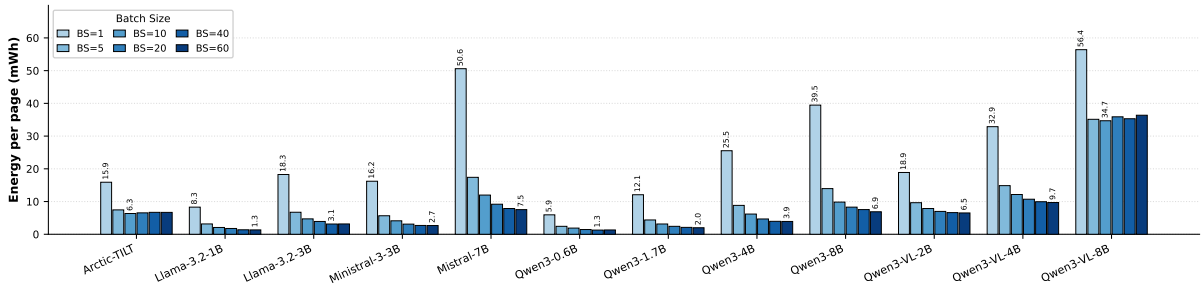


Figure 5: Inference energy per page on VRDU across models served with vLLM, unquantized, at batch sizes (BS) from 1 to 60; text-only models consume Tesseract output. Parsing energy is excluded. Data labels above the bars give the minimum and maximum energy per page observed for each model across the evaluated batch sizes.

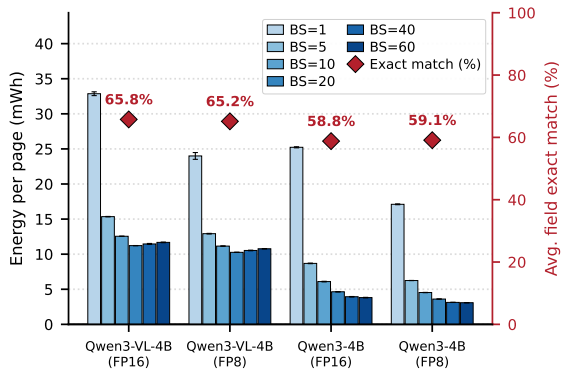


Figure 6: Energy per page and exact match for Qwen3-4B and Qwen3-VL-4B on VRDU, FP16 versus FP8, across batch sizes. Bars show the mean over three runs per configuration and error bars the standard deviation, which is mostly thinner than the bar outline. Exact match is averaged over runs and batch sizes, within which it varies by less than a point. It is Bench360's own field exact match rather than the recomputed metric of Tables 3 and 4, and in these runs both models used the vision arm's sampling settings (Appendix A), so absolute values differ slightly from the tables while the FP16-versus-FP8 difference is unaffected. The FP8 energy advantage is concentrated at batch size 1 and shrinks roughly tenfold by each model's best batch size.

act (76.9% at 7.9 mWh/pg), with Docling configurations occupying the low-energy half. The comparison is direct: on Kleister-NDA, Qwen3-4B + Docling is both more accurate and cheaper than Qwen3-VL-4B (75.4% at 5.0 mWh/pg against 70.1% at 8.1 mWh/pg), while on VRDU that ordering reverses (56.1% at 19.5 against 65.6% at 12.2).

Normalizing by page also makes the two corpora directly comparable, which per-document energy did not allow. A VRDU page is the more expensive of the two to process end to end: 11.5 mWh/pg against 7.9 for Qwen3-4B with Tesseract, and 12.2 against 8.1 for Qwen3-VL-4B. The higher per-document cost of the contracts is therefore a length effect — they run to three times as many pages — rather than evidence that a contract page is harder to process than a scanned form page.

The same numbers bear on the length confound (Section 4.2). If the vision-language models lost on Kleister-NDA because long visual contexts scale poorly, their per-page energy would rise on the longer corpus; it falls instead (Qwen3-VL-4B: 8.1 mWh/pg on Kleister-NDA against 12.2 on VRDU). The energy side of the inversion is therefore not a long-context effect. We cannot rule out a length effect on accuracy, and a length-matched subset would be needed to isolate it; our run reports store aggregate rather than per-document scores, so we leave this to future work.

Accuracy is also expensive at the margin. On VRDU, moving from Qwen3-VL-2B to Qwen3-VL-8B buys 6.2 exact-match points for 4.4× the energy per page. Tables 3 and 4 in Appendix C give the underlying numbers for every configuration.

6 Discussion

Naive single-document inference is the most expensive mistake available in this design space: batching alone removes 38–85% of the energy per page, most of it by batch size 10. FP8 recovers much of the same energy when serving one request at a time but less than one milliwatt-hour per page once batching is in place, so the two are largely substitutes: stacking FP8 on batching still helps, but by roughly a tenth as much.

Parsing affects accuracy unevenly across document types, but affects energy uniformly and by far more. Embedded-text parsers land within a few points of the alternatives on born-digital contracts at a fraction of the cost. On scanned forms neural OCR does buy real accuracy, yet at 18× Tesseract's per-page energy it costs an order of magnitude

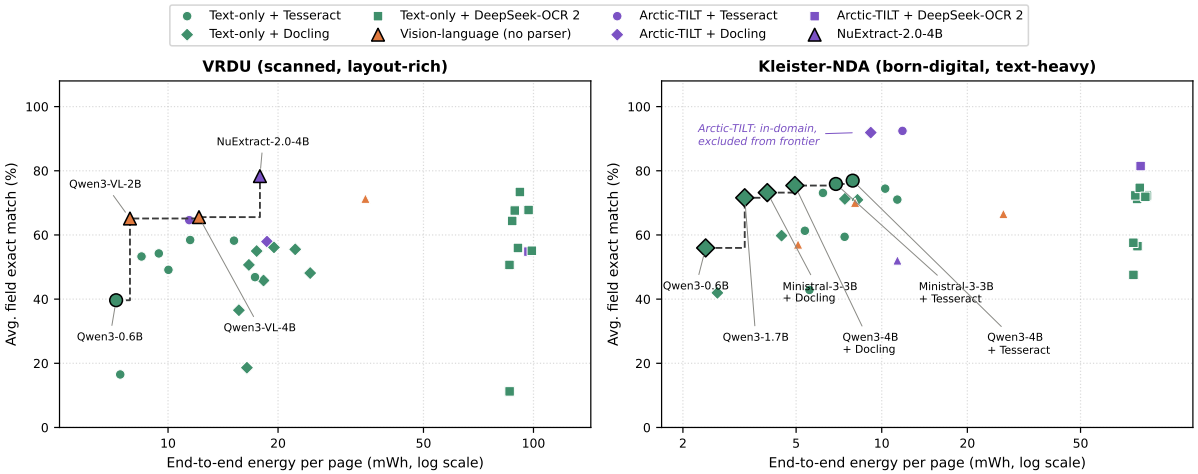


Figure 7: End-to-end accuracy–energy trade-off on VRDU (left) and Kleister-NDA (right). Energy per page includes both parsing and inference. Pareto-optimal configurations are enlarged, outlined in black, labelled, and joined by the dashed step line; all other measured configurations are shown at the same strength. The frontier is held by vision-language models on the layout-rich dataset and by text-only models with a cheap parser on the text-heavy one; no DeepSeek-OCR 2 configuration reaches either frontier. Frontiers cover zero-shot configurations only: Arctic-TILT is fine-tuned on Kleister-NDA and appears on that panel solely as an in-domain reference point, excluded from the frontier and from every zero-shot comparison.

more than the inference it feeds, reaches neither Pareto frontier, and delivers accuracy that a vision-language model reading the page directly supplies more cheaply. A parser is charged for every page of every document while the accuracy it enables is capped by the model downstream, which makes parsing the first place to look when energy matters.

Which architecture wins depends on the type of document rather than on model size or family; in our corpora layout and length co-vary, so we attribute the inversion to the document type as a whole rather than to layout alone. Vision-language models are best placed on registration forms, reading page images and skipping parsing entirely; on low-layout contracts this reverses, and small text-only models with a cheap parser are both more accurate and cheaper than every vision-language configuration we measured. Arctic-TILT does not settle the question — it leads on Kleister-NDA only because it was fine-tuned on it — but, read as an in-domain upper bound, it does size the prize for domain adaptation, at roughly 15 exact-match points over the best zero-shot pipeline at comparable energy. Parsing, model selection, and serving configuration must be tuned as one design, not in isolation.

7 Conclusion

We compared local document IE pipelines from an accuracy–energy perspective at two ends of the layout spectrum, and derive three guidelines for building accurate, energy-efficient, and privacy-compliant pipelines with small local models. First, *match parsing to layout:* embedded-text parsers or classical OCR for born-digital, text-heavy documents; neural OCR only for scanned forms, and only where its accuracy gain justifies an order of magnitude more parsing energy. Second, *align architecture with structure:* vision-language models for layout-rich documents, where reading page images removes the parsing stage entirely; small text-only models with an efficient parser for low-layout documents, where they were both cheaper and more accurate; and, where in-domain data exists, adaptation of a small model in preference to scaling a zero-shot one at equal energy. Last, *batch before you quantize:* increase batch size until memory pressure appears, worth 38–85% of energy per page at no cost in accuracy, and treat FP8 as a smaller second step: it saves about 30% for deployments that cannot batch but less than 1 mWh per page on top of batching, where its main value is memory headroom.

Limitations

Profiling of Energy Consumption

Relying on CodeCarbon to measure the energy consumption is a limitation, as the profiling was conducted entirely using software-based estimation rather than direct hardware power meters. Because it is a software tool, it does not capture the total energy consumption (Fischer, 2026). Systemic overheads, such as the energy required for hardware cooling, are excluded. This leads to an underestimation of the actual power consumption by 20-30% when using CodeCarbon (Fischer, 2026). However, because this underestimation represents a consistent deviation across all measurements, the comparative energy efficiency results and overall conclusions of this study remain robust.

Two further caveats apply. Parsing and specialized-model pipelines are profiled with CodeCarbon while vLLM-served models are profiled with Bench360. Both read the same NVML counter at the same 10 Hz, so the GPU term is directly comparable; what we have not cross-validated is the modeled CPU and RAM contribution, which each tool estimates independently. In addition, CPU and RAM contributions are modeled from utilization rather than metered. Disabling GPU tracking for Tesseract keeps the idle L4 out of its figures, but the absolute gap between CPU-bound parsing and GPU inference still inherits the error of that model (Section 4.3).

Single Measurements

The FP16-versus-FP8 comparison, whose differences at the best batch size fall below one milliwatt-hour per page, was measured three times per configuration. The standard deviation across runs was at most 0.5 mWh/pg, and below 0.05 at the best batch sizes, so the remaining FP8 saving is well outside run-to-run noise. The repeats also show why single runs call for caution: the original Qwen3-VL-4B FP16 runs at batch sizes 40 and 60 lay 13–17% below the repeat means. Every other configuration was measured once, apart from two that were run twice: exact match agreed within 0.9 and 2.2 points, and energy within 0.7% and 8.8%. The effects we base our recommendations on (38–85% from batching, an order of magnitude between parsers) are well outside any plausible noise band.

Evaluation Protocol

The text-only and vision-language arms use each family's recommended sampling defaults rather than one shared configuration. The bound in Section 4.4 (-1.6 to $+3.2$ exact-match points) shows that this is too small to explain the modality inversion. Greedy decoding would remove the difference entirely, together with a source of run-to-run variance, and is what we recommend for future benchmarks of this kind.

Arctic-TILT's Kleister-NDA results are in-domain: the released checkpoint is fine-tuned on that corpus, and we did not restrict our evaluation set to documents outside its training split, so the reported 92% exact match should be read as an upper bound obtained with target-domain supervision rather than as a comparable system result. We exclude it from best-configuration and Pareto claims on Kleister-NDA for this reason, and note that a clean in-domain versus zero-shot comparison would require either the exact training split of the released model or a model we fine-tune ourselves. We also deviate from the VRDU benchmark protocol by evaluating zero-shot instead of few-shot fine-tuned.

Finally, our headline exact-match metric does not penalize fields a model leaves unanswered; the value-level F1 in Appendix C does, and the two should be read together.

Normalization and Document Length

We report energy per page rather than per document, because the two corpora differ threefold in length and per-document figures would not be comparable across them. Two caveats follow. First, the divisor is a dataset-level average (5.87 and 1.83 pages per document), not a per-document page count, which the measurement data does not record. Dividing by it is therefore a constant rescale within each dataset, so every within-dataset result we report — the batching and quantization effects, the model rankings, and the composition of both Pareto frontiers — is identical to what per-document normalization would give. Only comparisons between the two datasets change.

Second, and more substantially, layout complexity and document length co-vary between our two corpora (Section 4.2), so the modality inversion is a property of the kind of document rather than of layout alone. Per-page energy argues against a long-context energy effect (Section 5.4), but length

may still affect accuracy, for instance through the page cap on vision inputs. Separating the two factors would require a length-matched subset or a long-form layout-rich corpus, neither of which the available benchmarks provide. Pages are also an imperfect denominator for text-only models, whose energy scales with token count rather than page count.

Hardware Configuration

The experiments were conducted using a NVIDIA L4 GPU (24 GB VRAM), which constrains the model selection. The available VRAM restricts evaluation to models of about $\leq$ 8B parameters without quantization. Prior work has shown that energy efficiency can vary substantially across GPU architectures and cooling solutions (Latif et al., 2025; Argerich et al., 2026a), and predictive models such as WattGPU (Argerich et al., 2026b) offer a way to estimate the inference side of our configurations on other GPUs without re-profiling. Thus, our findings should be interpreted as representative of a resource-constrained, single-GPU environment rather than universal across all hardware platforms. We expect the relative findings to transfer best: that batching saves energy by raising utilization, and the ordering of the parsers, which follows from how much work each does per page. The FP8 result is the most hardware-specific. The L4's Ada architecture has native FP8 tensor cores, whereas on GPUs without them vLLM falls back to weight-only FP8, so both the batch-size-1 saving and how much of it survives batching may differ on other platforms.

Quantization

In this study, we focus on online FP8 quantization as implemented in vLLM, which applies quantization at inference time without modifying the underlying checkpoints. While this approach already yields notable energy savings, prior work on activation-aware weight quantization (AWQ) (Lin et al., 2026) suggests that carefully pre-quantized models can further reduce memory footprint and compute while preserving accuracy. Integrating such pre-quantized variants into our benchmarking framework is a promising direction for achieving even lower energy per request, especially for larger models that currently exceed the VRAM budget on our single L4 GPU.

Prompting Techniques

The evaluations conducted in this study were limited to single-shot prompting. Advanced methods, such as chain-of-thought reasoning or other prompting strategies, were not evaluated. Consequently, the reported accuracy metrics should be interpreted as a lower bound of the models' capabilities.

Scope of the Energy Accounting

Model loading is excluded from all reported figures: energy is measured over the generation phase only. Loading a model costs 142–259 seconds depending on size, which is substantial in absolute terms but is amortized away in the high-volume batch setting this paper targets. Deployments that load a model per request would need to account for it separately.

Coverage

We study two English-language corpora representing two points on the layout spectrum. Invoices, receipts, multilingual forms such as the densely annotated regulatory documents of Colakoglu et al. (2026), and handwritten documents — where the modality dependence we report would be stressed hardest — are outside the scope of this evaluation. Adding such corpora, and in particular a long-form layout-rich one, is the most direct test of how far our guidelines generalize.

Ethical Considerations

We are not aware of any ethical concerns or violations of the ACL Code of Ethics arising from this work. By empirically characterizing accuracy–energy trade-offs for local LLM inference, our study aims to support more responsible and accessible deployment of on-premise models.

Use of Large Language Models (LLMs). We used LLMs to assist with the writing of this manuscript. In particular, we drafted sections ourselves and then employed cloud-based LLM services (e.g., Gemini, ChatGPT) to refine wording and improve clarity, with all technical content and experimental claims authored and verified by the authors.

References

Mauricio Fadel Argerich, Jonathan Fürst, and Marta Patiño-Martínez. 2026a. Watt counts: Energy-aware benchmark for sustainable LLM inference on heterogeneous GPU architectures. *Preprint*, arXiv:2604.09048.

Mauricio Fadel Argerich, Jonathan Fürst, and Marta Patiño-Martínez. 2026b. WattGPU: Predicting inference power and latency on unseen GPUs and LLMs. *Preprint*, arXiv:2607.02391.

Mauricio Fadel Argerich and Marta Patiño-Martínez. 2024. Measuring and improving the energy efficiency of large language models inference. *IEEE Access*, 12:80194–80207.

Łukasz Borchmann, Michał Pietruszka, Wojciech Jaskowski, Dawid Jurkiewicz, Piotr Halama, Paweł Józiak, Łukasz Garncarek, Paweł Liskowski, Karolina Szyndler, Andrzej Gretkowski, Julita Ołtusek, Gabriela Nowakowska, Artur Zawłocki, Łukasz Duhr, Paweł Dyda, and Michał Turski. 2025. Arctic-tilt: Business document understanding at sub-billion scale. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 264–283, Vienna, Austria. Association for Computational Linguistics.

Łukasz Borchmann, Rafał Powalski, Tomasz Stanislawek, Dawid Jurkiewicz, Wojciech Stokowiec, Michał Turski, Przemysław Biecek, and Przemysław Rokita. 2021. Due: End-to-end document understanding benchmark. In *NeurIPS 2021 Datasets and Benchmarks Track*.

Francisco Caravaca. 2026. Measuring energy consumption of llms inferences. *SIGMETRICS Perform. Eval. Rev.*, 53(3):18–19.

Gaye Colakoglu, Gürkan Solmaz, and Jonathan Fürst. 2025. Problem solved? information extraction design space for layout-rich documents using llms. In *EMNLP (Findings)*, pages 17908–17927.

Gaye Colakoglu, Gürkan Solmaz, and Jonathan Fürst. 2026. AgenticIE: An adaptive agent for information extraction from complex regulatory documents. *Preprint*, arXiv:2509.11773.

Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, and 11 others. 2024. mlco2/codecarbon: v2.4.1.

DeepSeek-AI. 2026. DeepSeek-OCR 2: Visual causal flow. *Preprint*, arXiv:2601.20552.

Julien Delavande, Regis Pierrard, and Sasha Luccioni. 2026. Understanding efficiency: Quantization, batching, and serving strategies in llm energy use. *Preprint*, arXiv:2601.22362.

Timo I. Denk and Christian Reisswig. 2019. Bertgrid: Contextualized embedding for 2d document representation and understanding. *Preprint*, arXiv:1909.04948.

Yihao Ding, Soyeon Caren Han, Jean Lee, and Eduard Hovy. 2025. Deep learning based visually rich document content understanding: A survey. *Preprint*, arXiv:2408.01287.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569, Vienna, Austria. Association for Computational Linguistics.

Raphael Fischer. 2026. Ground-truthing ai energy consumption: validating codecarbon against external measurements. *it - Information Technology*.

Aaron Grattafiori, Abhimanyu Dubey, and 1 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and 1 others. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Anoop Raveendra Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. *Preprint*, arXiv:1809.08799.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. OCR-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

Imran Latif, Muhammad Ali Shafique, Hayat Ullah, Alex C. Newkirk, Xi Yu, and Arslan Munir. 2025. Cooling matters: Benchmarking large language models and vision-language models on liquid-cooled versus air-cooled h100 gpu systems. *Preprint*, arXiv:2507.16781.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning (ICML)*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2026. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2025. Docling: An efficient open-source toolkit for ai-driven document conversion. *Preprint*, arXiv:2501.17887.

Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power hungry processing: Watts driving the cost of AI deployment? In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. 2024. Layoutllm: Layout instruction tuning with large language models for document understanding. *Preprint*, arXiv:2404.05225.

Mistral AI. 2025. Ministral-3-3B-Instruct-2512. Model weights, Hugging Face.

NuMind. 2026. Nuextract 2.0.

Soham Poddar, Paramita Koley, Janardan Misra, Sanjay Podder, Niloy Ganguly, and Saptarshi Ghosh. 2025. Towards sustainable nlp: Insights from benchmarking inference energy in large language models. *Preprint*, arXiv:2502.05610.

K. Pronk and Q. Zhao. 2025. Benchmarking energy efficiency of large language models using vllm. *Preprint*, arXiv:2509.08867.

Qwen Team. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qwen Team. 2025b. Qwen3-VL. Model weights, Hugging Face.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2021. *Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts*, page 564–579. Springer International Publishing.

Linus Stuhlmann, Mauricio Fadel Argerich, and Jonathan Fürst. 2026. Bench360: Benchmarking local llm inference from 360 degrees. *Preprint*, arXiv:2511.16682.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 5184–5193. ACM.

Toyohide Watanabe, Qin Luo, and Noboru Sugie. 1995. Layout recognition of multi-kinds of table-form documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):432–445.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Preprint*, arXiv:2312.17617.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1192–1200. ACM.

A Run Configuration

Decoding parameters follow each model family's recommended defaults and the token budget of the dataset:

Arm	Dataset	temp.	top-p	max tok.
Text-only	Kleister-NDA	0.6	0.9	128
Text-only	VRDU	0.6	0.9	256
Vision	Kleister-NDA	0.7	0.8	128
Vision	VRDU	0.7	0.8	128

Observed generation lengths averaged between 7.4 and 25.9 tokens across every run, so no configuration was budget-limited. For all Qwen3 and Qwen3-VL models, reasoning traces were disabled by passing `chat_template_kwargs={"enable_thinking": false}`; the Mistral and Ministral models do not expose such a mode. Vision-language runs pass at most the first 10 (Kleister-NDA) or 5 (VRDU) page images per document.

NuExtract Configuration of vLLM for NuExtract 2.0 4B:

```
vllm/vllm-openai:latest \
  --model numind/NuExtract-2.0-4B \
  --trust-remote-code \
  --dtype bfloat16 \
  --gpu-memory-utilization 0.8 \
  --max-model-len 32768 \
  --chat-template-content-format openai
```

Text-only models without quantization (vLLM). For text-only runs without quantization, vLLM was configured as follows:

```
vllm/vllm-openai:latest \
    --model "$MODEL" \
    --trust-remote-code \
    --max-model-len 31872 \
    --gpu-memory-utilization 0.95 \
    --port "$PORT"
```

Text-only models with FP8 quantization (vLLM). For text-only runs with FP8 quantization, we used:

```
vllm/vllm-openai:latest \
    --model "$MODEL" \
    --trust-remote-code \
    --max-model-len 31872 \
    --gpu-memory-utilization 0.95 \
    --port "$PORT" \
    --quantization fp8
```

Vision models without quantization (vLLM). For multimodal (vision) models without quantization, vLLM was configured as:

```
vllm/vllm-openai:latest \
    --model "$MODEL" \
    --trust-remote-code \
    --max-model-len 28000 \
    --port "$PORT" \
    --gpu-memory-utilization 0.95 \
    --no-enable-prefix-caching \
   --limit-mm-per-prompt '{"image": 15}'
```

Vision models with FP8 quantization (vLLM). For multimodal (vision) models with FP8 quantization, we used:

```
vllm/vllm-openai:latest \
    --model "$MODEL" \
    --trust-remote-code \
    --max-model-len 28000 \
    --port "$PORT" \
    --gpu-memory-utilization 0.95 \
    --no-enable-prefix-caching \
    --quantization fp8 \
   --limit-mm-per-prompt '{"image": 15}'
```

Arctic TILT configuration. For Arctic TILT, with `$BS` denoting the batch size, we used:

```
python examples/tilt_example.py \
    --model Snowflake/snowflake
    -arctic-tilt-v1.3 \
    --dataset "$DATASET" \
    --output-dir "$OUTPUT_DIR" \
    --gpu-memory-utilization 0.8 \
    --subset "" \
    --limit-documents 500 \
    --async \
    --max-num-seqs $BS \
    --enforce-eager
```

A.1 Bounding-box preprocessing for Arctic-TILT

Arctic-TILT (Borchmann et al., 2025) expects documents in the Document Understanding (DU) schema (Borchmann et al., 2021), which requires word-level bounding boxes. DeepSeek-OCR 2 emits bounding boxes at paragraph level instead. We approximate word-level boxes by interpolating their coordinates proportionally from the position of each token within the text of the enclosing paragraph box.

B Appendix: Prompt Templates

This section details the prompt templates used for the different information extraction tasks.

B.1 VRDU Text Information Extraction

For text-based extraction on the VRDU dataset, the system message establishes strict JSON output rules, and the user prompt directly injects the OCR text.

System Prompt:

```
You are an information extraction
    engine.
Your task is to read OCR text
   from a document and extract
   specific fields.
You must output ONLY one JSON
   object, with EXACTLY the
   requested keys.
Rules:
  - The JSON must be on a single
     line (no line breaks or
     indentation).
  - Each requested key MUST be
     present in the JSON.
  - If a field appears multiple
     times, use a JSON array of
     unique values in reading
     order.
  - If a field is not present in
     the OCR text, set its value
     to null.
  - Do NOT add any keys that were
      not requested.
  - Do NOT output any
     explanations, comments, or
     text outside the JSON object
     .
```

User Prompt:

```
Extract the requested keys from
   the OCR
OCR:
{ocr_text}

Requested keys:
{fields_str}

Output JSON (single line, no
   extra text):
```

B.2 VRDU Visual Information Extraction

For Vision-Language Models processing VRDU document images, the system message is minimized. The user prompt precedes a sequence of Base64-encoded images.

System Prompt:

```
You are an information extraction
    engine. Output ONLY JSON.
```

User Prompt:

```
Extract these fields from the
   images: {fields_str}
Output JSON on one line.
```

Note: This text is followed by the Base64 image payloads.

Dynamic Field Guidelines for Kleister-NDA

In the Kleister-NDA prompt templates below, the `{requested_descriptions}` placeholder is dynamically populated. To avoid confusing the model with irrelevant instructions, the pipeline only injects guidelines for fields that are actually present in the ground truth of the specific document being evaluated.

The possible field guidelines that can be injected are:

- **effective_date**: Extract the effective date. Strictly format as YYYY-MM-DD.
- **jurisdiction**: The state or country whose laws govern the agreement (e.g., 'New York', 'Delaware').
- **party**: The exact names of the companies, organizations, or individuals entering into the agreement.
- **term**: The duration of the agreement. Format as a number followed by the unit (e.g., '3 years', '1 year', '6 months').

B.3 Kleister-NDA Text Extraction

For text-based extraction on the Kleister-NDA dataset, the system message, hardcoded field guidelines, and the user prompt containing the document text are concatenated into a single block.

Combined Prompt:

```
You are an information extraction
    engine.
Your task is to extract specific
   fields from the provided Non-
   Disclosure Agreement (NDA)
   document text.
You must output ONLY one JSON
   object, with EXACTLY the
   requested keys.
```

```
Rules:
  - The JSON must be on a single
     line (no line breaks or
     indentation).
  - Each requested key MUST be
     present in the JSON.
  - If a field appears multiple
     times (e.g., multiple
     parties), use a JSON array
     of unique values.
  - Do NOT output any
     explanations, comments, or
     text outside the JSON object
     .

Field Guidelines:
{requested_descriptions}

--- DOCUMENT TEXT ---
{document_text}
--- END OF DOCUMENT ---

Requested keys:
{fields_str}

Output JSON:
```

B.4 Kleister-NDA Visual Extraction

For Vision-Language Models processing Kleister-NDA document images, the system message dynamically includes field guidelines only for the requested keys. The user prompt precedes the Base64-encoded images.

System Prompt:

```
You are an information extraction
    engine.
Your task is to extract specific
   fields from the provided Non-
   Disclosure Agreement (NDA)
   document images.
You must output ONLY one JSON
   object, with EXACTLY the
   requested keys.
Rules:
  - The JSON must be on a single
     line (no line breaks or
     indentation).
  - Each requested key MUST be
     present in the JSON.
  - If a field appears multiple
     times (e.g., multiple
     parties), use a JSON array
     of unique values.
  - Do NOT output any
     explanations, comments, or
     text outside the JSON object
     .

Field Guidelines:
{requested_descriptions}
```

User Prompt:

```
Extract these fields from the
   images: {fields_str}
Output JSON on one line.
```

Note: This text is followed by the Base64 image payloads.

C Detailed Results

Macro F1 Score (F1): Evaluates the overall extraction performance by calculating the harmonic mean of precision and recall. This is computed by assessing the value-level intersection of predicted and ground-truth sets for each document, which is then averaged across the entire dataset.

Fuzzy Match Score (FM): Accounts for minor OCR errors or generative variations by applying a token-sort ratio algorithm. This calculates a similarity score between 0 and 1 by alphabetically sorting tokens before comparison, ensuring robust matching even if the model outputs words in a slightly different order.

Table 3: Detailed benchmarking results for the VRDU dataset (Pareto plot configurations), over 500 documents. OCR (mWh/pg) is the parsing energy per page and E2E (mWh/pg) additionally includes model inference; both are normalized by the 1.83 pages per document of this dataset. The OCR text shipped with the VRDU benchmark is used for accuracy comparisons in Figure 4 but excluded here, since the energy of producing it is not attributable to our pipeline.

						Energy (mWh/pg)	
Model	BS	OCR	EM (%)	FM (%)	F1 (%)	OCR	E2E
Arctic-TILT	10	DeepSeek	54.8	74.5	55.0	83.5	96.8
Arctic-TILT	10	Docling	57.9	74.0	58.0	14.1	18.7
Arctic-TILT	10	Tesseract	64.6	74.7	64.6	5.3	11.4
Llama-3.2-1B	10	DeepSeek	11.2	13.4	9.8	83.5	86.0
Llama-3.2-1B	10	Docling	18.6	31.2	17.6	14.1	16.4
Llama-3.2-1B	10	Tesseract	16.5	23.3	15.6	5.3	7.4
Llama-3.2-3B	10	DeepSeek	55.9	77.5	57.4	83.5	90.7
Llama-3.2-3B	10	Docling	45.8	71.6	45.4	14.1	18.3
Llama-3.2-3B	10	Tesseract	49.1	73.1	49.3	5.3	10.0
Ministral-3-3B	10	DeepSeek	67.6	80.3	64.8	83.5	88.9
Ministral-3-3B	10	Docling	55.0	69.6	51.2	14.1	17.5
Ministral-3-3B	10	Tesseract	54.2	71.1	52.7	5.3	9.4
Mistral-7B	10	DeepSeek	55.1	71.0	53.1	83.5	99.0
Mistral-7B	10	Docling	48.1	67.2	45.7	14.1	24.5
Mistral-7B	10	Tesseract	46.9	63.4	45.3	5.3	17.3
NuExtract-2.0-4B	1	-	78.3	81.5	74.0	-	17.8
Qwen3-0.6B	10	DeepSeek	50.7	70.2	50.7	83.5	86.0
Qwen3-0.6B	10	Docling	36.5	64.4	36.5	14.1	15.6
Qwen3-0.6B	10	Tesseract	39.6	63.9	39.7	5.3	7.2
Qwen3-1.7B	10	DeepSeek	64.4	77.3	61.3	83.5	87.4
Qwen3-1.7B	10	Docling	50.7	70.4	48.7	14.1	16.6
Qwen3-1.7B	10	Tesseract	53.3	74.8	52.9	5.3	8.5
Qwen3-4B	10	DeepSeek	73.4	84.9	70.9	83.5	91.8
Qwen3-4B	10	Docling	56.1	75.0	53.8	14.1	19.5
Qwen3-4B	10	Tesseract	58.4	78.7	57.3	5.3	11.5
Qwen3-8B	10	DeepSeek	67.8	82.7	65.2	83.5	97.0
Qwen3-8B	10	Docling	55.5	72.2	51.9	14.1	22.3
Qwen3-8B	10	Tesseract	58.2	76.9	56.0	5.3	15.2
Qwen3-VL-2B	10	-	65.1	82.7	65.2	-	7.9
Qwen3-VL-4B	10	-	65.6	83.6	65.6	-	12.2
Qwen3-VL-8B	10	-	71.3	86.2	71.4	-	34.7

Table 4: Detailed benchmarking results for the Kleister-NDA dataset (Pareto plot configurations). Extraction is evaluated over 337 documents; parsing energy is amortized over the 500 documents of the parsing benchmark. Energy is per page, normalized by the 5.87 pages per document of this dataset. Arctic-TILT is fine-tuned on Kleister-NDA by its authors and is therefore evaluated in-domain, unlike every other model in the table; its rows are reported for reference and are excluded from best-configuration and Pareto comparisons.

						Energy (mWh/pg)	
Model	BS	OCR	EM (%)	FM (%)	F1 (%)	OCR	E2E
Arctic-TILT†	10	DeepSeek	81.5	91.2	84.8	75.1	81.3
Arctic-TILT†	10	Docling	91.9	98.0	93.8	1.5	9.2
Arctic-TILT†	10	Tesseract	92.4	98.1	93.8	4.4	11.8
Llama-3.2-1B	10	DeepSeek	47.6	70.7	58.4	75.1	76.7
Llama-3.2-1B	10	Docling	42.0	64.6	53.2	1.5	2.6
Llama-3.2-1B	10	Tesseract	42.9	65.2	53.9	4.4	5.6
Llama-3.2-3B	10	DeepSeek	56.5	74.9	66.2	75.1	79.3
Llama-3.2-3B	10	Docling	59.8	78.6	68.7	1.5	4.4
Llama-3.2-3B	10	Tesseract	59.4	77.2	68.6	4.4	7.4
Ministral-3-3B	10	DeepSeek	71.2	87.5	79.3	75.1	79.0
Ministral-3-3B	10	Docling	73.2	89.2	80.3	1.5	4.0
Ministral-3-3B	10	Tesseract	75.9	90.4	82.5	4.4	6.9
Mistral-7B	10	DeepSeek	72.3	89.0	79.1	75.1	85.6
Mistral-7B	10	Docling	70.9	88.1	78.3	1.5	8.2
Mistral-7B	10	Tesseract	71.0	87.3	79.0	4.4	11.3
NuExtract-2.0-4B	1	-	52.1	65.5	59.6	-	11.4
Qwen3-0.6B	10	DeepSeek	57.6	78.4	66.2	75.1	76.7
Qwen3-0.6B	10	Docling	56.0	77.5	65.1	1.5	2.4
Qwen3-0.6B	10	Tesseract	61.3	79.6	69.5	4.4	5.4
Qwen3-1.7B	10	DeepSeek	72.3	90.2	79.3	75.1	77.9
Qwen3-1.7B	10	Docling	71.6	89.7	78.6	1.5	3.3
Qwen3-1.7B	10	Tesseract	73.1	89.9	80.2	4.4	6.2
Qwen3-4B	10	DeepSeek	74.7	91.9	82.3	75.1	80.8
Qwen3-4B	10	Docling	75.4	92.4	82.4	1.5	5.0
Qwen3-4B	10	Tesseract	76.9	93.0	84.1	4.4	7.9
Qwen3-8B	10	DeepSeek	71.9	91.0	80.1	75.1	84.5
Qwen3-8B	10	Docling	71.2	90.2	79.0	1.5	7.4
Qwen3-8B	10	Tesseract	74.4	92.1	81.9	4.4	10.3
Qwen3-VL-2B	10	-	57.1	78.3	66.5	-	5.1
Qwen3-VL-4B	10	-	70.1	91.1	78.5	-	8.1
Qwen3-VL-8B	10	-	66.6	89.7	76.0	-	26.8

† In-domain: fine-tuned on Kleister-NDA by its authors; not comparable to the zero-shot rows.

Splitting Documents at Lower Cost: Multi-Split Boundary Decisions for LLM-Based Page Stream Segmentation

Nikhil Reddy Pottanigari* **Sepideh Kharaghani*** **Saverio Vadacchino**
Alejandro Posada **Ying Zhang**

ServiceNow Canada
{nikhilreddy.pottanigari, sepideh.kharaghani,
saverio.vadacchino, alejandro.posada,
yin.zhang}@servicenow.com

Abstract

Scanned mail, uploaded PDFs, and consolidated attachments often arrive as page streams that must be split into individual documents before downstream classification, extraction, or routing. Zero-shot large language models can detect document boundaries without task-specific training, but standard Page Classification (PC) and Boundary Decision (BD) formulations resolve only one boundary per model call. We introduce **Multi-Split Boundary Decision** (MSBD), which predicts multiple boundaries within a page window in a single call, reducing the number of inference requests. We evaluate MSBD across multiple language models, document collections, input modalities, and window sizes. The results reveal a model- and corpus-dependent operating range in which MSBD preserves strong segmentation accuracy while substantially improving inference efficiency, followed by a sharp decline at larger windows. MSBD provided the strongest overall accuracy–efficiency trade-off, while large windows expose distinct over- and under-segmentation behavior across models. These findings show that multi-boundary prediction can make zero-shot page stream segmentation more efficient when the window size is selected for the target corpus.

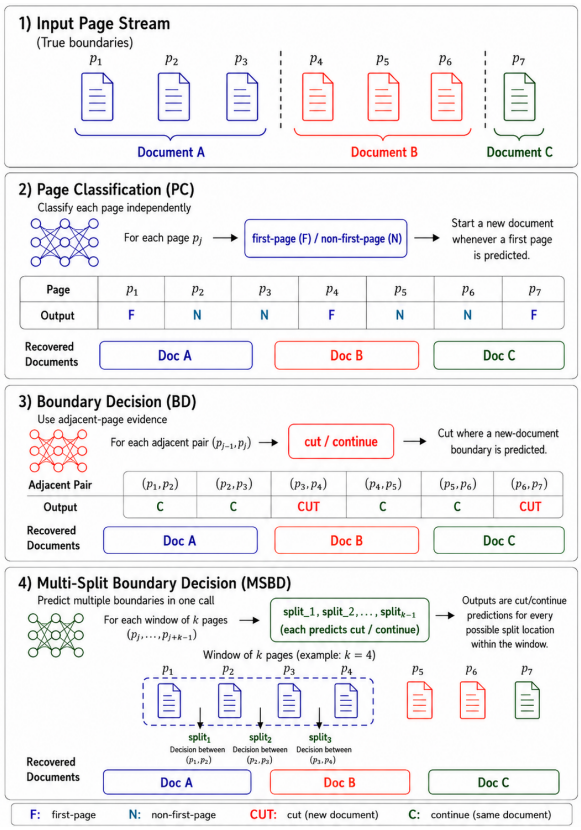


Figure 1: Zero-shot PSS formulations. PC classifies each page independently, BD evaluates one adjacent-page pair per call, and MSBD predicts all $K-1$ boundaries within a K-page window in a single call.

1 Introduction

Documents often arrive as batches of scanned pages, consolidated attachments, or multi-document PDF uploads rather than as individually separated files. Before classification, extraction, or routing, these page streams must be partitioned into their constituent documents. An incorrect boundary can either merge unrelated documents or split one document into fragments, affecting every downstream processing step.

This task is known as **Page Stream Segmentation** (PSS). Given an ordered page stream $P = (p_1, \ldots, p_N)$, the goal is to determine whether each page begins a new document (*cut*) or continues the preceding one (*continue*) (Gordo et al., 2013; Wiedemann and Heyer, 2021; Mungmeeprued et al., 2022). PSS is an important component of document-processing workflows in domains such as healthcare, finance, and logistics (Islam et al., 2026).

Most prior PSS systems rely on task-specific training using hand-crafted features, neural classifiers, or Transformer-based models (Gordo et al., 2013; Wiedemann and Heyer, 2021). Although

* Equal contribution.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 30–45
October 24, 2026 ©2026 Association for Computational Linguistics

effective within their target domains, these approaches require labelled boundary data and may need retraining when document distributions change. General-purpose language and vision-language models provide an alternative: they can perform PSS zero-shot, without training a dedicated boundary detector.

However, existing zero-shot formulations remain inefficient. **Page Classification** (PC) examines one page at a time, while **Boundary Decision** (BD) examines one adjacent-page pair at a time. Both resolve only one candidate boundary per model call, so processing cost grows linearly with stream length.

We introduce **Multi-Split Boundary Decision** (MSBD), which predicts all boundaries within a window of K consecutive pages in a single call. MSBD therefore reduces the number of model requests by sharing one inference call across multiple adjacent-page decisions. This creates a central trade-off: larger windows improve efficiency but may make the prediction task more difficult.

We study this trade-off in the *split-only* setting, where pages are already ordered and the task is limited to boundary detection rather than document classification or page reordering. We compare MSBD with PC, BD, and conventional trained and feature-similarity baselines across multiple models, datasets, input modalities, and window sizes.

Contributions

(1) We introduce **Multi-Split Boundary Decision**, a zero-shot PSS formulation that predicts multiple adjacent-page boundaries within a single model call.

(2) We systematically evaluate how window size affects segmentation accuracy, model calls, inference cost, latency, and output reliability across language models, datasets, and text and vision inputs.

(3) We identify a model- and corpus-dependent operating range in which MSBD preserves strong boundary accuracy while substantially reducing inference requests, and characterize the failure modes that emerge beyond this range.

(4) We compare zero-shot PC, BD, and MSBD with conventional supervised and feature-similarity baselines, clarifying the trade-offs between task-specific training and general-purpose prompting.

2 Background and Related Work

From hand-crafted features to decoder LLMs Early PSS systems used constrained clustering over inter-page similarity or supervised classifiers over hand-crafted structural, textual, and visual features (Collins-Thompson and Nickolov, 2002; Gordo et al., 2013; Daher and Belaïd, 2014; Agin et al., 2015). Later work introduced recurrent language models, CNNs, and RNNs over textual and visual page representations (Wiedemann and Heyer, 2021; Neche et al., 2020), and deep page embeddings enabled agglomerative clustering (Busch et al., 2023). Transformer-based methods remained limited: prior work combined LEGAL-BERT or LayoutLM with CNNs over consecutive pages (Guha et al., 2022; Mungmeeprued et al., 2022), and Braz et al. (2021) compared windows of one to three pages. More recently, Heidenreich et al. (2024) fine-tuned Mistral-7B and Phi-3 on TABME++, outperforming encoder models and an XGBoost baseline, and showed that zero-shot GPT-4o was competitive with trained encoders and that OCR quality strongly affects text-based PSS. Follow-up work in insurance reported similar fine-tuning gains while highlighting the difficulty of stream-level calibration (Heidenreich et al., 2025).

Boundaries, types, and order Islam et al. (2026) define *document packet splitting* as the joint problem of detecting boundaries, classifying document types, and reconstructing page order in shuffled packets; they find boundary detection to be the main bottleneck, motivating our split-only focus. Other work jointly models inter-page relations, segmentation, and classification (Demirtaş et al., 2022). Benchmark coverage has expanded from the synthetic streams in TABME (Mungmeeprued et al., 2022) to more realistic public datasets such as WooIR and OpenPSS (Van Heusden et al., 2022; van Heusden et al., 2024) and to domain-specific settings such as comic books (Ortega et al., 2025).

Multi-instance prompting Packing several task instances into one call to amortize API cost is known in general-purpose prompting (Cheng et al., 2023), where accuracy typically degrades as batch size grows. MSBD differs in one important respect: the $K-1$ decisions in a window are not independent instances but related decisions over a shared page sequence, so enlarging the window adds both prediction load and context. Consequently, accuracy can plateau rather than immediately degrade

at small values of K.

Positioning Unlike prior work that trains or fine-tunes models for PSS, we keep all models fixed and evaluate them entirely zero-shot, comparing text and vision inputs on TABME++ and on a form-heavy corpus constructed from VRDU (Wang et al., 2023).

3 Task Formulation and Methods

The three zero-shot formulations we study (Figure 1) all use the same fixed LLM or VLM without fine-tuning; they differ only in the context provided per call and the number of boundaries predicted at once. Our method, MSBD, is described next, while the single-call baselines against which we compare, PC and BD, are defined in §5.

MSBD: Multi-Split Boundary Decision MSBD processes a window of K consecutive pages and predicts all $K-1$ boundaries within that window in a single call. The model returns one JSON field per adjacent pair, from `split_1` to `split_`$(K-1)$, with each field labelled `cut_page` or `continuous_page` (Appendix C).

Windows are tiled with stride $K-1$, so consecutive windows overlap by one page and every adjacent pair is evaluated exactly once. This reduces the number of calls to approximately

$$\left\lceil \frac{N-1}{K-1} \right\rceil .$$

When $K=2$, MSBD is equivalent to BD. We evaluate $K \in \{3, 5, 7, 10, 20, 30, 50\}$.

The predicted fields are concatenated in page order to reconstruct the full boundary sequence $y_2, \ldots, y_N$, and documents are recovered by cutting at every predicted `cut_page`. A well-formed response contains one field for every candidate boundary in its window (normally $K-1$, with fewer in a final short window). The window size K therefore trades fewer calls against a harder multi-page prediction problem.

4 Experimental Setup

Datasets *TABME++* (Heidenreich et al., 2024) extends TABME (Mungmeeprued et al., 2022), a collection of synthetic streams formed by concatenating business documents, with improved commercial OCR. We use the dataset unmodified, as samples are already organized into page streams. It contains approximately 6.2k pages and a similar number of adjacent-page decisions.

VRDU (Wang et al., 2023) is a public, form-heavy dataset containing registration and advertising-buy forms, but it is not provided as page streams. We construct streams by concatenating complete documents and label the first page of each document as a `cut`. This construction makes many boundaries visually distinctive, which partly explains the strong PC performance on VRDU. The resulting corpus contains approximately 1.1k pages. Both datasets are reasonably balanced at the adjacent-pair level, with cut rates between approximately 36% and 46%.

Models and protocol We evaluate three models in a zero-shot setting: `gemini-3.1-flash-lite` (Google, 2026), `gpt-5.4-mini` (OpenAI, 2026), and `claude-haiku-4.5` (Amazon Web Services, 2026). Each formulation, PC, BD, and MSBD, is evaluated with OCR-text and page-image inputs on both datasets. All models are prompted to return structured JSON, decoded at temperature 0, and limited to 1024 output tokens. Page images are rendered at 300 DPI. We run one decoding pass per configuration.

Metrics We report Boundary F1 (B-F1), the F1 of the `cut` class over adjacent-page decisions, as the primary metric of the zero-shot study; the complete sweep results are provided in Appendix A. The available pre-LLM baseline results (§5.2) use per-class and overall (Micro) accuracy instead, so comparisons between the two families are directional rather than strictly controlled.

Cost accounting For each configuration, we report model calls, total test-set cost, and mean per-call latency. Call count depends on stream length and, for MSBD, window size K. Costs are computed from provider token usage and prices at evaluation time using LiteLLM [1]. Because prices may change, we also report cost per 1,000 pages and interpret values comparatively.

5 Baselines

We compare MSBD against two families of baselines: the single-call zero-shot prompts it is designed to replace (PC and BD), and pre-LLM systems that require task-specific training. Both fam-

[1] https://docs.litellm.ai/docs/completion/token_usage

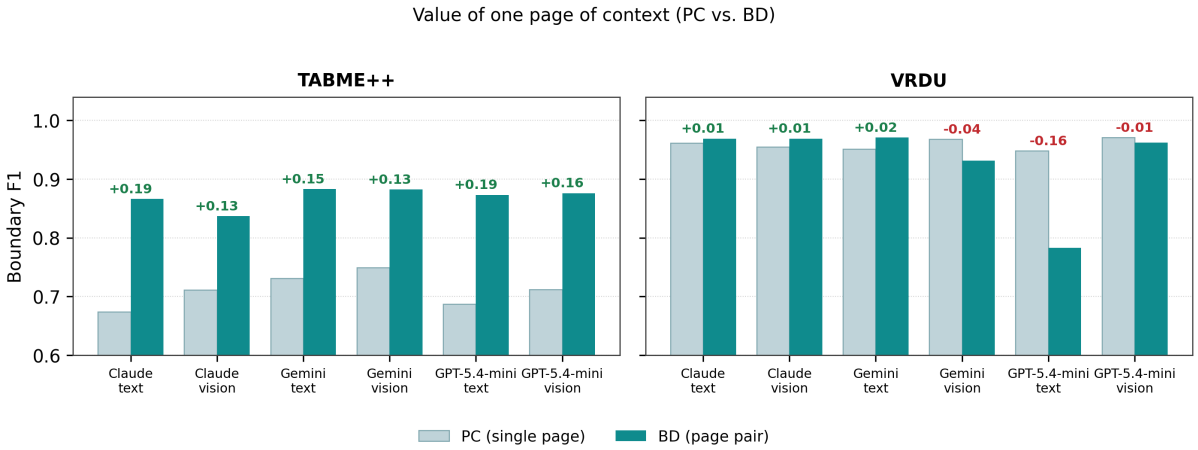


Figure 2: Boundary F1 for PC and BD. Adjacent-page context substantially improves TABME++, but provides limited benefit on the form-heavy VRDU dataset.

ilies are evaluated on the same TABME++ and VRDU streams described in §4.

5.1 Zero-shot single-call baselines

PC and BD use the same fixed models and prompting protocol as MSBD, but resolve a single boundary per call.

PC – Page Classification PC processes each page p_j independently, using either its OCR text or page image, and predicts `first_page` or `non_first_page` based on whether the page begins a new document. A boundary is inserted before each page predicted as `first_page`; thus, an N-page stream requires $N-1$ calls. Responses are parsed from a single JSON field (Appendix C).

BD – Boundary Decision BD processes each adjacent page pair (p_{j-1}, p_j) and predicts `cut_page` or `continuous_page`, indicating whether p_j begins a new document relative to p_{j-1}. Each decision therefore uses one neighbouring page as context but still requires one model call per candidate boundary. The stream is split at every pair labelled `cut_page`. Responses are parsed from a single JSON field (Appendix C).

5.2 Pre-LLM baselines

We compare zero-shot LLMs with two conventional PSS baselines: trained boundary classifiers and distance-based feature similarity. Both predict whether page p_j starts a new document (`cut`) or continues the document containing p_{j-1} (`continue`).

Trained boundary classifiers Let h_j denote the frozen visual representation of page p_j. We evaluate three trained configurations:

PC: classify h_j alone to determine whether p_j is a first page.

BD-Concat: concatenate $[h_{j-1}; h_j]$ and classify the boundary between the two pages.

BD-LSTM: process (h_{j-1}, h_j) as an ordered two-page sequence with a one-layer LSTM before classification.

PC uses only the current page. Both BD variants use the previous and current pages; BD-Concat combines their features directly, whereas BD-LSTM models their sequential relationship.

We use penultimate-layer features from a frozen, in-house *Vision Classifier*, a convolutional model trained to classify business-document pages. We evaluate linear, MLP, and LSTM heads and sweep Kullback–Leibler divergence regularization and report validation Micro accuracy on held-out TABME++ and VRDU streams.

Table 1 shows that adjacent-page context helps on TABME++: BD-Concat improves Micro accuracy from 0.757 to 0.769, and BD-LSTM further improves it to 0.817. The LSTM gain comes mainly from higher `continue` accuracy ($0.824 \rightarrow 0.926$), indicating better recognition of pages from the same document. On VRDU, PC performs best, suggesting that first pages are visually distinctive enough to identify independently.

Distance-based feature similarity We also infer boundaries from adjacent-page features without training a classifier. Using labelled page pairs, the

Dataset	Setting	Cut	Continue	Micro
TABME++	PC	0.637	0.824	0.757
	BD-Concat	0.619	0.830	0.769
	BD-LSTM	0.623	0.926	**0.817**
VRDU	PC	0.988	0.782	**0.878**
	BD-Concat	0.992	0.769	0.872
	BD-LSTM	0.992	0.750	0.862

Table 1: Accuracy of trained boundary classifiers using frozen Vision Classifier† features. PC uses one page, BD-Concat concatenates adjacent-page features, and BD-LSTM processes the same pair sequentially. *Cut* and *Continue* are per-class accuracies; *Micro* is overall accuracy. Best Micro per dataset is in **bold**. †In-house proprietary feature extractor; results are indicative rather than independently reproducible.

Dataset	Feature source	Cut	Continue	Micro
TABME++	Vision Classifier†	0.688	0.851	**0.792**
	FCOS	0.185	0.981	0.693
	GUSE	0.434	0.587	0.532
VRDU	Vision Classifier†	0.188	0.863	**0.550**
	FCOS	0.685	0.319	0.489
	GUSE	0.573	0.405	0.483

Table 2: Prototype-based feature-similarity results using Euclidean distance. Each adjacent-page pair is assigned to its nearest `cut` or `continue` prototype. *Cut* and *Continue* are per-class accuracies; *Micro* is overall accuracy. Best Micro per dataset is in **bold**. †In-house proprietary Vision Classifier features.

prototype method computes one representative prototype for each class and assigns each test pair to its nearest `cut` or `continue` prototype under Euclidean distance.

Table 2 compares visual features from the Vision Classifier and an FCOS (Fully Convolutional One-Stage) detector backbone with text embeddings from Google Universal Sentence Encoder (GUSE). Vision Classifier features achieve the highest Micro accuracy on both datasets. However, the class-wise results vary substantially across features and datasets. On TABME++, all three feature sources favour `continue`; FCOS detects only 18.5% of cuts but 98.1% of continuations. On VRDU, the Vision Classifier also favours `continue`, whereas FCOS and GUSE favour `cut`. Feature distance therefore does not provide a consistent boundary signal.

Implications for zero-shot PSS The two tables show that adjacent-page context can improve trained classifiers, but the benefit is dataset-dependent. They also show that the tested visual representations achieve higher Micro accuracy than GUSE embeddings. These methods nevertheless require labelled data to train classifiers or construct class prototypes, motivating our evaluation of zero-shot LLMs.

6 Results: Multi-Split Boundary Decision

We first compare PC and BD to isolate the value of adjacent-page context. We then examine how MSBD trades boundary accuracy for fewer model calls as the window size K increases.

PC versus BD: value of neighbouring context Table 3 and Figure 2 compare PC, which classifies the current page alone, with BD, which classifies the previous and current pages together. Both predict one boundary per call.

On TABME++, BD improves Boundary F1 by 0.13–0.19 over PC across the six model–modality combinations, showing that the previous page provides useful context. On VRDU, PC is already near ceiling because first pages are often visually distinctive, so BD offers little consistent improvement. The main exception is GPT-5.4-mini with text input, where Boundary F1 falls from 0.948 to 0.783.

Vision improves some PC results, but provides no consistent advantage over text once adjacent-page context is available and is more expensive in every BD setting. We therefore use text-based BD as the main reference for MSBD.

Model calls, cost, and latency Because MSBD predicts $K-1$ boundaries per call, the request count decreases rapidly as K grows. On TABME++, increasing K from 2 (equivalent to BD) to 50 reduces calls from $6{,}236$ to 127, a $49\times$ reduction.

Cost falls more slowly because each call contains more pages. For Gemini with text input, $K{=}10$ reduces calls by $9\times$ and test-set cost from \$2.79 to \$1.00, or approximately \$0.45 to \$0.16 per $1{,}000$ pages. Mean latency per call remains broadly stable for text through $K{=}10$ but rises at larger windows, particularly for vision input (Figure 5 in Appendix A). MSBD therefore improves throughput mainly by reducing the number of requests, not by making individual requests faster.

Accuracy and the knee Accuracy remains relatively stable at small windows and then declines sharply. We informally refer to the largest useful window before this decline as the *knee*.

TABME++						VRDU					
Model	Input	Method	B-F1	Lat	Cost	Model	Input	Method	B-F1	Lat	Cost
Claude Haiku 4.5	text	PC	0.674	2.12	10.33	Claude Haiku 4.5	text	PC	0.961	2.25	2.30
	text	BD	**0.866**	2.42	13.75		text	BD	**0.969**	2.50	3.26
	text	MSBD (K=3)	0.844	1.19	6.94		text	MSBD (K=3)	0.900	1.06	1.79
	text	MSBD (K=5)	0.817	1.30	**5.18**		text	MSBD (K=5)	0.795	1.15	**1.39**
	vision	PC	0.711	2.75	14.22		vision	PC	0.955	2.70	2.91
	vision	BD	0.837	2.91	21.25		vision	BD	0.969	3.15	4.50
	vision	MSBD (K=3)	0.844	1.53	12.63		vision	MSBD (K=3)	0.928	1.70	2.73
	vision	MSBD (K=5)	0.799	1.78	9.90		vision	MSBD (K=5)	0.755	2.08	2.15
Gemini 3.1 Flash-Lite	text	PC	0.731	1.10	1.97	Gemini 3.1 Flash-Lite	text	PC	0.951	1.02	0.44
	text	BD	0.883	1.20	2.79		text	BD	**0.971**	1.07	0.66
	text	MSBD (K=3)	0.847	1.08	1.61		text	MSBD (K=3)	0.940	0.91	0.41
	text	MSBD (K=5)	0.870	1.14	**1.22**		text	MSBD (K=5)	0.943	1.06	**0.32**
	vision	PC	0.749	1.70	2.96		vision	PC	0.968	1.51	0.53
	vision	BD	0.882	1.82	4.74		vision	BD	0.931	1.72	0.85
	vision	MSBD (K=3)	**0.887**	1.83	3.13		vision	MSBD (K=3)	0.938	1.58	0.56
	vision	MSBD (K=5)	0.881	2.14	2.49		vision	MSBD (K=5)	0.888	2.03	0.44
GPT-5.4-mini	text	PC	0.687	1.24	5.68	GPT-5.4-mini	text	PC	0.948	1.40	1.23
	text	BD	0.873	1.66	7.72		text	BD	0.783	1.30	1.84
	text	MSBD (K=3)	0.828	0.88	4.28		text	MSBD (K=3)	0.751	1.03	1.11
	text	MSBD (K=5)	0.793	0.99	**3.18**		text	MSBD (K=5)	0.609	2.41	**0.86**
	vision	PC	0.712	1.51	7.96		vision	PC	**0.971**	2.00	1.42
	vision	BD	**0.876**	1.68	12.39		vision	BD	0.962	2.34	2.22
	vision	MSBD (K=3)	0.846	1.47	7.84		vision	MSBD (K=3)	0.869	2.34	1.41
	vision	MSBD (K=5)	0.811	1.71	6.14		vision	MSBD (K=5)	0.800	2.77	1.11
GPT-4o†	zero-shot		0.859	–	–						
Phi-3-mini†	fine-tuned		0.973	–	–						
Mistral-7B†	fine-tuned		0.987	–	–						

Table 3: PC, BD, and MSBD results across datasets, models, and input modalities (B-F1 = Boundary F1; Lat = per-call latency in seconds; Cost = full test set, USD). MSBD is shown at K=3 and K=5 for both modalities; the complete sweep (K=2–50) with costs and call counts appears in Tables 4–5. Per dataset and model, the strongest B-F1 and the lowest cost among the rows shown are in **bold**; larger windows reduce cost further. †Page-level F1 reported by Heidenreich et al. (2024); not directly comparable.

On TABME++, Gemini with text input remains close to BD from K=5 to K=10: Boundary F1 is 0.870, 0.863, and 0.861 at K=5, 7, and 10, compared with 0.883 for BD. The corresponding gaps are 0.013, 0.020, and 0.022. Claude and GPT-5.4-mini begin degrading at smaller windows.

The useful range is narrower on VRDU. Gemini with text remains strong through K=7, but declines more clearly at K=10; Claude and GPT-5.4-mini degrade earlier. The knee therefore depends on both the model and the corpus.

At small windows, MSBD can occasionally match BD. For example, Gemini with vision reaches 0.887 at K=3, compared with 0.882 for BD. Because each configuration is evaluated with one decoding pass, this small difference should not be interpreted as a significant improvement.

Cost–accuracy trade-off Gemini provides the strongest overall trade-off between Boundary F1 and inference cost. In particular, Gemini with text input at K=5–10 retains most of BD's accuracy while substantially reducing calls and cost, making it the strongest MSBD configuration among those evaluated.

7 Failure Modes at Large Windows

Beyond the knee, MSBD fails in two distinct ways: invalid output structure and incorrect boundary behaviour.

Format errors versus silent degradation Gemini and GPT-5.4-mini return the expected number of boundary labels at all tested window sizes. Claude increasingly produces missing or extra fields as K grows, making some failures detectable through schema validation. GPT-5.4-mini shows the opposite behaviour: its outputs remain well formed while accuracy declines rapidly. Such silent degradation cannot be detected from output structure alone.

Over-segmentation versus under-segmentation Claude and Gemini predict too many cuts at large K, fragmenting documents through over-segmentation. GPT-5.4-mini predicts too few cuts, merging neighbouring documents through under-segmentation (Figure 3, right). Representative examples appear in Appendix B.

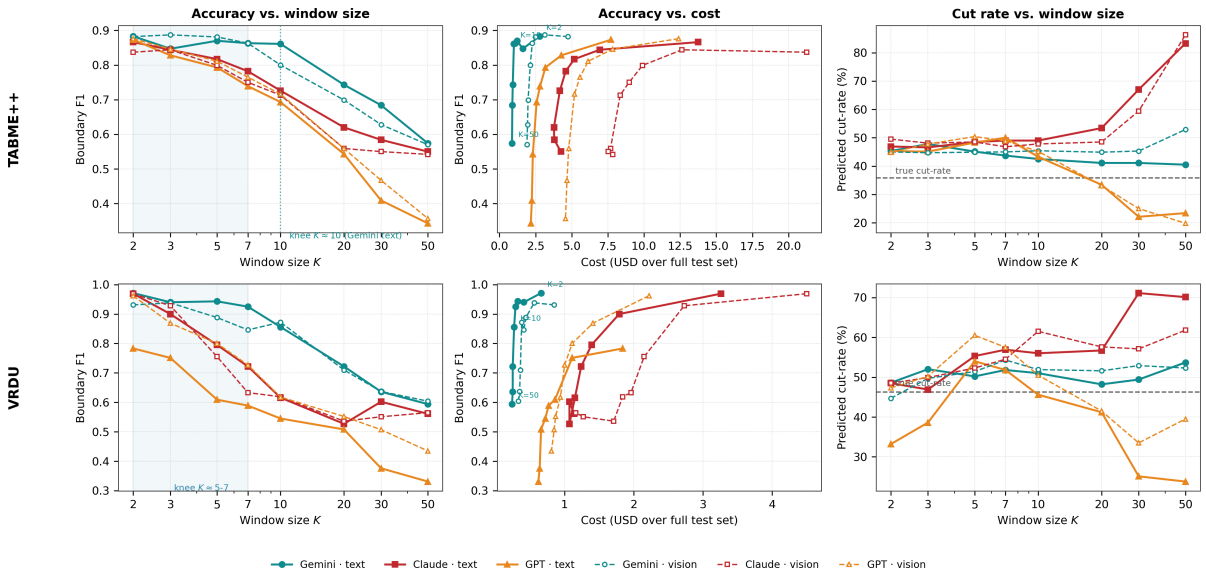


Figure 3: MSBD behavior across window sizes on TABME++ (top) and VRDU (bottom); solid and dashed curves show text and vision inputs. *Left*: Boundary F1 versus window size K—the useful window (knee) depends on the model and dataset, with GPT-5.4-mini degrading earliest. *Center*: Boundary F1 versus test-set cost as K increases—Gemini provides the strongest cost–accuracy trade-off. *Right*: predicted cut rate versus K—Claude and Gemini increasingly over-segment, whereas GPT-5.4-mini under-segments at large K; horizontal dashed lines mark the ground-truth cut rates.

8 Discussion

Recommended configuration Among the evaluated configurations, Gemini with text input provides the strongest accuracy–cost trade-off. TABME++ supports windows of approximately K=5–10, whereas the form-heavy VRDU corpus requires a more conservative range of approximately K=5–7. Because the knee depends on the model and corpus, deployments should select K using a labelled validation set.

Agentic and workflow integration MSBD can serve as a low-cost segmentation component in agentic and document-processing workflows. It segments a stream in approximately $(N-1)/(K-1)$ calls, after which the recovered documents can be routed to classification, extraction, or verification components. Window size provides an efficiency control: larger windows reduce requests, while smaller windows or BD offer a safer fallback when validation shows degraded accuracy.

9 Conclusion

Zero-shot LLMs and VLMs can perform PSS without task-specific training, but standard formulations require one model call per boundary. MSBD reduces this overhead by predicting multiple boundaries in each window while preserving strong accuracy up to a model- and corpus-dependent knee. Among the evaluated configurations, Gemini with text input provides the strongest accuracy–efficiency trade-off, particularly at moderate window sizes. Beyond the knee, accuracy declines and models exhibit malformed outputs or silent over- and under-segmentation. Reliable deployment therefore requires corpus-specific window selection, output validation, and monitoring for segmentation errors.

Limitations

We run a single pass per configuration at temperature 0 to control cost. We do not report variance across seeds or prompt rewording.

Our evidence covers two English-language datasets and three models; TABME++ is a synthetic concatenation and VRDU is concatenated by our own recipe, so the knees we report may shift on other corpora or construction recipes. Costs and latency reflect provider conditions at our run date, so both are relative comparisons rather than durable, absolute figures.

We use one prompt and JSON schema per formulation, with no per-model tuning. The large-window failure modes may be partly tied to this phrasing, and a per-model schema tweak could push each knee outward.

References

Onur Agin, Cagdas Ulas, Mehmet Ahat, and Can Bekar. 2015. An approach to the segmentation of multi-page document flow using binary classification. In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, volume 9443, pages 216–222. SPIE.

Amazon Web Services. 2026. Claude Haiku 4.5 – Amazon Bedrock. Accessed 2026-07-23.

Fabricio Ataides Braz, Nilton Correia da Silva, and Jonathan Alis Salgado Lima. 2021. Leveraging effectiveness and efficiency in page stream deep segmentation. *Engineering Applications of Artificial Intelligence*, 105:104394.

Lukas Busch, Ruben van Heusden, and Maarten Marx. 2023. Using deep-learned vector representations for page stream segmentation by agglomerative clustering. *Algorithms*, 16(5):259.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810.

Kevyn Collins-Thompson and Radoslav Nickolov. 2002. A clustering-based algorithm for automatic document separation. In *SIGIR 2002 Workshop on Information Retrieval and OCR: From Converting Content to Grasping, Meaning, Tampere, Finland*.

Hani Daher and Abdel Belaïd. 2014. Document flow segmentation for business applications. In *Document Recognition and Retrieval XXI*, volume 9021 of *Proc. SPIE*, page 90210G.

Mehmet Arif Demirtaş, Berke Oral, Mehmet Yasin Akpınar, and Onur Deniz. 2022. Semantic parsing of interpage relations. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1579–1585. IEEE.

Google. 2026. Gemini 3.1 Flash-Lite. Accessed 2026-07-23.

Albert Gordo, Marçal Rusiñol, Dimosthenis Karatzas, and Andrew D. Bagdanov. 2013. Document classification and page stream segmentation for digital mailroom applications. In *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 621–625. IEEE.

Abhijit Guha, Abdulrahman Alahmadi, Debabrata Samanta, Mohammad Zubair Khan, and Ahmed H. Alahmadi. 2022. A multi-modal approach to digital document stream segmentation for title insurance domain. *IEEE Access*, 10:11341–11353.

Hunter Heidenreich, Ratish Dalvi, Rohith Mukku, Nikhil Verma, and Neven Pičuljan. 2024. Large language models for page stream segmentation. *Preprint*, arXiv:2408.11981.

Hunter Heidenreich, Ratish Dalvi, Nikhil Verma, and Yosheb Getachew. 2025. Page stream segmentation with LLMs: Challenges and applications in insurance document automation. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 305–317, Abu Dhabi, UAE. Association for Computational Linguistics.

Md Mofijul Islam, Md Sirajus Salekin, Nivedha Balakrishnan, Vincil C. Bishop, III, Niharika Jain, Spencer Romo, Bob Strahan, Boyi Xie, and Diego A. Socolinsky. 2026. DocSplit: A comprehensive benchmark dataset and evaluation approach for document packet recognition and splitting. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 159–183, San Diego, California, USA. Association for Computational Linguistics.

Thisanaporn Mungmeeprued, Yuxin Ma, Nisarg Mehta, and Aldo Lipani. 2022. Tab this folder of documents: Page stream segmentation of business documents. In *Proceedings of the 22nd ACM Symposium on Document Engineering (DocEng '22)*, pages 1–10, San Jose, CA, USA. Association for Computing Machinery.

Chems Eddine Neche, Yolande Belaïd, and Abdel Belaïd. 2020. Use of language models for document stream segmentation. In *International Conference on Pattern Recognition Applications and Methods*.

OpenAI. 2026. GPT-5.4 mini model. Accessed 2026-07-23.

Marc Serra Ortega, Emanuele Vivoli, Artemis Llabrés, and Dimosthenis Karatzas. 2025. CoSMo: A multimodal transformer for page stream segmentation in comic books. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 7565–7573. IEEE.

Ruben Van Heusden, Jaap Kamps, and Maarten Marx. 2022. Wooir: a new open page stream segmentation dataset. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 24–33.

Ruben van Heusden, Jaap Kamps, and Maarten Marx. 2024. OpenPSS: An open page stream segmentation benchmark. In *Linking Theory and Practice of Digital Libraries: 28th International Conference on Theory and Practice of Digital Libraries*, volume 15177 of *Lecture Notes in Computer Science*, pages 413–429. Springer.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023. VRDU: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 5184–5193. Association for Computing Machinery.

Gregor Wiedemann and Gerhard Heyer. 2021. Multimodal page stream segmentation with convolutional neural networks. *Language Resources and Evaluation*, 55(1):127–150.

A Extended Results

This section reports the complete MSBD window-size sweep underlying the main-paper results. Table 4 presents Boundary F1, Table 5 reports model calls and test-set cost, and Figures 4 and 5 summarize the accuracy and latency trends. The $K=2$ setting is equivalent to the single-boundary BD baseline.

	TABME++						VRDU					
K	Cla-txt	Cla-vis	Gem-txt	Gem-vis	GPT-txt	GPT-vis	Cla-txt	Cla-vis	Gem-txt	Gem-vis	GPT-txt	GPT-vis
2	0.866	0.837	0.883	0.882	0.873	0.876	0.969	0.969	0.971	0.931	0.783	0.962
3	0.844	0.844	0.847	0.887	0.828	0.846	0.900	0.928	0.940	0.938	0.751	0.869
5	0.817	0.799	0.870	0.881	0.793	0.811	0.795	0.755	0.943	0.888	0.609	0.800
7	0.782	0.751	0.863	0.863	0.739	0.765	0.722	0.633	0.925	0.846	0.589	0.727
10	0.726	0.713	0.861	0.800	0.693	0.716	0.616	0.619	0.856	0.872	0.545	0.618
20	0.620	0.559	0.743	0.699	0.543	0.559	0.527	0.536	0.722	0.710	0.508	0.552
30	0.584	0.550	0.684	0.628	0.409	0.467	0.602	0.551	0.636	0.637	0.376	0.507
50	0.550	0.542	0.574	0.570	0.343	0.357	0.561	0.565	0.594	0.604	0.331	0.435

Table 4: Boundary F1 across the complete MSBD sweep. The $K=2$ setting is equivalent to BD. Cla, Gem, and GPT denote Claude Haiku 4.5, Gemini 3.1 Flash-Lite, and GPT-5.4-mini; txt and vis denote text and vision inputs. Results at large K on VRDU are based on relatively few windows and should therefore be interpreted cautiously. Costs and call counts for the same configurations appear in Table 5.

Accuracy across window sizes Gemini retains accuracy over larger windows than the other models. On TABME++, its text-input Boundary F1 remains between 0.861 and 0.870 for $K=5$–10, compared with 0.883 at $K=2$. With vision input, Gemini peaks at $K=3$ (0.887) and remains close to BD at $K=5$ (0.881). Claude and GPT-5.4-mini begin degrading at smaller windows.

The usable window is narrower on VRDU. Gemini with text remains strong through $K=7$ (0.925), but falls to 0.856 at $K=10$. Claude and GPT-5.4-mini decline more rapidly. The small non-monotonic changes at large K on VRDU should not be interpreted as recovery, because those settings contain few evaluation windows.

	TABME++							VRDU						
K	Calls	Cla-txt	Cla-vis	Gem-txt	Gem-vis	GPT-txt	GPT-vis	Calls	Cla-txt	Cla-vis	Gem-txt	Gem-vis	GPT-txt	GPT-vis
2	6236	13.75	21.25	2.79	4.74	7.72	12.39	1102	3.26	4.50	0.66	0.85	1.84	2.22
3	3118	6.94	12.63	1.61	3.13	4.28	7.84	551	1.79	2.73	0.41	0.56	1.11	1.41
5	1559	5.18	9.90	1.22	2.49	3.18	6.14	275	1.39	2.15	0.32	0.44	0.86	1.11
7	1039	4.58	8.97	1.09	2.27	2.81	5.57	183	1.24	1.96	0.29	0.41	0.77	1.00
10	692	4.18	8.35	1.00	2.13	2.57	5.19	122	1.15	1.84	0.27	0.38	0.72	0.94
20	328	3.77	7.68	0.92	1.98	2.31	4.79	58	1.07	1.71	0.25	0.36	0.66	0.87
30	215	3.79	7.52	0.89	1.94	2.24	4.67	38	1.07	1.27	0.25	0.35	0.64	0.85
50	127	4.25	7.84	0.87	1.90	2.17	4.57	22	1.14	1.16	0.24	0.33	0.62	0.81

Table 5: Test-set cost in USD and model-call counts across the complete MSBD sweep. The $K=2$ setting is equivalent to BD. Call counts depend only on the dataset and window size; individual runs may contain one or two fewer completed requests when a model call fails. Boundary F1 for the same configurations appears in Table 4.

Calls and cost Call count decreases approximately in proportion to $1/(K-1)$. On TABME++, increasing K from 2 to 50 reduces calls from 6,236 to 127; on VRDU, calls decrease from 1,102 to 22.

Cost falls much more slowly because each remaining call contains more pages. For Gemini with text input on TABME++, $K=5$ and $K=10$ reduce cost from \$2.79 to \$1.22 and \$1.00, respectively. At $K=50$, cost reaches \$0.87, only a $3.2\times$ reduction despite approximately $49\times$ fewer calls. Costs also flatten or occasionally increase at the largest windows, showing that larger request payloads can offset further reductions in request count. Vision input is more expensive than text input for every model and window shown.

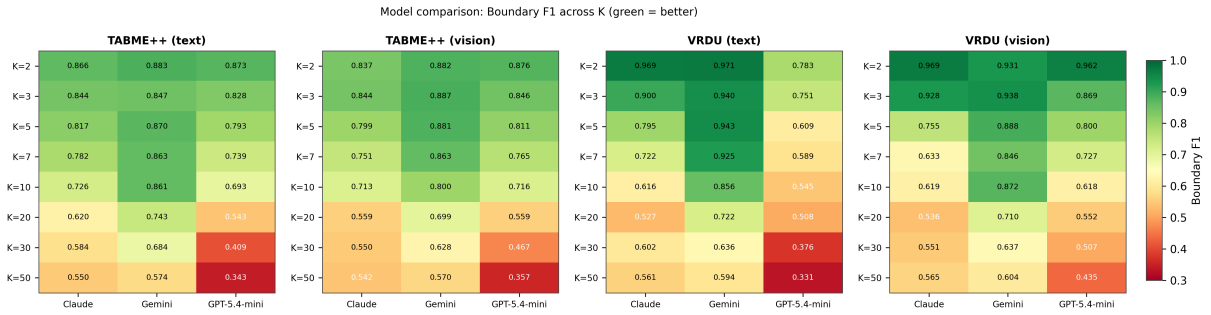


Figure 4: Boundary F1 across datasets, models, input modalities, and window sizes. Darker cells indicate higher F1. Gemini retains accuracy over the widest range of K, while GPT-5.4-mini generally degrades earliest.

Model and modality patterns Figure 4 highlights that Gemini provides the most stable performance across window sizes. The relative value of text and vision depends on the model, dataset, and window size: vision occasionally improves small-window accuracy, but it does not provide a consistent advantage and generally degrades faster as more pages are added. GPT-5.4-mini shows the earliest and steepest large-window decline.

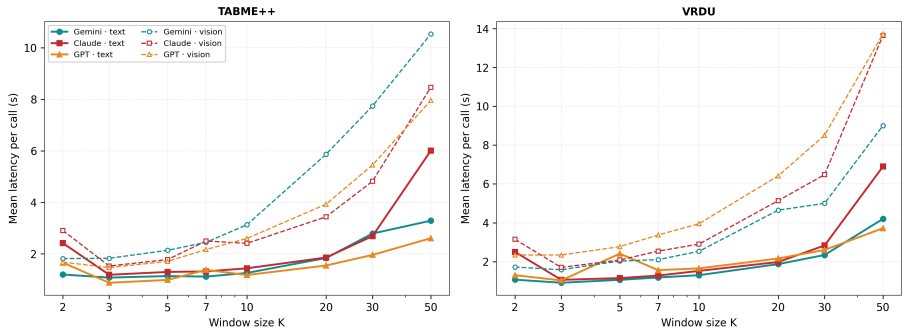


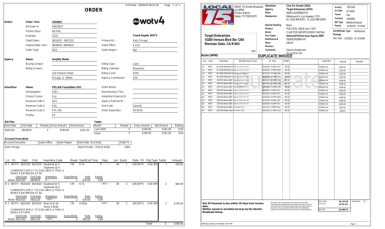
Figure 5: Mean latency per model call across the MSBD sweep. Text-input latency remains broadly stable at small and medium windows before increasing at larger K, while vision latency generally rises more quickly as additional pages are included.

Per-call latency Increasing K reduces the number of calls but does not make each call faster. Text-input latency remains broadly stable through the useful window and then increases as prompts become larger; vision-input latency rises more quickly at large windows. MSBD's throughput improvement therefore comes mainly from issuing fewer requests, rather than from reducing the latency of each request.

B Qualitative Error Gallery

The galleries below show MSBD windows (K=20, VRDU) illustrating the two failure directions from §7. Every panel displays the two pages that flank a single boundary decision: `truth` gives the ground-truth label (`cut` — a new document begins on the right-hand page; `continue` — both pages belong to one document), and the badges below report each model's prediction, green when it matches the truth and red when it errs. Panels are grouped by *who* errs: row headers mark whether all three models fail together or only one model fails while the other two stay correct. Figure 6 shows clear boundaries the models agree on; Figure 7 shows over-segmentation (false cuts inside one document, the dominant Claude/Gemini failure); and Figures 8 and 9 show under-segmentation (missed cuts that merge two documents), for Claude and Gemini in the former and for GPT-5.4-mini alone, its dominant failure mode, in the latter.

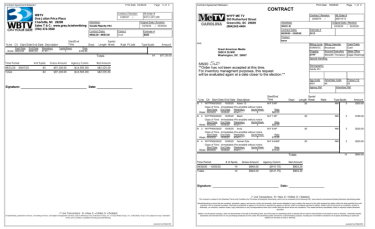
All three cut correctly



(1) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **cut** ✓

An order form ends and a different station's invoice begins.

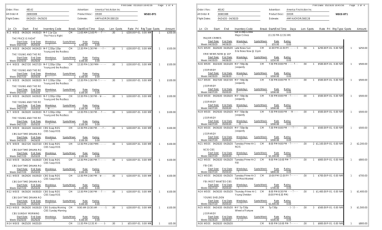


(2) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **cut** ✓

One contract ends and a different advertiser's contract begins.

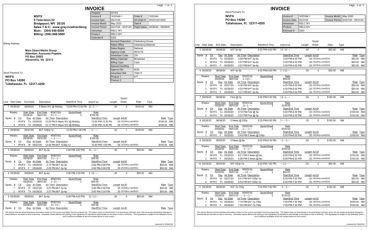
All three continue correctly



(3) truth: CONTINUE VRDU

Gemini **continue** ✓ Claude **continue** ✓ GPT **continue** ✓

Two pages of a single WSEE-DT1 order form.



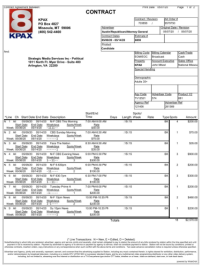
(4) truth: CONTINUE VRDU

Gemini **continue** ✓ Claude **continue** ✓ GPT **continue** ✓

Two pages of one WVFX invoice.

Figure 6: Correctly handled boundaries (VRDU). On clear transitions (distinct letterheads, advertisers, or form types), the models agree with the ground truth in both the `cut` and `continue` directions. These easy cases sit behind VRDU's high scores at small K.

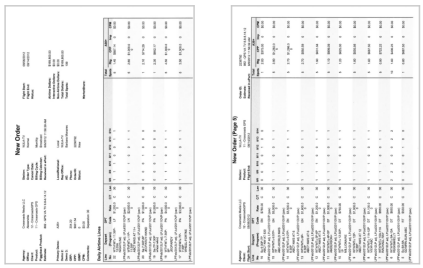
All three over-cut a continuation



(1) truth: CONTINUE VRDU

Gemini **cut** ✗ Claude **cut** ✗ GPT **cut** ✗

Both pages are one broadcast contract (same source file, pages 1–2 of 2).

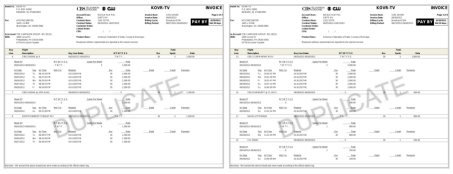


(2) truth: CONTINUE VRDU

Gemini **cut** ✗ Claude **cut** ✗ GPT **cut** ✗

A single multi-page broadcast order — a line-item table continuing onto its next page.

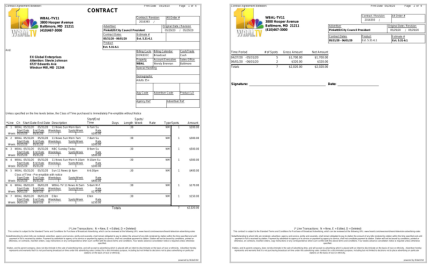
Only Claude over-cuts



(3) truth: CONTINUE VRDU

Gemini **continue** ✓ Claude **cut** ✗ GPT **continue** ✓

Two pages of a single KOVR-TV issue-ad invoice.

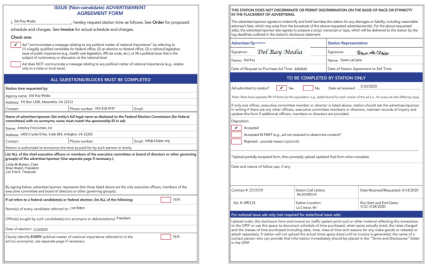


(4) truth: CONTINUE VRDU

Gemini **continue** ✓ Claude **cut** ✗ GPT **continue** ✓

Two consecutive pages of one WBAL-TV station contract.

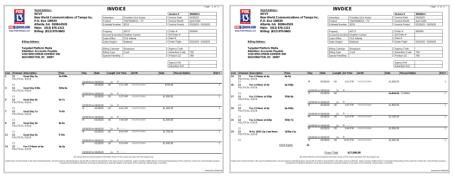
Claude and GPT over-cut (left) · Only Gemini over-cuts (right)



(5) truth: CONTINUE VRDU

Gemini **continue** ✓ Claude **cut** ✗ GPT **cut** ✗

An issue-advertising agreement form and its own terms page — a single filing.



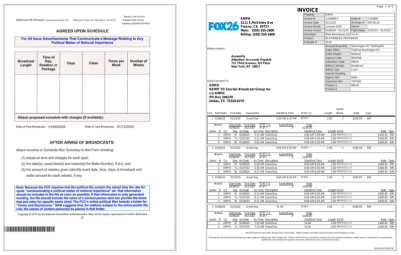
(6) truth: CONTINUE VRDU

Gemini **cut** ✗ Claude **continue** ✓ GPT **continue** ✓

Two pages of a single multi-page invoice — the line-item table continues onto the next page; the rare reverse with a lone over-cut.

Figure 7: Over-segmentation (VRDU): predicted cuts fall *inside* what is really a single document (ground truth `continue`), fragmenting one filing across several predicted documents. Top row: all three models over-cut a continuation page (a contract's second page; a line-item table continuing across pages). Middle row: only Claude over-cuts while Gemini and GPT-5.4-mini correctly continue. Bottom row: Claude and GPT-5.4-mini over-cut an agreement form's own terms page (left), and in the rare reverse only Gemini over-cuts a multi-page invoice (right).

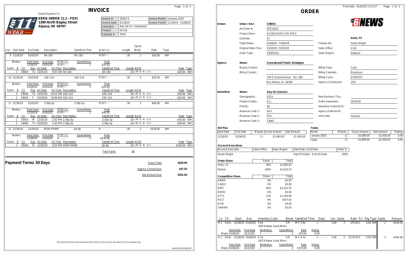
All three miss the boundary



(1) truth: CUT VRDU

Gemini **continue** ✗ Claude **continue** ✗ GPT **continue** ✗

An issue-ad schedule form ends and a new broadcast invoice begins.

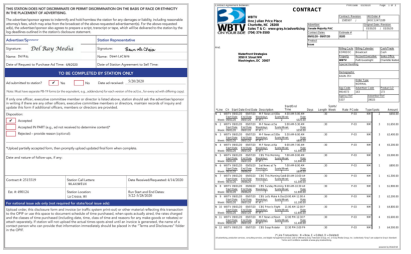


(2) truth: CUT VRDU

Gemini **continue** ✗ Claude **continue** ✗ GPT **continue** ✗

One invoice ends and a new station's order begins.

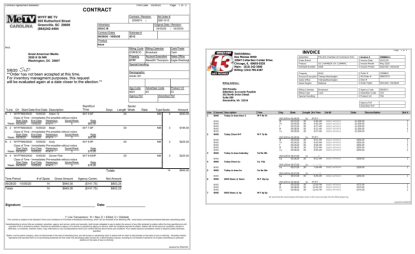
Only Claude merges (left) · Only Gemini merges (right)



(3) truth: CUT VRDU

Gemini **cut** ✓ Claude **continue** ✗ GPT **cut** ✓

A station agreement form ends and a new WBTV contract begins.

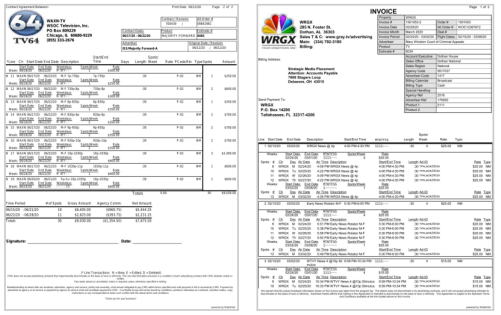


(4) truth: CUT VRDU

Gemini **continue** ✗ Claude **cut** ✓ GPT **cut** ✓

A contract ends and a new invoice begins.

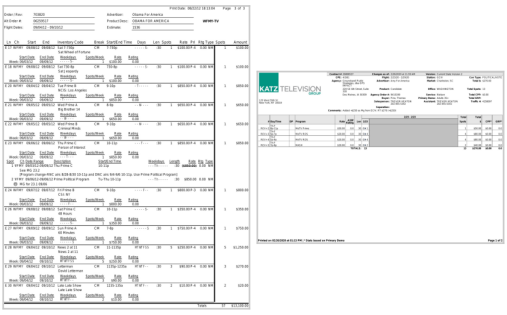
Figure 8: Under-segmentation (VRDU): a genuine boundary (ground truth `cut`) is predicted as `continue`, merging two documents into one. Top row: all three models miss the boundary on visually ambiguous transitions (a schedule form giving way to an invoice; an invoice to a new station's order). Bottom row: only Claude (left) or only Gemini (right) merges the documents while the other two models cut correctly.



(1) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **continue** ✗

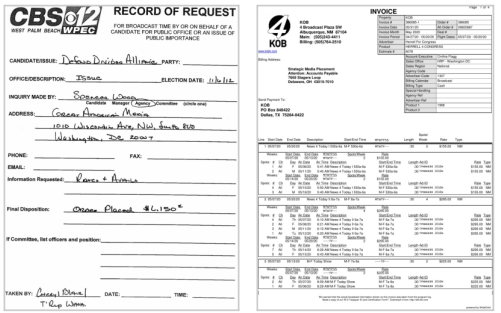
Only GPT merges; Gemini and Claude cut. A station contract ends and a new invoice begins.



(2) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **continue** ✗

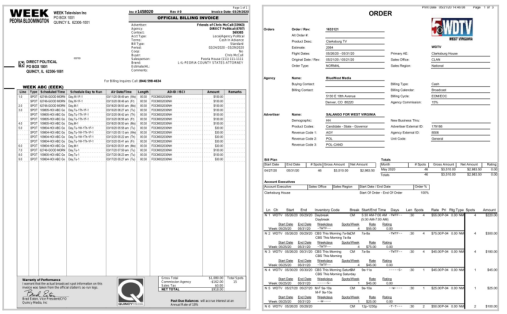
Only GPT merges; Gemini and Claude cut. A broadcast order line-item table ends and a new station's order begins.



(3) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **continue** ✗

Only GPT merges; Gemini and Claude cut. A political-file record of request ends and a new invoice begins.



(4) truth: CUT VRDU

Gemini **cut** ✓ Claude **cut** ✓ GPT **continue** ✗

Only GPT merges; Gemini and Claude cut. A billing invoice ends and a new order form begins.

Figure 9: Under-segmentation by GPT-5.4-mini (VRDU): in each of these windows GPT-5.4-mini alone predicts `continue` at a genuine boundary (ground truth `cut`), merging adjacent filings, whereas Gemini and Claude both cut correctly.

C Prompts

Prompts provided in this section are the exact PC, BD, and MSBD prompts used in all runs.

Page Classification (PC) Prompt

```
You are a document page classifier. Your task is to determine if a given document page is
the first page of a document (first_page), a subsequent/non-first page (non_first_page).

Analyze the page using its text and/or the page image (whichever is provided),
considering the following features:
1. Headers and Titles - Is there a prominent document title, organization name, or main
   heading at or near the top?
2. Page Numbers - Does the page have a page number like "1", "1/2", "1 in 3", or similar?
   First pages often use these formats or may lack a page number entirely.
3. Content Structure - Does the page contain introductory elements such as an executive
   summary, table of contents, or document metadata?
4. Document Identifiers - Are there document IDs, dates, reference numbers, or metadata
   typical of cover/first pages?
5. Formatting Elements - Is there distinctive formatting (centered or larger text, extra
   spacing, logos) suggesting a cover or title page?
6. Opening Content - Does the text start with introductory phrases, a document purpose
   statement, or a formal opening (e.g., "Dear [Name]," "To Whom It May Concern," or
   similar)?

The possible Classification Labels:
first_page, non_first_page

Output Format:
Return your response in the following JSON format:
```

```
{
"answer": "The predicted classification",
"explanation": "A brief justification for the classification"
}

Example output:
{
    "answer": "first_page",
    "explanation": "Page has a centered organization title, a document date at the top,
    and begins with an introductory letter opening, all of which indicate this is the
    first page"
}

Base your decision on the page text and/or the page image provided. Return only the JSON
object.
```

Boundary Decision (BD) Prompt

```
You are a document page classifier. Your task is to determine whether the last page in a
sequence of two or more pages belongs to the same document as the previous page(s)
(continuous_page) or starts a different document (cut_page).

Analyze the document pages using the page text and/or the page images (whichever is
provided), considering features such as:
1. Headers and footers - Do they match across pages?
2. Page numbers - Are they sequential?
3. Content flow - Does the text flow naturally from the previous page(s) to the last
   page?
4. Writing style - Is the language, terminology, and tone consistent?
5. Document identifiers - Are document IDs, reference numbers, or metadata consistent?
6. Sentence continuity - Do sentences break across pages and continue smoothly?
7. Visual layout - When page images are provided, compare letterhead, logos, margins,
   fonts, columns, and overall design; a new document usually begins with a distinct
   cover/first-page layout.

The possible classifications are:
continuous_page, cut_page

Output Format:
Return your response in JSON format with:
{
    "answer": "The predicted classification",
    "explanation": "A brief justification for the classification"
}

Example output:
{
    "answer": "continuous_page",
    "explanation": "Pages show sequential numbering (pages 3-4), matching headers with
    company name, and the sentence at the bottom of first page continues naturally at
    the top of the second page"
}

Remember that "continuous_page" means the last page continues from the previous page(s),
while "cut_page" means the last page starts a new document and differs from the previous
page(s).
Note: Don't include any additional text or formatting in your response. Just return the
JSON object.
```

Multi-Split Boundary Decision Prompt Template

```
You are a document page classifier tasked with analyzing a sequence of N consecutive
pages. Your job is to determine for each pair of adjacent pages whether they belong to
the same document (continuous_page) or whether there is a document boundary between them
(cut_page).

Analyze each pair of adjacent pages using the page text and/or the page images (whichever
is provided), considering features such as:
1. Headers and footers - Do they match across pages?
2. Page numbers - Are they sequential?
3. Content flow - Does the text flow naturally between adjacent pages?
4. Writing style - Is the language, terminology, and tone consistent?
5. Document identifiers - Are document IDs, reference numbers, or metadata consistent?
6. Sentence continuity - Do sentences break across pages and continue smoothly?
7. Visual layout - When page images are provided, compare letterhead, logos, margins,
   fonts, columns, and overall design; a new document usually begins with a distinct
   cover/first-page layout.
```

```
For the N consecutive pages, you need to make (N-1) decisions:
- Split 1: Between pages 1 and 2
- Split 2: Between pages 2 and 3
- Split 3: Between pages 3 and 4
- ... and so on until Split (N-1): Between pages (N-1) and N

The possible classifications for each split are:
continuous_page, cut_page

Output Format:
Return your response in JSON format with a decision for each split:
{
    "split_1": "continuous_page" or "cut_page",
    "split_2": "continuous_page" or "cut_page",
    "split_3": "continuous_page" or "cut_page",
    ... continue for all (N-1) splits
}

Example output for 5 pages:
{
    "split_1": "continuous_page",
    "split_2": "cut_page",
    "split_3": "continuous_page",
    "split_4": "continuous_page"
}

Remember:
- "continuous_page" means adjacent pages belong to the same document
- "cut_page" means there is a document boundary (the second page starts a new document)

Note: Don't include any additional text or formatting in your response. Just return the
JSON object.
```

Text, Pixels, or Both? Evaluating Input Representations for Multimodal Document QA

Nikhil Reddy Pottanigari* **Sepideh Kharaghani*** **Saverio Vadacchino**
Alejandro Posada **Kurt MacDonald** **Ying Zhang**

ServiceNow Canada
{nikhilreddy.pottanigari, sepideh.kharaghani,
saverio.vadacchino, alejandro.posada,
kurt.macdonald, yin.zhang}@servicenow.com

Abstract

Every document QA system begins with a choice that is rarely studied on its own: whether to feed the model page images, extracted text, or both. We isolate this choice, holding the prompt, judge, and scoring pipeline fixed, across four commercial model endpoints, two corpora, and two context regimes (gold evidence pages and the full document). On documents that fit the image budget, page images lead on accuracy at every document length on both corpora, but this advantage carries a growing latency and cost premium: text latency stays roughly flat as documents lengthen while image latency rises steadily. Text and images also fail on different questions, with exactly one representation correct on **19–25%** of items across the reported cells, so neither subsumes the other. Exploiting this complementarity, a lightweight TF-IDF router that reads only the question text gains **2.6 points** over always-text while cutting median latency **30%** relative to always-vision, on a document-disjoint held-out split.

1 Introduction

Before a document QA system can answer a question, a simple preprocessing choice sets how the document reaches the model: as page images, as extracted text, or both (Zhang et al., 2024). A knowledge-base article may encode evidence in a screenshot, diagram, or spatial arrangement that OCR text preserves only partially. Sending every page as an image retains this visual evidence but increases input size and inference latency on long documents. Combining both modalities can be worse than either channel alone, depending on the model.

This choice affects accuracy and latency together. Always sending images for a long report, or text for a screenshot-heavy knowledge base, can hurt answer quality or increase inference time. However, representation is often evaluated together with changes to the model, prompt, retrieval method, or evaluator, making its contribution difficult to identify.

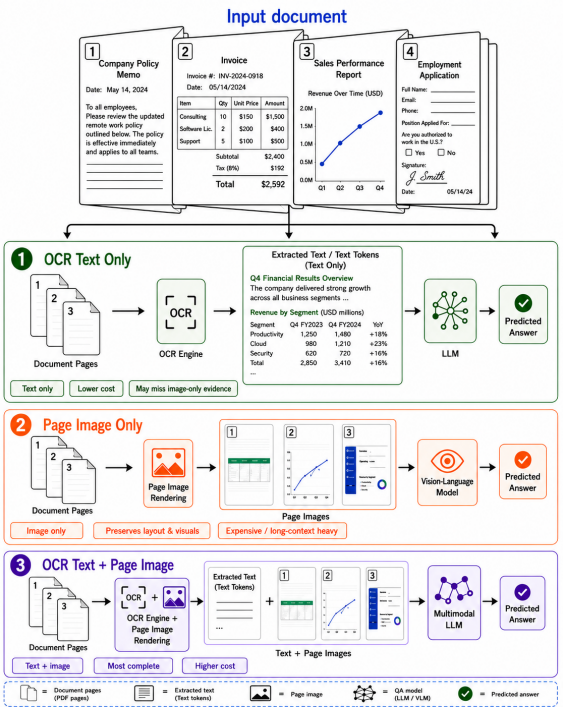


Figure 1: Three document representations use the same QA prompt and judge. IMAGE preserves layout and visual content. TEXT is compact and uncapped by page count. TEXT+IMAGE concatenates OCR text followed by page images at the cost of larger context.

We address this by comparing **IMAGE** (page images), **TEXT** (OCR text), and **TEXT+IMAGE** (text then images) across four commercial endpoints from two providers, one public benchmark (Ma et al., 2024b), and one enterprise knowledge base. Each model runs under two retrieval regimes: **ORACLE** (gold evidence pages) and **FULL** (full document). Within each model, corpus, and regime, the prompt, judge, and scoring pipeline are held

* Equal contribution.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 46–62
October 24, 2026 ©2026 Association for Computational Linguistics

constant; the representation and its resulting input size are the intended experimental differences.

Our contributions:

- A **controlled representation benchmark** across four models, two corpora, and two retrieval regimes with a fixed prompt and judge.
- A **systematic analysis of how document length interacts with input representation**, showing that IMAGE leads on accuracy within the image budget while their latency and cost grow with document length (Section 4.1).
- A **lightweight question-text router** that captures much of the accuracy gap to always-vision while retaining most of text's latency advantage, evaluated on a document-disjoint held-out split (Section 5).

2 Related Work

Document QA and representation. Document QA has expanded from single-page visual understanding (Mathew et al., 2021) to multi-page reasoning over tables, charts, and cross-page evidence (Van Landeghem et al., 2023; Ma et al., 2024b; Deng et al., 2025). Layout-aware models such as LayoutLMv3 (Huang et al., 2022) and Pix2Struct (Lee et al., 2023) learn joint text–image representations. Berghaus et al. (2025) compare image- and text-based processing for invoice extraction and report model- and dataset-dependent results. Our experiments compare representations under a common prompt and evaluation pipeline across four models, two corpora, and two regimes.

Multimodal document retrieval and benchmarks. Retrieval-augmented generation (Lewis et al., 2020) couples two decisions: which evidence to retrieve and how to represent it. PDF-Triage (Saad-Falcon et al., 2024) routes questions to document structure; M3DocRAG (Cho et al., 2024) and MMDocRAG (Dong et al., 2025) retrieve multimodal document evidence. Our ORACLE/FULL regimes compare gold-page input with no page retrieval. Long-context position effects are established for text (Liu et al., 2024), and prompt-compression methods reduce text context length (Jiang et al., 2023; Li et al., 2023). We examine how document representation interacts with full-document context and a fixed image budget. A parallel line of work asks whether page images can replace extracted text directly. ColPali (Faysse et al., 2025) and Document Screenshot Embedding (Ma et al., 2024a) embed rendered pages rather than parsed text, and VisRAG (Yu et al., 2025) builds a vision-language RAG pipeline over page images, each reporting that image representations avoid parsing loss on layout-rich documents. Our controlled comparison isolates this same text-versus-image choice at generation time rather than at retrieval time.

Parsing and evaluation. Text quality depends on the parsing pipeline. Neural parsers (Blecher et al., 2024; Nassar et al., 2025) produce structured markup from page images, while direct multimodal processing can retain non-textual evidence that text extraction omits (Shen et al., 2026). Our text condition uses a proprietary pipeline that outperforms Tesseract (Smith, 2007) on three OCR datasets (Appendix Table 8); this comparison does not eliminate extraction error as a possible source of the text–image gap. We use the same LLM judge and prompt in every condition, while recognizing that LLM-based evaluation can itself introduce error (Zheng et al., 2023; Liu et al., 2023; Bavaresco et al., 2025).

3 Setup

Representation strategies. IMAGE sends each page as a 300-DPI image; TEXT sends page text from a proprietary OCR pipeline whose comparison with few open-sourced OCR engines appears in Appendix Table 8; TEXT+IMAGE supplies the OCR text followed by the page images. All three use the same QA prompt (Appendix F).

Datasets. Two corpora span short, visually dense articles and longer mixed-format documents (Table 1). The enterprise knowledge base (KB) averages 5.7 pages per document and has 93.6% of its questions labeled as requiring visual or cross-modal evidence. MMLongBench-Doc (Ma et al., 2024b) contains longer PDFs (48.3 pages on average) with single-page, cross-page, and unanswerable questions. The KB is internal, whereas MMLongBench-Doc provides a public comparison corpus (Appendix C).

Models and retrieval regimes. We evaluate four models from two providers: Gemini 3.1 Flash-Lite and Gemini 3.1 Pro (Google via Vertex AI), GPT-5.4 and GPT-5.4-mini (OpenAI via Azure)[1]. Each model runs under two regimes that bracket realistic retrieval:

[1] Exact API identifiers in Appendix B.

Table 1: Evaluation corpora. The ≤**50-page** columns count the documents short enough to fit the vision page cap and their questions (the subset used in Table 3).

Corpus	Docs	≤50-page Docs	Questions	≤50-page Questions	Profile
Enterprise KB	1,447	1,445	5,169	5,163	Short support articles with screenshots, UI states, diagrams, and cross-modal references; median 4pp., max 237pp.; 99.86% of docs ≤50pp.
MMLongBench-Doc	135	101	1,091	798	Visually rich PDFs: reports, papers, manuals; cross-page and unanswerable questions (Ma et al., 2024b); median 28pp., max 468pp. (mean 48.3); 25% of docs >50pp.

Table 2: Evaluation metrics: accuracy as the primary outcome; pairwise disagreement, and latency

Metric	Definition	Interpretation
Accuracy	$\mathrm{Acc}(s, m, d, r) = \lvert Q\rvert^{-1} \sum_q c_q$, with $c_q \in [0, 1]$ the mean judged correctness across prediction and judge runs.	Average judged answer correctness; higher is better.
Pairwise disagreement	$\mathrm{Dis}(s_i, s_j) = \lvert Q\rvert^{-1} \sum_q \mathbf{1}[b_{q,i} \neq b_{q,j}]$, where $b_{q,s}$ is the binary correctness label used for paired analysis.	Higher values indicate more complementary correctness outcomes.
Latency	Average seconds per question from run telemetry, by representation and regime.	High latency can make an accurate route impractical.

- **ORACLE** supplies only the gold evidence pages. It measures answer generation when annotated evidence pages are provided, without evaluating page retrieval.
- **FULL** sends the full document with no retrieval. Here the model has to both find the relevant pages and extract the answer from a much larger, noisier context.

The gap between regimes measures sensitivity to replacing annotated evidence pages with the full document: a large positive gap indicates that the representation benefits from page selection, whereas a small or negative gap indicates comparable or better performance with full context.

Page cap and document scope. Due to model-provider limits, we cap all requests at **50 page images** and restrict the main analysis to documents of 50 pages or fewer, so that every representation sees the complete page set. We adopt this restriction specifically to isolate the page-cap truncation effect from any genuine modality effect. Applying the same cap to both providers keeps the IMAGE conditions comparable. For completeness, Appendix D reports results over *all* documents, still capped at 50 page images. In that setting, documents longer than 50 pages are truncated under IMAGE, while TEXT continues to see the full document, and the two effects are shown to be separable.

Scoring and metrics. Models produce free-form answers with a 3,000-token limit, temperature 0, and a fixed seed of 30. For each model, representation, dataset, and regime, we average correctness over **five prediction runs**. Each prediction is then evaluated by a fixed GPT-5.2 judge over **three judging runs**, and we report the mean score. Per-cell run-to-run standard deviations are reported alongside every accuracy and latency value in Tables 3 and 4. Reported ± values are standard deviations across the five prediction and three judging runs. Because decoding is greedy with a fixed seed, they quantify residual nondeterminism in the hosted endpoints and the judge rather than sampling uncertainty over questions. Table 2 defines the three metrics.

4 Results

We organize results around two questions: how the accuracy–latency trade-off changes with document length, and whether text and images miss the same questions. Table 3 and Figure 2 show that on the ≤50-page subset, image-bearing inputs (IMAGE, TEXT+IMAGE) lead on accuracy across document lengths on both corpora, while TEXT stays consistently lower but at a fraction of the latency: text latency is roughly flat in document length,

Table 3: Accuracy on the ≤**50-page subset** (documents short enough that every representation sees the full page set, i.e. no vision-input truncation). Cells show mean ± std across runs, quantifying endpoint and judge nondeterminism rather than question-level uncertainty; the *Avg.* row is the arithmetic mean of the four model means per column. Left block: ORACLE (gold evidence pages); right block: FULL (full document).

	ORACLE			FULL		
Model	TEXT	IMAGE	TEXT+IMAGE	TEXT	IMAGE	TEXT+IMAGE
Enterprise KB						
Gemini 3.1 Flash-Lite	0.576 ± 0.002	**0.729 ± 0.002**	0.724 ± 0.001	0.594 ± 0.001	0.711 ± 0.003	**0.714 ± 0.000**
Gemini 3.1 Pro	0.618 ± 0.003	**0.741 ± 0.004**	0.736 ± 0.002	0.634 ± 0.003	**0.728 ± 0.002**	0.728 ± 0.003
GPT-5.4	0.639 ± 0.003	0.677 ± 0.002	**0.733 ± 0.004**	0.663 ± 0.006	0.679 ± 0.004	**0.734 ± 0.005**
GPT-5.4-mini	0.612 ± 0.002	0.683 ± 0.003	**0.731 ± 0.002**	0.628 ± 0.003	0.672 ± 0.003	**0.715 ± 0.005**
Avg.	0.611	0.708	**0.731**	0.630	0.698	**0.723**
MMLongBench-Doc						
Gemini 3.1 Flash-Lite	0.529 ± 0.001	**0.625 ± 0.003**	0.613 ± 0.002	0.511 ± 0.001	0.615 ± 0.009	**0.620 ± 0.006**
Gemini 3.1 Pro	0.641 ± 0.007	**0.711 ± 0.008**	0.691 ± 0.003	0.628 ± 0.007	**0.690 ± 0.004**	0.623 ± 0.006
GPT-5.4	0.559 ± 0.007	0.667 ± 0.004	**0.673 ± 0.003**	0.599 ± 0.009	**0.686 ± 0.008**	0.683 ± 0.008
GPT-5.4-mini	0.527 ± 0.009	0.634 ± 0.005	**0.643 ± 0.007**	0.547 ± 0.005	0.633 ± 0.004	**0.642 ± 0.007**
Avg.	0.564	**0.659**	0.655	0.571	**0.656**	0.642

Table 4: Mean latency (seconds/question, ± std across runs) on the ≤**50-page subset**, where no representation is truncated by the vision page cap. Left block: ORACLE (gold evidence pages); right block: FULL (full document). The *Avg.* row is the arithmetic mean of the four model means per column. Lower is better.

	ORACLE			FULL		
Model	TEXT	IMAGE	TEXT+IMAGE	TEXT	IMAGE	TEXT+IMAGE
Enterprise KB						
Gemini 3.1 Flash-Lite	1.29 ± 0.00	2.42 ± 0.38	2.52 ± 0.31	1.40 ± 0.01	3.26 ± 0.60	3.20 ± 0.32
Gemini 3.1 Pro	5.57 ± 0.25	5.69 ± 0.36	5.36 ± 0.37	6.21 ± 0.25	5.68 ± 0.15	5.81 ± 0.22
GPT-5.4	3.46 ± 0.29	4.97 ± 0.42	4.48 ± 0.42	3.42 ± 0.18	5.43 ± 1.07	5.71 ± 0.43
GPT-5.4-mini	1.68 ± 0.20	2.55 ± 0.15	2.42 ± 0.15	1.78 ± 0.42	3.36 ± 0.13	3.26 ± 0.16
Avg.	3.00	3.91	3.70	3.20	4.43	4.50
MMLongBench-Doc						
Gemini 3.1 Flash-Lite	1.72 ± 0.09	3.59 ± 0.44	4.06 ± 0.16	2.10 ± 0.06	5.88 ± 0.52	6.93 ± 0.28
Gemini 3.1 Pro	6.10 ± 0.22	6.88 ± 0.69	6.80 ± 0.72	7.57 ± 0.31	8.40 ± 0.50	10.12 ± 2.31
GPT-5.4	4.21 ± 0.32	8.03 ± 0.75	7.83 ± 0.67	4.78 ± 0.24	13.37 ± 0.61	13.36 ± 0.51
GPT-5.4-mini	2.00 ± 0.23	5.05 ± 0.36	5.03 ± 0.31	2.22 ± 0.27	10.20 ± 0.31	10.09 ± 0.36
Avg.	3.51	5.89	5.93	4.17	9.46	10.13

whereas IMAGE and TEXT+IMAGE latency grows steadily. The accuracy gap narrows on the shortest and longest KB buckets but does not close. Section 5 then exploits the resulting complementarity with a lightweight router.

4.1 Accuracy–latency trade-off across representations

Page images retain layout and non-textual content that extracted text may omit (Shen et al., 2026), but any accuracy gain must be weighed against latency. We examine this trade-off under two regimes, on the ≤50-page subset where every representation sees the complete page set.

Oracle: image inputs lead with gold pages. Under ORACLE, either IMAGE or TEXT+IMAGE is best for every model on both corpora, with gains over TEXT of up to 14.8 percentage points on the KB and 11.6 points on MMLongBench-Doc (Table 3). This is consistent with image inputs retaining evidence omitted by the text pipeline, but it does not identify whether the difference arises from visual evidence, OCR errors, or model-specific processing. The gain comes at a latency cost: on the KB, image input raises average latency from 3.0 s (TEXT) to 3.9 s (IMAGE) and 3.7 s (TEXT+IMAGE), and on MMLongBench-Doc from 3.5 s to 5.9 s (Table 4). The larger ORACLE latency gap on MMLongBench-Doc is consistent with its questions citing more evidence pages per query (1.88 vs. 1.05 on the KB (Appendix C)), so each IMAGE or TEXT+IMAGE request carries more page images.

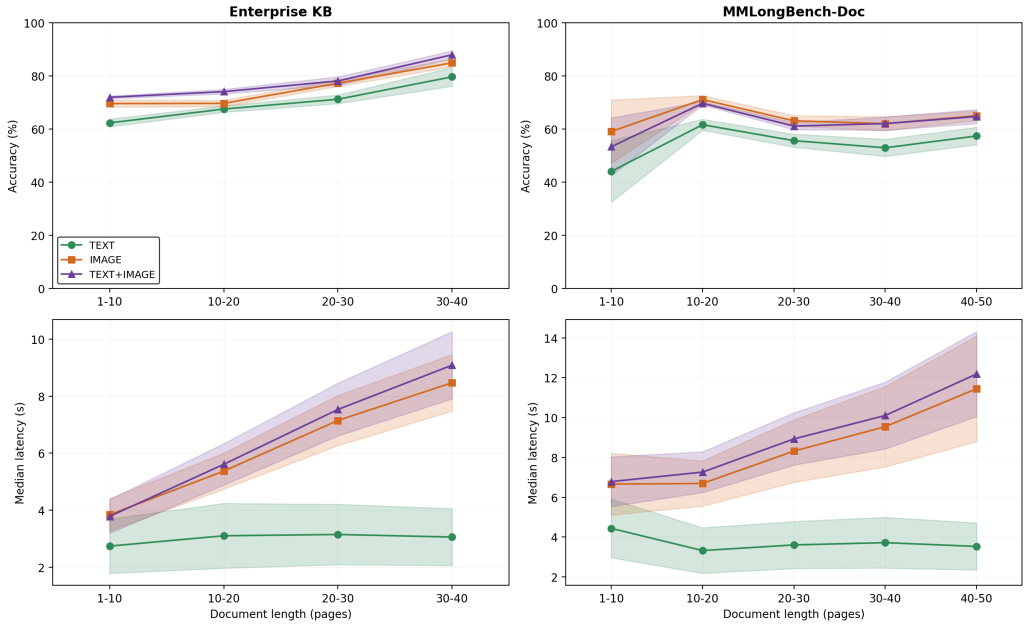


Figure 2: Accuracy (top) and median latency (bottom) versus document-length bucket on the ≤50-page subset under FULL retrieval, aggregated across all four models. The Enterprise KB has no 40–50 page bucket, as its documents are shorter (median 4 pages). Bands show ± std across models.

Full: image accuracy leads, but the latency gap widens. Under FULL, image-bearing inputs still lead on accuracy on both corpora: on the KB, TEXT+IMAGE and IMAGE average 0.72 and 0.70 versus 0.63 for TEXT; on MMLongBench-Doc, IMAGE averages 0.66 versus 0.57 for TEXT (Table 3). Figure 2 shows TEXT latency staying roughly flat in document length (∼3 s on the KB, ∼3–4 s on MMLongBench-Doc), while IMAGE and TEXT+IMAGE latency rise steadily, reaching ∼9 s on the longest KB bucket and ∼12 s on the longest MMLongBench-Doc bucket. Accuracy, by contrast, remains image-favored in every length bucket within the subset. The bucket-level view is included to show that the accuracy ordering does not reverse within the un-truncated range; the reversal reported in Appendix D appears only once the page cap binds.

Does combining both modalities help? Whether TEXT+IMAGE helps depends on the endpoint and corpus. GPT-5.4 and GPT-5.4-mini gain roughly 5 pp (5.6 and 4.8, respectively) from TEXT+IMAGE over IMAGE on the KB under ORACLE, and their IMAGE-vs-TEXT+IMAGE disagreement is the highest among the KB cells (13–14%, Table 6), showing that the additional text changes some correctness outcomes. Gemini Pro's IMAGE and TEXT+IMAGE outcomes differ less often (∼7–8% on the KB).

Table 5: Average cost (USD per 100 questions) and latency (seconds per question) per representation strategy, averaged across the four models and two datasets. Full per-model, per-dataset latency breakdowns are in Table 4.

Strategy	Cost ($/100q)	Latency (s/q)
ORACLE		
TEXT	0.61	3.25
IMAGE	0.91	4.90
TEXT+IMAGE	1.29	4.81
FULL		
TEXT	1.48	3.69
IMAGE	2.78	6.95
TEXT+IMAGE	3.51	7.31

On MMLongBench-Doc/FULL, Gemini Pro's TEXT+IMAGE falls behind its IMAGE result and roughly ties its TEXT result (0.623 vs. 0.690 and 0.628), while for the GPT endpoints TEXT+IMAGE is the most accurate condition on the KB. TEXT+IMAGE provides no consistent latency benefit and is the slowest condition on the longer corpus (Table 4).

Table 6: Pairwise disagreement rate (%) on the ≤**50-page subset**: share of questions where exactly one format is correct (i.e., the sum of text-only-correct and vision-only-correct questions). A rate of 10% in the TEXT/IMAGE column means that on 10% of questions one format answered correctly and the other did not. Coral ≥20%; mint ≤10%.

	Enterprise KB				MMLongBench-Doc			
	TEXT / IMAGE		IMAGE / TEXT+IMAGE		TEXT / IMAGE		IMAGE / TEXT+IMAGE	
Model	ORACLE	FULL	ORACLE	FULL	ORACLE	FULL	ORACLE	FULL
Gemini 3.1 Flash-Lite	24.7	22.2	10.7	10.4	22.8	23.2	12.3	13.5
Gemini 3.1 Pro	20.4	19.2	6.7	8.4	19.3	21.2	10.4	16.0
GPT-5.4	22.9	19.7	13.5	12.7	19.7	20.2	8.9	7.9
GPT-5.4-mini	24.6	22.6	14.0	14.2	23.4	20.6	9.6	9.5

Finding 1. Document length governs the *cost* of the trade-off, not its direction. Within the ≤50-page subset, image-bearing inputs (IMAGE or TEXT+IMAGE) lead on accuracy on both corpora, with no length bucket showing a text advantage; what grows with length is their latency, which rises steadily while text latency stays roughly flat.

The image advantage carries a cost and latency premium. Table 5 summarizes cost and latency across the four models and two corpora. Within each regime, TEXT is consistently the cheapest and fastest condition: under ORACLE, moving from TEXT to IMAGE raises cost 1.5× (0.61 to 0.91 $/100q) and latency from 3.3 to 4.9 s/q, and TEXT+IMAGE is more expensive still (1.29 $/100q).

The premium widens under FULL, where the full document is processed: IMAGE and TEXT+IMAGE cost 1.9× and 2.4× as much as TEXT, respectively, while their latency is roughly twice that of text (7.0 and 7.3 s/q vs. 3.7 s/q). Since IMAGE or TEXT+IMAGE also lead on accuracy (Section 4.1), no representation dominates on both axes. This trade-off motivates per-question routing, using image input only when it is likely to improve the answer.

4.2 Text and images answer different questions

TEXT and IMAGE give different correctness outcomes on approximately **19–25%** of questions across the reported corpus, model, and regime cells (Table 6). This disagreement decomposes into a text-only-correct slice and a larger vision-only-correct slice, reflecting vision's higher overall accuracy. On the short KB articles, ORACLE and FULL disagreement rates are nearly identical because the full document is already close to the evidence pages. On MMLongBench-Doc the two regimes diverge somewhat more, consistent with extra context amplifying representation differences. Because both exclusive-correct slices are present in every cell, an oracle that picks the correct format per question would beat either fixed policy.

Finding 2. Text and images answer different questions: 19–25% TEXT/IMAGE disagreement across the reported cells. Neither format subsumes the other; this establishes headroom for per-question selection.

5 Per-Question Representation Routing

Text and images each answer questions the other misses (Finding 2). This complementarity raises a concrete question: if a system could decide *per question* whether to send text or images, it could capture the accuracy of images where they help while keeping text's lower latency and cost elsewhere. We test whether this is achievable using only the question text, as a proof of concept that quantifies the potential benefit rather than a production-ready system.

Router design. We frame routing as binary classification. For each (question, model) pair under FULL, the label is $y = 1$ if IMAGE is correct and TEXT is incorrect, and $y = 0$ otherwise. When both representations are correct we default to TEXT because it is faster in aggregate; when both are incorrect we also assign TEXT, treating the example as a non-vision-win case. We exclude TEXT+IMAGE from the router: although often competitive, it adds a second input channel and closely tracks IMAGE in many settings (Table 3), so routing between TEXT and IMAGE targets the primary accuracy–latency trade-off.

Labels are obtained by running both representations on every question. This makes the router

Table 7: Held-out question-text router results under FULL, pooling all four models. The test partition is the 20% split described in the text. *Routing accuracy* = accuracy when every question uses the modality the strategy selects (cf. per-model accuracy in Table 3). *Implied med. latency* = median wall-clock seconds across the strategy's per-question routing decisions (cf. per-model latency in Table 4). Gains are computed from unrounded scores. **Bold** = best per column within each dataset.

Strategy	Routing acc.	Gain vs text	Lat.
Overall			
Always-text	63.6%	+0.0 pp	**2.5 s**
Always-vision	**67.7%**	+4.1 pp	4.4 s
TF-IDF router	66.2%	+2.6 pp	3.1 s
Enterprise KB			
Always-text	65.0%	+0.0 pp	**2.4 s**
Always-vision	**69.2%**	+4.2 pp	4.1 s
TF-IDF router	67.8%	+2.7 pp	2.9 s
MMLongBench-Doc			
Always-text	53.9%	+0.0 pp	**3.2 s**
Always-vision	**58.0%**	+4.1 pp	8.7 s
TF-IDF router	55.9%	+2.0 pp	5.6 s

a proof of concept: it measures how much a per-question policy *could* gain if it selected correctly, not the performance of a deployable classifier. In production, a similar router would be trained on a smaller labeled sample and applied to new questions. In our training set only 13.2% of examples are positive, so IMAGE uniquely beats TEXT on a small fraction of questions.

We represent each question with TF-IDF features and train a logistic regression classifier with balanced class weights, using an approximately 80/20 document-level split stratified by dataset and model. All questions from the same document stay in one partition, ensuring zero document leakage and a more reliable estimate of generalization to unseen documents.

All router numbers in Table 7 are computed on this 20% held-out test partition over the *full corpus* (all documents, image budget still capped at 50 pages) and pooled across the four models, so they are micro-averages over pooled questions rather than the per-model, ≤50-page macro-averages in Table 3; the two are not directly comparable. Always-vision is correspondingly lower on MMLongBench-Doc here, because the held-out set includes documents beyond the 50-page cap where image inputs are truncated (Appendix D).

Accuracy. On the document-disjoint split, always-vision remains the strongest fixed policy, reaching 69.2% on KB and 58.0% on MMLongBench-Doc (Table 7). The router does not beat it, but closes much of the gap over always-text, increasing accuracy by **2.7 pp** on KB (65.0 → 67.8%) and **2.0 pp** on MMLongBench-Doc (53.9 → 55.9%) while routing 26.8% of examples to vision. Overall, the router reaches 66.2%, gaining **2.6 pp** over always-text while remaining 1.5 pp below always-vision.

Latency. Since the router keeps most questions on the faster text path, median latency drops substantially relative to always-vision: from 4.4 s to 3.1 s overall, a savings of approximately **30%**, while routing only 26.8% of examples to vision.

Finding 3. On the held-out split with no document overlap, a lightweight TF-IDF router gains 2.02.7 pp over always-text while cutting median latency by 30% relative to always-vision. Because training labels require running both representations on every question, this is a proof-of-concept estimate of achievable headroom, not a deployable classifier.

6 Discussion

Document length as the organizing variable. Document length organizes the accuracy–latency trade-off, but not by changing which representation is most accurate. On the ≤50-page subset, IMAGE or TEXT+IMAGE leads on accuracy across all document lengths on both corpora; what scales with length is their cost and latency, which grow steadily while text stays roughly flat. The apparent accuracy crossover in favor of text on long documents is a consequence of the fixed 50-image cap truncating image inputs, and appears only in the full-corpus results (Appendix D); it is a property of the current deployment budget, not of the modality. Complementary errors provide a separate motivation for routing: text and images disagree on roughly a fifth to a quarter of questions in every cell, so neither representation subsumes the other. The current router, however, is evaluated only under FULL, and we treat it as a proof of concept rather than a tuned system.

Multimodal fusion is model-dependent, not additive. Sending both text and images does not uniformly help. The GPT endpoints gain approximately 5 pp over image-only input on the KB under ORACLE, whereas the Gemini endpoints show smaller or less consistent changes (Section 4.1).

These outcome differences do not reveal the models' internal fusion mechanisms, but they show that the preferred representation depends on the endpoint as well as the document setting.

Implications for RAG pipelines. The results motivate evaluating both text and image indices rather than assuming that one representation is uniformly best. The TF-IDF router is a proof of concept; a stronger evaluation would use realistic retrieval, and document features in addition to the question text.

7 Conclusion

We compared text, image, and combined document inputs under a common prompting and evaluation pipeline across four model endpoints, two corpora, and two context regimes. Image-bearing inputs outperform text on both corpora, with the largest gains on the short, visually dense Enterprise KB, while text remains cheaper and faster per question. The two representations also produce complementary correctness outcomes, with approximately 19–25% pairwise disagreement across the reported cells. On a held-out document-level split with no train–test leakage, a lightweight TF-IDF question router recovers 2.0–2.7 percentage points over always-text while reducing median latency by 30% relative to always-vision.

Limitations

First, ORACLE and FULL do not evaluate a realistic top-k retriever. They compare annotated evidence pages against full-document input, so the results bound two extremes and do not establish performance for intermediate retrieval quality.

Second, we report run-to-run variation but not question-level uncertainty. Our design fixes the question set and varies only prediction and judging runs, so the reported standard deviations capture endpoint and judge nondeterminism. Paired significance testing over the per-question correctness scores would sharpen these comparisons.

Third, the TEXT condition depends on our OCR pipeline. The pipeline outperforms Tesseract on standard document-OCR benchmarks (Appendix Table 8) but is not state of the art, and may extract text less faithfully than the frontier models themselves would from the same pages. Part of the text–image gap we report may therefore reflect extraction error rather than an intrinsic limitation of the text modality.

Ethics Statement

The Enterprise KB contains internal product-support documentation owned by the authors' organization and was accessed under existing internal data-use policies. Questions were created and reviewed by internal annotators as part of their regular work. All model endpoints, including the judge, were accessed under enterprise agreements with Vertex AI and Azure OpenAI, which exclude submitted prompts and outputs from provider model training. No data was retained beyond the terms of these agreements. The Enterprise KB and OCR pipeline cannot be released because they are proprietary. We therefore include MMLongBench-Doc to enable independent reproduction of the main text-versus-image comparison.

References

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, André F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K. Surikuchi, Ece Takmaz, and Alberto Testoni. 2025. LLMs instead of human judges? a large scale empirical study across 20 NLP evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–255. Association for Computational Linguistics.

David Berghaus, Armin Berger, Lars Hillebrand, Kostadin Cvejoski, and Rafet Sifa. 2025. Multimodal vision vs. text-based parsing: Benchmarking LLM strategies for invoice processing. In *2025 IEEE International Conference on Big Data*, pages 2519–2526. IEEE.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2024. Nougat: Neural optical understanding for academic documents. In *International Conference on Learning Representations*.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2024. M3DocRAG: Multimodal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma.

2025. PaddleOCR 3.0 technical report. *Preprint*, arXiv:2507.05595.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and Cheng-Lin Liu. 2025. LongDocURL: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1135–1159. Association for Computational Linguistics.

Kuicai Dong, Yujing Chang, Shijie Huang, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025. Benchmarking retrieval-augmented multimodal generation for document question answering. In *Advances in Neural Information Processing Systems*, volume 38.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations (ICLR)*. ArXiv:2407.01449.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091. Association for Computing Machinery.

Jaided AI. 2024. EasyOCR. `https://github.com/JaidedAI/EasyOCR`. Accessed: 2026.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. FUNSD: A dataset for form understanding in noisy scanned documents. *Preprint*, arXiv:1905.13538.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18893–18912. PMLR.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522. Association for Computational Linguistics.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. 2024a. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6492–6505, Miami, Florida, USA. Association for Computational Linguistics.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2024b. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems*, volume 37, pages 95963–96010.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2199–2208. IEEE.

Ahmed Nassar, Andres Marafioti, Matteo Omenetti, Maksym Lysak, Nikolaos Livathinos, Christoph Auer, Lucas Morin, Rafael Teixeira de Lima, Yusik Kim, A. Said Gurbuz, Michele Dolfi, Miquel Farré, and Peter W. J. Staar. 2025. SmolDocling: An ultra-compact vision-language model for end-to-end multi-modal document conversion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21972–21983. IEEE.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. 2019. CORD: A consolidated receipt dataset for post-OCR parsing. In *Workshop on Document Intelligence at NeurIPS 2019*.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, Seunghyun Yoon, Ryan A. Rossi, and Franck Dernoncourt. 2024. PDFTriage: Question answering over long, structured documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages

153–169. Association for Computational Linguistics.

Jiyuan Shen, Peiyue Yuan, Atin Ghosh, Yifan Mai, and Daniel Dahlmeier. 2026. OCR or not? rethinking document information extraction in the MLLMs era with real-world large-scale datasets. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 385–396. Association for Computational Linguistics.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition*, volume 2, pages 629–633. IEEE.

Stacey Svetlichnaya. 2020. DeepForm: Understand structured documents at scale.

Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Paweł Józiak, Rafał Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Ackaert, Ernest Valveny, Matthew B. Blaschko, Sien Moens, and Tomasz Stanisławek. 2023. Document understanding dataset and evaluation (DUDE). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19471–19483. IEEE.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *International Conference on Learning Representations (ICLR)*. ArXiv:2410.10594.

Qintong Zhang, Victor Shea-Jay Huang, Bin Wang, Junyuan Zhang, Zhengren Wang, Hao Liang, Shawn Wang, Matthieu Lin, Conghui He, and Wentao Zhang. 2024. Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. *arXiv preprint arXiv:2410.21169*.

Handong Zheng, Yumeng Li, Kaile Zhang, Liang Xin, Guangwei Zhao, Hao Liu, Jiayu Chen, Jie Lou, Qi Fu, Rui Yang, Shuo Jiang, Weijian Luo, Weijie Su, Weijun Zhang, Xingyu Zhu, Yabin Li, Yiwei Ma, Yu Chen, Yuqiu Ji, and 6 others. 2026. Multimodal OCR: Parse anything from documents. *Preprint*, arXiv:2603.13032.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

A OCR Text Extraction

We benchmark five OCR systems: **Tesseract** (v5.5.3) (Smith, 2007), a widely-used classical baseline; **PaddleOCR** (v3.7.0) (Cui et al., 2025) with PP-OCRv5 Latin models; **EasyOCR** (v1.7.2) (Jaided AI, 2024) combining CRAFT detection with LSTM/CTC recognition; and **dots.mocr** (Zheng et al., 2026), a 3B-parameter vision-language model that jointly performs layout detection and text recognition. Our **in-house engine** is a proprietary CNN-based detector and recognizer trained on public, private, and synthetic data. All systems use GPU inference with their Latin-script configurations.

We evaluate using End-to-End Bag-of-Words (EEBoW) recall, which measures textual completeness independent of spatial accuracy. Text is tokenized into whitespace-delimited words and represented as multisets; recall is the fraction of ground-truth words recovered.

Table 8: OCR results on three public benchmarks. End-to-End Bag-of-Words recall.

Model	CORD-1k	DeepForm	FUNSD
Tesseract	0.474	0.668	0.589
EasyOCR	0.511	0.855	0.428
PaddleOCR	0.801	0.956	0.746
dots.mocr	0.780	0.504	0.726
In-house	**0.881**	**0.970**	**0.815**

Our in-house engine leads across all three public benchmarks (CORD-1k (Park et al., 2019), DeepForm (Svetlichnaya, 2020), and FUNSD (Jaume et al., 2019)), with the largest margin on CORD-1k (0.881 vs. 0.474 for Tesseract) shown in Table 8. Notably, all benchmarked datasets (Appendix C) are inherently multimodal and require visual representations to answer questions. Even with stronger OCR, the gap between text-only and image-based approaches remains large, confirming that visual representations are fundamentally necessary. Using weaker open-source models would only widen this gap.

B Experimental Settings and Provenance

Input and inference. Images use **300-DPI PDF page renders**. Text uses **page text**, our proprietary OCR pipeline outperforming other open-source models (Table 8). All runs use the same QA prompt (Appendix F), no learned retriever, seed 30, a **3,000-token answer limit**, and **at most**

50 page images per request. Temperature is 0 for all four models.

Latency measurement. Provider-side automatic retries are disabled for all API calls. Reported latency is the wall-clock time of the single call that produced the answer, excluding any internal retry overhead the provider would otherwise add.

Judging. **GPT-5.2** judges answers across **three runs** (8,000-token limit, two parallel calls per batch) (Zheng et al., 2023; Liu et al., 2023; Bavaresco et al., 2025). Questions are batched in fixed document order.

Model Identifiers. Run identifiers are `gemini-3.1-flash-lite`, `gemini-3.1-pro-preview`, `gpt-5.4`, and `gpt-5.4-mini`. These are the exact strings passed at inference time.

C Dataset Details

MMLongBench-Doc. MMLongBench-Doc contains 135 PDFs with 6,522 pages, averaging 48.3 pages per document (median 28, range 9–468) (Ma et al., 2024b). We use the current 1,091-question release. Questions include 485 single-page (44.5%), 360 cross-page (33.0%), and 246 unanswerable (22.5%). On answerable questions, the evidence labels are multi-label: Pure-text 298 (35.3%), Table 218 (25.8%), Chart 178 (21.1%), Image 293 (34.7%), and Layout 119 (14.1%). Answer formats include Integer 34.2%, String 29.5%, Float 18.9%, and List 17.3%. Questions cite 1.88 evidence pages on average (9.08 distinct evidence pages per document).

Enterprise KB (internal). The enterprise KB contains 1,447 documents (8,296 pages; avg. 5.7 pages/doc) with 5,169 human-reviewed questions (avg. 3.6 questions/doc). Answer mode: visual 3,314 (64.1%), cross-modal 1,527 (29.5%), textual 328 (6.3%). Difficulty: easy 3,731 (72.2%), medium 1,425 (27.6%), hard 13 (0.3%). 93.6% of questions require visual or cross-modal interpretation. Questions cite 1.05 evidence pages on average (2.42 distinct evidence pages per document).

D Full-corpus results

This appendix reports the per-model results underlying the main text over *all* documents, with the image budget still capped at 50 pages. Documents longer than 50 pages are therefore truncated under IMAGE and TEXT+IMAGE, while TEXT sees the full document. We report the effect of the $\leq$50-page restriction on accuracy (Table 9) and latency (Table 10), the full length-bucket breakdown extending past the cap (Figure 3).

The restriction isolates the page-cap truncation effect. On the Enterprise KB, restricting to $\leq$50 pages changes every accuracy and latency cell by less than 0.001 (Tables 9, 10): only 6 of 5,169 questions come from longer documents, so the subset and the full corpus are effectively the same measurement. On MMLongBench-Doc, where 25% of documents exceed the cap, the restriction has a clear and localized effect. The accuracy gain from restricting concentrates entirely in IMAGE/FULL, where it reaches +0.041 to +0.058 across the four models, and is an order of magnitude smaller for TEXT/FULL and for every ORACLE cell. The latency change mirrors this: the large reductions (-1.0 to -1.6 s) fall on IMAGE and TEXT+IMAGE under FULL, while TEXT and all ORACLE cells barely move. Both patterns identify the same cause: only the image conditions on long documents are affected by the restriction, because only they are truncated by the page cap.

The text advantage appears only past the cap. Figure 3 extends the length-bucket view to the full corpus (shaded region marks buckets above the 50-page cap). Within the un-truncated range, image-bearing inputs lead at every length, matching the main text. The crossover in favor of TEXT emerges only in the truncated buckets: on MMLongBench-Doc, IMAGE accuracy falls to $\sim$42% at 100–150 pages and $\sim$39% beyond 150, while TEXT holds near 58%. This is the behavior that motivates reporting the $\leq$50-page subset in the main text: the crossover is a property of the fixed image budget, not of the modalities, and it disappears once every representation sees the complete page set.

E Error Category Analysis

Tables 11 and 12 break down errors by model, representation, and regime (tagged by judge; Appendix F). Categories are non-exclusive.

On the KB, `partial_correctness` is the most frequent reported category (44–57%), `extra_incorrect_information` follows (18–25%), and switching from ORACLE to FULL

Table 9: Effect of restricting to the ≤50-page subset. Accuracy is mean ± std across runs; $\Delta = \mathrm{acc}_{\leq 50} - \mathrm{acc}_{\mathrm{all}}$. **Bold** Δ marks $|\Delta| \geq 0.02$. Large positive Δ concentrates in IMAGE/FULL, reflecting vision-input truncation on documents longer than the page cap.

Model	Representation	Regime	$\mathrm{Acc}_{\leq 50}$	$\mathrm{Acc}_{\mathrm{all}}$	Δ
Enterprise KB					
GPT-5.4	TEXT	FULL	0.6634 ± 0.0059	0.6626 ± 0.0059	+0.0008
GPT-5.4	TEXT	ORACLE	0.6387 ± 0.0033	0.6382 ± 0.0033	+0.0005
GPT-5.4	IMAGE	FULL	0.6792 ± 0.0043	0.6788 ± 0.0043	+0.0004
GPT-5.4	IMAGE	ORACLE	0.6774 ± 0.0018	0.6769 ± 0.0018	+0.0005
GPT-5.4	TEXT+IMAGE	FULL	0.7344 ± 0.0051	0.7338 ± 0.0052	+0.0006
GPT-5.4	TEXT+IMAGE	ORACLE	0.7329 ± 0.0039	0.7324 ± 0.0039	+0.0005
GPT-5.4-mini	TEXT	FULL	0.6277 ± 0.0033	0.6269 ± 0.0033	+0.0007
GPT-5.4-mini	TEXT	ORACLE	0.6122 ± 0.0015	0.6116 ± 0.0015	+0.0006
GPT-5.4-mini	IMAGE	FULL	0.6720 ± 0.0027	0.6714 ± 0.0027	+0.0006
GPT-5.4-mini	IMAGE	ORACLE	0.6825 ± 0.0027	0.6821 ± 0.0028	+0.0005
GPT-5.4-mini	TEXT+IMAGE	FULL	0.7148 ± 0.0054	0.7142 ± 0.0053	+0.0006
GPT-5.4-mini	TEXT+IMAGE	ORACLE	0.7309 ± 0.0020	0.7304 ± 0.0020	+0.0005
Gemini 3.1 Pro	TEXT	FULL	0.6335 ± 0.0031	0.6330 ± 0.0031	+0.0005
Gemini 3.1 Pro	TEXT	ORACLE	0.6179 ± 0.0029	0.6174 ± 0.0028	+0.0004
Gemini 3.1 Pro	IMAGE	FULL	0.7280 ± 0.0016	0.7276 ± 0.0016	+0.0005
Gemini 3.1 Pro	IMAGE	ORACLE	0.7409 ± 0.0035	0.7406 ± 0.0035	+0.0003
Gemini 3.1 Pro	TEXT+IMAGE	FULL	0.7275 ± 0.0034	0.7271 ± 0.0035	+0.0005
Gemini 3.1 Pro	TEXT+IMAGE	ORACLE	0.7364 ± 0.0017	0.7361 ± 0.0017	+0.0003
Gemini 3.1 Flash-Lite	TEXT	FULL	0.5942 ± 0.0009	0.5938 ± 0.0008	+0.0004
Gemini 3.1 Flash-Lite	TEXT	ORACLE	0.5762 ± 0.0024	0.5757 ± 0.0024	+0.0005
Gemini 3.1 Flash-Lite	IMAGE	FULL	0.7112 ± 0.0025	0.7110 ± 0.0025	+0.0002
Gemini 3.1 Flash-Lite	IMAGE	ORACLE	0.7294 ± 0.0015	0.7292 ± 0.0015	+0.0003
Gemini 3.1 Flash-Lite	TEXT+IMAGE	FULL	0.7141 ± 0.0002	0.7138 ± 0.0002	+0.0004
Gemini 3.1 Flash-Lite	TEXT+IMAGE	ORACLE	0.7243 ± 0.0010	0.7241 ± 0.0010	+0.0003
MMLongBench-Doc					
GPT-5.4	TEXT	FULL	0.5990 ± 0.0093	0.5927 ± 0.0086	+0.0063
GPT-5.4	TEXT	ORACLE	0.5589 ± 0.0065	0.5615 ± 0.0077	−0.0026
GPT-5.4	IMAGE	FULL	0.6857 ± 0.0078	0.6348 ± 0.0082	**+0.0509**
GPT-5.4	IMAGE	ORACLE	0.6672 ± 0.0043	0.6585 ± 0.0039	+0.0087
GPT-5.4	TEXT+IMAGE	FULL	0.6830 ± 0.0080	0.6636 ± 0.0079	+0.0193
GPT-5.4	TEXT+IMAGE	ORACLE	0.6734 ± 0.0027	0.6643 ± 0.0027	+0.0091
GPT-5.4-mini	TEXT	FULL	0.5466 ± 0.0046	0.5430 ± 0.0038	+0.0036
GPT-5.4-mini	TEXT	ORACLE	0.5268 ± 0.0093	0.5340 ± 0.0097	−0.0072
GPT-5.4-mini	IMAGE	FULL	0.6333 ± 0.0038	0.5923 ± 0.0025	**+0.0410**
GPT-5.4-mini	IMAGE	ORACLE	0.6343 ± 0.0052	0.6279 ± 0.0025	+0.0065
GPT-5.4-mini	TEXT+IMAGE	FULL	0.6424 ± 0.0073	0.6236 ± 0.0040	+0.0187
GPT-5.4-mini	TEXT+IMAGE	ORACLE	0.6426 ± 0.0065	0.6354 ± 0.0054	+0.0072
Gemini 3.1 Pro	TEXT	FULL	0.6276 ± 0.0072	0.6053 ± 0.0084	**+0.0223**
Gemini 3.1 Pro	TEXT	ORACLE	0.6413 ± 0.0074	0.6320 ± 0.0066	+0.0093
Gemini 3.1 Pro	IMAGE	FULL	0.6902 ± 0.0040	0.6326 ± 0.0050	**+0.0576**
Gemini 3.1 Pro	IMAGE	ORACLE	0.7113 ± 0.0076	0.7039 ± 0.0063	+0.0073
Gemini 3.1 Pro	TEXT+IMAGE	FULL	0.6233 ± 0.0060	0.6077 ± 0.0042	+0.0156
Gemini 3.1 Pro	TEXT+IMAGE	ORACLE	0.6912 ± 0.0030	0.6849 ± 0.0013	+0.0064
Gemini 3.1 Flash-Lite	TEXT	FULL	0.5109 ± 0.0006	0.5108 ± 0.0004	+0.0000
Gemini 3.1 Flash-Lite	TEXT	ORACLE	0.5292 ± 0.0006	0.5228 ± 0.0004	+0.0065
Gemini 3.1 Flash-Lite	IMAGE	FULL	0.6145 ± 0.0093	0.5778 ± 0.0074	**+0.0367**
Gemini 3.1 Flash-Lite	IMAGE	ORACLE	0.6253 ± 0.0027	0.6205 ± 0.0033	+0.0048
Gemini 3.1 Flash-Lite	TEXT+IMAGE	FULL	0.6199 ± 0.0056	0.6101 ± 0.0011	+0.0097
Gemini 3.1 Flash-Lite	TEXT+IMAGE	ORACLE	0.6128 ± 0.0018	0.6101 ± 0.0017	+0.0026

barely changes the profile. On MMLongBench-Doc, `answered_when_unanswerable` is high (36–53%), consistent with the fact that 22.5% of questions are unanswerable by design; this category means the model *gave* an answer to an unanswerable question (over-answering), and its rate is largely stable across regimes rather than growing with the ORACLE→FULL shift. `numerical_imprecision` is also frequent (25–41%); Integer and Float are common answer formats in this benchmark.

Table 10: Effect of the $\leq$50-page restriction on latency (seconds/question, mean $\pm$ std); $\Delta = \text{lat}_{\leq 50} - \text{lat}_{\text{all}}$ (negative = subset is faster). **Bold** Δ marks $|\Delta| \geq 0.5$s. Large negative Δ concentrates in IMAGE/TEXT+IMAGE under FULL: the excluded long documents are the ones that inflate vision latency.

Model	Representation	Regime	$\text{Lat}_{\leq 50}$	Lat_{all}	Δ (s)
Enterprise KB					
GPT-5.4	TEXT	FULL	3.42 ± 0.18	3.42 ± 0.18	-0.003
GPT-5.4	TEXT	ORACLE	3.46 ± 0.29	3.46 ± 0.29	-0.001
GPT-5.4	IMAGE	FULL	5.43 ± 1.07	5.44 ± 1.08	-0.011
GPT-5.4	IMAGE	ORACLE	4.97 ± 0.42	4.98 ± 0.42	-0.009
GPT-5.4	TEXT+IMAGE	FULL	5.71 ± 0.43	5.73 ± 0.43	-0.013
GPT-5.4	TEXT+IMAGE	ORACLE	4.48 ± 0.42	4.49 ± 0.42	-0.004
GPT-5.4-mini	TEXT	FULL	1.78 ± 0.42	1.78 ± 0.42	-0.000
GPT-5.4-mini	TEXT	ORACLE	1.68 ± 0.20	1.68 ± 0.20	-0.001
GPT-5.4-mini	IMAGE	FULL	3.36 ± 0.13	3.37 ± 0.13	-0.011
GPT-5.4-mini	IMAGE	ORACLE	2.55 ± 0.15	2.55 ± 0.15	-0.002
GPT-5.4-mini	TEXT+IMAGE	FULL	3.26 ± 0.16	3.27 ± 0.16	-0.012
GPT-5.4-mini	TEXT+IMAGE	ORACLE	2.42 ± 0.15	2.43 ± 0.15	-0.000
Gemini 3.1 Pro	TEXT	FULL	6.21 ± 0.25	6.21 ± 0.25	-0.001
Gemini 3.1 Pro	TEXT	ORACLE	5.57 ± 0.25	5.57 ± 0.25	-0.001
Gemini 3.1 Pro	IMAGE	FULL	5.68 ± 0.15	5.68 ± 0.15	-0.005
Gemini 3.1 Pro	IMAGE	ORACLE	5.69 ± 0.36	5.69 ± 0.36	$+0.000$
Gemini 3.1 Pro	TEXT+IMAGE	FULL	5.81 ± 0.22	5.82 ± 0.22	-0.009
Gemini 3.1 Pro	TEXT+IMAGE	ORACLE	5.36 ± 0.37	5.36 ± 0.37	$+0.000$
Gemini 3.1 Flash-Lite	TEXT	FULL	1.40 ± 0.01	1.40 ± 0.01	-0.001
Gemini 3.1 Flash-Lite	TEXT	ORACLE	1.29 ± 0.00	1.29 ± 0.00	-0.000
Gemini 3.1 Flash-Lite	IMAGE	FULL	3.26 ± 0.60	3.27 ± 0.60	-0.008
Gemini 3.1 Flash-Lite	IMAGE	ORACLE	2.42 ± 0.38	2.42 ± 0.38	$+0.000$
Gemini 3.1 Flash-Lite	TEXT+IMAGE	FULL	3.20 ± 0.32	3.21 ± 0.32	-0.007
Gemini 3.1 Flash-Lite	TEXT+IMAGE	ORACLE	2.52 ± 0.31	2.52 ± 0.31	$+0.000$
MMLongBench-Doc					
GPT-5.4	TEXT	FULL	4.78 ± 0.24	4.99 ± 0.27	-0.207
GPT-5.4	TEXT	ORACLE	4.21 ± 0.32	4.15 ± 0.30	$+0.067$
GPT-5.4	IMAGE	FULL	13.37 ± 0.61	14.75 ± 0.65	-1.386
GPT-5.4	IMAGE	ORACLE	8.03 ± 0.75	7.99 ± 0.74	$+0.044$
GPT-5.4	TEXT+IMAGE	FULL	13.36 ± 0.51	14.92 ± 0.52	-1.555
GPT-5.4	TEXT+IMAGE	ORACLE	7.83 ± 0.67	7.83 ± 0.65	-0.004
GPT-5.4-mini	TEXT	FULL	2.22 ± 0.27	2.32 ± 0.30	-0.103
GPT-5.4-mini	TEXT	ORACLE	2.00 ± 0.23	1.99 ± 0.23	$+0.013$
GPT-5.4-mini	IMAGE	FULL	10.20 ± 0.31	11.53 ± 0.33	-1.332
GPT-5.4-mini	IMAGE	ORACLE	5.05 ± 0.36	5.07 ± 0.37	-0.020
GPT-5.4-mini	TEXT+IMAGE	FULL	10.09 ± 0.36	11.50 ± 0.36	-1.405
GPT-5.4-mini	TEXT+IMAGE	ORACLE	5.03 ± 0.31	5.04 ± 0.31	-0.014
Gemini 3.1 Pro	TEXT	FULL	7.57 ± 0.31	7.82 ± 0.48	-0.246
Gemini 3.1 Pro	TEXT	ORACLE	6.10 ± 0.22	6.12 ± 0.44	-0.015
Gemini 3.1 Pro	IMAGE	FULL	8.40 ± 0.50	9.45 ± 1.17	-1.046
Gemini 3.1 Pro	IMAGE	ORACLE	6.88 ± 0.69	7.06 ± 0.65	-0.180
Gemini 3.1 Pro	TEXT+IMAGE	FULL	10.12 ± 2.31	11.61 ± 2.86	-1.492
Gemini 3.1 Pro	TEXT+IMAGE	ORACLE	6.80 ± 0.72	7.01 ± 0.81	-0.211
Gemini 3.1 Flash-Lite	TEXT	FULL	2.10 ± 0.06	2.31 ± 0.08	-0.209
Gemini 3.1 Flash-Lite	TEXT	ORACLE	1.72 ± 0.09	1.74 ± 0.08	-0.022
Gemini 3.1 Flash-Lite	IMAGE	FULL	5.88 ± 0.52	6.58 ± 0.57	-0.694
Gemini 3.1 Flash-Lite	IMAGE	ORACLE	3.59 ± 0.44	3.74 ± 0.41	-0.152
Gemini 3.1 Flash-Lite	TEXT+IMAGE	FULL	6.93 ± 0.28	8.05 ± 0.20	-1.119
Gemini 3.1 Flash-Lite	TEXT+IMAGE	ORACLE	4.06 ± 0.16	4.19 ± 0.13	-0.124

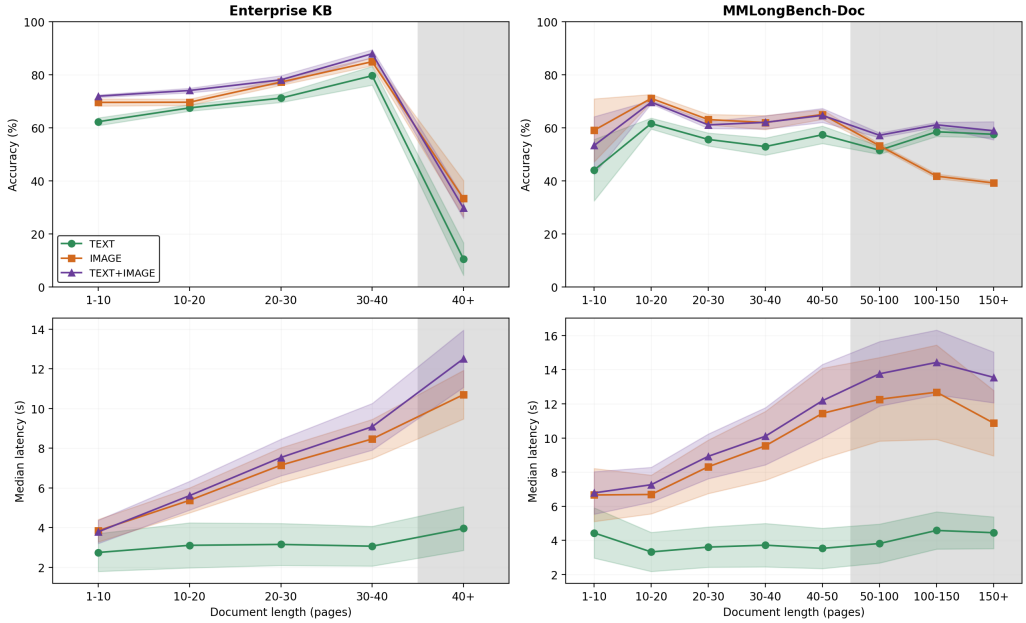


Figure 3: Full-corpus accuracy (top) and median latency (bottom) versus document-length bucket under FULL retrieval, aggregated across all four models. The shaded region marks buckets above the 50-page image cap, where IMAGE and TEXT+IMAGE inputs are truncated while TEXT sees the full document. Within the un-truncated range, image-bearing inputs lead on accuracy at every length; the crossover in favor of TEXT appears only in the truncated buckets, and image latency falls there as truncation caps the number of page images processed. Bands show $\pm$ std across models.

Table 11: Non-exclusive error-category shares (% of incorrect answers) by model, representation, and regime on Enterprise KB. Left block: ORACLE (gold evidence pages); right block: FULL (full document). Columns need not sum to 100%.

	ORACLE			FULL		
Model	TEXT	IMAGE	TEXT+IMAGE	TEXT	IMAGE	TEXT+IMAGE
partial_correctness						
Gemini 3.1 Flash-Lite	44.9	53.0	52.9	46.0	52.0	52.1
Gemini 3.1 Pro	50.5	56.8	56.0	49.5	52.0	52.0
GPT-5.4	51.5	49.1	54.7	52.4	45.5	52.0
GPT-5.4-mini	49.4	48.1	53.7	51.0	49.3	51.5
extra_incorrect_information						
Gemini 3.1 Flash-Lite	18.7	18.6	19.8	21.2	23.3	25.5
Gemini 3.1 Pro	18.1	17.7	20.4	20.4	22.9	22.4
GPT-5.4	20.8	19.9	22.4	22.6	21.1	23.7
GPT-5.4-mini	21.8	18.5	21.6	23.8	20.8	24.4
contradiction						
Gemini 3.1 Flash-Lite	19.1	18.3	17.1	19.2	19.6	21.4
Gemini 3.1 Pro	16.3	16.9	15.8	19.9	19.2	19.9
GPT-5.4	21.9	23.1	19.0	20.6	23.0	20.2
GPT-5.4-mini	21.7	21.3	19.5	21.9	21.1	20.0
answered_when_unanswerable						
Gemini 3.1 Flash-Lite	20.7	10.4	10.8	17.7	6.6	4.9
Gemini 3.1 Pro	14.8	8.3	9.8	15.4	10.6	9.3
GPT-5.4	9.3	8.3	6.1	9.6	9.1	7.2
GPT-5.4-mini	9.6	7.3	4.7	9.3	6.6	4.9
numerical_imprecision						
Gemini 3.1 Flash-Lite	6.0	4.6	5.3	5.9	5.6	5.9
Gemini 3.1 Pro	4.8	4.0	4.1	4.4	4.0	4.2
GPT-5.4	6.9	10.3	6.8	6.2	10.3	5.7
GPT-5.4-mini	8.4	11.7	7.9	7.1	10.6	7.5

Table 12: Non-exclusive error-category shares (% of incorrect answers) by model, representation, and regime on MMLongBench-Doc. Left block: ORACLE (gold evidence pages); right block: FULL (full document). Columns need not sum to 100%.

	ORACLE			FULL		
Model	TEXT	IMAGE	TEXT+IMAGE	TEXT	IMAGE	TEXT+IMAGE
partial_correctness						
Gemini 3.1 Flash-Lite	9.6	11.7	11.7	10.1	10.8	10.6
Gemini 3.1 Pro	9.3	11.7	10.5	10.8	8.5	10.6
GPT-5.4	12.3	10.4	10.2	8.7	7.6	9.1
GPT-5.4-mini	9.0	9.2	8.8	11.0	8.8	9.3
extra_incorrect_information						
Gemini 3.1 Flash-Lite	4.0	5.5	6.5	6.0	5.7	8.2
Gemini 3.1 Pro	4.4	5.7	4.4	8.2	6.7	7.8
GPT-5.4	6.1	5.3	5.5	5.8	5.1	5.5
GPT-5.4-mini	5.8	4.0	5.9	5.4	7.0	6.6
contradiction						
Gemini 3.1 Flash-Lite	6.7	8.6	8.4	14.0	8.9	8.9
Gemini 3.1 Pro	8.8	9.2	7.0	8.7	7.5	7.1
GPT-5.4	9.0	8.6	9.1	11.4	9.8	10.5
GPT-5.4-mini	10.8	8.2	9.5	12.2	11.1	9.5
answered_when_unanswerable						
Gemini 3.1 Flash-Lite	51.6	40.4	39.0	36.3	38.2	36.5
Gemini 3.1 Pro	52.9	42.9	45.3	52.7	44.3	42.4
GPT-5.4	43.6	46.8	44.6	40.5	47.2	39.7
GPT-5.4-mini	44.0	45.5	42.8	37.1	41.4	37.7
numerical_imprecision						
Gemini 3.1 Flash-Lite	30.5	35.6	37.4	37.6	39.3	40.7
Gemini 3.1 Pro	26.5	32.1	33.6	24.9	35.8	34.8
GPT-5.4	32.4	31.6	33.1	36.7	32.1	36.1
GPT-5.4-mini	33.7	33.9	34.7	38.6	35.3	40.6

F Prompt Templates

F.1 QA Prompt

All conditions use the prompt below; IMAGE and TEXT+IMAGE additionally append the rendered page images after the text context block.

QA Prompt Template

```
You are an expert question answering
assistant. Your task is to provide a concise
and accurate answer to the user's question,
strictly based on the provided context.

Instructions:
1. Read the question and the provided context
   carefully.
2. In the "explanation" field, think
   step-by-step to determine how the context
   answers the question.
3. Formulate an answer that directly addresses
   the question.
4. Your answer must be derived solely from the
   information present in the context. Do not
   use outside knowledge or invent facts.
5. If the context does not contain enough
   information, state "Not Answerable".
6. Keep the answer as short as possible:
   a single value, phrase, or list of items.

Output Format:
{
  "explanation": "<reasoning>",
  "answer": "<concise answer>"
}

Question: {{question}}
Context:
```

F.2 LLM-as-Judge Prompt

The judge receives batches of question-answer pairs through the `{{qa_pairs}}` placeholder, following reference-guided evaluation protocols for open-ended model outputs (Zheng et al., 2023; Liu et al., 2023; Bavaresco et al., 2025).

LLM-as-Judge Prompt Template

```
You are an expert evaluator assessing answers
in a Question and Answer system. For each
question-answer pair you receive:
1. A unique question ID
2. The question text
3. The predicted answer
4. The ground truth answer
Determine:
1. Whether the predicted answer is
   semantically equivalent to the ground truth.
2. All applicable error or warning categories.
Evaluation Instructions:
- Evaluate based on semantic meaning, not
  exact wording.
- Be flexible with phrasing, terminology,
  units, abbreviations, and formatting.
- Mark incorrect only if: (1) core meaning
  differs, (2) contains factually incorrect
  information, (3) contradicts the ground
  truth, or (4) omits key information.
- For numeric values: accept more-precise
  expansions of rounded ground truths; penalize
  rounding that loses accuracy.
- For lists: every item must match; order does
  not matter.
- For multiple-choice: accept the option
  letter, text, or both.
ERROR CATEGORIES (incorrect answers only):
  partial_correctness,
  extra_incorrect_information, contradiction,
```

```
  overly_vague, incorrect_yes_no,
  answered_when_unanswerable,
  numerical_imprecision, minor_spelling_error
WARNING CATEGORIES (correct answers only):
  ground_truth_rounding,
  ground_truth_spelling_error,
  redundant_information, extra_information,
  missing_information, other
Output Format:
{ "evaluations": [
    { "question_id": "<id>",
      "is_correct": true/false,
      "explanation": "<pred> vs <truth>",
      "errors": [{"class": "...",
                  "explanation": "..."}],
      "warnings": [{"class": "...",
                    "explanation": "..."}]
    }, ...
] }
```

AdaptiveDocRAG: Efficient Document Understanding with Multimodal Large Language Models

Surya Selvam
School of ECE, Purdue University
selvams@purdue.edu

Anand Raghunathan
School of ECE, Purdue University
raghunathan@purdue.edu

Abstract

Document understanding requires answering questions over long, multi-page documents that interleave text, figures, and tables. Recent methods adopt the Document Retrieval-Augmented Generation (DocRAG) paradigm, which indexes each document and feeds the top-K most relevant pages to a Multimodal Large Language Model (MLLM) per query. Because each page can be represented as extracted text or a rendered image, DocRAG systems expose two key knobs: an *indexing modality* choice when the documents are ingested and a *context modality* at query time. Existing systems either fix both knobs to the visual modality or use both modalities throughout, often incurring higher indexing and query costs than needed for a given document or query. We propose **AdaptiveDocRAG**, which adapts both knobs as well as the size of the visual context to optimize the trade-off between accuracy, query response time, and indexing time. **AdaptiveDocRAG** achieves this through three lightweight modules. First, *document-adaptive indexing modality selection* uses a text-density heuristic to route text-dominant documents to cheaper textual indexing. Second, *query-adaptive context modality selection* uses a learned per-query router to select textual or visual context. Third, *query-aware visual context compression* reuses the retriever's per-patch query–page similarity to crop relevant sub-page regions and fuse them with top-ranked text chunks. On MMLongBench-Doc and NQ-Text with Qwen3-VL-8B, AdaptiveDocRAG matches or improves upon the accuracy of visual-only DocRAG (up to $+\mathbf{1.1}$ points) while using $\sim\mathbf{50}\%$ of the MLLM tokens, resulting in $\mathbf{1.30}$–$\mathbf{1.79}\times$ end-to-end speedup and $\mathbf{1.34}\times$ faster indexing.

1 Introduction

Document question answering (DocQA) supports enterprise search, literature review, and the analysis of legal and financial documents (Mathew et al., 2021; Tito et al., 2023). Real-world documents are long and multimodal, interleaving text, figures, and tables in complex layouts that span hundreds of pages. Answering questions over them requires reasoning jointly across modalities and pages, a substantially harder challenge than single-page or text-only QA (Ma et al., 2024; Deng et al., 2025).

Recent Multimodal Large Language Models (MLLMs) are well suited to DocQA because they process and reason jointly over text and images (Anthropic, 2026; OpenAI, 2026; Google DeepMind, 2026). However, directly feeding entire documents to MLLMs does not scale in practice: context windows are limited, and both inference latency and token cost scale linearly with the number of pages. DocRAG (Cho et al., 2024; Yu et al., 2025) addresses this by indexing each document once with an embedding model and feeding only the top-K pages most relevant pages to a query to the MLLM, thereby reducing MLLM usage cost and latency.

A DocRAG system has three stages with distinct costs. *Indexing* runs once per document and embeds each page into a vector database; its cost scales with the number of pages and dominates time-to-first-response on a freshly ingested document. *Retrieval* runs once per query and scores each indexed page against the query to identify the top-K most relevant pages; while its cost depends on the database size, it is a very small part of query response latency in our workloads. *Question answering* runs once per query and feeds the top-K pages to the MLLM; its cost scales with K and dominates query response latency and token usage.

Each page in DocRAG is dual-modal: it can be represented as extracted text or as a rendered page image. This exposes two design knobs that together define the DocRAG modality design space. The *indexing modality* determines whether pages are embedded as text or as images, affecting both indexing latency and retrieval quality. The *context*

The Proceedings of Workshop on Document Intelligence and Understanding, pages 63–72
October 24, 2026 ©2026 Association for Computational Linguistics

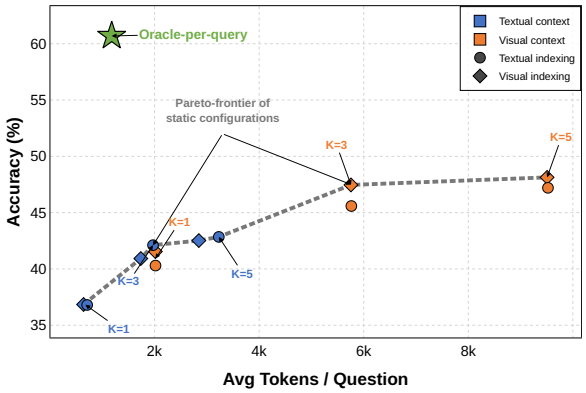


Figure 1: Accuracy vs. tokens-per-question on the LongDocURL dataset across the 12 static DocRAG configurations (2 indexing modalities $\times$ 2 context modalities $\times$ $K \in \{1, 3, 5\}$). Marker shape denotes the indexing modality, and marker color denotes the context modality. The star denotes the per-query oracle.

modality determines whether the retrieved pages are passed to the MLLM as text or as images, affecting token count, latency, and answer accuracy.

Existing DocRAG systems fix these knobs statically and adopt one of two strategies. Visual-only systems (Faysse et al., 2025; Cho et al., 2024; Yu et al., 2025) use the visual modality for both indexing and context, treating page images as a superset of text. Dual-modality systems (Han et al., 2025; Jain et al., 2025; Suri et al., 2025; Wang et al., 2025) index pages in both modalities and consume both at query time. Both strategies incur a cost that is often unnecessary: page images produce at least $3\times$ more tokens than their text, increasing both encoder compute at indexing and MLLM tokens at query time—yet many documents and queries (e.g., named-entity lookups, factoids) are answerable from text alone. A per-query oracle that selects the best static configuration for each question lies well above both fixed strategies (Figure 1), revealing substantial room for adaptation along these two axes. A third axis remains unaddressed: even when visual context is needed, prior work feeds the entire page image, though query relevance is typically concentrated in a small region. Together, these observations expose three inefficiencies in DocRAG systems.

We propose **AdaptiveDocRAG**, a query- and document-adaptive DocRAG framework that addresses these inefficiencies through the following techniques:

- *Document-adaptive indexing modality selection*, which uses text density to select between textual or visual indexing at ingestion time, reducing indexing time and time-to-first-response for text-heavy documents.
- *Query-adaptive context modality selection*, which uses a learned decision tree to select textual or visual context for each query, reducing MLLM token cost and response time.
- *Query-aware visual context compression*, which crops retrieved pages to query-relevant regions and fuses them with top-ranked text chunks, reducing MLLM token usage while retaining relevant visual evidence.

On MMLongBench-Doc and NQ-Text with Qwen3-VL-8B, AdaptiveDocRAG matches or improves upon visual-only DocRAG accuracy by up to 1.1 points while using approximately half as many MLLM tokens, running 1.30–1.79$\times$ faster end-to-end, and indexing documents 1.34$\times$ faster.

2 Related Work

DocRAG pipelines. Document retrieval has evolved from OCR-based pipelines with dense or late-interaction text retrievers (Khattab and Zaharia, 2020; Chen et al., 2024a) to vision-based retrievers that embed rendered page images directly. ColPali (Faysse et al., 2025) introduced patch-level late interaction over a vision-language backbone, and ColQwen (ColQwen2) extends it to Qwen2-VL. Multimodal DocRAG systems pair these retrievers with MLLMs for question answering. M3DocRAG (Cho et al., 2024) and VisRAG (Yu et al., 2025) use the visual modality for both indexing and context, treating page images as a superset of OCR text. MDocAgent (Han et al., 2025) and VisDoM (Suri et al., 2025) use parallel text and image RAG pipelines and consume both modalities at query time, increasing MLLM token use regardless of query type. SimpleDoc (Jain et al., 2025) and ViDoRAG (Wang et al., 2025) use iterative retrieval, whereas MoLoRAG (Wu et al., 2025) uses page-graph traversal and MAB-DQA (Xiang et al., 2026) decomposes queries via a multi-armed bandit; all require multiple MLLM calls per query. Adaptive RAG methods instead decide whether to retrieve or how many passages to retrieve (Jeong et al., 2024; Asai et al., 2024; Taguchi et al., 2025), but operate on a single modality. AdaptiveDocRAG differs by selecting the indexing modality per document, the context modality per query, and the extent of visual context per query.

Indexing	Time/doc (s)	Accuracy (%)
Textual	6.8	45.6
Visual	18.9	47.5

Table 1: Cost–accuracy trade-off of the indexing modality on LongDocURL.

Context	Accuracy (%)	Tokens	Resp. (s)
Textual	40.9	1,737	0.18
Visual	47.5	5,755	0.57

Table 2: Cost–accuracy trade-off of the context modality on LongDocURL.

Visual context compression. General-purpose token-pruning methods operate within the MLLM. FastV (Chen et al., 2024b) discards low-attention vision tokens in early decoder layers, LLaVA-PruMerge (Shang et al., 2025) compresses tokens in the vision encoder using CLS attention, and SparseVLM (Zhang et al., 2025) uses cross-modal attention for text-conditioned selection. RegionRAG (Li et al., 2026), the most closely related method to our third module, fine-tunes a vision-language retriever with region-level supervision, groups high-scoring patches into semantic regions, and crops them for MLLM input. AdaptiveDocRAG instead reuses an off-the-shelf retriever's per-patch query–page similarity without additional training, combines cropped regions with top-ranked text chunks, and operates outside the MLLM.

3 Motivation

DocRAG indexes each page as extracted text or a rendered image, retrieves the top-K pages relevant to a query, and supplies their content to an MLLM for question answering. The indexing and context modalities can each be textual or visual, yielding four choices in addition to the retrieval depth K. We characterize the cost–accuracy trade-offs of these choices to identify opportunities for improvement. To isolate modality effects, we use ColQwen2 for both modalities and Qwen3-VL-8B as the MLLM on LongDocURL.

Observation 1: Indexing modality presents a speed–accuracy trade-off. On LongDocURL, textual indexing is $2.8\times$ faster per document than visual indexing but sacrifices 1.9 percentage points in accuracy at $K = 3$ (Table 1). This trade-off suggests that textual indexing is suitable for text-heavy documents, whereas visual indexing remains important when layout and visual content provide retrieval evidence.

Visual context	Accuracy (%)	Tokens
Full page	47.5	5,755
Top 8 sub-pages	35.6	3,877
Top 4 sub-pages	27.8	1,997

Table 3: Effect of reducing visual context to query-relevant sub-pages on accuracy and MLLM input tokens.

Observation 2: Context modality presents a token–accuracy trade-off. Visual context uses $3.3\times$ as many MLLM input tokens as textual context but improves accuracy by 6.6 percentage points at $K = 3$ (Table 2). This loss exceeds the 1.9-point loss from textual indexing, indicating that visual evidence matters more for question answering than for retrieval. Nevertheless, text-answerable queries can use textual context, whereas queries involving charts, tables, and diagrams benefit from page images. These results motivate document- and query-adaptive modality selection.

Observation 3: Fixed visual-context budgets present a token–accuracy trade-off. Reducing visual context from full pages to the top 8 and top 4 query-relevant sub-pages lowers MLLM input tokens by 33% and 65%, respectively, but decreases accuracy by 11.9 and 19.7 percentage points at $K = 3$ (Table 3). This aggregate loss is unevenly distributed across queries: among questions answered correctly with full-page visual context, 70% and 54% remain correct with only the top 8 and top 4 query-relevant sub-pages, respectively. These results expose a cost–accuracy trade-off under fixed sub-page budgets, motivating query-aware visual context compression that selects the amount of visual evidence supplied to the MLLM for each query.

Observation 4: No static configuration matches a per-query oracle. No static configuration addresses all three trade-offs. Figure 1 compares the 12 static configurations (2 indexing modalities $\times$ 2 context modalities $\times$ $K \in \{1, 3, 5\}$) with a per-query oracle. The oracle achieves 60.7% accuracy with 1,180 tokens, whereas the best static configuration (visual context, $K = 5$) reaches 48.1% with 9,507 tokens. This gap motivates a framework that is adaptive to documents and queries.

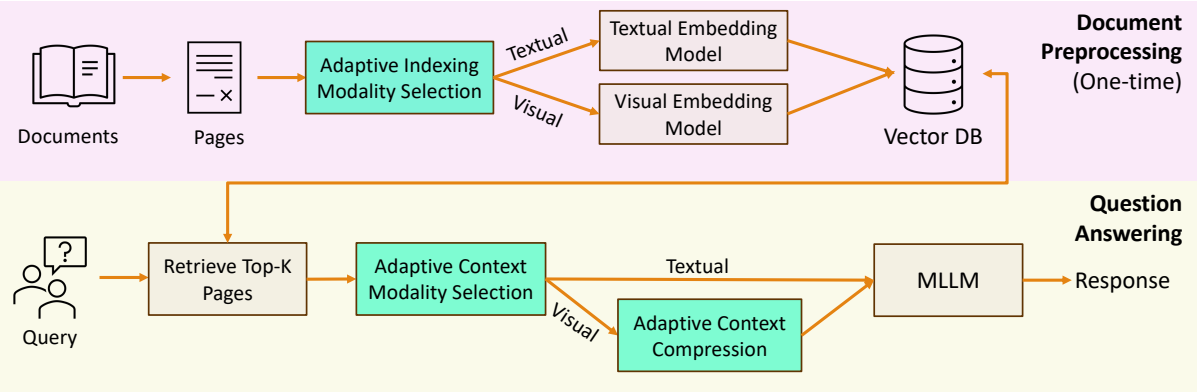


Figure 2: Overview of AdaptiveDocRAG with three adaptive modules (green boxes) that choose the indexing modality (per document), context modality (per query), and visual context compression.

4 AdaptiveDocRAG Methodology

AdaptiveDocRAG is a query- and document-adaptive DocRAG framework with three lightweight modules that address the trade-offs identified above (Figure 2). We first select an indexing modality for each document using text density. For each query, we then use a learned decision tree to select textual or visual context from features of the query and retrieved pages. When visual context is selected, we further compress it by retaining query-relevant sub-page regions and top-ranked text chunks. None of the modules requires fine-tuning of the retriever or MLLM, and each adds negligible computation relative to MLLM inference.

4.1 Document-Adaptive Indexing Modality Selection

We select an indexing modality once per document using its text density. We use text density as a proxy for the extent to which a document's retrieval evidence is available in extracted text. For text-heavy documents, textual embeddings are likely to capture much of the retrievable content while avoiding the cost of image encoding. Conversely, documents with little extracted text often contain visual elements, such as tables, figures, and diagrams, whose information is better preserved by visual indexing. During preprocessing, we divide each document D into pages and extract text from each page using PDF text extraction when applicable and using OCR otherwise. We define the text density d_{TD} as the average character count per page in D. Given a threshold τ, we assign the indexing modality as:

$$\mathcal{M}(D) = \begin{cases} \textsc{Textual} & d_{\text{TD}} \geq \tau \\ \textsc{Visual} & d_{\text{TD}} < \tau \end{cases} \quad (1)$$

4.2 Query-Adaptive Context Modality Selection

For each query, we select extracted text when it is likely sufficient to answer the question and rendered page images when answering requires layout or other visual evidence. Because both representations are retained at ingestion, this selection is independent of the indexing modality. We use a lightweight decision tree to make this choice from features of the query and its top-K retrieved pages. The routing decision asks whether the retrieved evidence is sufficient in textual form or whether answering the query requires visual evidence. To make this prediction, we use three complementary features:

Query intent. Some queries can be answered by locating textual facts, whereas others require visual reasoning over charts, tables, or layouts. We construct two reference sets of example questions: one containing questions answerable from text alone, such as named-entity lookups, and another containing questions requiring visual reasoning, such as chart interpretation. For each query seen at runtime, we compute its cosine similarity to the two sets; these similarity scores characterize whether the query is more likely to require textual or visual evidence.

Retrieved-page content. For each retrieved page, we compute text density from its extracted text and image entropy from its rendered image. Text density quantifies the amount of textual evidence available in the page, whereas image entropy characterizes the complexity of its visual content. These measurements, together with the document's selected indexing modality $\mathcal{M}(D)$, characterize

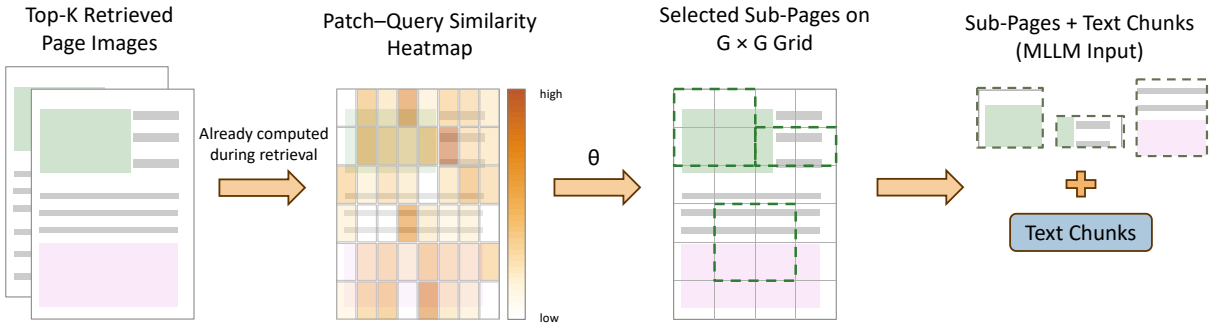


Figure 3: Query-Aware Visual Context Compression: thresholding normalized sub-page scores aggregated from patch–query similarities selects sub-pages, which are fused with top-T text chunks to form the MLLM input.

whether the retrieved evidence is better represented as text or images.

Query–page alignment. For each retrieved page, we compute the cosine similarity between the query and the page's extracted text. This similarity indicates whether the page contains query-relevant textual evidence.

Let $\mathcal{P}_q = \{p_1, \ldots, p_K\}$ denote the top-K retrieved pages for query q, and $\phi(q, \mathcal{P}_q)$ the feature vector defined above. We use the decision tree T to assign the context modality:

$$\mathcal{C}(q) = T(\phi(q, \mathcal{P}_q)) \in \{\textsc{Textual}, \textsc{Visual}\}. \tag{2}$$

Router training. We construct oracle modality labels on a held-out training set by running both textual and visual contexts for each query with the same retriever. We label each query with the higher-scoring modality, assigning ties to the lower-cost textual modality. We train the decision tree on these labels and the routing signals above. The decision tree provides interpretable routing decisions at negligible cost relative to MLLM inference.

4.3 Query-Aware Visual Context Compression

Visual context is token-intensive because each retrieved page is represented by many image tokens. However, a query often requires evidence from only a small part of a page, such as a chart, table, or figure. We therefore identify query-relevant sub-page regions and pair them with relevant text chunks, rather than passing all retrieved page images in full to the MLLM (Figure 3).

Patch–query similarity. To identify these regions, we divide each retrieved page into a $G \times G$ grid of sub-pages. We use the query–patch similarities produced by the late-interaction retriever to assign each sub-page c on page p a score $s(p, c)$ by summing its patch similarities.

Sub-page selection. To select the most relevant regions, we normalize sub-page scores jointly across all $K \times G^2$ candidates:

$$\hat{s}(p, c) = \frac{s(p, c) - s_{\min}}{s_{\max} - s_{\min}}, \tag{3}$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum scores across the candidate sub-pages. We retain sub-page (p, c) when $\hat{s}(p, c) \geq \theta$ and pass its cropped image to the MLLM; applying the same threshold to every candidate makes the number of retained sub-pages query-adaptive rather than fixed. For a query about a localized element, only a few sub-pages score above θ and are retained, whereas queries requiring evidence across the page retain more sub-pages.

Text chunk fusion. After selecting the visual regions, we add complementary text because the crops may omit textual evidence needed to interpret the visual content, including evidence located elsewhere in the retrieved pages. We therefore rank fixed-length chunks from the retrieved pages by their similarity to the query and append the top-T chunks to the selected crops.

5 Experimental Results

5.1 Experimental Setup

Datasets. We evaluate AdaptiveDocRAG on a 2,204-question test set combining MMLongBench-Doc (Ma et al., 2024) (1,091 questions across 135 chart/table/figure-heavy documents) and NQ-Text from the UDA benchmark suite (Hui et al.,

Method	Indexing Time (s/doc)	$K=1$		$K=3$		$K=5$	
		Accuracy (%)	Tokens	Accuracy (%)	Tokens	Accuracy (%)	Tokens
Textual RAG	2.10	46.2	792	53.1	2,256	55.7	3,656
Visual RAG	5.83	46.4	1,929	54.7	5,613	56.6	9,194
AdaptiveDocRAG ($\theta=0.5$)	**4.34**	45.4	963	54.5	2,724	**57.4**	**4,360**
AdaptiveDocRAG ($\theta=0.3$)	**4.34**	45.7	996	**54.7**	**2,909**	**57.7**	**4,721**
MDocAgent	7.93	53.1	7,169	58.4	17,663	61.4	27,688
SimpleDoc	124.9	60.0	13,820	63.8	15,561	65.4	17,152
Oracle	7.93	60.5	823	67.3	2,416	68.6	4,005

Table 4: AdaptiveDocRAG Compared with baselines on MMLongBench-Doc and NQ-Text with Qwen3-VL-8B.

2024) (1,113 questions across 274 text-heavy documents). We use LongDocURL (Deng et al., 2025) (2,325 questions across 396 documents) for the cost–accuracy analysis in Section 3 and for hyperparameter tuning.

Models. We use ColQwen2 (ColQwen2; Faysse et al., 2025) as the retriever for both textual and visual indexing, so comparisons isolate the effect of modality rather than retriever architecture. All experiments use Qwen3-VL-8B-Instruct (Qwen Team, 2025) as the MLLM and BGE-M3 (Chen et al., 2024a) for context-routing features and top-T text-chunk selection.

Implementation details. We render page images at 144 DPI and extract embedded text using either PyMuPDF (Artifex Software, 2024), or using Tesseract OCR (Smith, 2007) when embedded text is unavailable. The MLLM is served with vLLM (Kwon et al., 2023) on a single A100 80GB GPU. We use $\tau = 2{,}500$ chars/page for indexing modality routing, grid size $G=2$, sub-page similarity threshold $\theta=0.3$, and $T=10$ text chunks for visual context compression.

Metrics. Accuracy is measured by ANLS on MMLongBench-Doc and by an LLM-as-judge score on NQ-Text, using Qwen3-VL-8B-Instruct as the judge and reporting scores on a percentage scale. Cost is measured by indexing time per document and MLLM tokens per question.

5.2 Main Results

We compare AdaptiveDocRAG with Textual RAG, Visual RAG, a per-query Oracle, MDocAgent (Han et al., 2025), and SimpleDoc (Jain et al., 2025). Textual RAG fixes both indexing and context to text, while Visual RAG fixes both to page images. Oracle selects, for each query and K, the best of the four static (indexing, context) choices. For MDocAgent and SimpleDoc, we use ColQwen2 and Qwen3-VL-8B-Instruct to match AdaptiveDocRAG. Table 4 reports average accuracy and total MLLM tokens (context + response) per question at $K \in \{1, 3, 5\}$. Table 8 reports the effect of replacing the textual retriever at fixed $K = 5$. AdaptiveDocRAG matches Visual RAG's accuracy at $K = 3$ using 52% of its tokens, and Pareto-dominates it at $K=5$ (+1.1 points at 51% of its tokens). The more aggressive $\theta=0.5$ configuration provides a lower-token operating point: at $K=5$, it exceeds Visual RAG by 0.8 points while using 47% of its tokens. At $K=1$, AdaptiveDocRAG trails Visual RAG by 0.7 points, where a single routing or compression error cannot be offset by other retrieved pages. AdaptiveDocRAG remains within 1.21× of Oracle's token cost at every K. At $K \in \{3, 5\}$, MDocAgent and SimpleDoc obtain higher accuracy with approximately 4–6× more tokens.

At ingestion, AdaptiveDocRAG routes 38.9% of documents to textual indexing, yielding a 1.34× indexing speedup over Visual RAG. Oracle and MDocAgent require both text and visual indices, raising indexing time to 7.93s/doc. SimpleDoc requires substantially more indexing time because it summarizes each page with an MLLM, raising indexing time to 124.9s/doc.

At query time, AdaptiveDocRAG selects textual context for 75.4% of queries. At $K=3$, routing reduces MLLM tokens to 56% of Visual RAG before compression. Compression further reduces this cost to 52%, while the complete pipeline matches Visual RAG's 54.7% accuracy. This compression path is available for visually indexed documents with visual context, for which patch–query similarity scores are available.

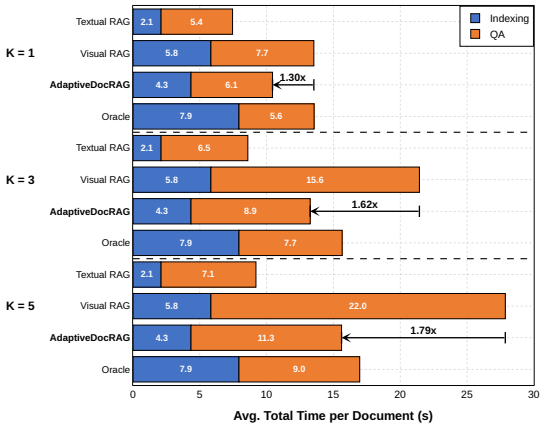


Figure 4: End-to-end wall time per document for Textual RAG, Visual RAG, Oracle, and AdaptiveDocRAG at $K \in \{1, 3, 5\}$.

Strategy	Accuracy (%)	Indexing Time (s/doc)
Textual Indexing	51.8	2.10
Visual Indexing	54.7	5.83
Adaptive Indexing	53.4	4.34

Table 5: Indexing modality ablation at $K = 3$ with context modality held at visual.

Figure 4 includes all adaptive-module overheads: routing features over the query and retrieved pages, tree inference, sub-page selection, and text-chunk ranking— totaling tens of milliseconds per query. AdaptiveDocRAG is $1.30\times$ / $1.62\times$ / $1.79\times$ faster than Visual RAG at $K \in \{1, 3, 5\}$. MDocAgent and SimpleDoc are omitted from the chart for legibility; at $K = 3$, they are $4.5\times$ and $17.0\times$ slower than AdaptiveDocRAG, respectively.

5.3 Ablation Studies

We ablate the three adaptive modules and evaluate sensitivity to the textual retriever.

For indexing, we hold context modality fixed to visual at $K = 3$ and vary only the indexing modality (Table 5). Adaptive indexing achieves 53.4% accuracy, within 1.3 points of visual indexing, while reducing indexing time from 5.83s/doc to 4.34s/doc. This reduction comes from routing 39% of documents to textual indexing; the remaining gap occurs when textual retrieval misses pages found by visual retrieval.

With adaptive indexing fixed, query-adaptive context routing provides the largest token reduction (Table 6). It improves accuracy from 53.4% to 55.2% relative to adaptive indexing with visual

Strategy	Accuracy (%)	Tokens
Adaptive Indexing + Textual Context	52.6	2,135
Adaptive Indexing + Visual Context	53.4	5,614
Adaptive Indexing + Adaptive Context	55.2	3,119
AdaptiveDocRAG	54.7	2,909
Oracle (per-query)	67.3	2,416

Table 6: Context modality and compression ablation at $K = 3$ on top of adaptive indexing.

Strategy	Accuracy (%)	Tokens
$\theta = 0, T = 0$ (Full Visual Page)	46.1	8,560
$\theta = 0.1, T = 10$	46.0	8,271
$\theta = 0.3, T = 0$ (sub-pages only)	40.7	5,724
$\boldsymbol{\theta = 0.3, T = 10}$	**46.1**	**6,552**
$\theta = 0.5, T = 10$	44.6	4,651
$\theta = 0.7, T = 10$	39.8	3,006

Table 7: Visual context compression ablation at $K = 5$ on queries routed to visual context with visual indexing ($n = 419$).

context and reduces tokens from $5{,}614$ to $3{,}119$ per query. Compression further reduces tokens to $2{,}909$ per query with a 0.5-point accuracy decrease, so the full pipeline remains 1.3 points above adaptive indexing with visual context while using 52% of its tokens.

We evaluate compression on the 419 queries routed to visual context from visually indexed documents at $K = 5$ (Table 7). At the default setting ($\theta = 0.3,\ T = 10$), compression matches full-page accuracy while using 77% of the uncompressed token budget. Sub-page selection without text fusion loses 5.4 accuracy points, whereas adding $T = 10$ text chunks recovers the full-page accuracy. Raising θ to 0.5 reduces tokens to 54% of the uncompressed budget, with a 1.5-point accuracy decrease.

The main comparison and module ablations use ColQwen2 for both textual and visual retrieval, which keeps the retriever architecture fixed when evaluating modality decisions. Table 8 tests whether AdaptiveDocRAG is sensitive to the textual retriever. We replace only the textual retriever with BM25 or BGE-M3, while keeping ColQwen2 for visual retrieval and using the same routing model, thresholds, and compression settings. BGE-M3 matches ColQwen2 within 0.1 accuracy points and slightly reduces both indexing time and token use. BM25 reduces Textual RAG indexing time from 2.10s/doc to 0.08s/doc, but the corresponding AdaptiveDocRAG variant loses 1.9 accuracy points, indicating that lower ingestion cost comes

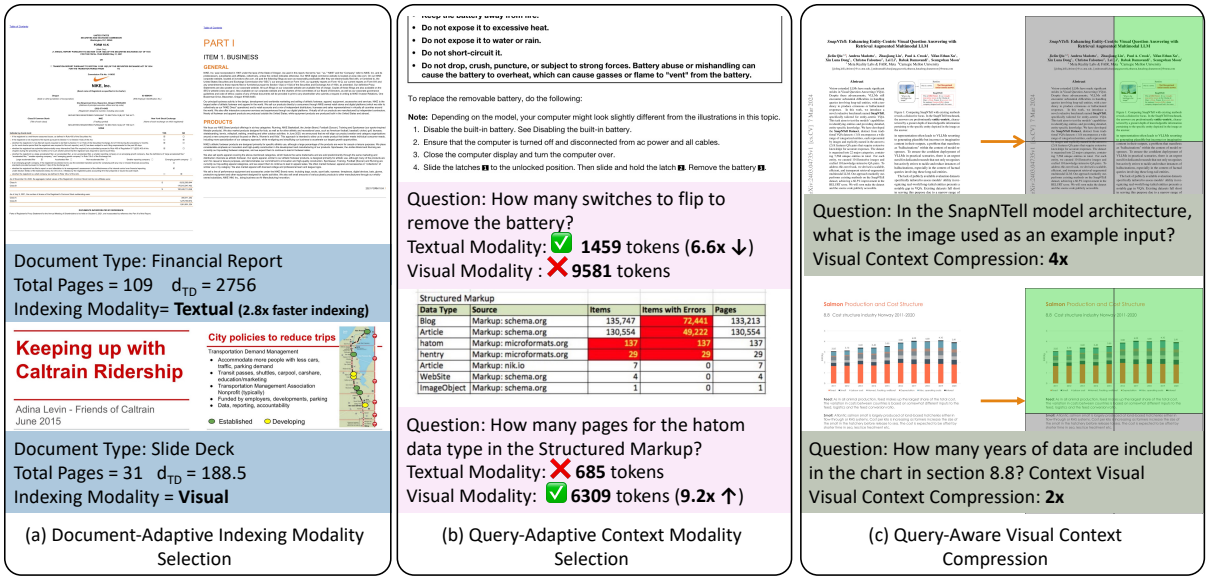


Figure 5: Qualitative examples of AdaptiveDocRAG's three techniques: (a) document-adaptive indexing modality selection, (b) query-adaptive context modality selection, and (c) query-aware visual context compression.

Retrievers	Index (s/doc)	Acc. (%)	Tokens
Textual RAG			
BM25	0.08	52.7	3,580
BGE-M3	1.82	55.7	3,280
ColQwen2	2.10	55.7	3,656
Visual RAG			
ColQwen2	5.83	56.6	9,194
AdaptiveDocRAG			
BM25 + ColQwen2	3.59	55.8	4,621
BGE-M3 + ColQwen2	4.27	57.6	4,472
ColQwen2 + ColQwen2	4.34	57.7	4,721

Table 8: Textual retriever ablation at $K = 5$; AdaptiveDocRAG rows list textual + visual retrievers.

with weaker retrieval accuracy.

5.4 Qualitative Analysis

Figure 5 illustrates representative decisions from the three adaptive modules. Document-level text density routes a text-heavy SEC form 10-K to textual indexing, which is 2.8× faster, and a sparse slide deck to visual indexing to preserve its layout (Fig. 5a). Query-level routing selects textual context for a procedural question, answering it correctly with 6.6× fewer tokens, but selects visual context for a table lookup that requires row–column structure (Fig. 5b). For visual-context queries, sub-page selection reduces visual context by 4× when relevant evidence is localized and by 2× when it spans a larger portion of the page (Fig. 5c).

6 Conclusion

Static DocRAG pipelines incur avoidable costs by fixing the indexing modality, context modality, and extent of visual context independently of the document and query. We introduced AdaptiveDocRAG, which adapts these three design knobs with lightweight modules: a text-density rule selects textual or visual indexing, a learned decision tree selects textual or visual context for each query, and patch–query similarity selects query-relevant sub-page regions that are paired with top-ranked text chunks. In our evaluations, AdaptiveDocRAG matches or improves upon the accuracy of visual-only DocRAG while substantially speeding up indexing and query response.

Limitations

Our evaluation is limited to MMLongBench-Doc and NQ-Text, a single MLLM (Qwen3-VL-8B), and thresholds tuned on LongDocURL. The extent to which the routing and compression decisions transfer across models, languages, and document domains remains to be evaluated.

Ethics Statement

By selecting the indexing modality per document and the context modality per query, AdaptiveDocRAG reduces MLLM token use and end-to-end latency. These reductions may lower the compute and deployment costs of multimodal document QA. Our evaluation uses publicly available research

benchmarks: MMLongBench-Doc, NQ-Text, and LongDocURL. Deployment over sensitive corpora may create privacy risks, and users may over-rely on automated answers. Appropriate data governance and human oversight are especially important in high-stakes settings.

Acknowledgment: This work was supported in part by CoCoSys, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and in part by the National Science Foundation under Award 2318101. The authors acknowledge Murugan Sankaradass for initial discussions regarding challenges in DocRAG.

References

Anthropic. 2026. Introducing Claude Opus 4.7. Anthropic news post. Released April 16, 2026.

Artifex Software. 2024. PyMuPDF: Python bindings for MuPDF. Software.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024b. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Computer Vision – ECCV 2024*, pages 19–35. Springer.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2024. M3DocRAG: Multi-modal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*.

ColQwen2. 2024. ColQwen2. `https://huggingface.co/vidore/colqwen2-v1.0`.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and Cheng-Lin Liu. 2025. LongDocURL: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1135–1159. Association for Computational Linguistics.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Google DeepMind. 2026. Gemini 3.1 Pro: A smarter model for your most complex tasks. The Keyword (Google blog). Released February 19, 2026.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. MDocAgent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964*.

Yulong Hui, Yao Lu, and Huanchen Zhang. 2024. UDA: A benchmark suite for retrieval augmented generation in real-world document analysis. In *Advances in Neural Information Processing Systems 37 (NeurIPS), Datasets and Benchmarks Track*.

Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. 2025. SimpleDoc: Multi-modal document understanding with dual-cue page retrieval and iterative refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 28410–28427. Association for Computational Linguistics.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 7036–7050. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 39–48.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pages 611–626. ACM.

Yinglu Li, Zhiying Lu, Zhihang Liu, Yiwei Sun, Chuanbin Liu, and Hongtao Xie. 2026. RegionRAG: Region-level retrieval-augmented generation for visual document understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pages 6662–6670.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2024. MMLongBench-Doc: Benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems 37 (NeurIPS), Datasets and Benchmarks Track*.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

OpenAI. 2026. Introducing GPT-5.5. OpenAI announcement. Released April 23, 2026.

Qwen Team. 2025. Qwen3-VL technical report. *Preprint*, arXiv:2511.21631.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2025. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22857–22867.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 629–633. IEEE.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A. Rossi, and Dinesh Manocha. 2025. VisDoM: Multi-document QA with visually rich elements using multimodal retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 6088–6109. Association for Computational Linguistics.

Chihiro Taguchi, Seiji Maekawa, and Nikita Bhutani. 2025. Efficient context selection for long-context QA: No tuning, no iteration, just adaptive-k. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 20105–20130. Association for Computational Linguistics.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multipage DocVQA. *Pattern Recognition*, 144:109834.

Qiuchen Wang, Ruixue Ding, Zehui Chen, Weiqi Wu, Shihang Wang, Pengjun Xie, and Feng Zhao. 2025. ViDoRAG: Visual document retrieval-augmented generation via dynamic iterative reasoning agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9113–9134, Suzhou, China. Association for Computational Linguistics.

Xixi Wu, Yanchao Tan, Nan Hou, Ruiyang Zhang, and Hong Cheng. 2025. MoLoRAG: Bootstrapping document understanding via multi-modal logic-aware retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14024–14045. Association for Computational Linguistics.

Yixin Xiang, Yunshan Ma, Xiaoyu Du, Yibing Chen, Yanxin Zhang, and Jinhui Tang. 2026. MAB-DQA: Addressing query aspect importance in document question answering with multi-armed bandits. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22973–22992, San Diego, California, United States. Association for Computational Linguistics.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. 2025. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 74840–74857. PMLR.

Most of a Document-VQA Score Is Answer Formatting

Srinivas Harish
New York University
srinivas.harish@nyu.edu

Abstract

DocVQA and InfographicVQA are scored with average normalized Levenshtein similarity (ANLS), and a DocVQA number is read as a measure of how well a model reads a page, but the `lmms-eval` and VLMEvalKit harnesses that compute it append a one-line terse-answer suffix to every question. ANLS divides edit distance by the length of the longer string, so a correct answer inside a sentence more than twice its length falls below the 0.5 threshold and scores zero, and removing only the suffix moved Qwen2.5-VL-3B's DocVQA ANLS from 94.2 to 7.5. On that model ANLS given a short answer (within two tokens of the longest gold) stayed between 93.9 and 95.7 across three suffixes whose scores spanned 94.2 to 65.3, so the reported score followed the share of short answers more than the share that contained the gold. Two suffixes that `lmms-eval` defines for the same task differed by 28.8 points, and under one of them 79% of the answers ANLS scored zero contained the gold answer, all 305 of which a judge with negated and two-candidate controls accepted as the committed answer. On a 317-question subset, four hosted models lost between 34.4 and 89.5 points without the suffix (Gemini 3.8 Flash fell from 92.0 to 2.5), and with the suffix in place, deleting Markdown emphasis markers raised Claude Opus 5.5 by 10.9 points. On Qwen2.5-VL-3B a length-gated span scorer moved 13.3 points across the five suffixes where ANLS moved 86.6, and it scored a submission of the page's whole OCR at 1.3, where best-subsequence exact match and the learned PEDANTS classifier gave 93.0 and 98.6. On ChartQA the suffix was worth 28.4 points of relaxed accuracy to Qwen2.5-VL-3B. We release the scorer and recommend reporting the prompt suffix and output post-processing with every document-VQA score.

1 Introduction

DocVQA (Mathew et al., 2021) is scored with ANLS, and a score of 94.2 is taken to mean that a model extracts the requested value from a scanned page almost every time, which is how the Qwen2.5-VL and Qwen3-VL technical reports use it (Bai et al., 2025b,a). Such numbers are computed with evaluation harnesses, the libraries that run a benchmark end to end, which append to every question a suffix that the reports' results tables do not show. When we removed only that suffix, Qwen2.5-VL-3B's DocVQA ANLS moved from 94.2 to 7.5, although 86.9% of the answers that ANLS then scored zero still contained the gold answer. In `lmms-eval` (Zhang et al., 2025) a DocVQA question receives `"\nAnswer the question using a single word or phrase."` by default and `" Answer:"` when the model name passed to the harness is `qwen_vl`; VLMEvalKit (Duan et al., 2024) appends the first string to every DocVQA question, and the Qwen3-VL report gives the same string as its DocVQA prompt in an appendix. Which suffix a model saw therefore depends on the harness and on how the model is registered in it (files and commits in Appendix B).

Without such a suffix, a correct answer from Qwen2.5-VL-3B or -7B was usually scored zero. ANLS takes the normalized Levenshtein similarity between the prediction and the nearest gold answer and zeroes anything below 0.5, and it normalizes by the length of the longer of the two strings, so a prediction that places the right answer inside a sentence is penalized in proportion to the sentence: *The date mentioned in the letter is 1/8/93.* against the gold *1/8/93* scores 0.14 and is set to zero. The suffix keeps predictions within the length ANLS tolerates, and its effect is included in every DocVQA score these harnesses report.

We measured that effect by changing the suffix and nothing else, on Qwen2.5-VL-3B, Qwen2.5-

The Proceedings of Workshop on Document Intelligence and Understanding, pages 73–80
October 24, 2026 ©2026 Association for Computational Linguistics

VL-7B and SmolVLM run locally and on GPT-6 Sol, Gemini 3.8 Flash, Claude Opus 5.5 and Qwen3.8-Max through an API, and scored the stored generations with ANLS, containment and the length-gated span scorer, and Qwen2.5-VL-3B's with all seven scorers of Table 5.

1. On Qwen2.5-VL-3B, removing the suffix cost 86.6 ANLS points and 13.3 points of length-gated span score on the same generations (the span scorer looks for the gold inside predictions at most m tokens longer than it, §3), and the two suffixes `lmms-eval` defines for DocVQA differed by 28.8 points on this model (§4).
2. Without a suffix 86.9% and 87.8% of the Qwen2.5-VL models' ANLS zeros contained the gold, and a judge checked on negated and two-candidate controls found that at least 97.7% of the contained zeros in every condition we audited committed to the gold as the answer (§4.1). ANLS given a short answer (within two tokens of the longest gold) stayed between 93.9 and 95.7 across three suffixes, so the reported score followed the share of short answers (§4.3).
3. The hosted models lost between 34.4 and 89.5 ANLS points without the suffix, and deleting Markdown emphasis markers raised Claude Opus 5.5 by 10.9 points with it (§4.2); a trailing full stop cost SmolVLM 52.5 points of exact match (§6).
4. We release a length-gated span scorer whose gate we set against a page-dump attack (the page's whole OCR submitted as the prediction); it scored that attack at 1.3, where best-subsequence exact match and PEDANTS (Li et al., 2024) scored it at 93.0 and 98.6 (§5).

2 ANLS and Whitespace Handling

For a gold set G and a prediction p, let $s = \max_{g \in G} \left(1 - d(g,p)/\max(|g|,|p|)\right)$ with d the Levenshtein distance; ANLS is s if $s \geq \tau = 0.5$ and 0 otherwise, averaged over questions and reported on a 0 to 100 scale. The denominator grows with the prediction, so a verbose answer is penalized for its length whether or not it is right.

We transcribed the scorer from the reference implementation of the ICDAR Robust Reading Competition (Biten et al., 2019a) and from `lmms-eval`, whose scoring functions are identical. Both compute the edit distance on lowercased, whitespace-normalized strings and the denominator on the raw strings, so leading or trailing whitespace enlarges the denominator without touching the numerator and can only raise a score, by an amount that depends on the model. Qwen2.5-VL-3B emitted almost no stray whitespace and gained 0.0002 points from this, while SmolVLM prefixed a space to every prediction and gained 1.85.

3 Experimental Setup

Data. We used the public 1,200-question DocVQA sample distributed with its OCR tokens (`nielsr/docvqa_1200_examples`), which the attack in §5 needs. Under the default suffix we measured 94.2 for Qwen2.5-VL-3B and 96.2 for the 7B, within 0.5 points of the 93.9 and 95.7 reported on the test split (Bai et al., 2025b). InfographicVQA validation (Mathew et al., 2022) (2,801 questions) is a second document corpus under the same metric and ChartQA test (Masry et al., 2022) (2,500 questions) a third benchmark with a different one.

Models. Run locally: Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct (Bai et al., 2025b), and SmolVLM-2B (`HuggingFaceTB/SmolVLM-Instruct`, Marafioti et al., 2025), an Idefics3-architecture model from a different family. Hosted: GPT-6 Sol, Gemini 3.8 Flash, Claude Opus 5.5 and Qwen3.8-Max, queried through the OpenRouter API on 26 September 2026 (Appendix F). The hosted models were compared on the 317 questions among the first 427 of the sample for which all eight runs (four models, with and without the suffix) returned a response; on these questions Qwen2.5-VL-3B scored 93.0 and 6.0 with and without the suffix, against 94.2 and 7.5 on all 1,200.

Decoding. Local models decoded greedily at batch size one (Appendix G), with a cap of 32 new tokens for terse suffixes, as in the `lmms-eval` configuration, and 64 where the prompt invites prose. Greedy decoding with a smaller cap produces a prefix of the same output, so we cut every generation made with the 64-token cap to its first 32 tokens and rescored; Qwen2.5-VL-3B's no-suffix ANLS moved from 7.5 to 7.6 and the 7B's from 3.6 to 4.0, and restricting to generations that stopped before the cap gave 7.8 and 4.0 (Appendix C). Hosted models ran at temperature 0 with the same 1,024-

Model	Suffix	tok	ANLS	span	EM	n_0	cg
Qwen2.5-VL-3B	default	3	94.2	95.5	86.0	53	18.9
Qwen2.5-VL-3B	`" Answer:"`	4	65.3	92.6	58.1	386	79.0
Qwen2.5-VL-3B	“briefly”	3	87.6	89.9	30.6	57	31.6
Qwen2.5-VL-3B	none	11	7.5	82.2	0.9	1057	86.9
Qwen2.5-VL-3B	“sentence”	3	83.6	94.1	62.2	161	71.4
Qwen2.5-VL-7B	default	2	96.2	96.7	90.5	35	8.6
Qwen2.5-VL-7B	none	13	3.6	73.0	0.0	1126	87.8
Qwen2.5-VL-7B	“sentence”	10	15.4	89.5	6.2	953	90.2
SmolVLM-2B	default	2	68.4	67.4	2.8	245	2.0
SmolVLM-2B	none	3	54.9	79.3	21.7	458	48.3

Table 1: DocVQA, $n = 1{,}200$ questions per row; only the suffix differs within a block. *tok* is the median prediction length in tokens, *span* the length-gated span scorer at m=20, *EM* exact match, n_0 the number of predictions ANLS scores exactly zero and *cg* the percentage of those that contain a gold answer.

token limit with and without the suffix, which no generation reached. Generations are stored verbatim, every number in the paper is recomputed from them, and differences are computed from unrounded values.

Scoring. All scorers lowercase and collapse whitespace first. *Exact match* compares the whole prediction with each gold. *Containment* splits both strings on spaces, strips the characters `.,;:!?"'()[]` from both ends of every token, and asks whether the token sequence of any gold occurs contiguously in the prediction's; *best-subsequence exact match* is containment used as a score. The *span scorer* applies when the prediction has at most m more tokens than the longest gold: it takes the best normalized Levenshtein similarity between any gold and any contiguous run of 1 to $|g_{\max}| + 2$ prediction tokens, $|g_{\max}|$ being the token length of the longest gold, and thresholds it at 0.5, and longer predictions fall back to whole-string ANLS with edit distance and length both computed on the normalized strings. *Token F1* is the bag-of-words F1 of Rajpurkar et al. (2016) against the best gold, and PEDANTS (Li et al., 2024) is a learned answer-equivalence classifier that takes the question, the gold and the prediction. For hosted models we delete Markdown emphasis markers (*, **, __ and backticks) before containment and the span scorer, and report ANLS on the raw output, as a harness computes it.

4 Results

Across the five suffixes Qwen2.5-VL-3B's ANLS ranged over 86.6 points and its span score over 13.3 (Table 1). Under a paired bootstrap over the same 1,200 items (5,000 replicates, 95% percentile intervals throughout), removing the suffix cost the 3B 86.6 ANLS points $[84.9, 88.2]$ and 13.3 points $[11.7, 15.1]$ of span score, so 73.3 of the ANLS points it lost came from answers the span scorer still credited, answers that contained the gold but did not give it as a bare string.

On Qwen2.5-VL-7B the same change cost 92.6 ANLS points $[91.2, 93.8]$. The 7B wrote longer no-suffix answers than the 3B, so it lost more, and the two models swapped places: with the suffix the 7B led, 96.2 to 94.2, and without it the 3B led, 7.5 to 3.6. On InfographicVQA validation the suffix was worth 71.9 ANLS points $[70.3, 73.4]$ to the 3B, from 72.5 to 0.7, and 60.3% of its zero-scored no-suffix answers contained the gold against 3.9% under the suffix.

Asked to *“Answer in a complete sentence.”*, an instruction a deployed document assistant might receive, the 7B complied at a median of 10 tokens and scored 15.4 ANLS with a span score of 89.5, while the 3B stayed at three tokens and scored 83.6, 68.2 points above the 7B. *“Answer briefly.”* left the 3B at a median of three tokens and 87.6 ANLS, 6.6 below the default. The `" Answer:"` suffix is the one `lmms-eval` applies to DocVQA for models passed to it under the name `qwen_vl`, and applied to Qwen2.5-VL-3B it cost 28.8 ANLS points against the default.

4.1 Gold Answers Inside ANLS Zeros

Containment is a property of two strings and does not depend on ANLS or on our scorer, so it checks the ANLS zeros without any other metric (Table 1, columns n_0 and *cg*). Under the default suffix 18.9% of the 3B's zeros contained the gold, so a zero there usually meant a wrong answer; under `" Answer:"` it was 79.0% and with no suffix 86.9%. For the 7B and SmolVLM the figures were 8.6% and 2.0% under the default and 87.8% and 48.3% without a suffix. Restricted to golds of at least two tokens the 3B's figures under `" Answer:"` and with no suffix were 79.2% and 86.3%, and a raw-substring definition agreed to within 2 percentage points everywhere (Appendix D). One of the 1,200 questions contains its own gold answer, and excluding it changes no *cg* value by more than 0.2 percentage points.

A prediction can contain the gold without giving it as the answer, as in *it is not 1/8/93, it is 1/9/93*. We gave each of the 2,434 zero-scored predictions

that contained a gold, with its question and gold, to Qwen3.8-27B (4-bit weights, run locally, temperature 0) and asked whether the response commits to the reference as its answer. On 600 control responses built from 200 of the 3B's no-suffix questions the judge accepted 100.0% of bare gold answers and 0.0% of negated (*The answer is not X; it is Y.*) and two-candidate responses. On the real predictions it accepted 100.0% of the 305 under `" Answer:"` and 99.0% without a suffix for the 3B, 99.6% for the 7B and 97.7% for SmolVLM, so contradictory or multi-candidate outputs were at most 2.3% of the contained zeros in any of these conditions (prompt in Appendix E).

4.2 Hosted Models

All four hosted models lost ANLS without the suffix, by 34.4 to 89.5 points, a factor of 2.6 between models whose suffix scores were within 17 points of each other (Table 2). Gemini 3.8 Flash fell from 92.0 to 2.5 while its span score fell by 18.7 to 77.0 and 92% of its zeros contained the gold, and GPT-6 Sol, whose median answer grew only from 3 to 4 tokens without the suffix, lost 34.4. The ranking changed without the suffix, GPT-6 Sol going from third of the four to first and Gemini 3.8 Flash from second to last. The span score fell by 4.5 to 18.7 points for the hosted models, and their zeros contained the gold in 64% to 94% of cases.

The hosted models also lost points to Markdown, which appeared in fewer than 1% of the locally run models' answers. Without the suffix every Gemini 3.8 Flash and Claude Opus 5.5 answer and 56% of Qwen3.8-Max answers contained Markdown bold, as in `**1/8/93**`, and ANLS counts each asterisk as an edit. Under the default suffix Claude Opus 5.5 still bolded 35% of its answers and scored 78.1; deleting the emphasis markers raised that to 89.0, a gain of 10.9 points [8.5, 13.5], which moved it from last of the four to 0.1 points below GPT-6 Sol's unstripped 89.2. That 78.1 was computed with the `lmms-eval` default suffix and ANLS on the raw output, so this model's DocVQA number depends on whether the evaluation strips Markdown, which neither harness configuration nor the model reports cited here state for DocVQA.

4.3 ANLS Decomposed by Answer Length

We call a prediction *short* if it is within two tokens of the longest gold, so that the prediction is at most two tokens longer than a gold. Under the default suffix ANLS and the span scorer nearly agreed (94.2 against 95.5 for the 3B), because 98.9% of predictions under that suffix were short, and the agreement disappeared under the other suffix the same harness defines (65.3 against 92.6 under `" Answer:"`).

ANLS given a short answer (Table 3) stayed between 93.9 and 95.7 across three suffixes whose scores spanned 94.2 to 65.3, and ANLS given a long answer was 6.6 to 9.6. The long answers therefore contributed 0.1, 1.2 and 2.3 points of the reported ANLS under those three suffixes, and on this model the reported number was the share of short answers times a factor between 93.9 and 95.7, plus at most 2.3 points from long answers. The span scorer gave 95.5 to 96.7 on short answers and 76.0 to 84.7 on long ones.

Under *"Answer in a complete sentence."* the 3B's median prediction stayed at three tokens, but the share of predictions over five tokens rose from 9.3% to 19.6% and the share that were not short from 1.1% to 12.3%, and ANLS scored those at 9.5, so the 3B lost 10.6 points.

5 A Scorer That Separates Formatting from Content

Asking whether the gold answer appears anywhere in the prediction removes the length penalty, but without a limit on length the span scorer gave 98.4 to a prediction consisting of the page's whole released OCR (best-subsequence exact match gave 93.0), since the answer is on the page. We therefore gate the span scorer (§3) at m extra tokens and set m by measurement (Table 4). The attack score rose monotonically with m, and the no-suffix span score rose from 69.2 at $m = 12$ to 82.2 at $m = 20$ and 87.0 at $m = 40$. We took $m = 20$, the largest value in the sweep at which the attack scored under 2 points (1.3), and the default-suffix span score had reached its ungated value of 95.5 by $m = 12$.

Table 5 compares the gated scorer with six others on the same five suffixes and the same attack. The three that compare the whole prediction with the gold (ANLS, exact match, token F1) scored the attack at 0 to 4 and ranged over 54.7 to 86.6 points across the five suffix conditions. The three that look for the answer inside the prediction (best-subsequence exact match, PEDANTS, the ungated span scorer) ranged over 0.7 to 6.7 points and scored the attack at 93.0 to 98.6. The gated scorer had a range of 13.3 and an attack score of 1.3, and every other scorer had a range above 50 points or

	ANLS		drop without suffix		bold
Model	suffix	none	ANLS	span	%
Qwen2.5-VL-3B	93.0	6.0	87.0 [83.6, 90.1]	10.4 [7.3, 13.6]	0
GPT-6 Sol	89.2	54.8	34.4 [29.8, 39.2]	4.5 [2.0, 6.9]	13
Gemini 3.8 Flash	92.0	2.5	89.5 [86.4, 92.3]	18.7 [14.3, 23.1]	100
Claude Opus 5.5	78.1	4.1	74.0 [69.9, 77.7]	16.0 [11.4, 20.5]	100
Qwen3.8-Max	94.9	40.4	54.5 [49.2, 59.6]	8.9 [5.2, 12.7]	56

Table 2: DocVQA, the same 317 questions for every row; Qwen2.5-VL-3B (local) is included for reference. *suffix* is the `lmms-eval` default and *none* no suffix. Drops are paired differences in points with 95% bootstrap intervals. *bold* is the share of no-suffix answers containing Markdown bold.

Suffix	short	ANLS	ANLS given	
	(%)		short	long
`lmms-eval` default	98.9	94.2	95.1	9.6
“complete sentence”	87.7	83.6	93.9	9.5
`" Answer:"`	65.9	65.3	95.7	6.6
no suffix	1.9	7.5	74.3	6.2

Table 3: Qwen2.5-VL-3B on DocVQA, $n = 1{,}200$. Across the top three suffixes ANLS given a short answer barely moved while the reported score fell by 28.8 points. The no-suffix row has 23 short predictions, too few for its conditional to be compared with the others.

m	default	no suffix	page-dump attack
0	94.3	7.7	0.0
5	94.9	17.1	0.0
8	95.4	48.0	0.3
12	95.5	69.2	0.6
20	**95.5**	**82.2**	**1.3**
40	95.5	87.0	4.1
ungated	95.5	88.8	98.4

Table 4: Setting the gate, Qwen2.5-VL-3B on DocVQA: span score under two suffixes and under the page-dump attack. Higher is better in the first two columns and lower in the third; we use m=20.

an attack score above 90.

Selected on DocVQA and applied unchanged to InfographicVQA, $m = 20$ held the 3B's no-suffix span score to 45.7 against 67.1 ungated, because no-suffix InfographicVQA predictions exceeded the longest gold by a median of 16 tokens against 8 on DocVQA, and 35.1% of them exceeded the gate against 7.5%. We therefore release the selection procedure with the scorer, which evaluates each m against a page-dump attack built from the target corpus's OCR and takes the largest m at which the attack stays under 2 points.

Scorer	default	none	range	attack
ANLS	94.2	7.5	86.6	0.0
exact match	86.0	0.9	85.1	0.0
token F1	94.2	39.5	54.7	4.0
best-subsequence EM	90.3	86.9	5.0	93.0
PEDANTS	96.7	96.2	0.7	98.6
span, ungated	95.5	88.8	6.7	98.4
span, gated m=20	95.5	82.2	13.3	1.3

Table 5: Qwen2.5-VL-3B on DocVQA. *range* is the spread in points across the five suffixes of Table 1 and *attack* the score of the page-dump attack.

6 Punctuation, Whitespace and Markdown

Adding the suffix raised SmolVLM's ANLS by 13.5 points and lowered its span score by 11.9 [10.2, 13.8], while on the two Qwen2.5-VL models it raised ANLS by more than 80. Under the default suffix SmolVLM ended every answer with a full stop, `' 1/8/93.'` against the gold `'1/8/93'`, and removing that one character raised its ANLS by 6.1 points and its exact match from 2.8 to 55.3; the same operation changed Qwen2.5-VL-3B's ANLS by −0.1 points.

7 Relaxed Accuracy on ChartQA

ChartQA (Masry et al., 2022) is scored by a relaxed accuracy inherited from PlotQA (Methani et al., 2020) that admits 5% relative error on numeric answers. Removing the suffix moved Qwen2.5-VL-3B from 83.5 [82.1, 84.9] to 55.2 [53.2, 57.2], a drop of 28.4, while a parser that first extracts the number from the prediction and ignores a percent sign that appears on only one of prediction and gold gave 70.4; 383 of the 2,500 no-suffix answers (15.3%) were wrong under the official parser and right under that one. Without the suffix the median answer was one token and the mean 11.3, and the 38% of answers that included reasoning accounted for the lost points. That parser restored 53.6% of

the points lost without the suffix, against 93.8% for the ungated span scorer on DocVQA, and the rest were reasoned answers that reached a wrong value.

The official parser divides by one hundred exactly when a string ends in `%`, separately for prediction and gold, and none of the 2,500 test golds ends in `%` while 1,915 are numeric, so `40%` against the gold `40` has a relative error of 0.99 and is wrong. Qwen2.5-VL-3B ended 1 of its 2,500 answers with a percent sign under the suffix and 17 without it, and neutralizing the rule changed its relaxed accuracy by 0.04 and 0.0 points.

8 Related Work

ANLS (Biten et al., 2019b) was designed for scene-text VQA, where a near-miss was a recognition error, and DocVQA and InfographicVQA adopted it while the leading systems were extractive and kept it once systems became generative. ANLS* (Peer et al., 2024) extends it to structured outputs and keeps compatibility with earlier scores; we hold the metric fixed and vary the suffix, which moved ANLS by up to 92.6 points (Qwen2.5-VL-7B). Nourbakhsh et al. (2025) score whether a DocVQA answer is grounded in the page, which an answer that contains the gold without being grounded in the page would fail.

The closest work is Ye et al. (2026), who audit official errors on six short-answer VQA benchmarks with a semantic judge under the standard terse-answer suffix, find that up to half of the errors on text-rich benchmarks are acceptable answers in the wrong surface form, and show that rewording the prompt flips item-level correctness. We change only the suffix the harness itself appends, which measures the effect on a reported number directly (94.2 to 7.5 on Qwen2.5-VL-3B, 92.0 to 2.5 on Gemini 3.8 Flash), decompose the score by answer length, and give a deterministic scorer tested against an attack. VLMEvalKit's optional `use_verifier` path (Duan et al., 2024) replaces string matching with a model judge the user supplies; PEDANTS (Li et al., 2024) makes a similar answer-equivalence judgement with a learned classifier, and it scored the page-dump attack at 98.6.

Sclar et al. (2024) show that semantically equivalent prompt formats move accuracy by margins comparable to differences between models, and Mizrahi et al. (2024) argue that single-template evaluation misstates the state of the art and propose evaluating over a distribution of prompts. Both place the sensitivity in the model's response to the prompt; in our data 73.3 of the 86.6 ANLS points the 3B lost without the suffix were credited by the span scorer on the same generations, so most of that change came from how ANLS scored the responses.

9 Conclusion

Removing the suffix that makes DocVQA answers terse cost Qwen2.5-VL-3B 86.6 ANLS points and Gemini 3.8 Flash 89.5, and under that suffix Markdown emphasis markers cost Claude Opus 5.5 10.9 points. Scoring the same generations for whether the answer is present, with a gate against the page-dump attack, reduced Qwen2.5-VL-3B's range across the five suffixes from 86.6 to 13.3 points. Because removing the suffix moved every hosted model's score by more than the 17-point spread between their suffix scores, and Markdown handling moved Claude Opus 5.5 by 10.9 points, we recommend reporting the prompt suffix and output post-processing with every document-VQA score.

Limitations

The gate reintroduces length sensitivity in smaller measure. Without a suffix the 7B scored 73.0 gated against the 3B's 82.2, although their ungated span scores were 89.0 and 88.8 and the shares of all their answers (not only zeros) containing a gold 88.1% and 86.9%; the gap comes from the gate applying to 17.4% of the 7B's answers against 7.5% of the 3B's. We selected m on the same DocVQA sample we report it on. The DocVQA results use a 1,200-question sample rather than the full test set, whose labels are available only through the evaluation server.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-VL technical report. *Preprint*, arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025b. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*.

Ali Furkan Biten, Rubèn Tito, Andres Mafla, Lluis Gomez, Marçal Rusiñol, Minesh Mathew, C.V. Jawahar, Ernest Valveny, and Dimosthenis Karatzas.

2019a. ICDAR 2019 competition on scene text visual question answering. In *International Conference on Document Analysis and Recognition (ICDAR)*. ArXiv:1907.00490.

Ali Furkan Biten, Rubèn Tito, Andres Mafla, Lluis Gomez, Marçal Rusiñol, Ernest Valveny, C.V. Jawahar, and Dimosthenis Karatzas. 2019b. Scene text visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. ArXiv:1905.13648.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, Dahua Lin, and Kai Chen. 2024. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201.

Zongxia Li, Ishani Mondal, Huy Nghiem, Yijun Liang, and Jordan Boyd-Graber. 2024. PEDANTS: Cheap but effective and interpretable answer equivalence. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9373–9398, Miami, Florida, USA. Association for Computational Linguistics.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. 2025. SmolVLM: Redefining small and efficient multimodal models. In *Second Conference on Language Modeling (COLM)*. ArXiv:2504.05299.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*. ArXiv:2203.10244.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. 2022. InfographicVQA. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. ArXiv:2104.12756.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. ArXiv:2007.00398.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2020. PlotQA: Reasoning over scientific plots. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. ArXiv:1909.00997.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Armineh Nourbakhsh, Siddharth Parekh, Pranav Shetty, Zhao Jin, Sameena Shah, and Carolyn Rose. 2025. Where is this coming from? Making groundedness count in the evaluation of document VQA models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5341–5361. Association for Computational Linguistics.

David Peer, Philemon Schöpf, Volckmar Nebendahl, Alexander Rietzler, and Sebastian Stabinger. 2024. ANLS* – a universal document processing metric for generative large language models. *arXiv preprint arXiv:2402.03848*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *International Conference on Learning Representations (ICLR)*. ArXiv:2310.11324.

Guanhua Ye, Niu Jingbin, Yan Li, Meiyu Liang, Zhe Xue, Yingxia Shao, and Yawen Li. 2026. What does your short-answer VQA score actually measure? Evaluator-dependent instability in multimodal short-answer benchmarks. *Preprint*, arXiv:2607.10240.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025. LMMs-Eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916. Association for Computational Linguistics.

A Reproducibility

All claims about the local models regenerate from the released generations with `python3 scripts/verify_claims.py`; each entry in `claims.json` names the function and the generation file that produce it, and the script exits non-zero if a recorded value fails to reproduce. The matched-cap, scorer, judge, hosted-model and canonicalization analyses regenerate from the scripts and generation files in the release's `analysis` directory.

B Harness Sources

`lmms-eval` at commit `42296f8`: in `lmms_eval/tasks/docvqa/_default_template_docvqa_yaml`, lines 13–19 set `post_prompt` to `"\nAnswer the question`

using a single word or phrase." under default and to " Answer:" under qwen_vl, with max_new_tokens: 32 and temperature: 0; in api/task.py, lines 888–891 select the entry keyed by the model name passed to the harness and otherwise fall back to default. The InfographicVQA template uses the same default string, and the ChartQA task uses "\nAnswer the question with a single word." with the same qwen_vl override. VLMEvalKit at commit 6f03707: in vlmeval/dataset/image_vqa.py, lines 86–91 append "\nAnswer the question using a single word or phrase." to every DocVQA question, and vlmeval/config.py runs Qwen2.5-VL-3B-Instruct with use_custom_prompt=False, so that string is what the model sees. Appendix B.4 of the Qwen3-VL technical report (Bai et al., 2025a) gives <image> {question} Answer the question using a single word or phrase. as the prompt for DocVQA, InfographicVQA (InfoVQA in the report) and ChartQA.

C Matched Token Cap

	at cap	ANLS	cut to 32	stopped
3B, no suffix	3.0	7.5	7.6	7.8
7B, no suffix	8.9	3.6	4.0	4.0

Table 6: ANLS of no-suffix generations (cap 64; *at cap* is the percentage that reached it), rescored after retokenizing each with the model's tokenizer and keeping the first 32 tokens, and restricted to generations that stopped before the cap. Default-suffix ANLS for comparison: 94.2 and 96.2.

D Stratified Containment Analysis

Restricting to golds of at least two tokens gave a contained share of 79.2% under " Answer:" and 86.3% with no suffix; restricting to golds of at least eight characters gave 77.0% and 85.7%. A raw-substring definition agreed with the token definition to within 2 percentage points in every condition. Canonicalizing numbers and dates before matching (thousands separators, currency and percent signs, number words up to twelve, and dates to month/day/two-digit year) raised the contained share by 0.6 to 2.8 percentage points in each condition with more than 150 zero-scored answers, for example *$19,902 million* against the gold *19,902* and *February 4, 1982* against *Feb. 4, 1982*; the main text uses no canonicalization.

E Containment Judge

The judge saw: *You are checking answers to questions about a document. Question: {q} Reference answer: {g} Response: {p} Does the response commit to the reference answer as its answer to the question? Formatting, extra words, and restating the question do not matter. Say NO if the response gives a different answer, says the reference is wrong or not the answer, or offers the reference as one of several candidates without choosing it. Reply with exactly one word: YES or NO.* Thinking was disabled and every reply parsed. The controls used 200 randomly chosen no-suffix items from the 3B, with the distractor taken from the gold of another question.

F Hosted Models

OpenRouter identifiers openai/gpt-6-sol, google/gemini-3.8-flash, anthropic/claude-opus-5.5 and qwen/qwen3.8-max-0902; every response reported the requested identifier. Each request carried the page as a PNG and the question with or without the default suffix, at temperature 0, with the lowest reasoning setting the API exposes and a 1,024-token output limit.

G Batch Size

Batched decoding with left padding changes Qwen2.5-VL's multimodal rotary position indices, and on five of eight spot-checked items it produced truncated prefixes such as '1/8/' for the gold *1/8/93*, while every single-item generation ended on an end-of-turn token before the cap. Batched decoding also added stray whitespace to 10.7% of outputs, which would enter the whitespace term of §2, so all local generations use batch size one.

Graph-Guided Passage Retrieval for Author-Centric Structured Feedback

Maitreya Prafulla Chitale[1*] **Ketaki Mangesh Shetye**[1*]
Harshit Gupta[2] **Manav Chaudhary**[3] **Manish Shrivastava**[1] **Vasudeva Varma**[1]
[1]IIIT Hyderabad [2]Microsoft, India [3]Sentisum
{maitreya.chitale, ketaki.shetye}@research.iiit.ac.in
harshit.g@alumni.iiit.ac.in
manav@sentisum.com
{m.shrivastava, vv}@iiit.ac.in

Abstract

Obtaining high-quality, pre-submission feedback is a critical bottleneck in the academic publication lifecycle for researchers. We introduce AutoRev, an automated author-centric feedback system that generates structured, actionable guidance prior to formal peer review. AutoRev employs a graph-based retrieval-augmented generation framework that models each paper as a hierarchical document graph, integrating textual and structural representations to retrieve salient content efficiently. By leveraging graph-based passage retrieval, AutoRev substantially reduces LLM input context length, leading to higher-quality feedback generation. Experimental results demonstrate that AutoRev significantly outperforms baselines across multiple automatic evaluation metrics, while achieving strong performance in human evaluations. We have made our code publicly available[1].

1 Introduction

Recent advancements have focused on assisting reviewers and directly generating peer reviews (Yu et al., 2024; Aitymbetov and Zorbas, 2025; D'Arcy et al., 2024; Idahl and Ahmadi, 2025). Concurrently, there has been growing interest in examining the strengths and limitations of such systems, particularly the use of large language models (LLMs) in this domain (Ye et al., 2024). Relying on LLMs to generate or inform final review decisions remains risky due to issues of bias, overconfidence, and superficial feedback and these decision-oriented tools lack utility for authors in the pre-submission phase. Consequently, we advocate for a new paradigm towards author-centric support. We introduce AutoRev, an author-centric system that generates structured, actionable pre-submission feedback containing *Summary*, *Strengths*, *Weaknesses*, and *Questions*, without producing acceptance recommendations or replacing human peer review.

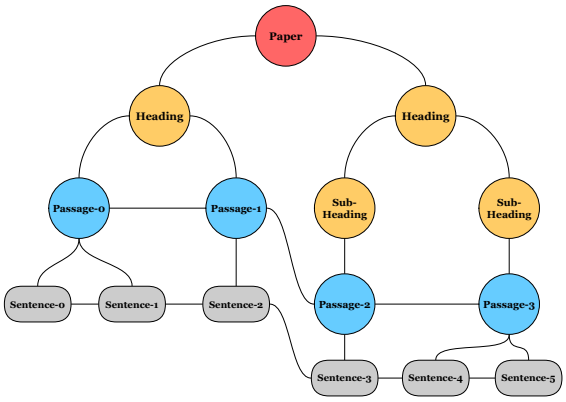


Figure 1: Example document graph with nodes for headings, sub-headings, passages, and sentences. This structure allows AutoRev to capture hierarchical and sequential dependencies across sections.

With the rise of Graph Neural Networks, the use of graphs as *extractors* or as *guides* for facilitating downstream tasks has recently gained traction in the NLP community (Chitale et al., 2025; Gao et al., 2025; Agarwal et al., 2025; Zhu et al., 2025). Inspired by this, we leverage the graph's capability as an extractor of passages to significantly reduce the input token length to the LLM. Reducing the token length of the input is shown to enhance LLM performance (Hsieh et al., 2024). To summarize our main contributions:

1. We propose a novel author-centric framework for generating concise, structured pre-submission feedback focused on providing actionable guidance for authors.

2. We model papers as hierarchical document graphs and learn to select feedback-relevant passages, enabling targeted context selection that reduces LLM input length, mitigates

[*]Equal contribution.
[1]https://github.com/Ketaki1906/AutoRev

The Proceedings of Workshop on Document Intelligence and Understanding, pages 81–99
October 24, 2026 ©2026 Association for Computational Linguistics

long-context degradation, and improves feedback quality, outperforming full-context approaches.

3. We validate our approach using a combination of automatic metrics, human evaluations, factual grounding analysis, and cross-venue generalization, providing a thorough assessment of both quality and practical utility.

2 Leveraging Peer Reviews as Proxy for Pre-Submission Feedback

In the absence of large-scale, publicly available datasets explicitly designed for pre-submission feedback, we use human-written peer reviews as a proxy, as they provide expert critiques of writing, methodology, and presentation aligned with the guidance authors seek prior to submission. In particular, the *Weaknesses* and *Questions* sections offer actionable critiques and clarifications. Since the dataset spans papers at varying stages of writing maturity, the associated reviews reflect feedback targeted at authors with differing levels of experience. We further support this choice through an impact-oriented user study (Section 6.3.4).

3 Related Works

Recent work has explored the use of NLP techniques to support scholarly writing and review workflows, including reviewer selection (Aitymbetov and Zorbas, 2025), meta-review generation (Kumar et al., 2023), and automated review generation (Yu et al., 2024). Many of these approaches rely on large language models, ranging from modular agent-based systems (D'Arcy et al., 2024) to fine-tuned review generators (Idahl and Ahmadi, 2025). In parallel, a separate line of work focuses on author-assistive tools for specific writing sub-tasks, such as limitation suggestion (Faizullah et al., 2024), citation recommendation and literature discovery (Gu and Hahnloser, 2023), and drafting individual sections like related work (Hu and Wan, 2014) or introductions (Liebling et al., 2025). While these are effective within their scoped objectives, they are either reviewer-facing or narrowly targeted, and often operate on full-paper inputs at inference time. As a result, they raise concerns around hallucination, superficial feedback, computational cost, and overconfidence in model-generated judgments (Ye et al., 2024), while failing to provide holistic, structured feedback that supports authors prior to submission.

Yu et al. (2024) introduce an automated paper reviewing framework (SEA) that standardizes reviews using GPT-4 (OpenAI et al., 2024) consolidation and proposes a self-correction strategy during fine-tuning. We adopt SEA as our baseline as it is trained on the publicly available ICLR 2024 dataset and generates the same structured fields as AutoRev, among others.

Studies have explored graph-based extensions to Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) to improve grounding and reasoning in LLMs. Zhu et al. (2025) provide a comprehensive taxonomy of Graph-RAG systems based on database construction, retrieval mechanisms, and processing pipelines. Within this framework, AutoRev represents a domain-specific Graph-RAG approach that constructs a hierarchical graph directly from a scientific paper and employs a learning-based Graph Attention Network (Veličković et al., 2018) for passage retrieval.

4 Dataset

The dataset used in our study is borrowed from Yu et al. (2024), comprising papers submitted to various conferences along with multiple peer reviews per paper. Among these, the ICLR 2024 and NeurIPS 2023 datasets are publicly released, though the GPT-4-consolidated reviews for NeurIPS 2023 are not available. Hence, our analysis focuses only on the ICLR 2024 papers. Table 1 summarizes statistics for the ICLR 2024 papers (Yu et al., 2024).

Attribute	Value
Total Papers	5,653
Average Tokens per Paper	9,815
Total Reviews	21,839
Acceptance Rate	37%
Domain	Machine Learning

Table 1: Dataset statistics for the ICLR 2024 dataset, as reported in Yu et al. (2024).

Each paper is associated with multiple reviews containing sections such as Summary, Strengths, Weaknesses, Questions, Soundness, Presentation, Contribution, Confidence, and Rating. We retain only the Summary, Strengths, Weaknesses, and Questions sections, as these encode constructive feedback for authors-centric pre-submission support (see Section 2), and exclude rating-based fields that corre-

spond to final decision-making. For training, we use the GPT-4 consolidated reviews released by Yu et al. (2024), which merge multiple human-written reviews into a single structured review per paper.

To assess cross-venue performance, we curate a supplementary dataset of 10 papers from COLM and NeurIPS 2025 (CNT_10). This set complements the ICLR 2024 test data (ICT) for the human evaluation detailed in Section 6.3.

5 AutoRev

In this section, we describe the implementation of our proposed framework. Section 5.1 describes the initial passage extraction using Dense Passage Retrieval (DPR) (Karpukhin et al., 2020). The extracted passages are used as training data in Section 5.3, to train a Graph Neural Network (GNN) that learns to extract relevant passages from a research paper. The trained GNN enables our model to perform competitively on the test set without the need for grounding on human-generated feedback, as discussed in Section 5.4. Post that, we describe our process of fine-tuning LLMs to generate the final structured feedback, in Section 5.5. An overview of our methodology can be found in Figure 2.

5.1 DPR-based Passage Extraction

Dense Passage Retrieval (DPR) is a neural retrieval model trained primarily on question-answering datasets that encodes queries and passages into dense vector representations. Given an input query, DPR retrieves relevant passages based on semantic similarity in the embedding space. We use a pre-trained DPR model to identify passages corresponding to the *Summary*, *Strengths*, *Weaknesses*, and *Questions* sections of reviews for each paper in the training set. Specifically, we use the context encoder[2] and query encoder[3] from the *multiset-base* DPR variant.

First, we segment each paper into distinct passages, where each passage is defined as a text segment split at newline characters (\n). Each of these passages is treated as a potential context candidate for DPR. Specifically, for each section of the feedback, we concatenate k consecutive sentences (with $k = 1$, 3, or 5) to form a single query, to capture a broader contextual meaning. For each query, the top-m most relevant passages (with $m = 3$ or 5) are retrieved using DPR. We then consolidate all unique passages retrieved across these multiple queries for each paper.

This final set of passages represents the most relevant excerpts from the full paper that are semantically linked to the feedback content. These extracted passages serve as the ground truth labels for training the GNN, as described in Section 5.3

5.2 Graph Construction for Representing a Paper

The idea of graph construction employed in our work is inspired by Zhao et al. (2024). We borrow their idea of representing a document as a hierarchical graph but extend it beyond sentence level. The graph is constructed directly from the parsed paper and is designed to generalize across diverse academic writing styles and section hierarchies.

Formally, each paper is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes correspond to textual units at different granularities, including the paper root, section headings, sub-headings, passages, and sentences. Edges encode two complementary relationships: (i) hierarchical links ($\mathcal{E}_{hier}$) that reflect the document's structural nesting and (ii) sequential links ($\mathcal{E}_{seq}$) that preserve the linear ordering of passages and sentences. Together, these connections allow information to propagate both vertically across document structure and laterally across adjacent text. A schematic illustration of the document graph is shown in Figure 1. Full details of node types, edge definitions, and construction procedures are provided in Appendix A.

This design explicitly mitigates over-squashing in tree-like graphs, where exponential neighborhood growth leads to information bottlenecks at the root (Alon and Yahav, 2021; Topping et al., 2022). Following the rewiring strategies of Topping et al. (2022), we augment the hierarchical backbone with sequential edges to facilitate lateral information flow and reduce reliance on hierarchical connections. Additionally, the inherent structure of scientific papers results in graphs of shallow depth (typically ≤ 5 hops), limiting long-range dependency compression and preserving fine-grained information from leaf nodes.

5.3 GNN Training for Passage Retrieval

To train a GNN on the graph constructed in Section 5.2, we take the data generated in Section 5.1, which maps each paper p_i to a set of important

[2] https://huggingface.co/facebook/dpr-ctx_encoder-multiset-base

[3] https://huggingface.co/facebook/dpr-question_encoder-multiset-base

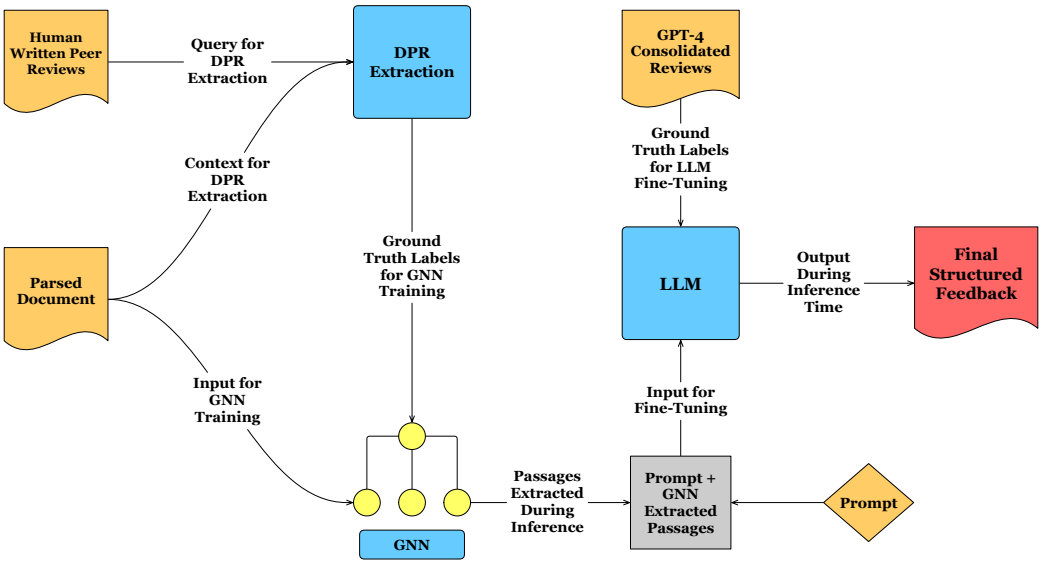


Figure 2: The system begins with a parsed document from which DPR extracts passages for GNN training. The GNN learns to identify key passages relevant to structured feedback generation. The trained GNN selects important passages without requiring human-generated feedback. These passages, combined with a structured prompt, are input to fine-tune an LLM. The fine-tuned LLM generates the final structured feedback, without external grounding.

passages, $\mathcal{O}_i = \{o_1, o_2, \ldots, o_n\}$. Each passage is important for generating a feedback for the paper it is associated with. This mapping is used as a training set for training the GNN. The node embeddings for sentence nodes are initialized with embeddings from a Sentence-Transformer (Reimers and Gurevych, 2019) model, namely *all-mpnet-base-v2*. The node embeddings for all other nodes are initialized with zero embeddings of dimension 768, matching that of the Sentence-Transformer model.

We train a Graph Attention Network (GAT) (Veličković et al., 2018) over the document graph to classify passage nodes as feedback-relevant or not. The GNN is trained using various configurations based on the values of k and m, as described in the DPR process in Section 5.1. These trained GNN models are denoted as GNN (k, m). Additional implementation details related to this section are available in Appendix C.

As illustrated in the example from our test set (Figure 3), the DPR-based approach fails to retrieve a key passage related to the main contributions of the paper, whereas our GNN successfully identifies it. Therefore, graph-based representation, which leverages both sequential and hierarchical node connections, effectively identifies the salient passages required to enhance the feedback generation process. The GNN extracted passages, along with a structured prompt, serve as the input in the LLM fine-tuning stage, as discussed in Section 5.5.

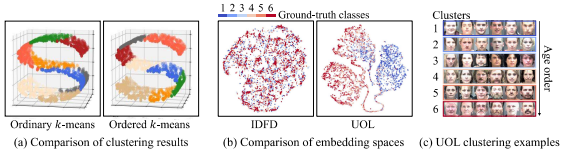
5.4 Grounding and Inference Efficiency

Unlike DPR-based approaches, which rely on human-generated feedback to act as queries, that may be unavailable in real-world scenarios, our system identifies salient passages directly from the document graph, thereby eliminating the dependency on external grounding. Thus, for practical deployment, our GNN-based method offers a crucial advantage by operating autonomously at inference time.

Unlike SEA baselines that process the full text, AutoRev generates feedback from a reduced set of GNN-selected passages. This approach significantly lowers computational cost and input length. As noted by Hsieh et al. (2024), shorter contexts enhance LLM performance, which is also reflected in our higher scores compared to baselines which operate of full paper inputs (Section 6). Table 2 details the specific reductions in tokens and passages.

5.5 Fine-Tuning Large Language Models for Structured Feedback Generation

General-purpose LLMs are not trained specifically to generate structured feedback for academic research papers. Our motivation for fine-tuning is to ensure that the models learn to generate qualita-



(a) Comparison of clustering results (b) Comparison of embedding spaces (c) UOL clustering examples

Figure 1: (a) Comparison of the ordinary k-means and the ordered k-means on data samples around an S curve in the 3D space. (b) Comparison of the embedding spaces for dividing the MORPH II dataset (Ricanek & Tesafaye, 2006) into six clusters, obtained by IDFD (Tao et al., 2021) and the proposed UOL algorithm. We group the ground-truth age classes into six age ranges, depicted by colors from blue to red. (c) Examples in each cluster of UOL on the MORPH II dataset.

- Furthermore, to design an even better space for ordered clustering, we train an embedding network based on the order desideratum that encourages objects to be arranged according to their cluster indices. We iterate the ordered k-means clustering and the network training alternately. For example, in Figure 1(b), the proposed UOL well arranges object instances in a facial dataset, MORPH II (Ricanek & Tesafaye, 2006), according to their ordered classes in the embedding space, while the deep clustering algorithm IDFD (Tao et al., 2021) locates the instances in different classes closely without distinction. Consequently, UOL discovers the underlying age order with no supervision, as shown in Figure 1(c). In other words, given only facial images, UOL sorts them roughly according to the ages.

Extensive experiments demonstrate that UOL provides reliable clustering results on various orderable datasets and also yields promising rank estimation results with no supervision.

The contributions of this paper can be summarized as follows.

- We propose the *first* deep clustering algorithm for orderable data, which discovers clusters together with their hidden order relationships.
- We develop the ordered k-means by extending the ordinary k-means. It not only groups object instances effectively but also sorts the clusters to form a meaningful order. We also prove the local optimality of the solution.
- The proposed algorithm provides promising clustering results on various types of orderable data for facial age estimation, medical assessment, facial expression recognition, and speech evaluation. Also, it shows decent estimation results as an unsupervised rank estimator.

2 Related Work

Deep clustering: In deep clustering, instances are grouped into clusters while a network is trained to embed them into a better space for clustering. Most deep clustering algorithms (Xie et al., 2016; Chang et al., 2017; Van Gansbeke et al., 2020; Wu et al., 2019; Yang et al., 2016; Tsai et al., 2021; Tao et al., 2021; Huang et al., 2022; Niu et al., 2022) adopt the alternate learning framework; they repeat clustering and network training alternately at regular intervals. Caron et al. (2018) employed k-means to cluster instances and trained a network to classify instances simply according to their cluster memberships. Tao et al. (2021) also used k-means, but they trained a network with a feature decorrelation constraint to construct a better embedding space. Recently, Van Gansbeke et al. (2020); Huang et al. (2022); Niu et al. (2020); Ji et al. (2019); Tsai et al. (2021) have adopted the contrastive loss for network training, as self-supervised learning techniques (Chen et al., 2020; He et al., 2020) have shown promising results for constructing discriminative embedding spaces.

Even though many deep clustering algorithms have been proposed, most of them are designed for ordinary data. Thus, when applied to orderable data, they may yield sub-optimal clusters because there is no clear distinction between classes in orderable data. In contrast, the proposed algorithm provides more reliable clustering results by considering the order — which is unknown but to be discovered — of underlying classes.

2

Figure 3: Excerpt of a test sample where passages highlighted in cyan colour refer to relevant passages identified by GNN only and yellow colour refer to relevant passages identified by both GNN and DPR

Method	Avg Token Len	Avg Passages
Full Paper	9445.92	47.07
GNN $(5, 5)$	4147.15	29.49
GNN $(5, 3)$	2899.59	21.00

Table 2: Average token length (Avg Token Len) and Average passage count (Avg. Passages) for a full paper, our best-configuration: GNN $(5, 5)$, and our fastest-configuration: GNN $(5, 3)$.

tively sound, factual and structured feedback.

We select *Mistral-Instruct-v0.2* (Mistral) (Jiang et al., 2023) to align with our SEA-EA baseline, and state-of-the-art (SOTA), open-weight model, *Llama 3.1 8B* (Llama) (Grattafiori et al., 2024) for its strong text generation capabilities. To efficiently adapt these models for structured feedback generation while ensuring computational feasibility, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) for finetuning.

We use the trained GNN, as described in Section 5.3, to retrieve relevant passages from each paper, which are provided as input to the LLM during fine-tuning. For supervision, we use the *Summary*, *Strengths*, *Weaknesses*, and *Questions* sections of human-written peer reviews for ICLR 2024 papers, consolidated using GPT-4 and released as part of the dataset introduced by Yu et al. (2024). These consolidated reviews serve as structured feedback targets for training the LLM. The dataset split is summarized in Table 3. Both models are fine-tuned

Dataset Split	Number of Samples
Training Set	1,765
Test Set	271

Table 3: Number of samples used for training and testing sourced from Yu et al. (2024)

on passages retrieved via varying GNN (k, m) configurations (Section 5.3), denoted as Mistral (k, m) and Llama (k, m). Prompts and hyperparameters are detailed in Appendix D.

6 Results and Analysis

This section presents a comprehensive evaluation of AutoRev across automatic, human, and robustness-focused analyses. We first benchmark feedback quality on the ICLR 2024 test set using ROUGE-1/2/L (Recall and F1) (Lin, 2004) and BERTScore-F1 (Zhang* et al., 2020) against the sections in the reference feedback, assessing both lexical overlap and semantic similarity. To evaluate feedback quality beyond surface similarity, we additionally report Aspect Coverage (ACov) (Yuan et al., 2022), which measures the breadth of scholarly aspects addressed in the generated feedback. Given the limitations of automated metrics, we conduct human evaluations assessing qualitative dimensions such as feedback depth, constructiveness, thoroughness, and helpfulness, alongside a

distinct assessment of factual validity of the generated feedback. We further examine cross-venue generalization on NeurIPS and COLM 2025 papers and assess practical utility via an impact-oriented user survey. Finally, we employ a scalable LLM-as-a-Judge framework and perform ablation studies to analyze the effects of passage-selection parameters.

6.1 Performance Comparison with Baselines

We compare AutoRev against SEA-based baselines from (Yu et al., 2024), including M-7B, SEA-E, and SEA-EA. All baselines operate on the full paper text during both training and inference, without any learned passage selection. M-7B performs direct inference using an instruction-tuned Mistral model, SEA-E leverages standardized fine-tuning to improve review quality, and SEA-EA further incorporates a mismatch score to assess consistency between paper content and generated review.

In contrast, AutoRev employs learned passage selection using a Graph Neural Network over a document graph, providing only salient passages to the LLM. As shown in Table 4 , AutoRev consistently outperforms all baselines across ROUGE (Recall and F1) and BERTScore metrics. Our best-performing configuration, Llama $(5, 5)$, achieves an average improvement of **58.72%** over the state-of-the-art baselines.

To isolate the effects of model capacity from retrieval quality, we also evaluate AutoRev using Mistral. The substantial gains observed with Mistral confirm that the performance improvements stem primarily from GNN-based passage selection rather than the choice of LLM. The graph-based retrieval enables the model to capture both hierarchical and sequential dependencies across the document, providing the focused context necessary for generating comprehensive feedback.

Finally, Llama $(5, 3)$ offers a strong efficiency-performance trade-off, reducing training time by 32.07% with only a 0.71% decrease in performance relative to Llama $(5, 5)$, while still outperforming all baselines.

6.2 Aspect Coverage Analysis

We evaluate the breadth of scholarly criteria addressed in the generated structured feedback using Aspect Coverage (ACov), following the aspect tagging framework of Yuan et al. (2022). Unlike surface-level similarity metrics, ACov measures whether feedback engages with the diverse dimensions typically considered in peer review.

Table 5 reports aspect coverage for our two best-performing configurations, Llama $(5, 3)$ and Llama $(5, 5)$. Both variants demonstrate consistently strong coverage across core feedback dimensions, except for Motivation/Impact and Replicability, indicating that AutoRev generates feedback that goes beyond summarization and engages with substantive and technical aspects of the paper.

Overall, these results suggest that AutoRev provides broad and balanced coverage of evaluation criteria, reinforcing its suitability as a comprehensive pre-submission feedback system.

6.3 Human Evaluation

Understanding the limitations of automatic evaluation metrics, as highlighted by Hanna and Bojar (2021) and Schluter (2017), we conduct human evaluations to assess the acceptability, factual grounding, and practical utility of the generated feedback. Our evaluation is structured around two complementary dimensions: (i) qualitative assessment of feedback quality and (ii) factual validity of individual claims. We perform these evaluations on both the ICLR 2024 test set and a small out-of-domain dataset drawn from NeurIPS and COLM 2025. The annotators selected are experienced authors accustomed to the publication process and academic paper writing. Due to the cost of expert annotation, human evaluations are conducted on representative subsets of the data. We report results for Llama $(5, 3)$ and Llama $(5, 5)$, our best-performing configurations.

6.3.1 Qualitative Assessment

A total of 30 papers are randomly selected from the ICLR 2024 test set (ICT_30). Each paper is associated with three feedback instances: one reference feedback corresponding to the ICT_30 paper and one generated by each of Llama $(5, 3)$ and Llama $(5, 5)$, yielding 90 feedback samples. A total of 27 annotators participate in the study, with each feedback annotated by three independent annotators. Feedback quality is evaluated across four criteria:

1. *Feedback Depth (FD)*: Evaluates the technical understanding and meaningful engagement with the paper's content.

2. *Feedback Thoroughness (FT)*: Measures the coverage of key components like methodology, results, and reproducibility.

3. *Feedback Constructiveness (FC)*: Assesses the utility of suggestions for improvement.

Model	ROUGE F1			ROUGE Recall			BS-F1	Training Hours
	R-1	R-2	R-L	R-1	R-2	R-L		
M-7B	22.55	8.89	11.28	13.48	5.29	16.73	83.79	–
SEA-E	34.29	11.71	14.98	23.28	7.90	10.13	84.04	–
SEA-EA	36.97	13.02	15.88	25.62	8.97	10.97	84.15	–
Mistral $(5, 3)$	53.44	15.47	22.39	53.18	15.42	22.31	86.44	4.34
Llama $(5, 3)$	<u>54.24</u>	<u>15.91</u>	22.59	**53.75**	15.77	22.41	<u>86.59</u>	3.60
Mistral $(5, 5)$	53.65	15.80	<u>22.74</u>	<u>53.60</u>	<u>15.79</u>	**22.74**	86.55	6.44
Llama $(5, 5)$	**54.47**	**16.26**	**22.87**	53.49	**15.98**	<u>22.48</u>	**86.68**	5.30

Table 4: Performance on the ICLR-24 test set comparing our models with baselines (M-7B, SEA-E, SEA-EA) from Yu et al. (2024). **Bold** and <u>underlined</u> values indicate best and second-best scores, respectively. BS-F1 denotes BertScore-F1. Please note that the GPU configurations differ between Llama and Mistral fine-tuning (details in Appendix D).

Model	**SUM**	**CLA**	**SOU**	**SUB**	**ORI**	**CMP**	**REP**	**MOT**
Llama $(5, 3)$	100.0	99.6	98.9	95.2	92.6	79.0	72.0	66.8
Llama $(5, 5)$	100.0	99.3	99.3	95.9	90.0	84.5	71.6	66.8

Table 5: **Aspect Coverage (%)** of model-generated structured feedback evaluated using the Aspect Tagger (Yuan et al., 2022) across aspects: **Summary (SUM)**, **Clarity (CLA)**, **Soundness/Correctness (SOU)**, **Substance (SUB)**, **Originality (ORI)**, **Meaningful Comparison (CMP)**, **Replicability (REP)**, and **Motivation/Impact (MOT)**.

4. *Feedback Helpfulness (FH)*: Evaluates how effectively important issues are identified and explained to the authors.

Annotators assign scores on a 1-5 scale following the guidelines detailed in Appendix E and snapshots of the annotation interface are shown in Appendix F. Scores are averaged across annotators for each feedback and then aggregated across all samples for each model.

As shown in Table 6, AutoRev-generated feedback closely approaches ICT_30 feedback in constructiveness and helpfulness, which are particularly critical for pre-submission support. While ICT_30 feedback remains superior in depth and thoroughness, the results show that AutoRev does not merely summarize paper content, but meaningfully engages with methodological and clarity issues, reinforcing its suitability as a structured feedback tool for authors.

Feedback	**FD**	**FT**	**FC**	**FH**
ICT_30	3.54	3.49	3.26	3.41
Llama $(5, 3)$	3.37	3.30	3.27	3.34
Llama $(5, 5)$	3.27	3.24	3.19	3.30

Table 6: Human Qualitative Assessment scores (scale of 1–5) of generated feedback using Llama models against reference feedback corresponding to ICT_30 paper across four criteria: Feedback Depth (FD), Feedback Thoroughness (FT), Feedback Constructiveness (FC), and Feedback Helpfulness (FH).

6.3.2 Factual Validity Assessment

To evaluate factual grounding, we randomly select 10 papers from the ICLR 2024 test set (ICT_10). Each paper is associated with three feedback instances: one reference feedback corresponding to the ICT_10 paper and outputs from Llama $(5, 3)$ and Llama $(5, 5)$. These feedbacks are annotated by 10 annotators, with each paper evaluated by two annotators.

Annotators assess each statement within a feedback based on its factual consistency with the source paper, using a four-point scale: Completely Agree (CA), Mostly Agree (MA), Mostly Disagree (MD), and Completely Disagree (CD). The evaluation guidelines are provided in Appendix G, and snapshots of the annotation interface are shown in Appendix F.

The percentage agreement scores summarized in Tables 7 and 12 indicate that AutoRev maintains strong factual grounding in the source document. While ICT_10 feedback achieves the highest

agreement, AutoRev-generated feedback receives a clear majority of positive ratings, with complete disagreements (potential hallucinations) remaining below 15%. These findings suggest that graph-based passage selection effectively anchors feedback generation in relevant evidence, mitigating hallucinated content.

6.3.3 Cross-Venue Generalization Assessment

To further examine robustness beyond the training distribution, we conduct an out-of-domain human evaluation on a curated set of 10 papers, comprising five papers each from COLM 2025 and NeurIPS 2025 (CNT_10). Each paper is evaluated using feedback generated by Llama $(5, 3)$ and Llama $(5, 5)$. Due to the absence of publicly available consolidated reference feedback for these venues and the high cost of expert annotation, reference feedback is not included.

A total of 10 annotators participate in the study, with each paper evaluated by two annotators following the same guidelines described in Section 6.3.2. The results, reported in Table 7 and 12, show comparable factual agreement across venues. This suggests that AutoRev generalizes beyond the ICLR 2024 training distribution, capturing structural and semantic regularities common to scientific writing rather than overfitting to venue-specific conventions.

Dataset	Feedback	CA	MA	MD	CD
ICT_10	Reference	39.12	37.85	16.40	6.62
	Llama $(5, 3)$	32.64	31.94	22.92	12.50
	Llama $(5, 5)$	32.27	32.98	23.05	11.70
CNT_10	Llama $(5, 3)$	22.00	39.86	24.57	13.57
	Llama $(5, 5)$	24.06	40.25	22.19	13.50

Table 7: Factual validity assessment (% agreement) on ICLR 2024 (ICT_10) and cross-venue CNT_10 datasets. CA/MA indicate grounded statements, while CD reflects clear factual errors (hallucinations).

6.3.4 Impact Assessment

To evaluate the impact of our system, we conduct a survey among the 10 annotators involved in the tasks discussed in Sections 6.3.2 and 6.3.3. Participants are asked: *"As an author, how likely are you to use AutoRev to get feedback for your own manuscript before submitting it to a conference?"* On a 5-point scale, the system received an average rating of **4.2**. This indicates that authors are strongly inclined to use our model for pre-submission critique, demonstrating the system's practical utility.

6.4 LLM-as-a-Judge Evaluation

As mentioned by Zheng et al. (2023), the LLM-as-a-Judge method offers an explainable and affordable way to approximate human preferences, which are otherwise costly to gather. We employ a large language model, Gemini-2.0-Flash[4] as an automated judge to evaluate generated feedback. Since LLM-as-a-Judge evaluation is scalable and much cheaper to undertake, we evaluate all our models against the reference feedback, over the entire ICT dataset. The evaluation criteria and prompting guidelines are adapted from those used in our human qualitative assessment (Section 6.3.1; Appendix E).

The results of this experiment, summarized in Table 8, show a strong correlation between LLM-as-a-Judge scores and human annotations (Table 6). This alignment validates the automated judge as a reliable proxy for assessing feedback quality in this domain.

Feedback	FD	FT	FC	FH
ICT	3.98	3.87	3.79	3.79
Mistral $(5, 3)$	3.50	3.35	3.44	3.32
Llama $(5, 3)$	<u>3.61</u>	**3.45**	**3.47**	<u>3.39</u>
Mistral $(5, 5)$	3.48	3.36	3.43	3.34
Llama $(5, 5)$	**3.63**	<u>3.44</u>	**3.47**	**3.41**

Table 8: LLM-as-a-Judge (Gemini-2.0-Flash) scores (scale of 1–5) comparing model-generated feedback against ICT across Feedback Depth (FD), Feedback Thoroughness (FT), Feedback Constructiveness (FC), and Feedback Helpfulness (FH). **Bold** and <u>underlined</u> values indicate the best and second-best model scores (excluding reference feedback), respectively.

6.5 Ablation Studies

We perform ablation studies by systematically varying the values of $k \in \{1, 3, 5\}$ and $m \in \{3, 5\}$ during LLM fine-tuning. The detailed results are reported in Appendix B. The results indicate that increasing k and m provides richer context to both Mistral and Llama to generate feedback, leading to consistent improvements across all evaluation metrics. Though, this comes at the downside of higher training times as m increases for a fixed k.

[4] https://deepmind.google/technologies/gemini/

7 Conclusion

In this work, we present AutoRev, an author-centric system for generating structured pre-submission feedback using graph-based passage retrieval. By modeling papers as hierarchical document graphs and leveraging a GNN to identify salient content, AutoRev substantially reduces LLM input length, overcoming the limitations of full-paper fine-tuning, stemming from long-context degradation (Hsieh et al., 2024), while operating without human-written grounding. Experiments on ICLR 2024 show that AutoRev demonstrates strong empirical performance through automatic metrics, with human evaluations confirming constructive, well-grounded feedback and cross-venue generalization. Notably, Llama $(5, 5)$ excels in generating comprehensive feedback for rigorous refinement of manuscripts, whereas Llama $(5, 3)$ offers a favorable efficiency-performance trade-off for large-scale deployment.

Limitations

While our framework provides a robust and high-performing solution for feedback generation, we acknowledge that no automatic system can fully replicate human judgment in complex tasks such as peer reviewing, as noted by Ye et al. (2024). Our goal is to assist authors in refining their papers and we aim for our system to support authors only in the pre-submission feedback phase. Currently, our system is primarily focused on ICLR 2024 papers due to the limited availability of public datasets containing consolidated reviews. However, the consistent positive performance observed in out-of-domain human evaluations (Section 6.3.3) suggests that our model generalizes well beyond the ICLR 2024 setting.

Another limitation is our reliance on human-written reviews consolidated using GPT-4. While consolidated reviews provide a clearer and more consistent supervision signal than multi-review datasets, which are often noisy and contradictory and can hinder model learning, this approach introduces its own challenges. In particular, relying on an LLM for consolidation may induce biases. Future work can explore hybrid strategies that reduce or eliminate reliance on LLMs for consolidating multiple reviews into a single supervision signal.

Ethical Considerations

While our framework provides a robust and high-performing solution for feedback generation, we acknowledge that no automatic system can fully replicate human judgment in complex tasks such as peer reviewing, as noted by Ye et al. (2024). Our goal is to assist authors in refining their papers and we aim for our system to support authors only in the pre-submission feedback phase.

The deployment of AutoRev raises ethical concerns regarding bias, misuse, and academic integrity. Critically, while the system is designed strictly as an author-centric aid, there is a risk of over-reliance or misuse in decision-making, which could undermine the nuances of human reviewing. Furthermore, by training on historical peer reviews and GPT-4 consolidated data, the model may inherit and amplify existing biases present in those texts, such as leniency toward well-known authors and their writing style or under-representation of certain research areas, potentially skewing evaluations. Additionally, the high fidelity of the generated text raises concerns about academic misconduct, where AI critiques could be misrepresented as original human work. To address these issues, we advocate for transparency and emphasize that AutoRev is intended solely as a supplementary tool to support, not replace, the scientific review process.

References

Ankush Agarwal, Chaitanya Devaguptapu, and Ganesh S. 2025. Hybrid graphs for table-and-text based question answering using LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 858–875, Albuquerque, New Mexico. Association for Computational Linguistics.

Nurmukhammed Aitymbetov and Dimitrios Zorbas. 2025. Autonomous machine learning-based peer reviewer selection system. In *Proceedings of the 31st International Conference on Computational Linguistics: System Demonstrations*, pages 199–207, Abu Dhabi, UAE. Association for Computational Linguistics.

Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages

214–217, Barcelona, Spain. Association for Computational Linguistics.

Maitreya Prafulla Chitale, Uday Bindal, Rajakrishnan P Rajkumar, and Rahul Mishra. 2025. DiscoGraMS: Enhancing movie screen-play summarization using movie character-aware discourse graph. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 954–965, Albuquerque, New Mexico. Association for Computational Linguistics.

Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *Preprint*, arXiv:2401.04259.

Abdur Rahman Bin Mohammed Faizullah, Ashok Urlana, and Rahul Mishra. 2024. Limgen: Probing the llms for generating suggestive limitations of research papers. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 106–124, Cham. Springer Nature Switzerland.

Shengxiang Gao, Fang Nan, Yongbing Zhang, Yuxin Huang, Kaiwen Tan, and Zhengtao Yu. 2025. A mixed-language multi-document news summarization dataset and a graphs-based extract-generate model. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9255–9265, Albuquerque, New Mexico. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Nianlong Gu and Richard H.R. Hahnloser. 2023. SciLit: A platform for joint scientific literature discovery, summarization and citation generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 235–246, Toronto, Canada. Association for Computational Linguistics.

Michael Hanna and Ondřej Bojar. 2021. A fine-grained analysis of BERTScore. In *Proceedings of the Sixth Conference on Machine Translation*, pages 507–517, Online. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yue Hu and Xiaojun Wan. 2014. Automatic generation of related work sections in scientific papers: An optimization approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1633, Doha, Qatar. Association for Computational Linguistics.

Maximilian Idahl and Zahra Ahmadi. 2025. OpenReviewer: A specialized large language model for generating critical scientific paper reviews. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pages 550–562, Albuquerque, New Mexico. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Asheesh Kumar, Tirthankar Ghosal, Saprativa Bhattacharjee, and Asif Ekbal. 2023. Towards automated meta-review generation via an nlp/ml pipeline in different stages of the scholarly peer review process. *Int. J. Digit. Libr.*, 25(3):493–504.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Daniel J. Liebling, Malcolm Kane, Madeleine Grunde-McLaughlin, Ian Lang, Subhashini Venugopalan, and Michael Brenner. 2025. Towards AI-assisted academic writing. In *Proceedings of the 1st Workshop on AI and Scientific Discovery: Directions and Opportunities*, pages 31–45, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Natalie Schluter. 2017. The limits of automatic summarisation according to ROUGE. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45, Valencia, Spain. Association for Computational Linguistics.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Rui Ye, Xianghe Pang, Jingyi Chai, Jiaao Chen, Zhenfei Yin, Zhen Xiang, Xiaowen Dong, Jing Shao, and Siheng Chen. 2024. Are we there yet? revealing the risks of utilizing large language models in scholarly peer review. *Preprint*, arXiv:2412.01708.

Jianxiang Yu, Zichen Ding, Jiaqi Tan, Kangyang Luo, Zhenmin Weng, Chenghua Gong, Long Zeng, RenJing Cui, Chengcheng Han, Qiushi Sun, Zhiyong Wu, Yunshi Lan, and Xiang Li. 2024. Automated peer reviewing in paper SEA: Standardization, evaluation, and analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10164–10184, Miami, Florida, USA. Association for Computational Linguistics.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. 2022. Can we automate scientific reviewing? *J. Artif. Int. Res.*, 75.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Chenlong Zhao, Xiwen Zhou, Xiaopeng Xie, and Yong Zhang. 2024. Hierarchical attention graph for scientific document summarization in global and local level. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 714–726, Mexico City, Mexico. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. 2025. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *Preprint*, arXiv:2504.10499.

A Graph Construction

Formally, each paper is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ represent text segments with semantically encoded features, and $\mathcal{E}$ denote the hierarchical and sequential relationships that form the document's structure. Our graph construction is designed to be generalizable across diverse academic paper structures, accommodating arbitrary levels of nesting and heterogeneous section organization.

A.1 Node Definitions

- **Paper Node** (v_{paper}): A single root node representing the entire document.
- **Heading Nodes** ($\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$): Nodes corresponding to top-level section headings.
- **Sub-heading Nodes** ($\mathcal{S}_i = \{s_{i1}, s_{i2}, \ldots, s_{ik}\}$): Nodes representing sub-headings under each heading h_i. This structure naturally extends to accommodate multiple levels of nesting, making it suitable for documents with deeper hierarchical organization.
- **Passage Nodes** ($\mathcal{O}_j = \{o_{j1}, o_{j2}, \ldots, o_{jn}\}$): Nodes representing text segments, split by newline characters (\n), capturing logical textual blocks.
- **Sentence Nodes** ($\mathcal{T}_{jk} = \{t_{jk1}, t_{jk2}, \ldots, t_{jkp}\}$): Nodes for individual sentences within each passage o_{jk}, tokenized using NLTK (Bird and Loper, 2004).

A.2 Edge Definitions

Edges capture both hierarchical and sequential relationships within the document.

- **Root Connections**: $(v_{\text{paper}}, h_i) \in \mathcal{E}_{\text{hier}}$ for all $h_i \in \mathcal{H}$.
- **Heading-SubHeading Connections**: $(h_i, s_{ij}) \in \mathcal{E}_{\text{hier}}$ for all $s_{ij} \in \mathcal{S}_i$.
- **Content Links**: $(s_{ij}, o_{jk}) \in \mathcal{E}_{\text{hier}}$ if $\mathcal{S}_i \neq \emptyset$; $(h_i, o_{jk}) \in \mathcal{E}_{\text{hier}}$ otherwise.
- **Passage-Sentence Connections**: $(o_{jk}, t_{jkq}) \in \mathcal{E}_{\text{hier}}$ for all $t_{jkq} \in \mathcal{T}_{jk}$.
- **Sequential Edges**:

  For passages: $(o_{jk}, o_{j(k+1)}) \in \mathcal{E}_{\text{seq}}$ for $1 \leq k < n$.

  For sentences: $(t_{jkq}, t_{jk(q+1)}) \in \mathcal{E}_{\text{seq}}$ for $1 \leq q < p$.

The final edge set is the union of hierarchical and sequential edges:

$$\mathcal{E} = \mathcal{E}_{\text{hier}} \cup \mathcal{E}_{\text{seq}}, \tag{1}$$

where $\mathcal{E}_{\text{hier}}$ captures the vertical (structural) relationships and $\mathcal{E}_{\text{seq}}$ captures the horizontal (textual progression) flow. This graph construction is agnostic to section depth and flexibly adapts to any nested structure within academic documents. A toy example of such a graph, built from a paper with 2 Headings, 2 Sub-Headings, 4 Passages, and 6 Sentences, is shown in Figure 1.

B Ablation studies

The quantitative performance of different combinations of k (number of review sentences concatenated; $k = 1, 3, 5$) and m (number of top DPR-retrieved passages used to train the GNN; $m = 3, 5$) for review generation is reported in Table 9 and Table 10, evaluated against the reference feedback. While calculating the ROUGE metrics, we enable the `use_stemmer` option to ensure consistent matching of words with different morphological forms, to improve the reliability of the evaluation.

C GNN Implementation Details

The passing of information from one node to another is enhanced by the GATConv (Equations 2, 3, and 4) layers, which involves the concept of attention to selectively propagate messages in the Message Passing layer of a GNN.

$$\mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right) \tag{2}$$

$$e_{ij} = \text{LeakyReLU}\Big(\mathbf{a}^T \big[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j\big]\Big) \tag{3}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})} \tag{4}$$

We train and test our model on a single NVIDIA RTX6000 GPU with 48GB of VRAM.

C.1 Model Architecture

The model contains a Graph Neural Network (GNN) module and a passage classifier.

GNN Module The GNN uses three graph attention (`GATConv`) layers. The first `GATConv` layer processes input features (of size *768 (embedding dimension of Sentence Encoder)*) into 128-dimensional features with 4 attention heads, producing a concatenated output of 128×4 dimensions. A residual connection is incorporated by applying a linear layer to project the input to 128×4, which is added to the `GATConv` output. This is followed by an ELU activation and dropout ($p = 0.2$).

The second `GATConv` layer maps the 128×4 input to 256 dimensional features using 2 attention heads (output dimension 256×2). A residual connection similarly projects the prior layer's output to 256×2, followed by ELU and dropout ($p = 0.2$). The final `GATConv` layer upscales the 256×2 features to the target output dimension (`1024 in our case`) without residual connections and dropout layers.

Passage Classifier For passage nodes, we employ a feedforward classifier to map the output embeddings from the GNN to the final label space. The classifier is implemented as a multi-layer perceptron (MLP) comprising the following sequence of operations:

- A linear transformation projecting the input to a 512 dimensional space.

Model	ROUGE-F1			ROUGE-Recall			BERTScore F1	Training Time (hrs)
	R-1	R-2	R-L	R-1	R-2	R-L		
Mistral $(1, 3)$	53.54	15.81	<u>22.68</u>	53.26	15.74	<u>22.59</u>	86.51	7.60
Mistral $(1, 5)$	53.54	15.76	22.64	52.81	15.56	22.36	86.51	10.22
Mistral $(3, 3)$	<u>53.86</u>	15.80	22.62	53.21	15.63	22.36	86.54	6.33
Mistral $(3, 5)$	**53.87**	**15.91**	22.74	<u>53.43</u>	<u>15.79</u>	22.57	<u>86.54</u>	7.64
Mistral $(5, 3)$	53.44	15.47	22.39	53.18	15.42	22.31	86.44	4.34
Mistral $(5, 5)$	53.65	<u>15.80</u>	**22.74**	**53.60**	**15.79**	**22.74**	**86.55**	6.44

Table 9: Overall performance of Mistral across different (k, m) settings. Metrics include ROUGE-F1 (R-1, R-2, R-L), ROUGE-Recall (R-1, R-2, R-L), BERTScore F1 and training time (in hours). Bold highlights the best scores, and underlined values indicate the second-best.

Model	ROUGE-F1			ROUGE-Recall			BERTScore F1	Training Time (hrs)
	R-1	R-2	R-L	R-1	R-2	R-L		
Llama $(1, 3)$	54.42	16.19	22.76	53.71	15.98	22.48	86.61	6.65
Llama $(1, 5)$	54.40	16.09	22.82	53.58	15.86	<u>22.50</u>	86.61	8.23
Llama $(3, 3)$	<u>54.52</u>	16.03	22.67	**53.79**	15.83	22.39	86.61	5.11
Llama $(3, 5)$	**54.61**	**16.30**	**22.88**	<u>53.78</u>	**16.06**	**22.56**	<u>86.66</u>	6.28
Llama $(5, 3)$	54.24	15.91	22.59	53.75	15.77	22.41	86.59	3.60
Llama $(5, 5)$	54.47	<u>16.26</u>	<u>22.87</u>	53.49	<u>15.98</u>	22.48	**86.68**	5.30

Table 10: Overall performance of Llama across different (k, m) settings. Metrics include ROUGE-F1 (R-1, R-2, R-L), ROUGE-Recall (R-1, R-2, R-L), BERTScore F1 and training time (in hours). Bold highlights the best scores, and underlined values indicate the second-best.

- A SiLU activation followed by dropout with a probability of $p = 0.3$.
- A second linear layer projecting to 256 dimensions.
- Layer normalization applied to the 256-dimensional features.
- Another SiLU activation.
- A final linear layer projecting to the number of target classes (typically 2 for binary classification).

This architecture enables the model to learn non-linear transformations of the graph-encoded passage representations while incorporating regularization and normalization for stable training.

C.2 Training

The model is trained for 30 epochs, with the best validation performance checkpoint retained for inference. The loss function used for training is the Cross Entropy Loss.

D Finetuning Implementation Details

We fine-tune *Llama-3.1-8B* using two NVIDIA L40S GPUs and *Mistral* using a single NVIDIA RTX 6000 Ada GPU. Both GPUs are equipped with 48GB of GDDR6 VRAM.

We apply Low-Rank Adaptation (LoRA) with a rank of 8 and a scaling factor ($\alpha = 16$). LoRA is integrated into the following transformer layers: *q_proj*, *k_proj*, *v_proj*, *o_proj*, *gate_proj*, *up_proj*, and *down_proj*. Models are fine-tuned for 3 epochs. The total training time for different configurations is reported in Table 9 and Table 10. The full set of hyperparameters used during training is listed in Table 11.

D.1 Prompt for Finetuning

The prompt used for fine-tuning both *Llama 3.1 8B* and *Mistral-Instruct-v0.2* is shown in Listing D.1.

Hyperparameter	Value
Batch Size	1
Gradient Accumulation Steps	1
16-bit Floating Point Precision	True
Optimizer	adamw_torch
Learning Rate	1×10^{-4}
Max Gradient Norm	0.3
Warmup Ratio	0.03
LoRA Rank	8
LoRA scaling factor	16
LoRA Dropout	0.05
Weight Decay	0.01
LR Scheduler Type	cosine

Table 11: Hyperparameters Used During Finetuning

Prompt Used During Finetuning

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
Instruction:
Generate a structured feedback for the research paper passages provided below. The feedback should include a summary of the paper, its strengths, weaknesses, and questions for the authors. Consider that the feedback is being given for a paper submitted to the ICLR conference.
Research Paper Passages:
Feedback for the paper:

```
**Summary** + "\n\n" +
**Strengths** + "\n\n" +
**Weaknesses** + "\n\n" +
**Questions** + "\n\n"
```

The extracted relevant passages are inserted ahead of the "Research Paper Passages" heading. Similarly, the corresponding content for each feedback section (Summary, Strengths, Weaknesses, and Questions) is added after their respective headings in the generated feedback.

E LLM-as-a-judge and Qualitative Assessment

The prompt used for LLM-as-a-judge with Gemini-2.0-Flash is given below. The corresponding model-generated feedback is inserted under the appropriate subheadings within the prompt.

For Qualitative Assessment, the prompt below is modified only slightly as to just remove the output format. This slightly modified prompt is given as the evaluation guideline to the annotators. It is ensured that the annotators selected are eligible to review papers for the particular conference.

Prompt Used LLM-as-a-judge

Instruction: You are an expert on research papers. Evaluate the following feedback according to the scoring methodology.
Given below is a structured feedback generated for a paper submitted to the ICLR conference.
Give rating for the categories described below on the full scale from 1 (lowest rating) to 5 (highest rating). Assign scores independently per category (they may differ significantly).

Category 1: Feedback Depth
1 - Unqualified: Admits lack of expertise | Fundamental misunderstandings | Fails to engage with technical content
2 - Limited: Surface-level comprehension | Misses >2 key concepts | Confuses established methods
3 - Functional: Grasps main contributions | Identifies 1-2 technical flaws | Limited literature context
4 - Qualified: Precise technical critique | Contextualizes in recent work (>3 citations) | Identifies methodological tradeoffs
5 - Authoritative: Demonstrates field leadership | Predicts follow-up research directions | Reveals fundamental limitations in approach

Category 2: Feedback Thoroughness
1 - Negligent: <3 substantive comments | Skips methodology/results | Fails to address claims
2 - Patchy: Only addresses obvious aspects | 0 analysis of experiments | No ethical consideration
3 - Standard: Checks main claims | Basic methodology feedback | 1 reproducibility concern noted
4 - Rigorous: Line-by-line methodology check | Statistical validity analysis | 2+ improvement suggestions
5 - Forensic: Alternative approach proposals | Re-analysis of key results | Compliance checklist (ethics, reproducibility, etc.)

Category 3: Feedback Constructiveness
1 - Destructive: Personal attacks | Unsubstantiated dismissal | 0 actionable feedback
2 - Demanding: Unrealistic requests ("redo all experiments") | Vague quality complaints
3 - Directive: General improvement areas | 1-2 specific examples | Mixed positive/negative tone
4 - Strategic: Priority-ranked suggestions | Alternative methodology paths | Conference-specific improvements
5 - Transformative: Step-by-step revision roadmap | Code/experiment snippets provided | Before-after examples

Category 4: Feedback Helpfulness
1 - Misleading: Factually incorrect suggestions | Contradicts paper's goals | Harmful recommendations
2 - Abstract: "Improve writing" without examples | "Add experiments" without design
3 - Guided: Section-specific edits | 2-3 implementable suggestions | Partial error diagnosis
4 - Impactful: Directly address weaknesses | Templates/examples provided | Clear priority levels
5 - Revelatory: Exposes fundamental flaws | Provides benchmark comparisons | Supplies missing literature

Scoring Methodology
For EACH category:

1. Find the HIGHEST level (5-1) where ALL criteria are FULLY satisfied (Base Level)
2. Check criteria in the NEXT HIGHER LEVEL (Base Level + 1):
    - Calculate partial credit: (Satisfied Criteria in Higher Level / Total Criteria in Higher Level)
3. Final Score = Base Level + Partial Credit
4. Round to 2 decimal places (0.33 increments for 3-criteria levels, 0.5 for 2-criteria levels)

Structured Feedback: {structured_feedback}

Scores for the paper feedback:
Respond ONLY with this JSON format:

```
{
  "feedback_scores": {
    "Category 1": "",
```

```
    "Category 2": "",
    "Category 3": "",
    "Category 4": ""
  }
}
```

F Human Evaluation UI

Our annotation UI is developed using Streamlit [5].

F.1 Qualitative Assessment

Examples of the user interface used for the Qualitative Assessment are shown in Figure 4 and 5.

F.2 Factual Validity Assessment

Dataset	Feedback	Positive Rating (%)
ICLR 2024	ICT_10	76.97
	Llama $(5, 3)$	65.25
	Llama $(5, 5)$	64.58
Other	Llama $(5, 3)$	61.86
	Llama $(5, 5)$	64.31

Table 12: Overall Positive Rating Percentage (sum of "Completely Agree" and "Mostly Agree") for test dataset of ICLR 2024 as well as NeurIPS and COLM 2025 (Other).

Examples of the user interface used for Factual Validity Assessment can be seen in Figures 6a and 6b.

G Evaluation Guidelines for Factual Validity Assessment

The evaluation guidelines for the factual validity assessment are shown in Section G. It is ensured that the annotators selected are eligible to review papers for the particular conference.

Evaluation Guidelines for Factual Validity Assessment

Instructions:
To assess a structured feedback and check how accurately it reflects the paper's content.
Evaluation Process:

1. Read the entire paper first.
   - If you need to download the paper for some reason, do ensure that you only download the paper using the provided UI.
2. Read the structured feedback and identify every **DISCRETE CLAIM**.
   - For the "Summary" section of the review, treat the entire content of the Summary as **ONE discrete claim**.
   - For the "Strengths", "Weaknesses", and "Questions" sections of the review, treat **EACH bulleted point** as a separate discrete claim.
3. For each claim, choose the best category from the dropdown.

Claim Categorization:
You must rate the **Validity Agreement** of every claim on a scale of 1–4, where 1 indicates "Completely Disagree" and 4 indicates "Completely Agree".
Completely Disagree (Invalid Claim) [1]:

- The claim is factually wrong, contradicts the paper, or is hallucinated.

Mostly Disagree (Largely Invalid Claim) [2]:

- The claim is mostly incorrect, though it may contain minor truth.

Mostly Agree (Largely Valid Claim) [3]:

- The claim is mostly correct, with minor gaps or oversights.

Completely Agree (Fully Valid Claim) [4]:

- The claim is entirely accurate.

H Example of a Generated Structured Feedback

We present a structured feedback generated by our system for a paper from our test set in Section H.

Output of the generated feedback by AutoRev for the paper *On the Role of Edge Dependency in Graph Generative Models*[6]

Summary
The paper introduces a novel framework for evaluating generative models of graphs, focus-

[5] https://streamlit.io/

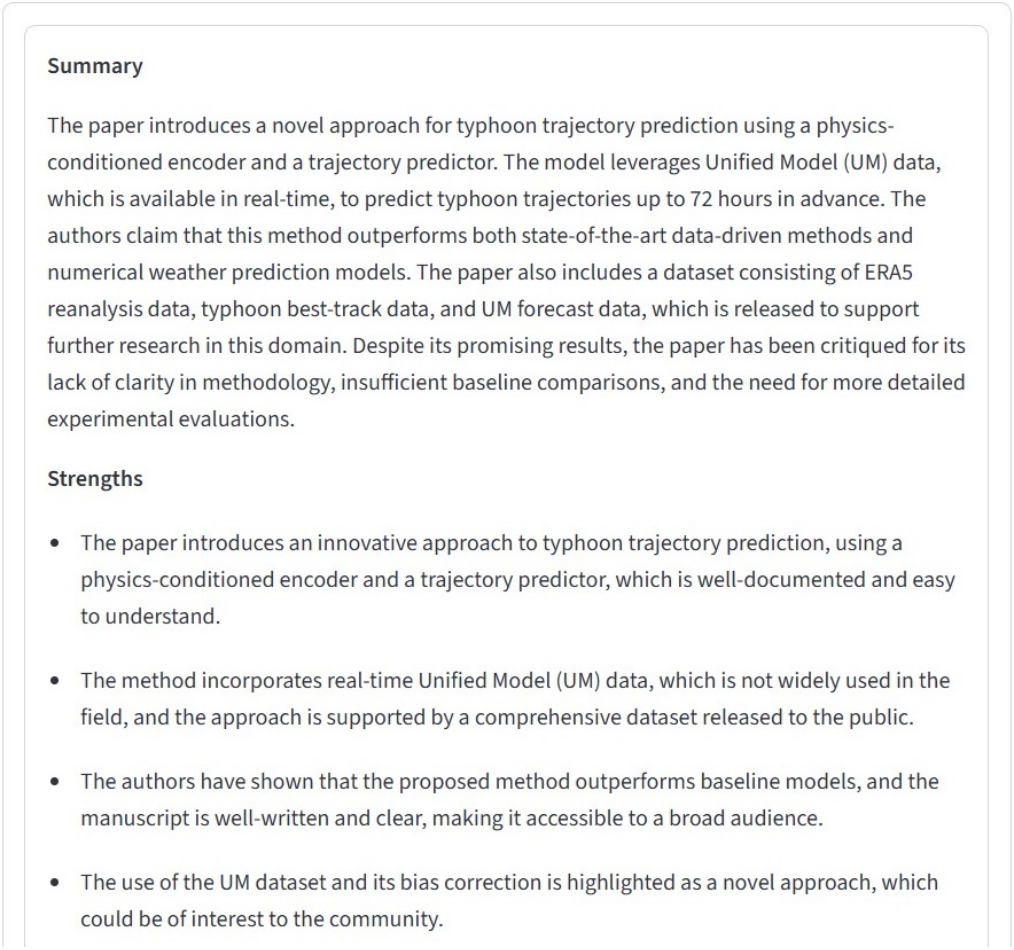


Figure 4: This interface component corresponds to the Qualitative Assessment task described in Section 6.3.1. It marks the beginning of the feedback presentation. Due to space constraints, only the summary and a subset of the strengths from one feedback are shown here.

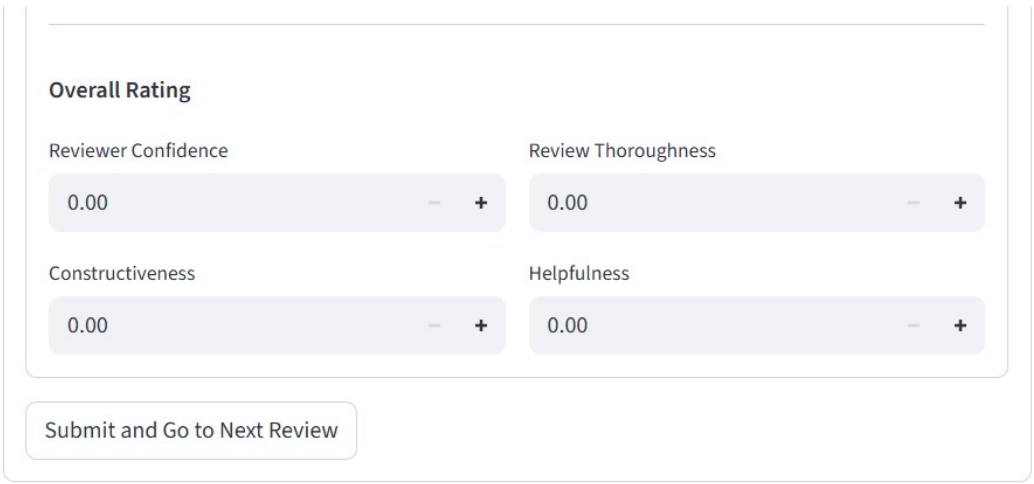


Figure 5: This interface component corresponds to the task described in Section 6.3.1. In this part, annotators provide a final rating across the four categories and submit their annotation.



(a) The original paper is displayed in a PDF viewer to provide factual context.

(b) Each point of the review is rated for accuracy using a dropdown menu.

Figure 6: The user interface for the Factual Validity Assessment task described in Section 6.3.2. On the left (a), the full research paper is presented to the annotator. On the right (b), the annotator rates each point of the feedback based on its consistency with the source document.

ing on the concept of "overlap" to ensure both accuracy and edge diversity. It categorizes graph generative models into three levels of complexity: edge independent, node independent, and fully dependent models. The paper derives theoretical bounds on triangle counts and other short-length cycles based on overlap and provides new generative models for each level. However, the paper's novelty and practical contributions are questioned due to the use of existing methods and the lack of significant empirical validation. The experimental section is seen as limited and does not convincingly demonstrate the superiority of the proposed methods over existing ones.

Strengths

- The paper introduces a novel evaluation framework that considers the importance of model-generated graph overlap to ensure both accuracy and edge diversity, which is a significant contribution to the field of graph generative models.
- The paper presents a hierarchy of graph generative models categorized into three levels of complexity, which encapsulates a wide range of prevalent methods and provides a structured framework for evaluating graph generative models.
- The theoretical bounds on the number of triangles and other short-length cycles are well-presented and support the findings with instances demonstrating the asymptotic optimality of these bounds.
- The authors introduce new generative models for each of the three hierarchical levels, leveraging dense subgraph discovery (Gionis & Tsourakakis, 2015), which is a practical application of the theoretical findings.

Weaknesses

- The novelty of the paper is limited as it primarily relies on existing methods and theoretical frameworks, and the experimental results do not convincingly demonstrate the superiority of the proposed methods over existing ones.
- The empirical evaluation is weak and does not provide sufficient evidence to support the claims made in the paper. The experimental results are not robust and the paper does not provide enough comparisons with other models to establish the effectiveness of the proposed methods.
- The paper has several presentation issues, including unclear definitions and notations, which could hinder the understanding of the content for readers not familiar with graph generative models.
- The practical applications and the significance of the proposed methods are not clearly demonstrated, and the paper lacks a detailed discussion on the limitations and potential future work.

Questions

- Can the authors clarify the definitions and notations used in the paper, particularly the ones related to the overlap and triangle density, to ensure a clearer understanding of the content?
- How do the authors justify the novelty of their work given the reliance on existing methods and theoretical frameworks?
- What specific contributions does the paper make to the field of graph generative models, and can these be clearly articulated?
- Why was the empirical evaluation conducted in a limited manner, and how can the paper be improved to provide more robust and convincing experimental results?
- Could the authors discuss the practical applications of the proposed methods and the significance of their work in more detail?
- Are there any plans to extend the theoretical bounds to higher-order cycles, and how might this affect the overall conclusions drawn in the paper?

[6]https://openreview.net/forum?id=LCQ7YTzgRQ

GraphER: An Efficient Graph-Based Enrichment and Reranking Method for Retrieval-Augmented Generation

Ruizhong Miao, Yuying Wang, Rongguang Wang, Chenyang Li, Tao Sheng, Sujith Ravi, Dan Roth

Oracle AI

Correspondence: ruizhongmiao@gmail.com

Abstract

Semantic search in retrieval-augmented generation (RAG) systems is often insufficient for complex information needs, particularly when relevant evidence is scattered across multiple sources, because it may fail to retrieve the *complete* set of evidence. Existing approaches to addressing this problem either rely on iterative agentic retrieval, which can be computationally inefficient, or maintain additional structures such as knowledge graphs, which introduce storage and maintenance overhead. In this paper, we propose **GraphER**, a graph-based enrichment and reranking framework that (1) leverages the organizational structure of data to capture proximity relationships beyond semantic similarity, (2) constructs a graph at query time based on these proximities, and (3) applies graph-based ranking to surface the top candidate documents. Experiments across table retrieval, multi-hop retrieval, and long-document retrieval benchmarks demonstrate consistent improvements in terms of *retrieval completeness*. Additionally, GraphER requires no additional graph infrastructure and integrates seamlessly with standard vector stores. The framework is retriever-agnostic, supports multiple forms of proximity, and introduces minimal query-time latency.

1 Introduction

In retrieval-augmented generation (RAG; Lewis et al., 2020; Yasunaga et al., 2023) systems, incomplete retrieval is often a primary bottleneck to answer quality. Modern neural retrievers estimate query-document relevance based on semantic similarity. For example, embedding-based bi-encoders and cross-encoders map queries and documents into vector representations and rank documents according to their proximity in the embedding space. Despite their scalability and effectiveness, these models often struggle with complex queries that require integrating information from multiple sources. This limitation arises mainly due to (1) they are point-wise rankers that evaluate each query-document pair in isolation without considering the relationships among candidate documents; and (2) semantic proximity alone may not fully capture the organizational structure of the data within the retrieval corpus.

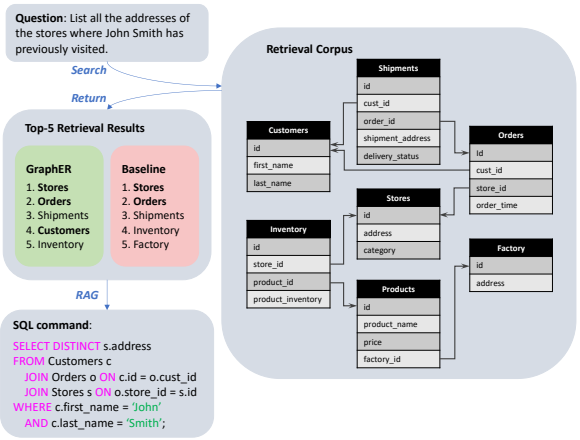


Figure 1: Example of RAG for SQL generation: Semantic search misses the Customers table in the top-5 results, whereas GraphER reranks the candidates to retrieve all relevant tables.

Consider the table retrieval example in Figure 1. To generate the correct SQL query, the retriever must return all relevant tables: *Customers*, *Orders*, and *Stores*. Semantic search ranks *Orders* and *Stores* highly because of their semantic proximity to the question, but fails to return *Customers* in the top-5 results. Although *Customers* is required to generate the correct SQL query, its relevance stems primarily from its foreign-key relationship with *Orders* rather than its semantic similarity to the question.

These observations motivate retrieval methods that consider not only semantic proximity between query-document pairs but also relationships among

The Proceedings of Workshop on Document Intelligence and Understanding, pages 100–111
October 24, 2026 ©2026 Association for Computational Linguistics

candidate documents. Prior work has explored several approaches to improve retrieval completeness in RAG. Agentic retrieval methods use LLM-guided iterative search to decompose queries and gather evidence across multiple retrieval steps (Yao et al., 2023; Khot et al., 2023; Zhou et al., 2023; Trivedi et al., 2023; Asai et al., 2024). The agentic approach is well-suited for retrieving information from external, open-ended environments (e.g., web search), where the data organization is largely unknown and users lack control over the indexing process. However, for private or enterprise databases, where users do have control over the organization of the available data, these methods can incur unnecessary latency and cost due to repeated retrieval and reasoning. Alignment-oriented approaches (Chen et al., 2025b) attempt to jointly reason over content and structure, but they also rely on computationally expensive online LLM calls and constrained decoding, which limit their practicality in many production settings.

Another class of methods augments retrieval with auxiliary data structures, such as knowledge graphs or hierarchical indexes, to capture relationships beyond semantic proximity (Sun et al., 2024; Luo et al., 2024; Gutiérrez et al., 2024, 2025; Sarthi et al., 2024; Sourati et al., 2026). These approaches require maintaining additional infrastructure alongside the base retriever. While effective in their respective settings, such infrastructure introduces computational and storage overhead, limiting their applicability in general-purpose RAG systems.

To address these limitations, we introduce **GraphER**, a **Graph**-based **E**nrichment and **R**eranking method for RAG systems. GraphER is capable of incorporating various forms of proximity relationships beyond semantic proximity. Its objective is to improve the completeness of the top-K results by exploiting these proximity relationships among candidate objects. GraphER is designed to be a standalone and computationally efficient reranking stage that is compatible with other upstream and downstream reranking stages. Therefore, GraphER can be combined with more computationally intensive semantic rerankers, such as cross-encoder rerankers.

From an efficiency perspective, unlike knowledge graph-based approaches, GraphER requires no additional infrastructure beyond a base retriever. Moreover, any LLM calls in GraphER are confined to the offline phase, while online retrieval is entirely LLM-free and incurs minimal query-time latency. These properties make GraphER straightforward to integrate into production RAG systems.

In summary, our contributions are as follows:

- We introduce **GraphER**, a graph-based enrichment and reranking framework that improves retrieval completeness in RAG systems.
- We propose **Graph Cohesive Smoothing (GCS)**, a lightweight graph-based ranking algorithm that mitigates hub-node bias.
- We demonstrate that GraphER naturally supports graph neural network rerankers.
- We conduct extensive experiments on three distinct types of retrieval tasks to validate the effectiveness of GraphER.

2 Methodology

GraphER consists of two main components: (1) an offline enrichment process that incorporates graph information during indexing, and (2) an online graph-based reranking process that leverages the enriched graph information to refine the initial search results. Figure 2 gives an overview of the GraphER methodology.

2.1 Offline indexing enriched with graph information

During offline indexing, we enrich each data object[1] with metadata that encode potential graph relationships to other data objects. In this work, we consider three types of proximity relationships that capture distinct sources of non-semantic relationships commonly found in structured and unstructured data, thereby covering a broad range of use cases:

Structural proximity Metadata are derived from predefined relationships such as foreign-key links between tables or hyperlinks between webpages.

Conceptual proximity We extract named entities from each data object using an instruction-tuned LLM and store them as metadata. Two objects are considered conceptually close if they share one or more named entities.

[1]The term data object here can refer to either a text passage or a table that has been serialized into text format.

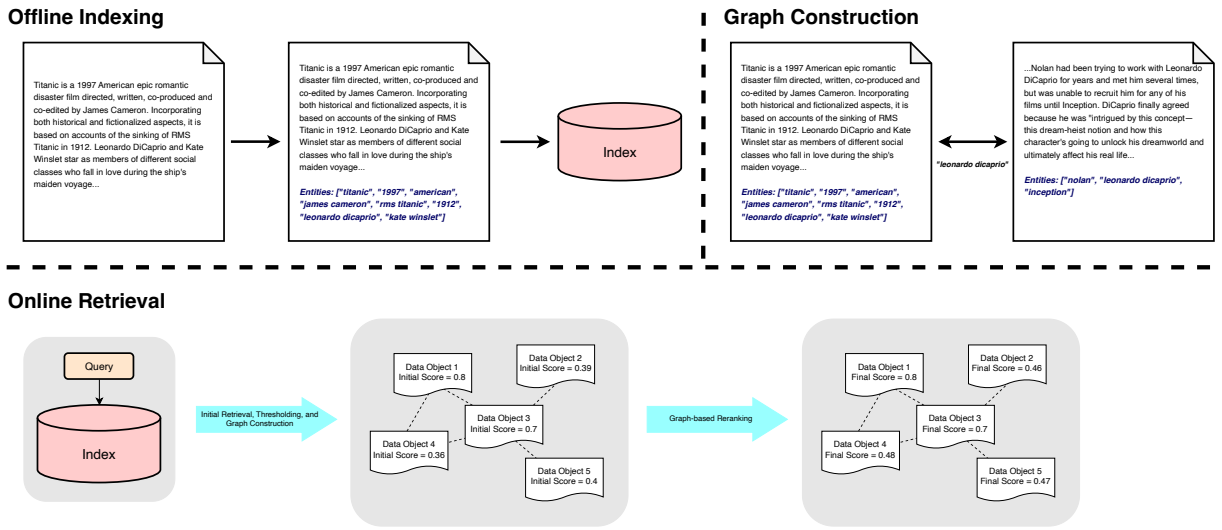


Figure 2: During offline indexing, each data object is enriched with metadata that informs edge relationships and then indexed using standard retrieval methods. During online retrieval, a base retriever retrieves candidate objects and assigns initial scores, after which a graph is constructed from the candidates based on the enrichments. A graph-based ranking algorithm is then applied to produce the final relevance scores.

Contextual proximity We record document and chunk identifiers to link adjacent chunks from the same document.

Using these metadata, a graph edge can be created between two data objects when they satisfy the corresponding proximity criterion. Since each object is enriched independently, the enrichment process requires only a single pass through the corpus and has linear complexity with respect to the size of the index.

2.2 Online graph construction and graph-based reranking

The online retrieval process consists of four steps: candidate retrieval, graph construction, graph-based reranking, and outputting the final ranked list. The candidate retrieval step is a standard procedure in RAG systems: the base retriever retrieves the top-n candidate objects and assigns an initial relevance score to each of them.

Graph construction Given the retrieved candidates, we construct a graph in which each node represents a candidate data object. An edge is added between two nodes if their metadata indicate a proximity relationship, as described in Section 2.1. For example, two passages enriched with named entities are connected if they share at least one named entity.

Note that the graph here differs from a knowledge graph in two key respects. First, graph construction occurs only for candidate objects during query time, and no knowledge graph is maintained in the index. This enables GraphER to integrate seamlessly with standard vector stores. Second, unlike a knowledge graph, where nodes represent entities and edges represent relationships, in GraphER a node represents a data object, which is typically a text string containing multiple entities, and the edges are unlabeled.

Graph-based ranking We then run graph-based ranking algorithms on the constructed graph. Personalized PageRank (PPR) has been used in various RAG systems (Gutiérrez et al., 2024, 2025; Xu et al., 2026) to rank candidate objects. As a comparative graph ranking algorithm, we also include PPR in our experiments. However, PPR has a tendency to favor hub nodes, i.e., nodes with a large number of connections to other nodes. This occurs even when both the hub node itself and its neighbors have low initial scores. This is due to PPR's random walk mechanism, which causes the hub node to be frequently visited and accumulate probability flows from its numerous neighbors. As a result, the hub node tends to be ranked highly, regardless of its actual relevance to the query.

To address this limitation of PPR, we draw on the principle of graph cohesion (Li et al., 2019; Le and Li, 2022), which assumes that connected nodes

	Spider1_train	Spider1_dev	Spider1_test	Bird_train	Bird_dev	Beaver_dev	HotpotQA	2WikiMultihopQA	MuSiQue	BEIR-NQ
# queries	8650	1032	2147	9398	1534	209	2000	2000	2417	3452
# queries with >1 rel. object	3658	378	787	7679	1218	206	2000	2000	2417	666
# data objects	876	876	1056	522	75	463	19323	12601	21100	117683

Table 1: Dataset statistics.

Algorithm 1 Graph Cohesive Smoothing

Input: Adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with $\boldsymbol{A}_{ij} \geq 0$, seed score vector $\boldsymbol{s} \in \mathbb{R}^n$, damping factor $\alpha \in (0, 1)$, tolerance $\epsilon > 0$.
Output: Relevance score vector $\boldsymbol{p} \in \mathbb{R}^n$
Definition: Let $\boldsymbol{W}$ be the row-normalized transition matrix, where

$$\boldsymbol{W}_{ij} = \begin{cases} \frac{\boldsymbol{A}_{ij}}{\sum_{k=1}^{n} \boldsymbol{A}_{ik}} & \text{if } \boldsymbol{A}_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

initialize $\boldsymbol{p}^{(0)} = \boldsymbol{s}$, $t = 0$.
repeat
  $\boldsymbol{p}^{(t+1)} = \alpha \boldsymbol{s} + (1 - \alpha)\, \boldsymbol{W}\, \boldsymbol{p}^{(t)}$
  Compute residual $\delta = \|\boldsymbol{p}^{(t+1)} - \boldsymbol{p}^{(t)}\|_1$
  $t = t + 1$
until $\delta < \epsilon$
return $\max(\boldsymbol{p}^{(t)}, \boldsymbol{s})$ (element-wise maximum)

tend to exhibit similar characteristics. Based on this principle, we propose an iterative ranking algorithm, termed Graph Cohesive Smoothing (GCS). The algorithm is described in Algorithm 1.

Compared with PPR, GCS uses row normalization, which smooths scores across neighboring nodes rather than propagating probability mass through random walks. Furthermore, the final element-wise maximum operation ensures that each node's score is at least as high as its initial score, which prevents highly relevant nodes from being penalized by averaging with lower-scoring neighbors.

Graph attention network ranker The ranking of candidate data objects can be viewed as a node-level prediction task. When the base retriever involves an embedding model, we also train a graph attention network (GAT) (Veličković et al., 2018) to leverage the nodal features and the graph structure defined on the candidate data objects. Specifically, the GAT architecture used in our experiments consists of five GATv2 layers (Brody et al., 2022), followed by two fully connected layers. For each graph node, the GAT takes as input a feature vector formed by concatenating the GCS score, the query embedding, and the data object embedding, and outputs a scalar value that serves as the final ranking score for that node.

3 Experiments

3.1 Experimental Setup

Datasets We evaluate GraphER on three task types, using a number of different datasets for each. The three task types are: (1) *table retrieval*, for which we use the Spider 1.0 (Yu et al., 2018), Bird (Li et al., 2023), and Beaver (Chen et al., 2025a) datasets; (2) *multi-hop QA retrieval*, for which we use the HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022) datasets. For HotpotQA and 2WikiMultihopQA, we randomly sampled 2000 queries from their dev branches and included the corresponding passages of the sampled queries as the retrieval corpus. For MuSiQue, we use the entire `musique_ans_v1.0_dev` branch; (3) *chunked documents retrieval*, for which we use the BEIR-NQ (Thakur et al., 2021) dataset. For this dataset, we only keep documents that contain at least one chunk identified as a relevant object for one or more queries.

Each of these three task types corresponds to a specific type of proximity described in Section 2.1. Specifically, in addition to semantic proximity, we consider structural proximity for table retrieval tasks, conceptual proximity for multi-hop QA retrieval tasks, and contextual proximity for chunked document retrieval tasks.

In each of the datasets considered, a significant portion of the queries have more than one relevant object for retrieval. The detailed statistics of these datasets are listed in Table 1.

Base retriever We use a hybrid search approach as the base retriever, which combines semantic embedding and keyword search. For embedding models, we use **Llama-Embed-Nemotron-8B** (Babakhin et al., 2025) and **Cohere-Embed-4** (Cohere, 2025). For the BEIR-NQ dataset in particular, we use **Llama-Embed-Nemotron-8B**

Retriever	Dataset				
	Spider1_train	Spider1_dev	Spider1_test	Bird_dev	Beaver_dev
			PR@5 (%)		
Llama-Embed-Nemotron-8B+BM25	23.7 (42.5)	37.3 (57.3)	22.2 (41.8)	21.8 (30.8)	14.1 (14.8)
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	**29.7** (44.4)	44.7 (58.8)	26.9 (41.6)	33.0 (40.3)	11.7 (12.4)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	29.4 (**44.9**)	47.6 (61.0)	29.2 (44.6)	29.4 (36.6)	**17.0** (**17.7**)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	*N/A*	**49.7** (**62.5**)	**40.9** (**54.4**)	**36.9** (**43.2**)	15.0 (15.8)
GraphER-GAT's improvement	*N/A*	+12.4 (+5.2)	+18.7 (+12.5)	+15.2 (+12.3)	+1.0 (+1.0)
Cohere-Embed-V4+BM25	46.6 (71.0)	75.9 (89.2)	50.4 (75.3)	74.2 (79.3)	11.2 (12.4)
Cohere-Embed-V4+BM25 + GraphER-PPR	**55.0** (73.1)	**85.2** (**92.4**)	63.7 (78.9)	**86.2** (**88.4**)	10.7 (12.0)
Cohere-Embed-V4+BM25 + GraphER-GCS	52.9 (**73.7**)	80.4 (91.1)	63.0 (80.3)	80.0 (83.9)	12.6 (**13.9**)
Cohere-Embed-V4+BM25 + GraphER-GAT	*N/A*	80.2 (90.8)	**65.7** (**81.8**)	83.4 (86.7)	**13.1** (**13.9**)
GraphER-GAT's improvement	*N/A*	+4.2 (+1.6)	+15.2 (+6.5)	+9.2 (+7.4)	+1.9 (+1.4)
			PR@10 (%)		
Llama-Embed-Nemotron-8B+BM25	38.8 (57.5)	58.7 (73.6)	38.5 (59.6)	70.4 (75.7)	30.6 (31.1)
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	44.3 (59.1)	66.7 (75.7)	46.1 (61.1)	**78.2** (**81.4**)	20.9 (21.5)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	**45.1** (**59.9**)	69.0 (77.4)	50.4 (64.2)	72.8 (77.0)	**31.1** (**31.6**)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	*N/A*	**70.1** (**78.5**)	**59.1** (**71.0**)	77.0 (80.4)	29.6 (30.1)
GraphER-GAT's improvement	*N/A*	+11.4 (+4.8)	+20.6 (+11.4)	+6.6 (+4.6)	−1.0 (−1.0)
Cohere-Embed-V4+BM25	60.3 (80.4)	86.0 (93.8)	65.2 (84.8)	89.3 (91.5)	19.9 (21.1)
Cohere-Embed-V4+BM25 + GraphER-PPR	**67.4** (82.6)	94.2 (97.3)	75.7 (88.2)	95.9 (96.6)	19.4 (20.6)
Cohere-Embed-V4+BM25 + GraphER-GCS	66.5 (**83.0**)	95.2 (97.5)	75.2 (88.4)	95.5 (96.3)	21.4 (22.5)
Cohere-Embed-V4+BM25 + GraphER-GAT	*N/A*	**96.3** (**97.8**)	**77.4** (**89.7**)	**96.6** (**97.3**)	**23.3** (**24.4**)
GraphER-GAT's improvement	*N/A*	+10.3 (+4.1)	+12.2 (+4.9)	+7.3 (+5.8)	+3.4 (+3.3)

Table 2: Retrieval performance on table retrieval benchmarks. We highlight the **best** and second-best results within each dataset-base retriever setting. Numbers in parentheses denote PR@K over all queries; numbers outside parentheses denote PR@K for queries with more than one relevant object.

and **Multilingual-E5-large** (Wang et al., 2024). For each of these embedding models, we combine it with the **BM25** algorithm (Robertson and Zaragoza, 2009; Trotman et al., 2014) by calculating a weighted average of the normalized BM25 score and the cosine similarity between the query and document embeddings. The weights used are 0.3 for BM25 and 0.7 for cosine similarity, respectively.

Our evaluation aims to isolate the contribution of GraphER beyond semantic retrieval. Therefore, we evaluate GraphER on top of strong semantic retrieval backbones rather than more sophisticated semantic rerankers. This allows us to measure the incremental benefit of the proposed methodology beyond the underlying semantic retriever.

In all experiments, the base retriever retrieves 200 candidate objects, or all available objects if there are fewer than 200. A graph is then constructed from these candidate objects, and GraphER's reranking algorithm is applied.

Metrics To evaluate retrieval completeness, we use *perfect* recall@K (PR@K). For a given question, PR@K is 1 if *all* relevant objects are included in the top-K search results, and 0 otherwise. In contrast to regular recall@K, which measures the proportion of relevant objects retrieved, PR@K is particularly important for questions that require information from multiple sources, as failing to retrieve even a single relevant object can result in incorrect answers.

For multi-hop QA retrieval tasks, we also use the top-10 retrieved data objects to generate answers for both GraphER-GAT and the baseline using GPT-5 (OpenAI, 2025). The generated answers are evaluated by prompting GPT-5 to compare them against the ground-truth answers. When available, we report the percentages of queries answered correctly, as judged by GPT-5. The prompt templates for answer generation and evaluation are provided in Appendices A.2 and A.3.

Hyper-Parameter Tuning and Model Training Protocols Both PPR and GCS have one hyper-parameter, namely the damping factor α. For each base retriever setting, specifically Llama-Embed-Nemotron-8B+BM25 and Cohere-Embed-4+BM25, we perform a grid search over α, evaluating candidate values from 0.1 to 0.9 in increments of 0.1 on the `Bird_train` dataset. The best-performing value of α on this dataset is then selected and fixed for the remaining datasets. For the Multilingual-E5-large+BM25 setting, which is

Retriever	Dataset		
	HotpotQA	2WikiMultihopQA	MuSiQue
	PR@5 (%)		
Llama-Embed-Nemotron-8B+BM25	78.5	42.6	24.2
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	72.9	42.1	17.3
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	**78.8**	43.8	25.4
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	78.0	**44.1**	**25.6**
GraphER-GAT's improvement	−0.5	+1.5	+1.4
Cohere-Embed-V4+BM25	74.2	42.0	23.0
Cohere-Embed-V4+BM25 + GraphER-PPR	71.6	**45.7**	22.1
Cohere-Embed-V4+BM25 + GraphER-GCS	74.4	43.3	24.0
Cohere-Embed-V4+BM25 + GraphER-GAT	**74.7**	43.8	**24.5**
GraphER-GAT's improvement	+0.4	+1.8	+1.5
	PR@10 (%)		
Llama-Embed-Nemotron-8B+BM25	88.2 (QA Acc. = 84.2)	49.4 (QA Acc. = 43.1)	35.5 (QA Acc. = 44.1)
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	84.5	49.6	27.7
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	88.7	51.1	36.9
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	**88.9** (QA Acc. = 84.9)	**53.0** (QA Acc. = 45.4)	**37.4** (QA Acc. = 45.2)
GraphER-GAT's improvement	+0.6 (QA Acc. +0.7)	+3.6 (QA Acc. +2.3)	+1.9 (QA Acc. +1.0)
Cohere-Embed-V4+BM25	85.5 (QA Acc. = 82.2)	48.3 (QA Acc. = 43.5)	34.1 (QA Acc. = 42.2)
Cohere-Embed-V4+BM25 + GraphER-PPR	85.8	**52.8**	34.6
Cohere-Embed-V4+BM25 + GraphER-GCS	86.4	50.1	35.5
Cohere-Embed-V4+BM25 + GraphER-GAT	**87.0** (QA Acc. = 83.7)	50.7 (QA Acc. = 44.9)	**36.3** (QA Acc. = 43.5)
GraphER-GAT's improvement	+1.4 (QA Acc. +1.5)	+2.4 (QA Acc. +1.4)	+2.2 (QA Acc. +1.3)

Table 3: Retrieval performance on multi-hop QA benchmarks. Numbers in parentheses denote the percentages of queries answered correctly, as evaluated by GPT-5.

only applied to the BEIR-NQ dataset, we use the same value of α as in Llama-Embed-Nemotron-8B+BM25.

In both PPR and GCS, edge weights of the adjacency matrix $\boldsymbol{A}$ are user-specified hyper-parameters that are predetermined and not subject to tuning. Throughout our experiments, we use the same edge weights within each task type. Specifically, for table retrieval datasets, we assign an edge weight of 1 between two tables if they are linked through a foreign-key relationship, and 0 otherwise. Similarly, for chunked document retrieval, we set the edge weight between two chunks to 1 if they are adjacent chunks from the same un-chunked document, and 0 otherwise. For multi-hop QA retrieval, we compute the edge weight $\boldsymbol{A}_{ij}$ as the number of unique named entities shared by objects i and j, divided by the total number of unique named entities in object j. This normalization helps control the influence of passages that contain a large number of entities by diluting their emphasis on any single entity in proportion to the total number of entities they contain. The extraction of named entities for each data object is done by prompting GPT-4o (OpenAI, 2024). The prompt template is provided in Appendix A.1.

Retriever	Dataset
	BEIR-NQ
	PR@5 (%)
Llama-Embed-Nemotron-8B+BM25	50.9 (77.4)
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	53.3 (76.8)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	52.3 (78.1)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	**54.4** (**78.2**)
GraphER-GAT's improvement	+3.5 (+0.8)
Multilingual-E5-large+BM25	10.7 (25.2)
Multilingual-E5-large+BM25 + GraphER-PPR	7.2 (20.9)
Multilingual-E5-large+BM25 + GraphER-GCS	**11.4** (**26.0**)
GraphER-GCS's improvement	+0.8 (+0.8)
	PR@10 (%)
Llama-Embed-Nemotron-8B+BM25	71.3 (86.6)
Llama-Embed-Nemotron-8B+BM25 + GraphER-PPR	71.3 (86.1)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GCS	**73.4** (**87.9**)
Llama-Embed-Nemotron-8B+BM25 + GraphER-GAT	73.0 (87.4)
GraphER-GAT's improvement	+1.7 (+0.7)
Multilingual-E5-large+BM25	17.0 (32.9)
Multilingual-E5-large+BM25 + GraphER-PPR	13.8 (29.1)
Multilingual-E5-large+BM25 + GraphER-GCS	**19.8** (**34.5**)
GraphER-GCS's improvement	+2.9 (+1.6)

Table 4: Retrieval performance on BEIR-NQ. Numbers in parentheses denote PR@K over all queries; numbers outside parentheses denote PR@K for queries with more than one relevant object.

For the GAT rankers, we first train the neural network on `Bird_train` while monitoring its performance on a sample of `Spider1_train`. The val-

idation results on this sample of `Spider1_train` are used to select the learning rate, L2 regularization coefficient, and number of training epochs. After fixing these hyper-parameters, we retrain the network using the combined set of `Bird_train` and `Spider1_train`, and report their evaluation metrics on the `dev` and `test` splits of these two datasets, as well as on other datasets. The GAT rankers learn edge weights automatically via the attention mechanism, so users do not need to specify edge weights in this case.

3.2 Retrieval Performance

In Tables 2, 3, and 4, we list the PR@5 and PR@10 for the three retrieval tasks. In terms of PR@10, the GraphER-GAT method improves upon the baseline in 14 out of 15 dataset-base retriever settings where it was applied. The GraphER-GCS method improves upon the baseline in all 18 dataset-base retriever settings where it was applied. These improvements are particularly pronounced for queries with more than one relevant object.

The PPR-variant of GraphER outperforms the baseline in 12 out of 18 dataset-base retriever settings. Moreover, in two of these settings, it leads to regressions of more than 7% compared to the baseline. Our examination of its failure cases reveals that this occurs because hub objects are almost always ranked at the top under the PPR algorithm, regardless of their true relevance to the query.

4 Discussions

4.1 Capturing Proximities Beyond Semantics

Our experiments show that the improvements provided by GraphER-GCS and GraphER-GAT are robust to the choice of semantic embedding model used in the base retriever. This can be attributed to the fact that GraphER does not alter the object representations within the semantic space. Instead, it leverages proximity relationships that extend beyond semantic proximity. These additional proximity relationships capture complementary information that is not fully reflected by semantic similarity measures such as cosine or L2 distance between semantic embeddings.

GraphER incorporates these additional proximity relationships by explicitly adding edges between data objects that are proximal to each other. In this work, we identified and experimented with three commonly encountered proximity types beyond semantic proximity: structural proximity in table retrieval, conceptual proximity in multi-hop QA retrieval, and contextual proximity in chunked document retrieval. It is worth noting that these three types are not exhaustive, and the proximity relationships can be adapted to accommodate specific use cases or to inject domain-specific business logic.

4.2 Learning Higher-Order Dependencies

Both PPR and GCS are linear transformations of the initial retrieval scores. To account for higher-order interactions among node features, we additionally employ a graph attention network (GAT) as the ranking component within the GraphER framework. Across the evaluated settings, GraphER-GAT outperforms GraphER-GCS in terms of PR@10 in 13 out of the 15 dataset-base retriever settings in which it is applied.

To determine whether these gains arise from message passing rather than from additional nonlinear transformations, we conduct an ablation study in which the message-passing component of the GAT is disabled while the number of layers, hidden dimensions, and input features remain unchanged. This modification reduces the GAT to a per-node multilayer perceptron (MLP) applied independently to each node, which we refer to as GraphER-MLP. The MLP models are then trained using the same protocol. The comparison of its retrieval performance to GraphER-GCS and GraphER-GAT is reported in Table 5.

GraphER-GAT achieves the best PR@10 on 5 of the 8 datasets, whereas GraphER-MLP achieves the best result on only one. These results suggest that GraphER-GAT benefits from message passing across graph edges, rather than relying solely on per-node feature transformations.

4.3 Retrieval Latency

GraphER involves the online construction of an $n \times n$ adjacency matrix, where n is the number of candidate objects. The additional time and space complexity introduced by GraphER beyond the base retriever can be controlled by adjusting this parameter. When GraphER-GAT is used, the GAT model incurs an additional forward pass.

Our experiments were conducted on a compute node with two AMD EPYC 7J13 processors, 1 TB of RAM, and a single NVIDIA A100-SXM4-40GB GPU. Empirically, GraphER-GCS and GraphER-GAT required 0.49 and 0.55 seconds, respectively,

Reranker	Dataset							
	Spider1_dev	Spider1_test	Bird_dev	Beaver_dev	HotpotQA	2WikiMultihopQA	MuSiQue	BEIR-NQ
				PR@5 (%)				
GraphER-GCS	47.6 (61.0)	29.2 (44.6)	29.4 (36.6)	**17.0** (**17.7**)	78.8	43.8	25.4	52.3 (78.1)
GraphER-GAT	**49.7** (**62.5**)	**40.9** (**54.4**)	**36.9** (**43.2**)	15.0 (15.8)	78.0	**44.1**	**25.6**	**54.4** (**78.2**)
GraphER-MLP	45.2 (60.5)	36.7 (52.7)	35.2 (42.5)	15.0 (15.8)	**78.9**	42.5	21.6	50.3 (76.2)
				PR@10 (%)				
GraphER-GCS	69.0 (77.4)	50.4 (64.2)	72.8 (77.0)	**31.1** (**31.6**)	88.7	51.1	36.9	**73.4** (**87.9**)
GraphER-GAT	**70.1** (**78.5**)	**59.1** (**71.0**)	77.0 (80.4)	29.6 (30.1)	**88.9**	**53.0**	**37.4**	73.0 (87.4)
GraphER-MLP	67.2 (76.0)	55.1 (69.2)	**79.0** (**82.1**)	28.2 (28.7)	87.2	51.1	32.4	71.2 (85.6)

Table 5: Comparison of GraphER-GCS, GraphER-GAT, and GraphER-MLP on evaluation benchmarks. The base retriever is Llama-Embed-Nemotron-8B+BM25.

to rerank 200 candidates, which is negligible compared with downstream LLM inference latency.

4.4 Compatibility with Semantic Rerankers

GraphER is not intended to replace neural rerankers such as cross-encoder models. Instead, it addresses a complementary problem. Cross-encoder rerankers refine semantic relevance by jointly encoding each query-document pair, whereas GraphER exploits relationships among candidate objects that are not captured by the point-wise semantic scoring. Because GraphER is computationally lightweight and operates independently of the underlying semantic scorer, it can be combined with cross-encoder rerankers in a multi-stage retrieval pipeline. Evaluating such hybrid pipelines is an important direction for future work.

4.5 Limitations and Future Work

GraphER relies on the availability and quality of metadata that encode proximity relationships among data objects. When such relationships are noisy, incomplete, or weakly aligned with the information need, graph-based reranking may provide limited benefits to retrieval performance. Furthermore, although the GraphER framework is designed to support arbitrary forms of proximity, our experiments cover only three representative types: structural, conceptual, and contextual proximity. Additional studies are needed to evaluate the effectiveness of other enrichment strategies across different domains. Finally, like other reranking approaches, GraphER operates only on the candidate set returned by the base retriever and therefore cannot recover relevant objects that are absent from the initial retrieval results. Consequently, its effectiveness depends on the base retriever achieving sufficient recall during candidate retrieval.

Several directions remain for future work. First, although we demonstrate that GraphER can be combined with graph neural networks, exploring more expressive graph learning architectures and alternative graph construction strategies may further improve retrieval quality. Second, while this work evaluates GraphER as a standalone reranking component to isolate its contribution, it is also compatible with downstream semantic rerankers, such as cross-encoder models. An empirical evaluation of such hybrid pipelines would help quantify the complementary benefits of improving candidate completeness and fine-grained semantic relevance. Finally, GraphER currently uses user-defined or extracted proximity relationships. Learning proximity relationships automatically from retrieval or user interaction data may enable the framework to adapt more effectively to different domains and applications.

5 Conclusion

We presented GraphER, a graph-based enrichment and reranking framework for improving retrieval completeness in RAG systems. GraphER incorporates proximity relationships beyond semantic similarity through lightweight metadata enrichment and query-time graph construction, without requiring additional graph infrastructure. Across table retrieval, multi-hop QA retrieval, and chunked-document retrieval tasks, GraphER consistently improves retrieval completeness over strong semantic retriever baselines. We also introduced Graph Cohesive Smoothing, which provides more robust gains than Personalized PageRank, and showed that a graph attention network can further improve

performance by modeling higher-order interactions among candidate objects. These results demonstrate that leveraging organizational structure beyond semantic similarity is an effective and practical approach for improving retrieval in RAG systems.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, volume 2024, pages 9112–9141.

Yauhen Babakhin, Radek Osmulski, Ronay Ak, Gabriel Moreira, Mengyao Xu, Benedikt Schifferer, Bo Liu, and Even Oldridge. 2025. Llama-embed-nemotron-8b: A universal text embedding model for multilingual and cross-lingual tasks. *Preprint*, arXiv:2511.07025.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Mike Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2025a. BEAVER: An enterprise benchmark for text-to-SQL. In *The 4th Table Representation Learning Workshop at ACL 2025*.

Peter Baile Chen, Yi Zhang, Mike Cafarella, and Dan Roth. 2025b. Can we retrieve everything all at once? ARM: An alignment-oriented LLM-based retrieval method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30298–30317.

Cohere. 2025. Introducing embed 4: Multimodal search for business. https://cohere.com/blog/embed-4.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 21497–21515. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Can M. Le and Tianxi Li. 2022. Linear regression and its inference on noisy network-linked data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(5):1851–1885.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*, volume 36, pages 42330–42357. Curran Associates, Inc.

Tianxi Li, Elizaveta Levina, and Ji Zhu. 2019. Prediction models for network-linked data. *The Annals of Applied Statistics*, 13(1):132 – 164.

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, volume 2024, pages 14400–14423.

OpenAI. 2024. Gpt-4o system card. https://openai.com/index/gpt-4o-system-card/.

OpenAI. 2025. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pages 32628–32649.

Zhivar Sourati, Zheng Wang, Marianne Menglin Liu, Yazhe Hu, Mengqing Guo, Sujeeth Bharadwaj, Kyu Han, Tao Sheng, Sujith Ravi, Morteza Dehghani, and Dan Roth. 2026. Lad-rag: Layout-aware dynamic rag for visually-rich document understanding. *Preprint*, arXiv:2510.07233.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pages 3868–3898.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, page 58–65, New York, NY, USA. Association for Computing Machinery.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. 2026. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *The Fourteenth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Richard James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-Tau Yih. 2023. Retrieval-augmented multimodal language modeling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39755–39769. PMLR.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Appendix

A.1 Prompt Template for Named Entity Extraction

```
Identify the key entities (people, organizations, locations, dates, etc.) from
the following paragraph.
Return the result as a valid Python list of strings, with no explanations,
only the list.

Example:
Text:
"Barack Obama was born in Honolulu, Hawaii, and served as the 44th President
of the United States."
Output:
["Barack Obama", "Honolulu", "Hawaii", "United States", "44th President"]

Text:
{content}
Output:
```

A.2 Prompt Template for Retrieval-Augmented Answer Generation

```
You are an AI assistant. Use ONLY the information provided in the context
below to answer the user's question.

If the context does not contain enough information to answer the question, say:
"I don't have enough information in the provided context to answer this."

<context>
{retrieved_objects}
</context>

Question:
{query}

Answer:
```

A.3 Prompt Template for RAG Answer Evaluation

```
You are an impartial evaluator tasked with judging the correctness of an
answer generated by an LLM.

=== Task ===
Evaluate whether the Ground Truth Answer is present in the Generated Answer.

Your goal is to determine if the essential meaning and key facts of the Ground
Truth Answer are correctly captured in the Generated Answer.

=== Instructions ===
1. Carefully compare the Generated Answer with the Ground Truth Answer.
2. Focus on substance and meaning, not exact wording.
3. Treat paraphrases, synonyms, or logically equivalent statements as correct
if they preserve the same factual meaning.
4. Ignore irrelevant extra information unless it directly contradicts the
Ground Truth.
5. If any vital fact from the Ground Truth is missing or incorrect, the
decision should be "no".
6. If all essential information from the Ground Truth is present and correct,
the decision should be "yes".

=== Input Data ===
- Question: {query}
- Generated Answer: {generated_answer}
- Ground Truth Answer: {ground_truth_answer}

=== Output Format ===
Provide your final evaluation strictly in the following format:

Explanation: <Brief explanation of how you made the decision>
Decision: <yes or no>

Proceed carefully and base your judgment strictly on the criteria above.
```

LongDA: Benchmarking LLM Agents for Long-Document Data Analysis

Yiyang Li[1], Zheyuan Zhang[1], Tianyi Ma[1], Zehong Wang[1], Keerthiram Murugesan[2], Chuxu Zhang[3], Yanfang Ye[1†]
[1]University of Notre Dame, [2]IBM Research, [3]University of Connecticut
[†]Corresponding Author {yli62, yye7}@nd.edu

huggingface.co/datasets/Yiyang-Ian-Li/LongDA

Abstract

We introduce LongDA, a data analysis benchmark for evaluating LLM-based agents under documentation-intensive analytical workflows. In contrast to existing benchmarks that assume well-specified schemas and inputs, LongDA targets real-world settings in which navigating long documentation and complex data is the primary bottleneck. To this end, we manually curate raw data files, long and heterogeneous documentation, and expert-written publications from 17 publicly available U.S. national surveys, from which we extract 505 analytical queries grounded in real analytical practice. Solving these queries requires agents to first retrieve and integrate key information from multiple unstructured documents before performing multi-step computations and writing executable code, which remains challenging for existing data analysis agents. To support the systematic evaluation under this setting, we develop LongTA, a tool-augmented agent framework that enables document access, retrieval, and code execution, and evaluate a range of proprietary and open-source models. Our experiments reveal substantial performance gaps even among state-of-the-art models, highlighting the challenges researchers should consider before applying LLM agents for decision support in real-world, high-stakes analytical settings.

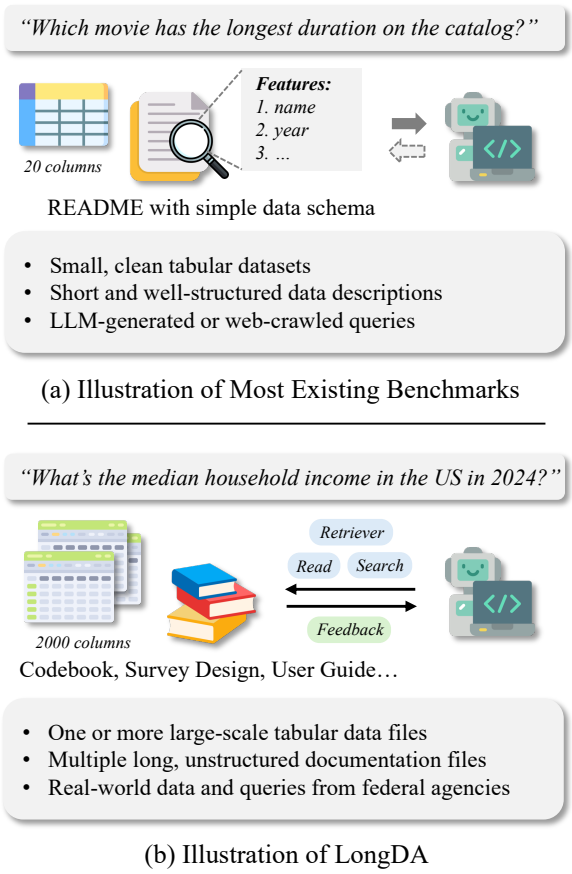


Figure 1: Comparison of existing data analysis benchmarks and LongDA.

1 Introduction

Recent advances in large language models (LLMs) (Chang et al., 2024; Ye et al., 2025) have enabled the development of LLM agents capable of assisting with data analysis tasks, including writing Python code (Hollmann et al., 2023; Guo et al., 2024; Sun et al., 2025a), executing statistical computations (Hong et al., 2025), and generating analytical reports (Zhang et al., 2025b). These agents show strong potential to reduce the manual effort required in data-driven research and decision-making, particularly in areas such as social science, economics, and public policy, where analytical workloads commonly involve complex and large-scale datasets (Sun et al., 2025b; Rahman et al., 2025; Wang et al., 2025). Accordingly, benchmarking the data analysis capabilities of LLM agents has become an important and active research direction (Zhang et al., 2024b; Hu et al., 2025).

Despite these advances, real-world data analysis remains fundamentally challenging. In practice, analytical workflows rarely begin with clean, well-documented datasets. Instead, analysts typically first navigate lengthy and heterogeneous documentation to understand dataset structures,

The Proceedings of Workshop on Document Intelligence and Understanding, pages 112–137
October 24, 2026 ©2026 Association for Computational Linguistics

Table 1: Comparison of related benchmarks. Context denotes the approximate number of context tokens per query. Syn., Hum., and Web denote synthetic, human, and web sources, respectively. ✓ indicates interactive environment.

Benchmark	Size	Data source	Query source	Interaction	Context
DS-1000 (Lai et al., 2023)	1,000	Web	Web	✗	900
Text2Analysis (He et al., 2024)	2,249	Hum.	Syn. / Hum.	✗	N/A
DSCodeBench (Ouyang et al., 2025)	1,000	Web	Syn. / Hum.	✗	640
DataSciBench (Zhang et al., 2025a)	222	N/A	Syn. / Hum. / Web	✗	200
InfiAgent-DABench (Hu et al., 2024)	257	Web	Syn.	✓	160
DSEval (Zhang et al., 2024b)	825	Web	Syn. / Hum.	✓	1,300
DACO (Wu et al., 2024)	1,942	Web	Syn. / Hum.	✓	890
DA-Code (Huang et al., 2024)	500	Web	Hum.	✓	370
DSBench (Jing et al., 2024)	466	Web	Web	✓	1,060
DABstep (Egg et al., 2025)	450	Company	Company	✓	5,200
LongDA (ours)	**505**	**Gov. agencies**	**Expert pubs.**	✓	**263,500**

variable definitions, and data preprocessing procedures (Koesten et al., 2021). This information is commonly scattered across multiple unstructured documents, such as codebooks, technical reports, and methodological appendices, which can span hundreds of thousands of tokens (Ale et al., 2025). Analysts generally need to complete this information-gathering stage before writing code to extract relevant variables and compute the results. For both humans and LLM agents, this documentation navigation stage is frequently the dominant bottleneck in real-world data analysis.

However, most existing data analysis benchmarks do not adequately capture this reality. As illustrated in Figure 1 and summarized in Table 1, prior benchmarks primarily evaluate data analysis under well-specified inputs, implicitly assuming that relevant variables, schemas, and data files are already known. Consequently, they overlook a critical component of real-world analysis: the ability to retrieve, interpret, and integrate information from long, unstructured documentation. Moreover, many benchmarks rely on LLM-generated or web-crawled queries, which may not reflect the analytical questions practitioners actually seek to answer. A more detailed discussion of related work is provided in Appendix A.

Motivated by these limitations, we introduce **LongDA**, the first data analysis benchmark that evaluates LLM agents under documentation-intensive analytical workflows. LongDA is constructed from federal agency–released data files, extensive documentation with an average length of *263k tokens*, and expert-written analytical publications. From these publications, we extract *505 real analytical queries* whose reported results are reproducible using publicly released data. To solve these queries, agents need to identify relevant variables, assign weights, and define populations before performing multi-step computations to obtain final answers. This remains a fundamental challenge for existing data analysis agents that are not designed to operate in this specific setting.

To enable systematic and controlled study of this challenge, we further develop **LongTA**, a lightweight tool-augmented agent framework that enables document access, retrieval, and code execution, serving as an evaluation scaffold and baseline for LongDA. Using LongTA, we conduct extensive experiments on a range of proprietary and open-source models, revealing substantial performance gaps and underscoring the limitations of current LLM agents in real-world analytical workflows.

Our contributions are summarized as follows:

- We introduce **LongDA**, a benchmark designed to evaluate data analysis agents on realistic datasets with long, complex documentation.
- We develop **LongTA**, a lightweight tool-augmented baseline for controlled document access and code execution for LongDA.
- We extensively evaluate state-of-the-art models, revealing substantial performance gaps and identifying information retrieval and tool-use strategy as primary bottlenecks.

2 LongDA Benchmark

2.1 Data Collection

To construct a realistic and practically meaningful data analysis benchmark, we collect seventeen U.S. national surveys spanning a broad range

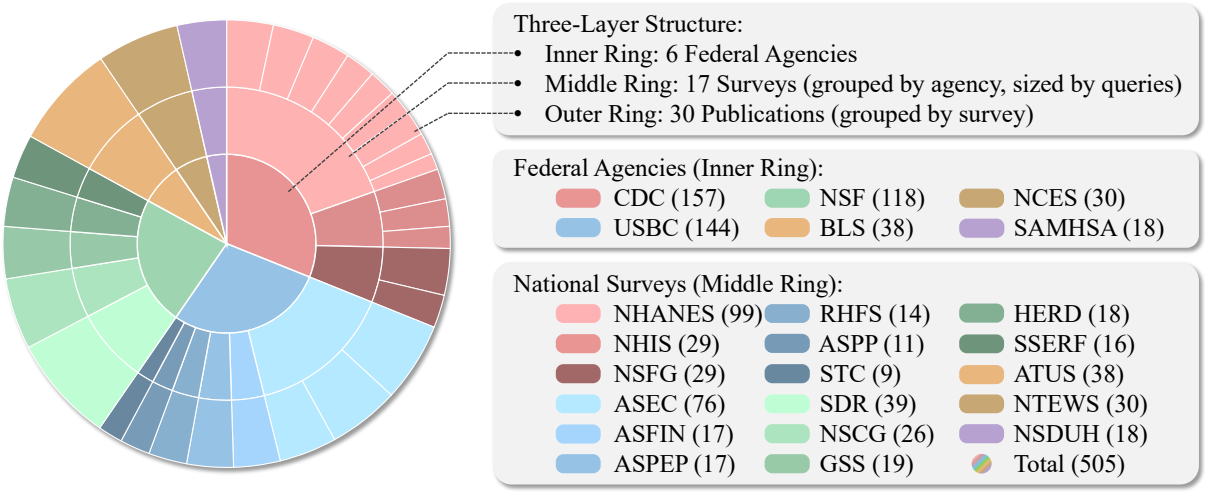


Figure 2: Hierarchical composition of the LongDA benchmark. Numbers in parentheses indicate the number of queries associated with each entity, with a total of 505 queries. Publications share the same color as their parent survey, and surveys belonging to the same federal agency are rendered in the same color family.

of domains, including public health, labor and employment, education, and the scientific workforce. These surveys are meticulously designed, conducted, and released by six U.S. federal agencies. They are routinely utilized by governments, researchers, and policymakers to provide valuable insights that inform public policy, social programs, and scientific research. For the complete list of survey abbreviations, official names, and source links, refer to Appendix B.1.

For each survey, we obtain three types of official resources from the corresponding agency websites: *(1) raw data files*, *(2) survey documentation*, and *(3) analytical publications* produced by domain experts. The documentation typically includes codebooks, methodological descriptions, and survey design reports, which are written for human analysts and commonly span long, unstructured texts. To ensure consistency and reproducibility, we collect data, documentation, and publications from a fixed survey year for each survey and organize all resources into a unified directory structure without modifying their original content. For the complete mapping between surveys and their associated publications, refer to Appendix B.2.

While all three resource types are collected during dataset construction, they play different roles in the benchmark: the publications are used exclusively for query extraction and are not accessible to agents during evaluation, while raw data files and the associated documentation are exposed to the agents when solving tasks. This design ensures that benchmark queries are grounded in real analytical practice while preventing agents from directly retrieving answers from expert-written reports.

2.2 Query and Ground Truth Extraction

After data collection, we extract queries and ground truths from the official analytical publications. Each query in LongDA consists of three components: *(1) a question*, *(2) an answer structure*, and *(3) additional information*. The *question* specifies the analytical goal based on the given survey. The *answer structure* defines the expected format of the response, which is either a single numerical value or a fixed-length list with predefined semantic elements (e.g., [*men*, *women*], [*under18*, *age18_64*, *age65plus*]). The *additional information* provides auxiliary context necessary for correctly interpreting the question. Together, these three components are the minimal information required to make each query both unambiguous for automatic evaluation and faithful to real-world analytical practice.

For each publication, we first verify that the reported results can be reproduced using the publicly released survey data, which is typically stated explicitly in the publication. We then construct queries based on reported numerical findings, together with their corresponding answer structures. When specifically noted, we extract relevant definitions or clarifications provided in the publication and include them as additional information to resolve potential linguistic ambiguities. This design ensures that queries are grounded in real-world analytical practice while being technically solvable, and encourages agents to perform genuine reasoning over data and documentation.

2.3 Benchmark Statistics

LongDA contains 505 queries in total. Figure 2 illustrates the hierarchical composition of LongDA, organized across three levels: federal agencies, surveys, and publications. In total, LongDA covers 6 U.S. federal agencies, 17 national surveys, and 30 official publications, reflecting the broad coverage of real-world data analysis scenarios.

A key characteristic of LongDA is the complexity of its accompanying documentation: with an average length of 263k tokens and a maximum length exceeding 735k tokens, LongDA possesses substantially longer contexts than existing benchmarks. The distribution of documentation lengths is shown in Figure 6. LongDA queries also exhibit structured answer formats: 221 queries (44%) require a single numerical answer, while the remaining 284 queries (56%) involve fixed-length lists with predefined semantic elements.

2.4 Evaluation Protocol

To control evaluation cost and better reflect realistic analytical workflows, queries derived from the same publication are presented to the agent in *blocks*, where each block contains all queries associated with a single publication (see Appendix C.2). Queries within a block typically revolve around shared themes or variables, allowing the agent to reuse intermediate understanding across related questions. Note that during evaluation, agents are restricted to accessing the raw data files and survey documentation corresponding to the given survey.

We evaluate agent performance along two primary dimensions: *effectiveness* and *efficiency*. Effectiveness is measured by **coverage rate** and **match rate**. Efficiency is characterized by total token consumption, total runtime over the full benchmark, and the average number of interaction steps per block. We describe each metric in detail below.

To demonstrate the agents' ability to follow task specifications and produce reliable, structured output, we compute the coverage rate, defined as the proportion of queries for which the answer syntactically conforms to the specified answer structure. Formally, given a query set $\mathcal{Q}$ and an agent's output set $\hat{\mathcal{A}}$, the coverage rate is defined as

$$\text{Coverage} = \frac{|\{\, q \in \mathcal{Q} \mid \hat{a}_q \text{ is structurally valid} \,\}|}{|\mathcal{Q}|},$$

where $\hat{a}_q$ denotes the agent's answer to query q.

The match rate measures numerical correctness under a predefined tolerance. For each query q with ground truth answer a_q and agent prediction $\hat{a}_q$, we define a tolerance parameter ϵ and set tolerance as

$$\tau(a_q) = \max(\epsilon \cdot |a_q|, 1),$$

ensuring that small-magnitude targets are not overly penalized. The prediction is considered correct if $|\hat{a}_q - a_q| \leq \tau(a_q)$ (applied element-wise for list answers). The match rate is the proportion of queries for which this condition holds.

3 LongTA Framework

To enable systematic study of LLM-based data analysis agents under documentation-intensive settings, we introduce LongTA, a lightweight tool-augmented agent framework that serves as an evaluation scaffold and baseline for LongDA. LongTA consists of a ReAct-style (Yao et al., 2022) coding agent and a set of specialized tools for interacting with raw data files and documentation.

3.1 ReAct Coding Agent

The coding agent follows an iterative ReAct loop in which the LLM interleaves tool calls for document navigation and Python code execution for numerical computation. This design matches realistic analysis workflows, where an analyst repeatedly consults documentation to identify variables and then runs code to examine the statistical results.

Block-level prompt. LongDA presents queries in blocks. For each block, we construct a prompt that contains: (i) global instructions, (ii) a list of questions, (iii) the expected answer structure for each question, (iv) additional information for each question, and (v) the list of available data files and documentation files for this survey. The prompt template is provided in Appendix C.1.

Iterative reasoning-and-action loop. Given the prompt, the agent runs for at most a fixed step budget. At each step, the agent generates a Python snippet to call tools or run analysis code. The loop terminates when the agent signals task completion or when the maximum step budget is reached.

Execution environment and safety. To support common survey analysis workloads while keeping the execution controlled, we configure the executor with an allowlist of commonly-used Python libraries (e.g., `pandas`, `numpy`, `scipy`, `statsmodels`) and standard library modules (e.g., `json`, `re`), balancing practicality and containment.

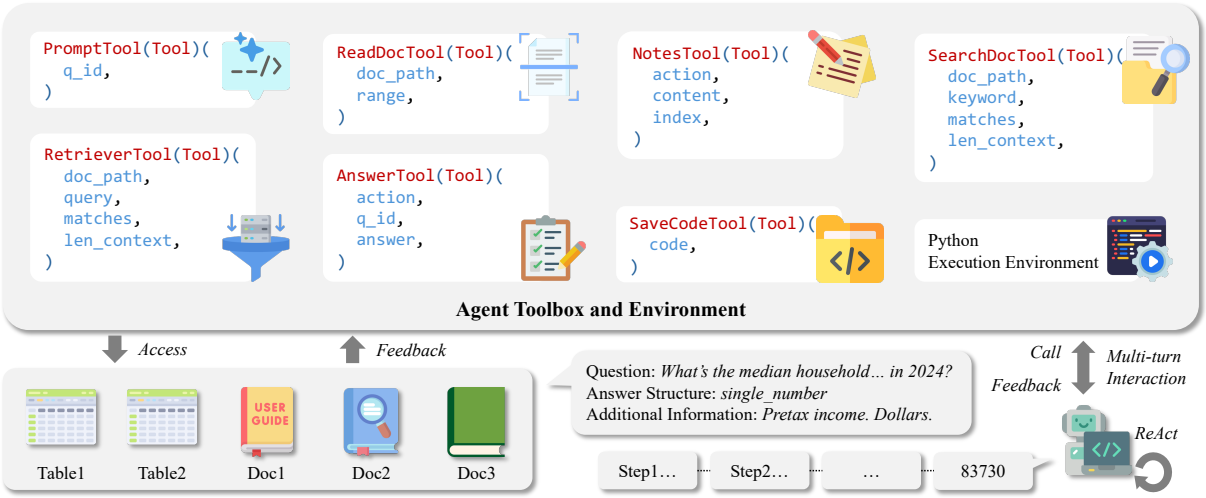


Figure 3: Illustration of LongTA. The agent solves queries through a ReAct-style multi-turn interaction, coordinating multiple tools for long-document navigation and code execution over heterogeneous data sources.

3.2 Tool Design

LongDA tasks are bottlenecked by documentation navigation: the agent must discover variable definitions, survey weights, and file schemas from long, heterogeneous documents before writing accurate analysis code. Accordingly, we design a small set of tools for document interaction, context management, and answer recording, built on top of a unified internal representation of documentation.

Unified document representation. All documentation files are exposed through a unified "document unit" abstraction. Given a document file, we parse it into a sequence of units depending on file type: PDFs are split into pages; text-like formats (TXT/PY/JSON) are split into lines; tabular formats (CSV/XLSX) are split into rows. Each unit has an identifier (e.g., "Page 3", "Line 120", "Row 15") along with the normalized text content. This design supports both keyword search and retrieval over large files while keeping tool outputs compact.

Document access and retrieval. To navigate long, heterogeneous documentation, we provide tools that support both targeted inspection and scalable retrieval. **(1)** **`read_doc`** enables direct inspection by returning either a short preview (when no range is specified) or user-selected units via one-based ranges (e.g., `1-3` or `2,5,7`), which is useful for reading codebook sections, methodological descriptions, or specific line/page ranges. Complementing this, **(2)** **`search_doc`** performs exact keyword search across units and returns snippets with a configurable context window, allowing the agent to quickly locate candidate variable names, weighting terms, or table captions. Since exact matching may be insufficient when the agent does not know the precise phrasing used in documentation, we further implement **(3)** **`retriever`**, a lightweight lexical retriever based on BM25 (Robertson et al., 2009): it splits each unit into overlapping chunks with configurable chunk size and overlap, builds index over all chunks, and returns the top-k relevant chunks.

Prompt revisiting and note-taking. To mitigate context loss in long interactions, we provide **`prompt`** for revisiting task specifications and **`notes`** for externalizing discovered information such as variable definitions and weight usage, which can be reused across queries within a block.

Answer recording and auditing. Automatic evaluation in LongDA requires answers to be structurally valid. We therefore enforce that final outputs are emitted through an **`answer`** tool. The tool validates that each answer is either a single number or a list of numbers, and stores the raw serialized value. This design separates reasoning traces from final outputs, enabling reliable automatic evaluation and post-hoc analysis. Finally, agents can use **`save_code`** to store the Python code used for a block, providing an audit trail that supports reproducibility checks and qualitative error analysis.

4 Experiments

4.1 Experiment Settings

We evaluate LongDA on a diverse set of state-of-the-art proprietary and open-source LLMs, including GPT-5 (OpenAI, 2025a), DeepSeek-V3.2 (Liu et al., 2025), Qwen3 series (Yang et al., 2025),

Table 2: Model performance comparison. Best results for each category are highlighted in **bold**. In (M), Out (M), and Total (M) denote the numbers of input tokens, output tokens, and total tokens (in millions), respectively.

Model	Coverage (%)	Match (%)	Avg. Steps	In (M)	Out (M)	Total (M)	Time (h)
GPT-5 (High)	**94.65**	68.91	6.20	7.38	0.87	8.25	5.74
GPT-5	91.09	**69.16**	5.50	5.75	0.65	6.40	3.58
GPT-5-mini	69.50	40.81	8.63	10.10	0.56	10.66	3.42
DeepSeek-V3.2	**67.33**	**53.00**	66.30	68.50	0.40	68.91	5.22
DeepSeek-V3.2-Thinking	61.58	47.82	37.60	35.55	0.83	36.38	9.08
Kimi-K2-0905	50.50	28.27	36.13	37.78	0.43	38.21	5.22
Qwen3-235B-A22B-Instruct	48.91	27.17	42.30	61.20	0.53	61.73	10.38
Qwen3-Coder-480B-A35B-Instruct	51.49	23.44	53.27	75.13	0.36	75.49	5.08
Qwen3-Next-80B-A3B-Instruct	49.50	21.85	21.50	22.00	0.37	22.36	1.64
GLM-4.7	27.13	19.18	81.17	119.42	0.59	120.01	8.42
GPT-OSS-120B	46.53	12.15	18.60	13.98	0.68	14.66	3.67

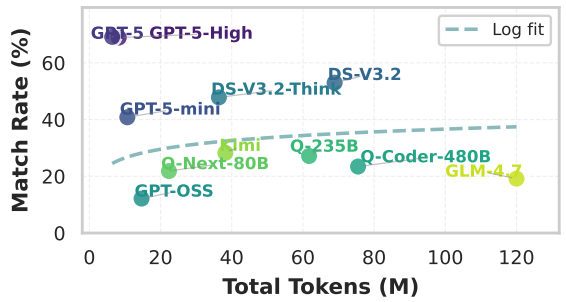


Figure 4: Match rate vs. total token consumption.

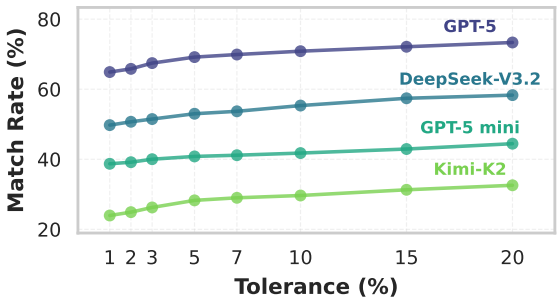


Figure 5: Match rate sensitivity to tolerance threshold.

Kimi-K2 (Team et al., 2025), GLM-4.7 (Z.ai, 2025), and GPT-OSS-120B (OpenAI, 2025b). All models are evaluated over LongTA with a budget of 100 steps and a tolerance threshold of 5% when computing the match rate.

4.2 Main Results

Overall Performance and Efficiency. Table 2 reveals substantial differences in both effectiveness and efficiency across models. The GPT-5 family achieves the highest coverage and match rates while requiring significantly fewer reasoning steps, fewer total tokens, and less wall-clock time. We attribute the strong performance of GPT-5 to three complementary factors. First, GPT-5 appears to possess richer prior knowledge of survey conventions and variable nomenclature, which often allows it to rapidly infer or approximate variable abbreviations before consulting documentation. This behavior is also reflected in our case study (Appendix D), where GPT-5 identifies many relevant keywords within the first few steps. Second, GPT-5 demonstrates substantially stronger capability in strategically leveraging the `search_doc` tool for information retrieval, as further analyzed in Section 4.3. Third, GPT-5 tends to execute larger code blocks per step (combining multiple keyword searches and sub-tasks within a single execution) rather than issuing one fine-grained tool call step by step. This execution style increases the effective information density of the context and avoids repeatedly injecting system prompts and intermediate states, thereby achieving both higher efficiency and stronger performance. Notably, even GPT-5 (High) achieves only 68.91% match rate, indicating substantial remaining headroom.

Among open-source models, DeepSeek-V3.2 exhibits comparatively stable generation behavior. However, we also observe that other models (e.g., Kimi and GLM series) sometimes fall into repetitive generation patterns. Such repetition substantially inflates output token usage and runtime, contributing to lower overall efficiency.

Scaling Behavior. Figure 4 illustrates the relationship between match rate and total token consumption. To analyze scaling behavior, we fit a logarithmic curve to the data. Note that GPT-5 and GPT-5 (High) are excluded from the fit as outliers to avoid distortion of the overall trend. Performance improves with increased token budget, though the gains exhibit diminishing returns, reflecting the intrinsic difficulty of LongDA.

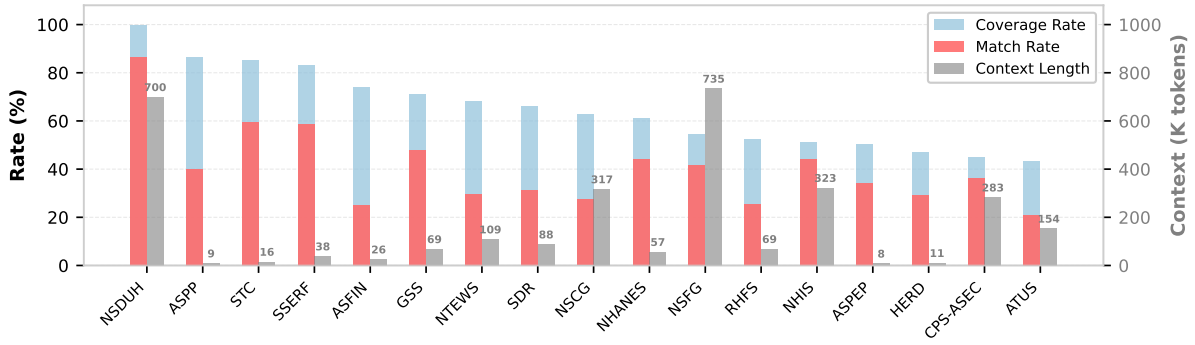


Figure 6: Survey-level performance averaged across all evaluated models.

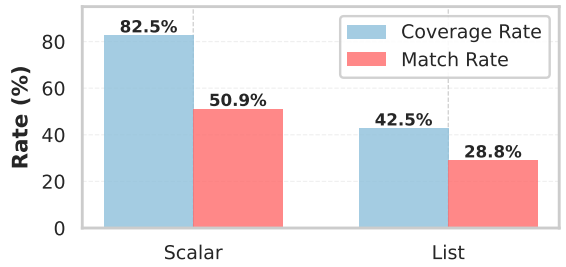


Figure 7: Performance by answer type (scalar vs. list).

Tolerance Sensitivity. Figure 5 studies sensitivity to the tolerance threshold. While match rate increases monotonically with tolerance, the relative ranking of models and the overall performance gaps remain stable, indicating that the evaluation is not overly sensitive to the exact tolerance choice.

Survey-Level Analysis. Figure 6 reports survey-level performance aggregated across all evaluated models, showing a mild negative correlation between documentation length and model performance. NSDUH is a notable exception: despite extremely long contexts, it achieves comparatively strong performance. This is likely because its publicly available microdata already includes many pre-coded variables derived from multiple raw fields, allowing models to answer queries once relevant variables are identified without requiring complex downstream computation.

Scalar vs. List Queries. Figure 7 compares scalar and list-structured queries. List queries are consistently more difficult than scalar ones, due to both increased computational complexity and the requirement to follow specific answer structures.

Effect of Explicit Reasoning. Comparing GPT-5 and DeepSeek-V3.2 with and without explicit reasoning variants, we observe that explicit reasoning does not provide clear performance improvements on LongDA. This suggests that the core difficulty of LongDA lies not in logical inference, but in discovering and extracting sufficient information from long, complex documentation. Moreover, reasoning-oriented models exhibit higher tool usage, longer outputs, and increased runtime (Table 2 and 3). We hypothesize that cautious reasoning induces excessive tool calls without substantially improving document understanding, introducing overhead with limited accuracy gains.

4.3 Tool Usage Analysis

Table 3 reports aggregate tool usage statistics. Additionally, to visualize tool-use dynamics, we extract full tool-call traces from agent execution logs and represent each block's trajectory as a sequence of tool invocations. Each trajectory is normalized to a $[0, 1]$ progress scale based on its maximum step count and discretized into 20 bins. For each bin, we compute the empirical distribution of tool usage across all trajectories of a model, producing the heatmaps in Figure 8.

Behavioral Analysis. GPT-5 variants strongly favor the `search` tool over the `retriever` and consistently invoke the `prompt` tool at the beginning of each block. Open-source models display more exploratory tool usage patterns with heavier reliance on retrieval and iterative refinement. These results indicate that successful performance on LongDA depends not only on language modeling and coding ability, but also on effective tool-use strategies for navigating long, complex documentation.

We further conduct a tool ablation study, as reported in Table 4. According to the results, removing `search_doc` leads to a substantial degradation for GPT-5, indicating that GPT-5 is particularly effective at leveraging `search_doc` for agentic problem solving in LongDA. In contrast, removing `retriever` causes only marginal fluctuations for DeepSeek-V3.2 and Qwen3-Coder. This sug-

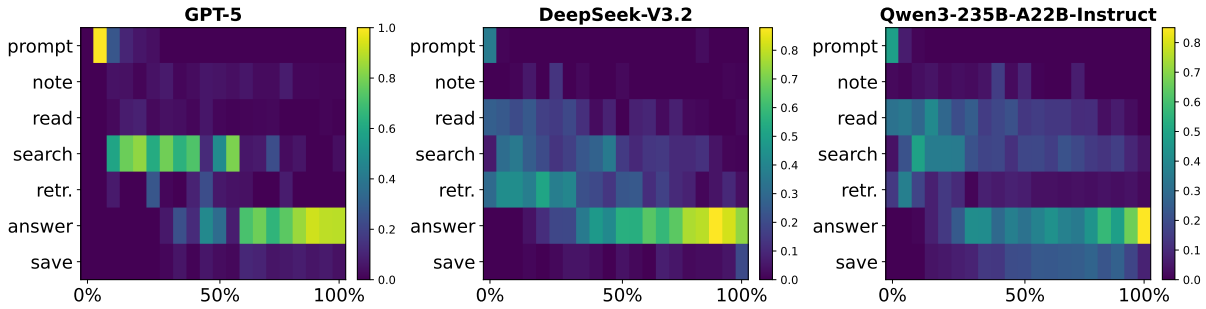


Figure 8: Normalized tool usage trajectories across three agent models. Each heatmap shows the probability distribution of tool calls over the normalized problem-solving progress (0–100%).

Table 3: Tool usage count by models.

Model	Prompt	Read	Search	Retr.	Note	Answer	Save
GPT-5	30	46	1,704	114	79	704	79
GPT-5 (High)	30	121	2,392	117	77	754	74
DeepSeek-V3.2	31	146	207	269	38	526	39
DeepSeek-V3.2-Thinking	30	122	206	393	39	893	95
Qwen3-235B-A22B-Instruct	30	309	371	146	57	596	293
Qwen3-Coder-480B-A35B-Instruct	84	187	231	15	17	528	18

Table 4: Ablation study on tool availability.

Model	Setting	Coverage (%)	Match (%)	Steps	In (M)	Out (M)	Time (h)
GPT-5	Full tools	**91.09**	**69.16**	5.50	5.75	0.65	3.58
GPT-5	w/o `search_doc`	80.20	58.30	6.03	6.09	0.65	2.84
DeepSeek-V3.2	Full tools	**67.33**	53.00	66.30	68.50	0.40	5.22
DeepSeek-V3.2	w/o `retriever`	66.93	**54.03**	65.07	67.04	0.40	5.36
Qwen3-Coder-480B-A35B-Instruct	Full tools	51.49	**23.44**	53.27	75.13	0.36	5.08
Qwen3-Coder-480B-A35B-Instruct	w/o `retriever`	**53.27**	21.52	48.00	79.48	0.40	4.27

gests that the performance gap between GPT-5 variants and other models cannot be explained solely by a preference for the `search_doc` tool. Instead, LongDA primarily challenges models' ability to strategically extract relevant evidence from long documentation and integrate retrieved information into coherent multi-step analytical workflows.

4.4 Summary of Findings

Overall, our experiments lead to three key observations. **First**, current state-of-the-art models remain far from solving documentation-intensive data analysis reliably. Even the strongest model, GPT-5 (High), achieves only 68.91% match rate, indicating substantial headroom. **Second**, success on LongDA is driven primarily by information retrieval and tool-use strategy rather than by pure reasoning capability. Models that are better at retrieving relevant evidence from long documentation and orchestrating tool calls consistently outperform others. **Third**, longer contexts and more complex answer structures amplify difficulty, reflecting the challenges of information retrieval, instruction following, and multi-step computation.

5 Conclusion

We introduce LongDA, the first benchmark for evaluating data analysis agents under documentation-intensive analytical workflows. By grounding tasks in large-scale government datasets and long, unstructured documentation, LongDA exposes fundamental challenges to current LLM-based agents, particularly in information retrieval and document navigation. Our extensive experiments show that even state-of-the-art models struggle with documentation-intensive analytical pipelines, where performance depends critically on effective evidence retrieval and strategic tool use under long-context constraints. We hope LongDA will serve as a challenging and practical testbed for advancing the development of reliable data analysis agents.

Limitations

LongDA has several limitations that should be considered when interpreting results and when using the benchmark.

Scope and representativeness. LongDA is constructed from 17 U.S. national surveys and associated publications, which provide breadth across agencies and domains but still represent a small slice of the full landscape of real-world data analysis. The benchmark may under-represent domains with substantially different data modalities (e.g., proprietary enterprise data) or workflows (e.g., streaming analytics, database-centric ETL).

Evaluation focuses on final numbers. Our primary metrics (coverage and match) evaluate whether agents produce structurally valid outputs and numerically correct answers within a tolerance. This design supports scalable automatic evaluation, but it does not directly measure intermediate analytical quality (e.g., whether the agent's code follows best practices, produces confidence intervals, or correctly handles edge cases) nor does it capture the quality of explanations and reporting narratives.

Survey design complexity is only partially exercised. While LongDA tasks require reading documentation and applying sample weights, many survey analyses in practice also involve variance estimation and design-based uncertainty quantification (e.g., strata/PSU adjustments, replicate weights, finite population corrections, and survey-specific variance estimators). LongDA currently does not include queries that explicitly evaluate variance estimation (e.g., standard errors or confidence intervals). As a result, strong performance on LongDA does not imply full competence in general complex-survey inference beyond point estimation.

Answer memorization risk. Although LongDA is designed such that agents are expected to derive answers through document retrieval and multi-step computation, this requirement is not strictly enforced. Consequently, a model may occasionally produce the correct answer from information memorized during pretraining rather than by actually performing the intended analysis. Future evaluations could mitigate this risk by verifying intermediate computation traces and required tool calls before accepting the final answer.

Ethical Considerations

LongDA is constructed from publicly released U.S. government survey data and documentation. The benchmark does not include personally identifiable information (PII) and does not require any individual-level inference. Given the high-stakes nature of real-world data analysis, outputs produced by LLM agents should not be used for automated decision-making without human verification. We report token usage and runtime to make evaluation cost explicit and to encourage more efficient and reproducible methods.

Acknowledgements

This work was partially supported by the NSF under grants IIS-2533550, IIS-2321504, IIS-2217239, CNS-2426514, and CMMI-2146076, Notre Dame Strategic Framework Research Grant (2025), and Notre Dame Poverty Research Package (2025). Any expressed opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

References

Laha Ale, Robert Gentleman, Christopher Endres, Sam Pullman, Nathan Palmer, Rafael Goncalves, and Deepayan Sarkar. 2025. Enhancing statistical analysis of real world data. *Database*, 2025:baaf073.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. 2025. Dabstep: Data agent benchmark for multi-step reasoning. *arXiv preprint arXiv:2506.23719*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*.

Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, and 1 others. 2024. Text2analysis: A benchmark of table question answering with advanced data analysis

and unclear queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18206–18215.

Noah Hollmann, Samuel Müller, and Frank Hutter. 2023. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36:44753–44775.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, and 1 others. 2025. Data interpreter: An llm agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821.

Chuxuan Hu, Dwip Dalal, and Xiaona Zhou. 2025. A dataset-centric survey of llm-agents for data science.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, and 1 others. 2024. Infiagent-dabench: Evaluating agents on data analysis tasks. *arXiv preprint arXiv:2401.05507*.

Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and 1 others. 2024. Da-code: Agent data science code generation benchmark for large language models. *arXiv preprint arXiv:2410.07331*.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2024. Dsbench: How far are data science agents from becoming data science experts? *arXiv preprint arXiv:2409.07703*.

Laura Koesten, Kathleen Gregory, Paul Groth, and Elena Simperl. 2021. Talking datasets–understanding data sensemaking behaviours. *International journal of human-computer studies*, 146:102562.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, and 1 others. 2024. Autokaggle: A multi-agent framework for autonomous data science competitions. *arXiv preprint arXiv:2410.20424*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

OpenAI. 2025a. Gpt-5 system card. Technical report, OpenAI.

OpenAI. 2025b. Introducing gpt-oss and gpt-oss-120b.

Shuyin Ouyang, Dong Huang, Jingwen Guo, Zeyu Sun, Qihao Zhu, and Jie M Zhang. 2025. Dscodebench: A realistic benchmark for data science code generation. *arXiv preprint arXiv:2505.15621*.

Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. 2025. Llm-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. 2025a. Lambda: A large model based data agent. *Journal of the American Statistical Association*, pages 1–13.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. 2025b. A survey on large language model-based agents for statistics and data science. *The American Statistician*, pages 1–14.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2024. Automl-agent: A multi-agent llm framework for full-pipeline automl. *arXiv preprint arXiv:2410.02958*.

Peiran Wang, Yaoning Yu, Ke Chen, Xianyang Zhan, and Haohan Wang. 2025. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Xueqing Wu, Rui Zheng, Jingzhen Sha, Te-Lin Wu, Hanyu Zhou, Mohan Tang, Kai-Wei Chang, Nanyun Peng, and Haoran Huang. 2024. Daco: Towards application-driven and comprehensive data analysis via code generation. *Advances in Neural Information Processing Systems*, 37:90661–90682.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Yanfang Ye, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Yiyang Li, Shifu Hou, Weixiang Sun, Kaiwen Shi, Yijun Ma, Wei Song, and 1 others. 2025. Llms4all: A review of large language models across academic disciplines. *arXiv preprint arXiv:2509.19580*.

Z.ai. 2025. Glm-4.7: Advancing the coding capability. Technical report, Zhipu AI.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. 2025a. Datascibench: An llm agent benchmark for data science. *arXiv preprint arXiv:2502.13897*.

Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. 2024a. Mlcopilot: Unleashing the power of large language models in solving machine learning tasks. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2931–2959.

Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. 2025b. Deepanalyze: Agentic large language models for autonomous data science. *arXiv preprint arXiv:2510.16872*.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automl-gpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499*.

Yuge Zhang, Qiyang Jiang, XingyuHan XingyuHan, Nan Chen, Yuqing Yang, and Kan Ren. 2024b. Benchmarking data science agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5677–5700.

Yizhang Zhu, Shiyin Du, Boyan Li, Yuyu Luo, and Nan Tang. 2024. Are large language models good statisticians? *Advances in Neural Information Processing Systems*, 37:62697–62731.

A Related Work

A.1 Data Analysis Benchmarks

Existing data analysis benchmarks can be broadly categorized into single-turn and interactive settings. Single-turn benchmarks such as DS-1000 (Lai et al., 2023), DSCodeBench (Ouyang et al., 2025), and DataSciBench (Zhang et al., 2025a) primarily evaluate one-shot code generation or analytical reasoning with well-specified prompts and structured inputs, placing limited demands on document-level language understanding.

More recent interactive benchmarks (e.g., InfiAgent-DABench (Hu et al., 2024), DSEval (Zhang et al., 2024b), DA-Code (Huang et al., 2024), DSBench (Jing et al., 2024)) allow agents to perform multi-step reasoning and interact with execution environments. However, most of these benchmarks are still built on relatively small datasets with limited or simplified documentation, and therefore do not fully capture the challenge of navigating large-scale, unstructured data descriptions. In contrast, LongDA evaluates data analysis agents on real-world datasets with extensive and complex documentation, requiring language understanding, information retrieval, and tool-assisted computation.

A.2 LLM Data Analysis Agents

Recent work has increasingly explored using large language models as autonomous agents for data analysis. Early efforts primarily focused on leveraging LLMs for code generation and computational reasoning, simplifying complex analytical workflows through program synthesis and tool execution (Gao et al., 2023; Zhu et al., 2024; Wang et al., 2024). Building on this foundation, a new generation of data science agents has emerged, integrating function-calling, code interpreters, and external tools to support end-to-end analytical processes, including Data Interpreter (Hong et al., 2025), DS-Agent (Guo et al., 2024), LAMBDA (Sun et al., 2025a), and related systems (Zhang et al., 2023, 2024a; Li et al., 2024; Trirat et al., 2024; Jiang et al., 2025).

Collectively, these systems demonstrate that LLM agents can effectively automate many components of data science pipelines, such as feature engineering, hyper-parameter tuning, model training, and evaluation. However, most existing approaches assume relatively clean, well-documented inputs and place limited emphasis on navigating large-scale, heterogeneous documentation. As a result, they under-exercise the document understanding and information discovery capabilities that dominate real-world analytical workflows. Moreover, scalability, long-context management, and robust multi-step planning remain open challenges for current agent frameworks.

Table 5: Survey Abbreviations, Official Names, and Data Sources

Abbrev.	Official Survey Name	Agency	Year	Data Source
ATUS	American Time Use Survey	Bureau of Labor Statistics	2024	Link
NHANES	National Health and Nutrition Examination Survey	National Center for Health Statistics	2021–2023	Link
NHIS	National Health Interview Survey	National Center for Health Statistics	2023	Link
NSFG	National Survey of Family Growth	National Center for Health Statistics	2022–2023	Link
GSS	Graduate Student Survey	National Science Foundation	2023	Link
HERD	Higher Education R&D Survey	National Science Foundation	2023	Link
NSCG	National Survey of College Graduates	National Science Foundation	2023	Link
SDR	Survey of Doctorate Recipients	National Science Foundation	2023	Link
SSERF	Survey of Science & Engineering Research Facilities	National Science Foundation	2023	Link
NTEWS	National Training, Education, and Workforce Survey	National Center for Education Statistics	2024	Link
NSDUH	National Survey on Drug Use and Health	SAMHSA	2023	Link
ASFIN	Annual Survey of State Government Finances	U.S. Census Bureau	2021	Link
ASPEP	Annual Survey of Public Employment & Payroll	U.S. Census Bureau	2024	Link
ASPP	Annual Survey of Public Pensions	U.S. Census Bureau	2019	Link
CPS–ASEC	Current Population Survey – ASEC	U.S. Census Bureau & BLS	2024	Link
RHFS	Rental Housing Finance Survey	U.S. Census Bureau	2023	Link
STC	Survey of State Tax Collections	U.S. Census Bureau	2020	Link

Table 6: Mapping Between Surveys and Source Publications

Survey	Publication	Agency	Year
ATUS	American Time Use Survey: 2024 Results	Bureau of Labor Statistics	2024
NHANES	Data Brief No. 508: Obesity Prevalence	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 511: Hypertension Prevalence	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 515: Sleep Duration	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 516: Physical Activity	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 519: Anemia Prevalence	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 527: Depression Prevalence	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 533: Fast Food Consumption	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 538: Prescription Opioid Use	National Center for Health Statistics	2021–2023
NHANES	Data Brief No. 540: Diabetes Prevalence	National Center for Health Statistics	2021–2023
NHIS	Data Brief No. 518: Chronic Pain	National Center for Health Statistics	2023
NHIS	Data Brief No. 528: Depression Medication	National Center for Health Statistics	2023
NHIS	Data Brief No. 529: COPD Prevalence	National Center for Health Statistics	2023
NSFG	Data Brief No. 520: Contraceptive Use	National Center for Health Statistics	2022–2023
NSFG	Data Brief No. 539: Pregnancy Intentions	National Center for Health Statistics	2022–2023
GSS	NSF 25–316: Graduate Students and Postdoctorates in SEH	National Science Foundation	2023
HERD	NSF 25–313: Higher Education R&D Expenditures	National Science Foundation	2023
NSCG	NSF 25–331: National Survey of College Graduates	National Science Foundation	2023
SDR	NSF 25–320: Survey of Doctorate Recipients	National Science Foundation	2023
SSERF	NSF 25–318: Survey of S&E Research Facilities	National Science Foundation	2023
NTEWS	NSF 25–352: National Training, Education, and Workforce Survey	National Center for Education Statistics	2024
NSDUH	2023 NSDUH Annual National Report	SAMHSA	2023
ASFIN	State Government Finances Summary	U.S. Census Bureau	2021
ASPEP	Public Employment & Payroll Summary Report	U.S. Census Bureau	2024
ASPP	Public Pensions Summary	U.S. Census Bureau	2019
CPS–ASEC	Income in the United States	U.S. Census Bureau & BLS	2024
CPS–ASEC	Poverty in the United States	U.S. Census Bureau & BLS	2024
CPS–ASEC	Health Insurance Coverage in the United States	U.S. Census Bureau & BLS	2024
RHFS	Rental Housing Finance Survey	U.S. Census Bureau	2023
STC	State Government Tax Collections Summary Report	U.S. Census Bureau	2020

B Survey Metadata

B.1 Survey Abbreviations, Full Names, and Official Links

Table 5 provides the complete list of survey abbreviations used throughout the paper, together with their official full names, releasing agencies, and public data access links. This information is provided to ensure reproducibility and facilitate independent verification of the data sources.

B.2 Survey–Publication Mapping

Table 6 summarizes the correspondence between surveys and the official analytical publications from which benchmark queries are extracted. Each query in LongDA is derived from exactly one publication associated with a specific survey, and publications are never exposed to agents during evaluation.

C Prompt Design

C.1 Prompt Template

This section presents the complete prompt template used by the ReAct-style agent in all experiments. The prompt explicitly specifies the global task description, available tools, question blocks, answer formats, and execution constraints, and is shown verbatim in Figure 9 to ensure full reproducibility of the experimental setup.

C.2 Question Block Example

To further illustrate the structure of the task input provided to the agent, Figure 10 presents a concrete example of a question block drawn from the `ASFIN` survey. The example contains multiple queries, their required answer formats, and the associated data and documentation files.

Prompt Template

Task: Analyze the survey data and associated documentation to answer the following questions.

Survey: `{survey}`

Questions and Expected Answer Structures: `{qa_block}`

Available Data Files: `{paths/to/the/data/files}`

Available Documentation Files: `{paths/to/the/doc/files}`

Instructions:

1. **Understand the Task**
   - Use the `prompt` tool to review all questions, expected answer structures, and additional information.
   - Identify which data files and documentation are required.
2. **Gather Information**
   - Use `retriever` or `search_doc` to locate relevant documentation.
   - Use `read_doc` to inspect specific files or sections.
   - Record key information (e.g., variable definitions, column meanings, survey weights) using the `notes` tool.
   - Do *not* rely on prior knowledge; always verify from documentation.
3. **Analyze Data**
   - Write and execute Python code to load and process the data.
   - Apply sampling weights when required by the survey design.
4. **Format and Save Answers**
   - Format each answer *exactly* according to the specified answer structure.
   - Respect unit requirements (e.g., percentages, counts, thousands, millions).
   - Report all numerical values with **one decimal place**.
   - Use the `answer` tool with `action='add'` to store each answer by question ID.
   - Use `save_code` to save the analysis code after completing each block.
5. **Working Strategy**
   - Questions may be solved incrementally; solving all at once is not required.
   - Use `answer` with `action='view'` to inspect saved answers.
   - Use `notes` with `action='list'` to review recorded information.

Figure 9: Prompt template used by the agent for solving LongDA tasks.

Example Question Block (ASFIN 2021)

Survey: ASFIN — Annual Survey of State Government Finances (2021)

Questions and Expected Answer Structures

- **Q1:** What was the total state government revenue in 2021?
  Answer Structure: `single_number`
  Additional Info: Trillions of dollars.

  ...

- **Q4:** What percentage of total 2021 revenue came from intergovernmental revenue and insurance trust revenue?
  Answer Structure: `[intergovernmental_revenue, insurance_trust_revenue]`
  Additional Info: NA.

  ...

- **Q11:** What was liquor store revenue, utility revenue, miscellaneous general revenue, and current charge revenue in 2021?
  Answer Structure: `[liquor_store_revenue, utility_revenue, miscellaneous_general_revenue, current_charges_revenue]`
  Additional Info: Billions of dollars.

  ...

Available Data Files

- `benchmark/ASFIN/data/21state35.txt`

Available Documentation Files

- `benchmark/ASFIN/docs/2021 F13_Form.pdf`
- `benchmark/ASFIN/docs/statetechdoc2021.pdf`
- `benchmark/ASFIN/docs/public-use-file-layout.xlsx`
- `benchmark/ASFIN/docs/itemcodes.xlsx`
- `benchmark/ASFIN/docs/government-ids.xlsx`

Instructions: ...

Figure 10: Concrete example of a LongDA question block.

D Case Study: Agent Interaction Example

To illustrate how an agent solves LongDA tasks end-to-end, we present a real interaction trace produced by GPT-5 on a block from the NHANES survey (NCHS Data Brief No. 508). The block contains multiple related prevalence queries with strict answer-structure constraints. Throughout the interaction, the agent has access only to the NHANES microdata and accompanying documentation. In this example, GPT-5 successfully answers the entire block.

D.1 Overview of the Block

This NHANES block asks for adult (age $\geq$ 20) obesity and severe obesity prevalence, both overall and stratified by sex, age group, and education group. Solving the block requires the agent to (i) identify the correct BMI variable, (ii) determine the appropriate sampling weight for examination-based analysis, (iii) map demographic variables to subgroup definitions, and (iv) implement weighted estimators in Python while satisfying the required output ordering.

D.2 Step-by-Step Interaction

Step 1: Re-establishing task context. The agent begins by invoking the `prompt` tool to re-check the full task specification, including question IDs, answer structures, and auxiliary constraints. This behavior is consistent with the strategy we observe for GPT-5 across blocks: it refreshes global task requirements early to minimize downstream formatting errors.

Step 2–4: Documentation navigation and variable identification. The primary challenge of this block lies in identifying the correct variables and analytic guidance from long, heterogeneous documentation. GPT-5 first uses `search_doc` to locate the BMI field in the anthropometry codebook `BMX_L.pdf`. It identifies `BMXBMI` and confirms its definition as body mass index (kg/m^2), already rounded to one decimal place in the released data.

In parallel, it queries the demographics codebook `DEMO_L.pdf` to locate:

- `RIDAGEYR` for age,
- `RIAGENDR` for sex (coded as 1/2),
- `DMDEDUC2` for adult education categories.

The agent then explicitly searches for weight variables in `DEMO_L.pdf`, discovering both `WTMEC2YR` and `WTINT2YR`, and correctly determines that `WTMEC2YR` is the appropriate 2-year MEC examination weight for analyses involving examination measures such as BMI. This distinction is critical: using interview weights would produce biased estimates.

Finally, GPT-5 consults pregnancy-related documentation and records the discovered variable semantics, coding schemes, subgroup mappings, and weight selection using the `notes` tool. This externalization of key analytical decisions enables clean, reusable code across all queries in the block.

Step 5: Single-pass computation via executable Python. After resolving all documentation-dependent uncertainties, GPT-5 performs most of the computation in a single extended code execution. Specifically, it:

1. Loads `DEMO_L.xpt` and `BMX_L.xpt` and merges them on respondent identifier `SEQN`;
2. Filters to adults aged $\geq$ 20 with valid BMI and positive MEC weights (`WTMEC2YR` > 0);
3. Constructs indicator variables for obesity (BMI ≥ 30) and severe obesity (BMI ≥ 40);
4. Defines subgroup variables:
   - age groups: 20–39, 40–59, 60+;
   - education groups: `hs_or_less` (DMDEDUC2 $\in \{1, 2, 3\}$), `some_college` (4), `bachelors_plus` (5), excluding refused/unknown.
5. Implements a weighted prevalence estimator

$$\widehat{p} \;=\; 100 \cdot \frac{\sum_i w_i y_i}{\sum_i w_i},$$

   using `WTMEC2YR` as w_i and the indicator variable as y_i.

The agent then computes each query by slicing the dataset into the required subpopulation and applying the same estimator. All outputs are rounded to one decimal place to match benchmark formatting constraints.

Answer recording and audit trail. Upon completing the computations, GPT-5 records each result using the `answer` tool under the corresponding

question ID, guaranteeing strict adherence to the required output structures. It also invokes `save_code` to preserve the full analysis pipeline, enabling reproducibility, auditing, and qualitative error analysis.

Step 6: Finalization. Finally, the agent emits a structured summary of all answers via `final_answer`, serving as a human-readable recap and signaling termination of the interaction. Benchmark evaluation, however, relies exclusively on the structured outputs recorded through the `answer` tool.

D.3 Implications for LongDA Evaluation

This example illustrates several core properties that LongDA is explicitly designed to evaluate:

- **Long-document grounding.** Correct solutions require discovering variable definitions, coding schemes, and weight selection rules from documentation rather than relying on prior knowledge alone.
- **Tool-assisted analytical workflow.** Successful execution depends on iterative tool use for information retrieval followed by executable computation.
- **Block-level reuse of intermediate knowledge.** Once variables and weights are identified, the same intermediate understanding supports multiple queries within the block.
- **Strict output compliance.** The agent must satisfy rigid structural and ordering constraints to enable automatic evaluation.

D.4 Summary

Overall, this interaction exemplifies the intended LongDA workflow: the agent first resolves documentation-dependent uncertainties and then executes a coherent analysis program to answer multiple related questions correctly. Such tasks go beyond code synthesis alone and require reliable agentic document navigation and tool use under realistic constraints.

Example Trace: GPT-5 on NHANES (Data Brief No. 508)

This appendix shows an execution trace (abridged only by formatting) for a LongDA block. The agent alternates between tool calls (e.g., `prompt`, `search_doc`, `read_doc`) and Python code execution, and records final answers via the `answer` tool.

Step 1

Code Action

```
# Step 1: Review the full prompt to confirm
↪  tasks
full_prompt = prompt()
print(full_prompt)
```

Observations

```
Analyze the survey data and relevant
↪  documents to answer the following
↪  questions.
Survey: NHANES

Questions and Expected Answer Structures:
Question ID: NHANES_NCHS Data Brief No.
↪  508_q1
Q1: What was the overall prevalence of
↪  obesity among adults age 20 and older?
Expected Answer Structure: single_number
Additional Info: Body mass index (BMI):
↪  Calculated as weight in kilograms divided
↪  by height in meters squared, rounded to
↪  one decimal place. Obesity: Defined as a
↪  BMI of greater than or equal to 30.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q2
Q2: What were the obesity prevalences for
↪  men and women?
Expected Answer Structure: [men, women]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q3
Q3: What were the obesity prevalences for
↪  age groups 20-39, 40-59, and 60+?
Expected Answer Structure: [age20_39,
↪  age40_59, age60plus]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q4
Q4: What were the obesity prevalences by
↪  education level?
Expected Answer Structure: [hs_or_less,
↪  some_college, bachelors_plus]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q5
Q5: What was the overall prevalence of
↪  severe obesity among adults?
Expected Answer Structure: single_number
Additional Info: Severe obesity: Defined as
↪  a BMI of greater than or equal to 40.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q6
Q6: What were the severe obesity prevalences
↪  for age groups 20-39, 40-59, and 60+?
Expected Answer Structure: [age20_39,
↪  age40_59, age60plus]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q7
```

```
Q7: What were the severe obesity prevalences
↪  for men and women?
Expected Answer Structure: [men, women]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q8
Q8: What were the severe obesity prevalences
↪  for men by age group?
Expected Answer Structure: [men20_39,
↪  men40_59, men60plus]
Additional Info: NA.

Question ID: NHANES_NCHS Data Brief No.
↪  508_q9
Q9: What were the severe obesity prevalences
↪  for women by age group?
Expected Answer Structure: [women20_39,
↪  women40_59, women60plus]
Additional Info: NA.


Available data files:
[Omitted for simplicity]

Available documentation files:
[Omitted for simplicity]

Instructions:
You are analyzing survey data to answer
↪  specific questions. Follow this workflow:

1. UNDERSTAND THE TASK
   - Use `prompt` tool to review all
   ↪  questions, expected answer structures,
   ↪  and additional info
   - Identify which data files and
   ↪  documentation you need

2. GATHER INFORMATION
   - Use `retriever` or `search_doc` to find
   ↪  relevant documentation about the
   ↪  dataset
   - Use `read_doc` to read specific
   ↪  documentation files when needed
   - Record key information (column names,
   ↪  meanings, important variables) using
   ↪  `notes` tool
   - Do NOT rely on your own knowledge -
   ↪  always seek information from the
   ↪  documentation

3. ANALYZE DATA
   - Write Python code to load and analyze
   ↪  the data
   - Apply sampling weights if required by
   ↪  the survey design

4. FORMAT AND SAVE ANSWERS
   - Format your answer EXACTLY as specified
   ↪  in the "Expected Answer Structure"
   - Pay attention to unit requirements
   ↪  (e.g., percentages, counts, thousands,
   ↪  millions, billions)
   - All numbers should have ONE decimal
   ↪  place (e.g., 45.3, not 45 or 45.27)
   - Use `answer` tool with action='add' to
   ↪  save each answer with its question ID
   - Use `save_code` to store the analysis
   ↪  code for each block after finishing
   ↪  the block

5. WORKING STRATEGY
   - You can answer questions one by one - no
   ↪  need to solve all at once
   - Use `answer` tool with action='view' to
   ↪  check answers you've already saved
   - Use `notes` tool with action='list' to
   ↪  review your documented findings
```

Step 2

Code Action

```
# Step 2: Gather documentation info: search
↪  for BMI and weight variables in BMX and
↪  DEMO docs

bmx_doc = "benchmark/NHANES/docs/Examinatio⌋
↪  n/BMX_L.pdf"
demo_doc = "benchmark/NHANES/docs/Demograph⌋
↪  ics/DEMO_L.pdf"
exam_overview_doc = "benchmark/NHANES/docs/⌋
↪  Examination/August 2021-August 2023
↪  Examination Data - Continuous NHANES.pdf"
demo_overview_doc = "benchmark/NHANES/docs/⌋
↪  Demographics/August 2021-August 2023
↪  Demographics Data - Continuous
↪  NHANES.pdf"

# Search for BMI variable in BMX_L.pdf
bmi_search = search_doc(doc_path=bmx_doc,
↪  keyword="BMXBMI", max_matches=5,
↪  context_chars=200)
print("BMX BMI search:\
", bmi_search)

# Search for what BMI means text
bmi_text_search =
↪  search_doc(doc_path=bmx_doc,
↪  keyword="Body Mass Index", max_matches=5,
↪  context_chars=200)
print("\
BMX Body Mass Index search:\
", bmi_text_search)

# Search for exam weights in examination
↪  overview (expect WTMEC...)
exam_weight_search =
↪  search_doc(doc_path=exam_overview_doc,
↪  keyword="weight", max_matches=10,
↪  context_chars=200)
print("\
Exam overview weight search:\
", exam_weight_search)

# Search for MEC weight variable name
↪  specifically
mec_search =
↪  search_doc(doc_path=exam_overview_doc,
↪  keyword="WTMEC", max_matches=10,
↪  context_chars=200)
print("\
Exam overview WTMEC search:\
", mec_search)
```

```
# Search demographic variables: age, sex,
↪  education in DEMO_L.pdf
age_search = search_doc(doc_path=demo_doc,
↪  keyword="RIDAGEYR", max_matches=5,
↪  context_chars=200)
print("\
DEMO age search:\
", age_search)

sex_search = search_doc(doc_path=demo_doc,
↪  keyword="RIAGENDR", max_matches=5,
↪  context_chars=200)
print("\
DEMO sex search:\
", sex_search)

educ_search = search_doc(doc_path=demo_doc,
↪  keyword="DMDEDUC2", max_matches=5,
↪  context_chars=250)
print("\
DEMO education search:\
", educ_search)

# Also search the demographics overview for
↪  weights guidance
demo_weight_search =
↪  search_doc(doc_path=demo_overview_doc,
↪  keyword="weight", max_matches=10,
↪  context_chars=200)
print("\
Demographics overview weight search:\
", demo_weight_search)
```

Observations

```
Execution logs:
BMX BMI search:
 [benchmark/NHANES/docs/Examination/BMX_L.p⌋
 ↪  df] Found 3 matches for
 ↪  'BMXBMI'.
Page 3: ...Body Mass Index (BMXBMI) : Body
↪  Mass Index (BMI) w as calculated as
↪  weight in kilogr ams divided b y height
↪  in meters squared, and then rounded to
↪  one decimal place. BMI Category -
↪  Children/Adolescents (BMDBMIC) : This v
↪  ar...
Page 14: ...Variable Name: BMXBMI SAS Label:
↪  Body Mass Index (kg/m**2) English Text:
↪  Body Mass Index (kg/m**2) Target: Both
↪  males and females 2 YEARS - 150
↪  YEARSBMXBMI - Body Mass Index (kg/m**2)
↪  Code or Value Value Description Count...
Page 14: ...Variable Name: BMXBMI SAS Label:
↪  Body Mass Index (kg/m**2) English Text:
↪  Body Mass Index (kg/m**2) Target: Both
↪  males and females 2 YEARS - 150
↪  YEARSBMXBMI - Body Mass Index (kg/m**2)
↪  Code or Value Value Description Count
↪  Cumulative Skip to Item 11.1 to 74.8
↪  Range of V alues 8471 8471 . Missing 389
↪  8860...

BMX Body Mass Index search:
 [benchmark/NHANES/docs/Examination/BMX_L.p⌋
 ↪  df] Found 5 matches for 'Body Mass
 ↪  Index'.
Page 3: ...Body Mass Index (BMXBMI) : Body
↪  Mass Index (BMI) w as calculated as
↪  weight in kilogr ams divided b y height
↪  in meters squared, and then rounded to
↪  one decimal place. BMI Category -
↪  Children/Adolescents (BMDBMIC) : T...
Page 3: ...Body Mass Index (BMXBMI) : Body
↪  Mass Index (BMI) w as calculated as
↪  weight in kilogr ams divided b y height
↪  in meters squared, and then rounded to
↪  one decimal place. BMI Category -
↪  Children/Adolescents (BMDBMIC) : This v
↪  ariable w as created ...
Page 14: ...Variable Name: BMXBMI SAS Label:
↪  Body Mass Index (kg/m**2) English Text:
↪  Body Mass Index (kg/m**2) Target: Both
↪  males and females 2 YEARS - 150
↪  YEARSBMXBMI - Body Mass Index (kg/m**2)
↪  Code or Value Value Description Count
↪  Cumulative Skip to Item 11...
Page 14: ...Variable Name: BMXBMI SAS Label:
↪  Body Mass Index (kg/m**2) English Text:
↪  Body Mass Index (kg/m**2) Target: Both
↪  males and females 2 YEARS - 150
↪  YEARSBMXBMI - Body Mass Index (kg/m**2)
↪  Code or Value Value Description Count
↪  Cumulative Skip to Item 11.1 to 74.8
↪  Range of V alues 8471 8471 . ...
Page 14: ...Variable Name: BMXBMI SAS Label:
↪  Body Mass Index (kg/m**2) English Text:
↪  Body Mass Index (kg/m**2) Target: Both
↪  males and females 2 YEARS - 150
↪  YEARSBMXBMI - Body Mass Index (kg/m**2)
↪  Code or Value Value Description Count
↪  Cumulative Skip to Item 11.1 to 74.8
↪  Range of V alues 8471 8471 . Missing 389
↪  8860...

Exam overview weight search:
 [benchmark/NHANES/docs/Examination/August
 ↪  2021-August 2023 Examination Data -
 ↪  Continuous NHANES.pdf] No matches for
 ↪  'weight'.

Exam overview WTMEC search:
 [benchmark/NHANES/docs/Examination/August
 ↪  2021-August 2023 Examination Data -
 ↪  Continuous NHANES.pdf] No matches for
 ↪  'WTMEC'.

DEMO age search:
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 4 matches for
 ↪  'RIDAGEYR'.
Page 2: ...entifies whether a participant w
↪  as both interviewed at home or b y
↪  telephone and examined in the mobile
↪  examination center (MEC), or w as only
↪  interviewed but nev er participated in
↪  the examination. RIDAGEYR: Age in y ears
↪  at the time of the screening interview
↪  for surv ey participants between the
↪  ages of 1 and 79 y ears of age. Due to
↪  disclosure concerns, all responses of
↪  participants aged 80 y ears and...
```

```
Page 2: ...sure concerns, all responses of
↪  participants aged 80 y ears and older
↪  are coded as "80. " In NHANES A ugust
↪  2021- August 2023, the weighted mean age
↪  for participants 80 y ears and older is
↪  85 y ears. RIDAGEYR w as calculated
↪  based on the participant 's date of
↪  birth. In r are cases, if the actual
↪  date of birth w as missing but the
↪  participant 's age in y ears w as pro
↪  vided, the reported age w as used. RI...
Page 10: ...Variable Name: RIDAGEYR SAS
↪  Label: Age in y ears at screening English
↪  Text: Age in y ears of the participant at
↪  the time of screening. Individuals 80
↪  and o ver are top-coded at 80 y ears of
↪  age. Target: Both males and femal...
Page 10: ...ars at screening English Text:
↪  Age in y ears of the participant at the
↪  time of screening. Individuals 80 and o
↪  ver are top-coded at 80 y ears of age.
↪  Target: Both males and females 0 YEARS -
↪  150 YEARSRIDAGEYR - Age in years at
↪  screening Code or Value Value
↪  Description Count Cumulative Skip to
↪  Item 0 to 79 Range of V alues 11408 11408
↪  80 80 years of age and o ver 525 11933 .
↪  Missing 0 11933...

DEMO sex search:
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 2 matches for
 ↪  'RIAGENDR'.
Page 9: ...Variable Name: RIAGENDR SAS Label:
↪  Gender English Text: Gender of the
↪  participant. Target: Both males and
↪  females 0 YEARS - 150 YEARSRIAGENDR -
↪  Gender Code or Value Value Description
↪  Count Cumulative Skip to Item 1 Male
↪  557...
Page 9: ...Variable Name: RIAGENDR SAS Label:
↪  Gender English Text: Gender of the
↪  participant. Target: Both males and
↪  females 0 YEARS - 150 YEARSRIAGENDR -
↪  Gender Code or Value Value Description
↪  Count Cumulative Skip to Item 1 Male
↪  5575 5575 2 Female 6358 11933 . Missing
↪  0 11933...

DEMO education search:
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 5 matches for
 ↪  'DMDEDUC2'.
Page 3: ...ded into 6 categories: 1 = Less
↪  than 1 y ear, 2 = 1 to 4 y ears, 3 = 5 to
↪  9 y ears, 4 = 10 to 14 y ears, 5 = 15 to
↪  19 y ears, and 6 = 20 y ear or more. F or
↪  participants who did not report the month
↪  of their arriv al, a v alue of 7 (July) w
↪  as used. DMDEDUC2: the highest gr ade or
↪  lev el of education completed b y adults
↪  20 y ears and older . DMDMARTZ: The
↪  marital status question w as ask ed of
↪  people 14 y ears of age and older . Due
↪  to disclosure risk, marital status is
↪  only released for people 20 y ea...
```

```
Page 17: ...SAS Label: Country of birth
↪  English Text: Country of birth Target:
↪  Both males and females 0 YEARS - 150
↪  YEARSDMDBORN4 - Country of birth Code or
↪  Value Value Description Count Cumulative
↪  Skip to Item 1 Born in 50 US states or W
↪  ashington, 10039 10039 DMDEDUC2 2 Others
↪  1875 11914 77 Refused 0 11914 DMDEDUC2
↪  99 Don't know 0 11914 DMDEDUC2 . Missing
↪  19 11933...
Page 17: ...y of birth Target: Both males
↪  and females 0 YEARS - 150 YEARSDMDBORN4 -
↪  Country of birth Code or Value Value
↪  Description Count Cumulative Skip to Item
↪  1 Born in 50 US states or W ashington,
↪  10039 10039 DMDEDUC2 2 Others 1875 11914
↪  77 Refused 0 11914 DMDEDUC2 99 Don't know
↪  0 11914 DMDEDUC2 . Missing 19 11933...
Page 17: ...nd females 0 YEARS - 150
↪  YEARSDMDBORN4 - Country of birth Code or
↪  Value Value Description Count Cumulative
↪  Skip to Item 1 Born in 50 US states or W
↪  ashington, 10039 10039 DMDEDUC2 2 Others
↪  1875 11914 77 Refused 0 11914 DMDEDUC2
↪  99 Don't know 0 11914 DMDEDUC2 . Missing
↪  19 11933...
Page 19: ...Variable Name: DMDEDUC2 SAS
↪  Label: Education lev el - Adults 20+
↪  English Text: What is the highest gr ade
↪  or lev el of school {y ou ha ve/SP has}
↪  completed or the highest degree {y ou ha
↪  ve/s/he has} receiv ed? English
↪  Instructions: HAND CARD DMQ1 READ HAND
↪  CARD CA TEGORIE...

Demographics overview weight search:
 [benchmark/NHANES/docs/Demographics/August
 ↪  2021-August 2023 Demographics Data -
 ↪  Continuous NHANES.pdf] Found 1 matches
 ↪  for 'weight'.
Page 1: ...rview of Sample Design,
↪  Nonresponse Bias Assessment, and
↪  Analytic Guidelines SAS Universal Viewer
↪  Data User Agreement Data File Name Doc
↪  File Data File Date Published
↪  Demographic Variables and Sample Weights
↪  DEMO_L Doc DEMO_L Data [XPT - 2.5 MB]
↪  September 2024...
Last output from code snippet:
None
```

Step 3

Code Action

```
# Step 2 (continued): find weight variables
↪  in DEMO_L documentation
demo_doc = "benchmark/NHANES/docs/Demograph⌋
↪  ics/DEMO_L.pdf"

wts_mec_search =
↪  search_doc(doc_path=demo_doc,
↪  keyword="WTMEC", max_matches=10,
↪  context_chars=240)
print("DEMO_L search for WTMEC:\
", wts_mec_search)
```

```
wts_int_search =
↪  search_doc(doc_path=demo_doc,
↪  keyword="WTINT", max_matches=10,
↪  context_chars=240)
print("\
DEMO_L search for WTINT:\
", wts_int_search)

wts_overall_search =
↪  search_doc(doc_path=demo_doc,
↪  keyword="weight", max_matches=10,
↪  context_chars=240)
print("\
DEMO_L search for \
```

Observations

```
Execution logs:
DEMO_L search for WTMEC:
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 3 matches for
 ↪  'WTMEC'.
Page 5: ...protect confidentialit y and to
↪  reduce disclosure risk and are described
↪  in NHANES Analytic Guidelines . Analysts
↪  should review the Guidelines carefully
↪  prior to analyzing surv ey data. Sample
↪  Weights: The 2- year sample weights
↪  (WTINT2YR, WTMEC2YR, depending on the
↪  analysis) should be used for all NHANES
↪  A ugust 2021- August 2023 analyses.
↪  Please refer to the NHANES Analytic
↪  Guidelines and the on-line NHANES T
↪  utorial for further details on the use
↪  of sample weights and other ana...
Page 29: ...Variable Name: WTMEC2YR SAS
↪  Label: Full sample 2- year MEC exam
↪  weight English Text: Full sample 2 y ear
↪  MEC exam weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTMEC2YR -
↪  Full sample 2-year MEC exam weight Code
↪  or Value Value Description Count Cumu...
Page 29: ...Variable Name: WTMEC2YR SAS
↪  Label: Full sample 2- year MEC exam
↪  weight English Text: Full sample 2 y ear
↪  MEC exam weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTMEC2YR -
↪  Full sample 2-year MEC exam weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4581.595095 to
↪  227108.296958 Range of V alues 8860 8860
↪  0 Not MEC Examined 3073 11933 . Missing
↪  0 11933...

DEMO_L search for WTINT:
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 3 matches for
 ↪  'WTINT'.
Page 5: ...e used to protect confidentialit
↪  y and to reduce disclosure risk and are
↪  described in NHANES Analytic Guidelines .
↪  Analysts should review the Guidelines
↪  carefully prior to analyzing surv ey
↪  data. Sample Weights: The 2- year sample
↪  weights (WTINT2YR, WTMEC2YR, depending
↪  on the analysis) should be used for all
↪  NHANES A ugust 2021- August 2023
↪  analyses. Please refer to the NHANES
↪  Analytic Guidelines and the on-line
↪  NHANES T utorial for further details on
↪  the use of sample weights and...
Page 28: ...Variable Name: WTINT2YR SAS
↪  Label: Full sample 2- year interview
↪  weight English Text: Full sample 2 y ear
↪  interview weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTINT2YR -
↪  Full sample 2-year interview weight Code
↪  or Value Value Description Count C...
Page 28: ...Variable Name: WTINT2YR SAS
↪  Label: Full sample 2- year interview
↪  weight English Text: Full sample 2 y ear
↪  interview weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTINT2YR -
↪  Full sample 2-year interview weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4584.463196 to
↪  170968.343177 Range of V alues 11933
↪  11933 . Missing 0 11933...

DEMO_L search for 'weight':
 [benchmark/NHANES/docs/Demographics/DEMO_L⌋
 ↪  .pdf] Found 10 matches for
 ↪  'weight'.
Page 1: ...cific to this cy cle, please see
↪  the "Analytic Notes" section below .
↪  This demogr aphics file pro vides
↪  information on the following topics:
↪  Survey participant 's household
↪  interview and examination status;
↪  Interview and examination sample weights;
↪  Masked variance units; The six -month
↪  time period when the examination w as
↪  performed; Pregnancy status; The ratio
↪  of family income to po verty guidelines;
↪  Total number of people in the Household;
↪  Demogr aphic information about the
↪  house...
Page 2: ...of the screening interview for
↪  surv ey participants between the ages of
↪  1 and 79 y ears of age. Due to disclosure
↪  concerns, all responses of participants
↪  aged 80 y ears and older are coded as
↪  "80. " In NHANES A ugust 2021- August
↪  2023, the weighted mean age for
↪  participants 80 y ears and older is 85 y
↪  ears. RIDAGEYR w as calculated based on
↪  the participant 's date of birth. In r
↪  are cases, if the actual date of birth w
↪  as missing but the participant 's age in
↪  y ears w as pro vided,...
```

```
Page 5: ...sample design variance units.
↪  MVUs are used to protect confidentialit
↪  y and to reduce disclosure risk and are
↪  described in NHANES Analytic Guidelines .
↪  Analysts should review the Guidelines
↪  carefully prior to analyzing surv ey
↪  data. Sample Weights: The 2- year sample
↪  weights (WTINT2YR, WTMEC2YR, depending
↪  on the analysis) should be used for all
↪  NHANES A ugust 2021- August 2023
↪  analyses. Please refer to the NHANES
↪  Analytic Guidelines and the on-line
↪  NHANES T utorial for further deta...
Page 5: .... MVUs are used to protect
↪  confidentialit y and to reduce disclosure
↪  risk and are described in NHANES
↪  Analytic Guidelines . Analysts should
↪  review the Guidelines carefully prior to
↪  analyzing surv ey data. Sample Weights:
↪  The 2- year sample weights (WTINT2YR,
↪  WTMEC2YR, depending on the analysis)
↪  should be used for all NHANES A ugust
↪  2021- August 2023 analyses. Please refer
↪  to the NHANES Analytic Guidelines and
↪  the on-line NHANES T utorial for further
↪  details on the use of sample wei...
Page 5: ...ghts (WTINT2YR, WTMEC2YR,
↪  depending on the analysis) should be used
↪  for all NHANES A ugust 2021- August 2023
↪  analyses. Please refer to the NHANES
↪  Analytic Guidelines and the on-line
↪  NHANES T utorial for further details on
↪  the use of sample weights and other
↪  analytic issues. Disclosure risks and
↪  issues pertaining to confidentialit y
↪  protection prev ent NCHS from releasing
↪  all the NHANES demogr aphic v ariables
↪  publicly . Additional information and
↪  instructions for requesting use of ...
Page 28: ...Variable Name: WTINT2YR SAS
↪  Label: Full sample 2- year interview
↪  weight English Text: Full sample 2 y ear
↪  interview weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTINT2YR -
↪  Full sample 2-year interview weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4584.463196 to
↪  170968.343177...
Page 28: ...Variable Name: WTINT2YR SAS
↪  Label: Full sample 2- year interview
↪  weight English Text: Full sample 2 y ear
↪  interview weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTINT2YR -
↪  Full sample 2-year interview weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4584.463196 to
↪  170968.343177 Range of V alues 11933
↪  11933 . Missing 0 11933...
Page 28: ...Variable Name: WTINT2YR SAS
↪  Label: Full sample 2- year interview
↪  weight English Text: Full sample 2 y ear
↪  interview weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTINT2YR -
↪  Full sample 2-year interview weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4584.463196 to
↪  170968.343177 Range of V alues 11933
↪  11933 . Missing 0 11933...
```

```
Page 29: ...Variable Name: WTMEC2YR SAS
↪  Label: Full sample 2- year MEC exam
↪  weight English Text: Full sample 2 y ear
↪  MEC exam weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTMEC2YR -
↪  Full sample 2-year MEC exam weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4581.595095 to
↪  227108.296958 R...
Page 29: ...Variable Name: WTMEC2YR SAS
↪  Label: Full sample 2- year MEC exam
↪  weight English Text: Full sample 2 y ear
↪  MEC exam weight. Target: Both males and
↪  females 0 YEARS - 150 YEARSWTMEC2YR -
↪  Full sample 2-year MEC exam weight Code
↪  or Value Value Description Count
↪  Cumulative Skip to Item 4581.595095 to
↪  227108.296958 Range of V alues 8860 8860
↪  0 Not MEC Examined 3073 ...

DEMO_L page 19 (likely DMDEDUC2 details):
 [benchmark/NHANES/docs/Demographics/DEMO_L
 ↪  .pdf] Pages: Page
 ↪  19
Page 19: Variable Name: DMDEDUC2 SAS Label:
↪  Education lev el - Adults 20+ English
↪  Text: What is the highest gr ade or lev
↪  el of school {y ou ha ve/SP has}
↪  completed or the highest degree {y ou ha
↪  ve/s/he has} receiv ed? English
↪  Instructions: HAND CARD DMQ1 READ HAND
↪  CARD CA TEGORIES IF NECES SARY ENTER
↪  HIGHEST LEVEL OF SCHOOL Target: Both
↪  males and females 20 YEARS - 150
↪  YEARSDMDEDUC2 - Education level - Adults
↪  20+ Code or ValueValue Description Count
↪  Cumulative Skip to Item 1 Less than 9th
↪  gr ade 373 373 2 9-11th gr ade (Includes
↪  12th gr ade with no diploma)666 1039 3
↪  High school gr aduate/GED or equiv alent
↪  1749 2788 4 Some college or AA degree
↪  2370 5158 5 College gr aduate or abo ve
↪  2625 7783 7 Refused 0 7783 9 Don't know
↪  11 7794 . Missing 4139 11933
Last output from code snippet:
None
```

Step 4

Code Action

```
# Step 2 (additional): check for pregnancy
↪  variable in DEMO_L to see if exclusions
↪  are required
demo_doc = "benchmark/NHANES/docs/Demograph
↪  ics/DEMO_L.pdf"
preg_search = search_doc(doc_path=demo_doc,
↪  keyword="preg", max_matches=10,
↪  context_chars=200)
print("DEMO_L pregnancy search:\
", preg_search)

# Record key notes about variables and
↪  weights
```

```
notes(action="add", content="Use BMX_L for
↪  BMI: BMXBMI is Body Mass Index (kg/m**2),
↪  rounded to one decimal. Use DEMO_L for
↪  demographics: RIDAGEYR (age in years),
↪  RIAGENDR (1=Male, 2=Female), DMDEDUC2
↪  (education for adults 20+). Use WTMEC2YR
↪  for exam-based analyses per DEMO_L
↪  (2-year MEC exam weight).", index=0)
notes(action="add", content="Education
↪  grouping for adults 20+: hs_or_less =
↪  DMDEDUC2 in [1,2,3]; some_college = 4;
↪  bachelors_plus = 5. Exclude DMDEDUC2 in
↪  [7,9] or missing for education-specific
↪  estimates.", index=0)
notes(action="add", content="Age groups:
↪  20-39, 40-59, 60+ using RIDAGEYR (80+
↪  top-coded to 80, acceptable).", index=0)
print(notes(action="list", content="",
↪  index=0))
```

Observations

```
Execution logs:
DEMO_L pregnancy search:
 [benchmark/NHANES/docs/Demographics/DEMO_L
 ↪  .pdf] Found 10 matches for
 ↪  'preg'.
Page 1: ...ics: Survey participant 's
↪  household interview and examination
↪  status; Interview and examination sample
↪  weights; Masked variance units; The six
↪  -month time period when the examination w
↪  as performed; Pregnancy status; The
↪  ratio of family income to po verty
↪  guidelines; Total number of people in
↪  the Household; Demogr aphic information
↪  about the household reference person;
↪  and Other selected demogr aphic...
Page 3: ...d for people 20 y ears of age and
↪  older , and recoded from the original 6
↪  categories to 3 categories (1 =
↪  Married/Living with partner; 2 =
↪  Widowed/Div orced/Separ ated; 3 = Nev er
↪  married). RIDEXPRG: Pregnancy status at
↪  the time of the health examination w as
↪  ascertained for females 8-59 y ears of
↪  age. Due to disclosure risk, pregnancy
↪  status is released only for women 20-44
↪  y ears of age. RIDEXPRG v ...
Page 3: ...rtner; 2 = Widowed/Div orced/Separ
↪  ated; 3 = Nev er married). RIDEXPRG:
↪  Pregnancy status at the time of the
↪  health examination w as ascertained for
↪  females 8-59 y ears of age. Due to
↪  disclosure risk, pregnancy status is
↪  released only for women 20-44 y ears of
↪  age. RIDEXPRG v alues are based on self
↪  -reported pregnancy status and urine
↪  pregnancy test results. P eople who
↪  reported they were pregnant at ...
Page 3: ...alth examination w as ascertained
↪  for females 8-59 y ears of age. Due to
↪  disclosure risk, pregnancy status is
↪  released only for women 20-44 y ears of
↪  age. RIDEXPRG v alues are based on self
↪  -reported pregnancy status and urine
↪  pregnancy test results. P eople who
↪  reported they were pregnant at the time
↪  of exam were assumed to be pregnant
↪  (RIDEXPRG=1). Those who reported they
↪  were not pregnant or did no...
Page 3: ...tained for females 8-59 y ears of
↪  age. Due to disclosure risk, pregnancy
↪  status is released only for women 20-44
↪  y ears of age. RIDEXPRG v alues are
↪  based on self -reported pregnancy status
↪  and urine pregnancy test results. P
↪  eople who reported they were pregnant at
↪  the time of exam were assumed to be
↪  pregnant (RIDEXPRG=1). Those who
↪  reported they were not pregnant or did
↪  not know their pregnancy stat...
Page 3: ...e risk, pregnancy status is
↪  released only for women 20-44 y ears of
↪  age. RIDEXPRG v alues are based on self
↪  -reported pregnancy status and urine
↪  pregnancy test results. P eople who
↪  reported they were pregnant at the time
↪  of exam were assumed to be pregnant
↪  (RIDEXPRG=1). Those who reported they
↪  were not pregnant or did not know their
↪  pregnancy status were further classified
↪  based on the results of the ...
Page 3: ...men 20-44 y ears of age. RIDEXPRG
↪  v alues are based on self -reported
↪  pregnancy status and urine pregnancy
↪  test results. P eople who reported they
↪  were pregnant at the time of exam were
↪  assumed to be pregnant (RIDEXPRG=1).
↪  Those who reported they were not
↪  pregnant or did not know their pregnancy
↪  status were further classified based on
↪  the results of the urine pregnancy test.
↪  If the respondent reported...
Page 3: ...elf -reported pregnancy status and
↪  urine pregnancy test results. P eople
↪  who reported they were pregnant at the
↪  time of exam were assumed to be pregnant
↪  (RIDEXPRG=1). Those who reported they
↪  were not pregnant or did not know their
↪  pregnancy status were further classified
↪  based on the results of the urine
↪  pregnancy test. If the respondent
↪  reported "no" or "don' t know" and the
↪  urine test result w as po...
Page 3: ...and urine pregnancy test results.
↪  P eople who reported they were pregnant
↪  at the time of exam were assumed to be
↪  pregnant (RIDEXPRG=1). Those who
↪  reported they were not pregnant or did
↪  not know their pregnancy status were
↪  further classified based on the results
↪  of the urine pregnancy test. If the
↪  respondent reported "no" or "don' t
↪  know" and the urine test result w as
↪  positiv e, the respondent w as co...
```

```
Page 3: ...t the time of exam were assumed to
↪  be pregnant (RIDEXPRG=1). Those who
↪  reported they were not pregnant or did
↪  not know their pregnancy status were
↪  further classified based on the results
↪  of the urine pregnancy test. If the
↪  respondent reported "no" or "don' t
↪  know" and the urine test result w as
↪  positiv e, the respondent w as coded as
↪  pregnant (RIDEXPRG=1). If the respondent
↪  reported "no" and the urine...
Notes:
1. Use BMX_L for BMI: BMXBMI is Body Mass
↪  Index (kg/m**2), rounded to one decimal.
↪  Use DEMO_L for demog...
2. Education grouping for adults 20+:
↪  hs_or_less = DMDEDUC2 in [1,2,3];
↪  some_college = 4; bachelors_plu...
3. Age groups: 20-39, 40-59, 60+ using
↪  RIDAGEYR (80+ top-coded to 80,
↪  acceptable).
Last output from code snippet:
None
```

Step 5

Code Action

```
# Check BMX documentation for any mention of
↪  excluding pregnant women
bmx_doc = "benchmark/NHANES/docs/Examinatio⌋
↪  n/BMX_L.pdf"
preg_search_bmx =
↪  search_doc(doc_path=bmx_doc,
↪  keyword="preg", max_matches=10,
↪  context_chars=200)
print("BMX_L pregnancy search:\
", preg_search_bmx)

# Proceed to data analysis
import pandas as pd
import numpy as np

# Load data
demo_path = "benchmark/NHANES/data/Demograp⌋
↪  hics/DEMO_L.xpt"
bmx_path = "benchmark/NHANES/data/Examinati⌋
↪  on/BMX_L.xpt"

demo = pd.read_sas(demo_path,
↪  format="xport")
bmx = pd.read_sas(bmx_path, format="xport")

# Keep only necessary columns to reduce
↪  memory
demo_cols = ["SEQN", "RIDAGEYR", "RIAGENDR",
↪  "DMDEDUC2", "WTMEC2YR"]
demo = demo[demo_cols]
bmx_cols = ["SEQN", "BMXBMI"]
bmx = bmx[bmx_cols]

# Merge
df = demo.merge(bmx, on="SEQN", how="inner")

# Convert to numeric (ensure dtypes)
for col in ["RIDAGEYR", "RIAGENDR",
↪  "DMDEDUC2", "WTMEC2YR", "BMXBMI"]:
    df[col] = pd.to_numeric(df[col],
    ↪  errors="coerce")

# Restrict to adults 20+ and with valid
↪  weight and BMI
adults = df[(df["RIDAGEYR"] >= 20) &
↪  (df["WTMEC2YR"] > 0) &
↪  df["BMXBMI"].notna()]

# Define obesity and severe obesity
adults = adults.copy()
adults["obese"] = (adults["BMXBMI"] >=
↪  30).astype(int)
adults["severe_obese"] = (adults["BMXBMI"]
↪  >= 40).astype(int)

# Age groups
def age_group(age):
    if age >= 20 and age <= 39:
        return "20-39"
    elif age >= 40 and age <= 59:
        return "40-59"
    else:
        return "60+"
adults["age_group"] =
↪  adults["RIDAGEYR"].apply(age_group)

# Education groups: hs_or_less=[1,2,3],
↪  some_college=4, bachelors_plus=5
def educ_group(x):
    if x in [1, 2, 3]:
        return "hs_or_less"
    elif x == 4:
        return "some_college"
    elif x == 5:
        return "bachelors_plus"
    else:
        return np.nan
adults["educ_group"] =
↪  adults["DMDEDUC2"].apply(educ_group)

# Helper to compute weighted prevalence (%)
def weighted_pct(data, indicator_col,
↪  weight_col="WTMEC2YR"):
    w = data[weight_col]
    y = data[indicator_col]
    num = (w * y).sum()
    den = w.sum()
    if den == 0 or np.isnan(den):
        return np.nan
    return float(100.0 * num / den)

# Q1: overall prevalence of obesity among
↪  adults 20+
q1_val = weighted_pct(adults, "obese")
q1 = round(q1_val, 1) if pd.notna(q1_val)
↪  else np.nan
print("Q1 overall obesity prevalence:", q1)

# Q2: obesity prevalences for men and women
↪  [men, women]
men = adults[adults["RIAGENDR"] == 1]
women = adults[adults["RIAGENDR"] == 2]
q2_vals = [round(weighted_pct(men, "obese"),
↪  1), round(weighted_pct(women, "obese"),
↪  1)]
print("Q2 obesity men, women:", q2_vals)
```

```
# Q3: obesity by age groups [20-39, 40-59,
↪  60+]
q3_vals = []
for grp in ["20-39", "40-59", "60+"]:
    d = adults[adults["age_group"] == grp]
    q3_vals.append(round(weighted_pct(d,
    ↪  "obese"), 1))
print("Q3 obesity by age groups:", q3_vals)

# Q4: obesity by education level [hs_or_less,
↪  some_college, bachelors_plus]
educ_order = ["hs_or_less", "some_college",
↪  "bachelors_plus"]
q4_vals = []
for eg in educ_order:
    d = adults[adults["educ_group"] == eg]
    q4_vals.append(round(weighted_pct(d,
    ↪  "obese"), 1))
print("Q4 obesity by education level:",
↪  q4_vals)

# Q5: overall prevalence of severe obesity
↪  among adults
q5_val = weighted_pct(adults,
↪  "severe_obese")
q5 = round(q5_val, 1) if pd.notna(q5_val)
↪  else np.nan
print("Q5 overall severe obesity
↪  prevalence:", q5)

# Q6: severe obesity by age groups [20-39,
↪  40-59, 60+]
q6_vals = []
for grp in ["20-39", "40-59", "60+"]:
    d = adults[adults["age_group"] == grp]
    q6_vals.append(round(weighted_pct(d,
    ↪  "severe_obese"), 1))
print("Q6 severe obesity by age groups:",
↪  q6_vals)

# Q7: severe obesity prevalences for men and
↪  women [men, women]
q7_vals = [round(weighted_pct(men,
↪  "severe_obese"), 1),
↪  round(weighted_pct(women,
↪  "severe_obese"), 1)]
print("Q7 severe obesity men, women:",
↪  q7_vals)

# Q8: severe obesity prevalences for men by
↪  age group [men20-39, men40-59, men60+]
q8_vals = []
for grp in ["20-39", "40-59", "60+"]:
    d = men[men["age_group"] == grp]
    q8_vals.append(round(weighted_pct(d,
    ↪  "severe_obese"), 1))
print("Q8 severe obesity men by age:",
↪  q8_vals)

# Q9: severe obesity prevalences for women
↪  by age group [women20-39, women40-59,
↪  women60+]
q9_vals = []
for grp in ["20-39", "40-59", "60+"]:
    d = women[women["age_group"] == grp]
    q9_vals.append(round(weighted_pct(d,
    ↪  "severe_obese"), 1))
print("Q9 severe obesity women by age:",
↪  q9_vals)

# Save answers
ans1 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q1",
↪  answer=str(q1))
ans2 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q2",
↪  answer=[q2_vals[0], q2_vals[1]])
ans3 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q3",
↪  answer=[q3_vals[0], q3_vals[1],
↪  q3_vals[2]])
ans4 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q4",
↪  answer=[q4_vals[0], q4_vals[1],
↪  q4_vals[2]])
ans5 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q5",
↪  answer=str(q5))
ans6 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q6",
↪  answer=[q6_vals[0], q6_vals[1],
↪  q6_vals[2]])
ans7 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q7",
↪  answer=[q7_vals[0], q7_vals[1]])
ans8 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q8",
↪  answer=[q8_vals[0], q8_vals[1],
↪  q8_vals[2]])
ans9 = answer(action="add",
↪  q_id="NHANES_NCHS Data Brief No. 508_q9",
↪  answer=[q9_vals[0], q9_vals[1],
↪  q9_vals[2]])

print(ans1, ans2, ans3, ans4, ans5, ans6,
↪  ans7, ans8, ans9)

# Save the analysis code
save_code("""
Loaded DEMO_L and BMX_L, merged by SEQN.
↪  Filtered adults 20+ with valid BMI and
↪  WTMEC2YR > 0.
Defined obesity (BMXBMI >= 30) and severe
↪  obesity (BMXBMI >= 40).
Constructed age groups (20-39, 40-59, 60+)
↪  and education groups (hs_or_less=[1,2,3],
↪  some_college=4, bachelors_plus=5).
Computed weighted prevalences using WTMEC2YR
↪  for overall and subgroups, expressed as
↪  percentages with one decimal place.
Recorded answers using the answer tool.
""")
```

Observations

```
Execution logs:
BMX_L pregnancy search:
 [benchmark/NHANES/docs/Examination/BMX_L.p⌋
 ↪  df] Found 3 matches for
 ↪  'preg'.
```

```
Page 2: ... considered identifiable and pose
↪  a disclosure concern. The body weight
↪  data for participants who had limb
↪  amputations were set to "missing. " This
↪  data file includes body measures for
↪  women who were pregnant at the time of
↪  their health examination. Pregnancy
↪  status at the time of the health
↪  examination is indicated b y the
↪  variable, RIDEXPRG, in the Demogr aphic
↪  data file. RIDEXPRG v alues are report...
Page 2: ...cern. The body weight data for
↪  participants who had limb amputations
↪  were set to "missing. " This data file
↪  includes body measures for women who
↪  were pregnant at the time of their
↪  health examination. Pregnancy status at
↪  the time of the health examination is
↪  indicated b y the variable, RIDEXPRG, in
↪  the Demogr aphic data file. RIDEXPRG v
↪  alues are reported for women 20-44 y
↪  ears of age. RIDEXPRG for sev...
Page 2: ...status at the time of the health
↪  examination is indicated b y the
↪  variable, RIDEXPRG, in the Demogr aphic
↪  data file. RIDEXPRG v alues are reported
↪  for women 20-44 y ears of age. RIDEXPRG
↪  for sev eral pregnant women who were
↪  outside of this age range are not
↪  reported due to disclosure concerns. The
↪  body measures data for these
↪  participants are also not reported. The
↪  NHANES Anthropometry Procedures Manu...
Q1 overall obesity prevalence: 40.3
Q2 obesity men, women: [39.2, 41.3]
Q3 obesity by age groups: [35.6, 46.4, 38.9]
Q4 obesity by education level: [44.5, 45.2,
↪  31.6]
Q5 overall severe obesity prevalence: 9.4
Q6 severe obesity by age groups: [9.6, 11.9,
↪  6.6]
Q7 severe obesity men, women: [6.7, 12.1]
Q8 severe obesity men by age: [6.1, 9.2, 4.3]
Q9 severe obesity women by age: [13.0, 14.6,
↪  8.4]
Answer recorded for question NHANES_NCHS
↪  Data Brief No. 508_q1 Answer recorded
↪  for question NHANES_NCHS Data Brief No.
↪  508_q2 Answer recorded for question
↪  NHANES_NCHS Data Brief No. 508_q3 Answer
↪  recorded for question NHANES_NCHS Data
↪  Brief No. 508_q4 Answer recorded for
↪  question NHANES_NCHS Data Brief No.
↪  508_q5 Answer recorded for question
↪  NHANES_NCHS Data Brief No. 508_q6 Answer
↪  recorded for question NHANES_NCHS Data
↪  Brief No. 508_q7 Answer recorded for
↪  question NHANES_NCHS Data Brief No.
↪  508_q8 Answer recorded for question
↪  NHANES_NCHS Data Brief No. 508_q9
Last output from code snippet:
Code stored for block NHANES_NCHS Data Brief
↪  No. 508.
```

Step 6

Code Action

```
summary = {
    "NHANES_NCHS Data Brief No. 508_q1": q1,
    "NHANES_NCHS Data Brief No. 508_q2":
    ↪  q2_vals,
    "NHANES_NCHS Data Brief No. 508_q3":
    ↪  q3_vals,
    "NHANES_NCHS Data Brief No. 508_q4":
    ↪  q4_vals,
    "NHANES_NCHS Data Brief No. 508_q5": q5,
    "NHANES_NCHS Data Brief No. 508_q6":
    ↪  q6_vals,
    "NHANES_NCHS Data Brief No. 508_q7":
    ↪  q7_vals,
    "NHANES_NCHS Data Brief No. 508_q8":
    ↪  q8_vals,
    "NHANES_NCHS Data Brief No. 508_q9":
    ↪  q9_vals
}
final_answer(summary)
```

Observations

```
Execution logs:
Last output from code snippet:
{'NHANES_NCHS Data Brief No. 508_q1': 40.3,
↪  'NHANES_NCHS Data Brief No. 508_q2':
↪  [39.2, 41.3], 'NHANES_NCHS Data Brief No.
↪  508_q3': [35.6, 46.4, 38.9], 'NHANES_NCHS
↪  Data Brief No. 508_q4': [44.5, 45.2,
↪  31.6], 'NHANES_NCHS Data Brief No.
↪  508_q5': 9.4, 'NHANES_NCHS Data Brief No.
↪  508_q6': [9.6, 11.9, 6.6], 'NHANES_NCHS
↪  Data Brief No. 508_q7': [6.7, 12.1],
↪  'NHANES_NCHS Data Brief No. 508_q8':
↪  [6.1, 9.2, 4.3], 'NHANES_NCHS Data Brief
↪  No. 508_q9': [13.0, 14.6, 8.4]}
```

Reflect and Retrieve: Multimodal Agentic Retrieval with Reflection and Query Agents for Document Search

Tryambak Gangopadhyay, Md Sirajus Salekin, Yiyue Qian,
Nkechinyere Agu, Shinan Zhang, Rinne Han
Amazon Web Services

Abstract

Retrieving relevant pages from visually rich documents, with evidence in charts, tables, and figures rather than body text, remains difficult: a single-pass query against a fixed index misses content in visual elements, and reordering the same candidates cannot recover a page never retrieved. We propose Reflect and Retrieve, an agentic retrieval approach centered on a Reflection Agent. After an initial page-level retrieval, the agent uses a multimodal large language model to visually reason over the retrieved page images, judge whether the evidence is sufficient, and, when gaps remain, an orchestrator issues targeted follow-up retrievals to recover the missing pages. Evaluating on two benchmarks, MMDocIR and ViDoRe V3, the Reflection Agent delivers the dominant agentic gain: it is the only configuration to beat single-step retrieval for every reasoning model on both, improving Recall@N from 0.628 to 0.663 on MMDocIR and 0.566 to 0.644 on ViDoRe V3. The improvement comes from recovering relevant pages absent in initial retrieval. Our approach indicates a clear design order: selecting an optimal embedding approach for the retrieval tool is the first step, after which the Reflection Agent contributes more value than a query decomposition agent, positioning visual reflection as the key component for document search.

1 Introduction

Real-world documents convey information through charts, tables, diagrams, and page layout as much as through text: financial reports present results in bar charts, research papers embed findings in figures, and contracts organize obligations in tables. Retrieval-Augmented Generation (RAG) systems that rely on extracted text (Lewis et al., 2020) lose this visual context. Page-level retrieval, which returns the most relevant pages for a query, is a natural granularity that preserves the full multimodal content of each page (Dong et al., 2025a; Faysse et al., 2025).

Recent work makes substantial progress here. ColPali (Faysse et al., 2025) introduces late-interaction retrieval directly from page images using a vision-language model (VLM), and VisRAG (Yu et al., 2025) reports 20 to 40% end-to-end gains over text-only RAG by embedding pages as images; other approaches embed VLM-generated page descriptions (Dong et al., 2025a; Ma et al., 2024). All treat retrieval as a *single-step* operation: embed the query, return the top-K, and stop. This one-shot paradigm caps performance: our best single-step configuration reaches only 0.628 Recall@N on MMDocIR and 0.566 on ViDoRe V3, with no mechanism to detect or recover from retrieval failures.

In parallel, the agentic RAG literature shows that iterative, self-correcting retrieval improves accuracy in text-only settings: Self-RAG (Asai et al., 2024) decides when to retrieve and self-critiques via reflection tokens, Corrective RAG (Yan et al., 2024) adds a retrieval evaluator that triggers corrective actions when confidence is low, Adaptive-RAG (Jeong et al., 2024) routes queries by complexity, and a recent survey (Singh et al., 2025) identifies reflection, planning, tool use, and multi-agent collaboration as core patterns. These methods operate exclusively on text and cannot inspect a retrieved chart, table, or diagram to judge its relevance, a fundamental limitation for multimodal documents.

We propose **Reflect and Retrieve**, an agentic retrieval approach for multimodal page-level document search with three components: (1) a *retrieval tool*, whose optimal embedding strategy and VLM we first identify; (2) a *Query Agent* that decomposes or refines queries before retrieval; and (3) a multimodal *Reflection Agent* that visually inspects the retrieved page images and decides whether to accept them or issue targeted follow-up

The Proceedings of Workshop on Document Intelligence and Understanding, pages 138–154
October 24, 2026 ©2026 Association for Computational Linguistics

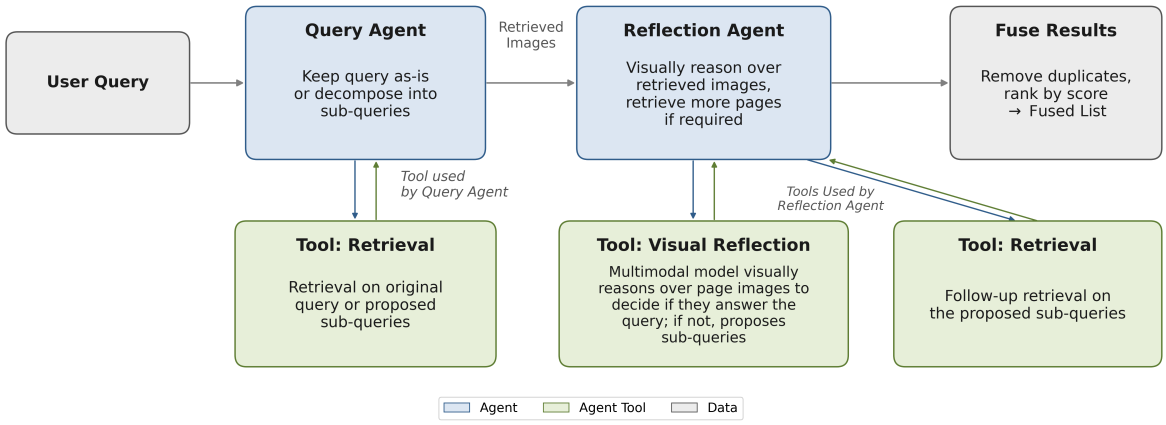


Figure 1: Reflect and Retrieve workflow centered on Reflection Agent. The Query Agent first keeps the user query as-is or decomposes it and retrieves an initial set of page images, which are sent to the Reflection Agent. The reflection agent calls a `visual reflection` tool, which uses a multimodal model to visually reason over the page images to decide whether they answer the query and, if not, proposes sub-queries; a `retrieval` tool then issues follow-up retrievals on those sub-queries. All retrieved items are fused by removing duplicates and ranking by score into the final fused list.

retrievals. The Reflection Agent is the key differentiator: by *seeing* the retrieved pages rather than only reading their text, it detects content gaps that text-only evaluation misses. Across enterprise document-search deployments in hospitality, security services, finance, and healthcare, first-pass retrieval repeatedly missed evidence inside charts, tables, and drawings that reranking cannot recover. Reflect and Retrieve emerged from those engagements: optimize the embedding strategy first, then add agentic retrieval with reflection and query agents. Our contributions are threefold: (1) a controlled comparison of five embedding configurations on two benchmarks to find optimal retrieval tool settings, establishing that VLM-generated descriptions outperform image-only embeddings by +16.7 to +19.7 Recall points; (2) the REFLECT AND RETRIEVE architecture, which introduces a multimodal Reflection Agent that delivers consistent gains (+3.4 to +7.9 points) on both benchmarks while query decomposition helps conditionally; and 3) a systematic study of agentic retrieval across five backbone LLMs on MMDocIR (1,658 queries) and ViDoRe V3 (1,487 queries) that yields practical guidance on when each agent helps, with a design distilled from production multimodal-search systems we built for enterprise customers.

2 Related Work

Multimodal Document Retrieval: Traditional retrieval discards visual signals by operating on extracted text (Lewis et al., 2020). Recent methods embed pages directly as images: ColPali (Faysse et al., 2025) adapts ColBERT (Khattab and Zaharia, 2020) late-interaction to page images, DSE (Ma et al., 2024) uses single-vector screenshot embeddings, VLM2Vec (Jiang et al., 2025) trains VLMs as multimodal encoders, and VisRAG (Yu et al., 2025) reports 20 to 40% gains over text-only RAG. Alternatively, embedding VLM-generated page descriptions yields strong results (Dong et al., 2025a), with extensions to cross-modal and multi-page settings (Li et al., 2025; Cho et al., 2025; Dong et al., 2025b). All treat retrieval as single-step with no self-correction mechanism.

Agentic and Iterative Retrieval: A growing body of work makes retrieval adaptive and self-correcting. Self-RAG (Asai et al., 2024) uses reflection tokens to decide when to retrieve and to assess relevance; Corrective RAG (Yan et al., 2024) adds a retrieval evaluator that triggers corrective actions when confidence is low; Adaptive-RAG (Jeong et al., 2024) routes queries by complexity; and Auto-RAG (Yu et al., 2024) plans multi-turn retrieval autonomously. On the query side, HyDE (Gao et al., 2023) retrieves with a hypothetical answer document, and RAPTOR (Sarthi et al., 2024) builds recursive passage trees. A recent survey (Singh et al., 2025) distils four agentic patterns: reflection, planning, tool use, and multi-agent collaboration. All of these operate in the text domain and cannot inspect charts, tables, or diagrams. REFLECT AND RETRIEVE bridges this

gap with a multimodal Reflection Agent that judges relevance from the page image itself, for example verifying that a retrieved financial page actually contains the queried chart rather than merely sharing keywords, a capability no prior agentic retrieval system provides.

3 Method: Reflect and Retrieve

REFLECT AND RETRIEVE places agents on top of a shared retrieval tool backed by a dense vector index. We describe the tool, then the two agents and their composition.

3.1 Retrieval Tool

The retrieval tool performs a single-step nearest neighbour search over a dense vector index. Its central design variable is the modality of the indexed page representation, governed by two dimensions: (i) the VLM used to generate a per-page textual description, and (ii) the embedding strategy applied to each page.

Each page is described by a VLM in up to 1,600 tokens using a structured prompt (Appendix A.2). We compare **Claude Sonnet 4.5** and **Qwen3-VL 235B**, both receiving the same prompt. We derive three variants using Amazon Nova Multimodal Embeddings (1,024-d): **Image-Only** from the raw page image; **Desc.-Only** from the VLM-generated description; and **Image+Desc.** from both jointly.

When the retrieval tool is called multiple times (because the query was decomposed or because the Reflection Agent issued follow-up retrievals), the returned lists are *fused*: concatenated, sorted by descending cosine similarity score, and deduplicated by page identity, yielding a single ranked list of up to N unique pages. Each retrieval call returns its top $k = 10$ pages; we use this cutoff in every scenario.

3.2 Reflection Agent

The Reflection Agent is the core of our approach (Figure 1). It operates downstream of an initial retrieval and performs *multimodal* reasoning over the retrieved page images (the standalone reflection workflow is shown in Appendix Figure 4). It comprises two components: an **orchestrator** agent that coordinates the workflow, and a `visual_reflection` tool, backed by a multimodal LLM, that inspects page images and returns a retrieval adequacy verdict.

Initial retrieval first produces a ranked list of k page images. The orchestrator passes these images together with the user query to the `visual_reflection` tool, which visually reasons over the rendered pages and returns either an *adequate* verdict, indicating the set sufficiently covers the query and triggering immediate output, or a *not adequate* verdict accompanied by up to two refined sub-queries that target the identified gaps. In the latter case the orchestrator issues follow-up retrieval calls on the suggested sub-queries, and all items (initial retrieval plus follow-ups) are fused and ranked by score.

By reasoning over rendered page images along with the user query, the reflection agent detects visual gaps, for example a chart referenced in the query but absent from the retrieved pages, and formulates targeted follow-up queries to recover the missing pages. To bound latency and cost, the agent is limited to at most two `visual_reflection` calls and two follow-up retrieval calls per question. We decouple the orchestrator (fixed to Claude Haiku 4.5 for efficiency) from the `visual_reflection` model (swept across four multimodal LLMs of varying capacity). We provide the orchestrator and `visual_reflection` prompts in Appendix.

3.3 Query Agent

The Query Agent is a text-only LLM that uses the retrieval tool. For each user query it chooses one of two actions: (A) *pass-through*, issuing the query unchanged as a single retrieval call, or (B) *decomposition*, splitting a multi-faceted query into at most two sub-queries and issuing one retrieval call each. With $m \leq 2$ (sub-)queries the fused list holds up to $N = m \times k$ items before deduplication. The agent only retrieves: it never answers from its own knowledge, rephrases beyond decomposition, or summarizes the returned pages. We evaluate it by sweeping the backbone LLM across five models of varying capacity; The query agent prompt appear in Appendix.

3.4 Query and Reflection Agent

In the most agentic configuration, the two agents are composed sequentially: the Query Agent first rewrites and potentially decomposes the query to produce a seed retrieval set, and the Reflection Agent then reviews and augments that set through visual inspection (Figure 1). This configuration tests whether query decomposition provides gains

complementary to the reflection loop.

4 Experimental Setup

4.1 Datasets

We evaluate on two page-level visual document retrieval benchmarks; each question references one or more ground-truth (GT) page images, and a retrieval is scored by how well its returned page set covers those GT pages. **MMDocIR** (Dong et al., 2025a) provides 1,658 questions over 313 source documents spanning 10 professional domains (academic papers, financial reports, news, laws, guidebooks, and others). 81% of questions have exactly one GT page (mean 1.27), making it primarily a "find the single right page" task. **ViDoRe V3** (Loison et al., 2026) is a page-level benchmark with human-verified annotations spanning 5 enterprise domains (finance, HR, industrial, pharmaceuticals, computer science). It ships each query in six languages; we use the English subset of 1,487 queries over 96 source documents and 12,968 pages. Its GT density is roughly four times that of MMDocIR (mean 5.09 pages per question; only 25% of questions are single page), making it a multi-page retrieval challenge. Per-domain question and page counts for both benchmarks are given in Tables 5 and 6.

4.2 Evaluation Metrics

All scenarios are scored on the fused, deduplicated ranked list of page images, formed by concatenating the per-query retrieval blocks, sorting by descending k-NN cosine similarity, and keeping the first occurrence of each page. For single step retrieval no fusion is needed, so the list has length k; agentic scenarios add pages, so the realized list length N can exceed k. We report six standard information retrieval metrics computed against this list: *Recall@N* (primary), *Precision@N*, *F1@N*, *NDCG@N*, *MAP*, and *MRR*, all at k=10 in the main body. We use k=10 because ViDoRe V3 averages 5.09 ground-truth pages per question; MMDocIR (mean 1.27 ground-truth pages) is unaffected by this choice, and a common k keeps the two benchmarks directly comparable. Unless stated otherwise, a reported value is the *micro* (per-question) average, computed per question then averaged with equal weight over all questions. Because the fused list length grows for agentic scenarios, we report the average fused N alongside *Recall@N*.

4.3 Models and Implementation

All indexes use Amazon Nova Multimodal Embeddings at 1,024 dimensions over OpenSearch Serverless with an HNSW index (cosine similarity). Per-page descriptions are generated by one of two VLMs, Claude Sonnet 4.5 and Qwen3-VL 235B. For the agentic scenarios we sweep five backbone LLMs for the Query Agent: Claude Haiku 4.5, Claude Sonnet 4.6, Claude Opus 4.8, Qwen3-VL 235B, and Qwen3-Next 80B. For the Reflection Agent the orchestrator is fixed to Claude Haiku 4.5 and the `visual_reflection` model is swept across four multimodal models: Haiku 4.5, Sonnet 4.6, Opus 4.8, and Qwen3-VL 235B. We fix the per-call retrieval cutoff at k=10 and cap the Query Agent at two sub-queries and the Reflection Agent at two `visual_reflection` calls and two follow-up retrievals.

5 Results

5.1 Retrieval Tool Optimization (Single-Step Retrieval)

We first evaluate the five embedding configurations in single-step retrieval mode, where each query is embedded once and the top-k nearest neighbours are returned directly. This identifies the optimal retrieval tool that agentic workflows will use.

Table 1 presents the single-step retrieval results. The most striking finding is the dominance of text based embeddings: Desc.-Only under Claude Sonnet 4.5 outperforms Image-Only by 16.7 Recall points on MMDocIR (0.461 to 0.628) and 19.7 points on ViDoRe V3 (0.369 to 0.566), the largest single gain in the entire pipeline. Fusing image and text signals does not help: Image+Desc. trails Desc.-Only by 3.0 (MMDocIR) and 4.7 (ViDoRe) Recall points, suggesting that averaging the two vectors at 1,024 dimensions dilutes the text signal rather than complementing it. VLM quality also proves consequential: Claude Sonnet 4.5 descriptions yield 3.8 (MMDocIR) and 5.0 (ViDoRe) Recall points over Qwen3-VL 235B under identical conditions, a gap attributable solely to description quality, since Image-Only does not depend on the VLM. The three embedding strategies are illustrated in Appendix (Figure 2), and the full per-metric results are repeated in Appendix (Table 7). Based on these results, we fix the Desc.-Only (Claude Sonnet 4.5) index as the retrieval tool for all subsequent agentic experiments.

VLM	Strategy	R@N	NDCG	MAP	MRR
MMDocIR (N=1,658)					
N/A	Image-Only	0.461	0.342	0.297	0.317
Claude Sonnet 4.5	Desc.-Only	**0.628**	**0.505**	**0.455**	**0.486**
	Image+Desc.	0.598	0.478	0.429	0.456
Qwen3-VL 235B	Desc.-Only	0.591	0.470	0.423	0.449
	Image+Desc.	0.595	0.469	0.420	0.447
ViDoRe V3 (N=1,487)					
N/A	Image-Only	0.369	0.321	0.231	0.417
Claude Sonnet 4.5	Desc.-Only	**0.566**	**0.511**	**0.399**	**0.624**
	Image+Desc.	0.519	0.462	0.352	0.572
Qwen3-VL 235B	Desc.-Only	0.516	0.463	0.349	0.588
	Image+Desc.	0.511	0.450	0.338	0.562

Table 1: Single-step retrieval *Recall@N*. **Bold** = best per benchmark. Desc.-Only with Claude Sonnet 4.5 is optimal embedding approach and used in the retrieval tool for agentic retrieval experiments.

visual_reflection	R@N	NDCG	MAP	MRR	Fused N	Cost/q ($)	Time/q (s)
MMDocIR (baseline R@N = 0.628)							
Sonnet 4.6	**0.663**	**0.521**	**0.467**	**0.495**	13.25	0.076	30.3
Opus 4.8	0.659	0.519	0.465	0.492	12.90	0.093	33.5
Qwen3-VL 235B	0.653	0.508	0.452	0.481	13.43	0.044	27.7
Haiku 4.5	0.650	0.505	0.450	0.478	13.34	0.048	24.1
ViDoRe V3 (baseline R@N = 0.566)							
Opus 4.8	**0.644**	0.496	0.364	0.571	18.73	0.080	37.7
Qwen3-VL 235B	0.632	0.490	0.361	0.568	18.12	0.039	34.1
Sonnet 4.6	0.608	**0.503**	**0.379**	**0.596**	15.74	0.055	36.0
Haiku 4.5	0.601	0.498	0.376	0.587	14.69	0.036	22.0

Table 2: Reflection Agent *Recall@N*, with rank-aware metrics (NDCG@N, MAP, MRR) and per-query cost and latency. The orchestrator is fixed to Haiku 4.5; the `visual_reflection` model is swept. Fused N is the average deduplicated list length after follow-up retrievals. **Bold** = best per benchmark (Recall@N determines the leading configuration; the best value in each rank-aware column is also bolded). Per-domain breakdowns are in Appendix Tables 13 and 14.

5.2 Reflection Agent

Having fixed Desc.-Only (Claude Sonnet 4.5) as the optimal embedding strategy (Section 5.1), we evaluate the Reflection Agent on top of it. The orchestrator is fixed to Claude Haiku 4.5 and the `visual_reflection` model is swept across four multimodal models; all configurations share the same index and k=10, so the only variable is the visual reasoning model.

Table 2 reports the results. Recall@N is our primary selection metric; the leading `visual_reflection` model per benchmark is therefore Sonnet 4.6 on MMDocIR and Opus 4.8 on ViDoRe V3. We report NDCG@N, MAP, and MRR alongside for completeness. On ViDoRe V3 these rank-aware metrics favor Sonnet 4.6, but we select Opus 4.8 on the basis of the primary Recall@N metric. On **both** benchmarks, *every* `visual_reflection` model beats the single-step baseline. On MMDocIR the gains range from +0.022 (Haiku 4.5) to +0.034 (Sonnet 4.6); on ViDoRe V3 they range from +0.035 (Haiku 4.5) to +0.079 (Opus 4.8). This is the only configuration that reliably exceeds the non-agentic baseline for every reasoning model on both benchmarks.

The improvement comes from *recovering relevant pages the first pass misses*. When the visual evaluator judges the seed set inadequate, it issues targeted follow-up retrievals that surface ground-truth pages absent from the original top-10, expanding the fused list to 12.90–13.43 pages on average on MMDocIR and 14.69-18.73 on ViDoRe V3. The gain is not merely an effect of the longer list: for example, single-step retrieval at k=13 reaches only 0.648 on MMDocIR, still below the reflection agent's 0.663 at a comparable

fused length of 13 pages. The gain scales with how effectively each evaluator drives this recovery: the two strongest models on ViDoRe (Opus 4.8 and Qwen3-VL 235B) surface the most relevant pages (fused N near 18 to 19). The visual modality is essential: inspecting page images lets the agent detect missing charts, tables, and figures that text-only query rewriting cannot reason about.

The advantage of reflection is not confined to a small retrieval budget. At a larger budget of k=30, where single-step already recovers substantially more pages, adding the Reflection Agent on the same retrieval tool still improves Recall on both benchmarks (0.7095 to 0.7424 on MMDocIR and 0.7075 to 0.7756 on ViDoRe V3; Appendix Table 15), confirming the gain is not an artifact of the small-k regime.

Table 2 reports the per-query cost and latency of each configuration: the Reflection Agent adds one to two multimodal calls, costing $0.036 to $0.093 and 22 to 38 seconds per query. The best evaluator per benchmark is not always the most expensive one: on MMDocIR, Sonnet 4.6 gives the top Recall at $0.076/query, below Opus 4.8 ($0.093), and the cheaper Qwen3-VL and Haiku evaluators retain most of the gain at roughly half the cost while also being faster. On ViDoRe V3, Qwen3-VL 235B achieves the second-best Recall (0.632, close to Opus 4.8's 0.644) while retaining most of the benefit at roughly half the cost ($0.039 vs $0.080). The recall-cost-latency trade-off, not recall alone, can guide the choice of visual reasoning model in deployment.

5.3 Query and Reflection Agent

We next study whether adding a Query Agent in front of the reflection loop yields gains complementary to reflection. On its own, the Query Agent helps only conditionally: on ViDoRe V3, where questions average 5.09 ground-truth pages, Claude Opus 4.8 is the one backbone to beat single-step (0.585, +0.020) by decomposing multi-faceted queries into genuinely separable facets; on MMDocIR, where 81% of questions have a single ground-truth page, no backbone beats the baseline and recall falls as decomposition rises. Full Query-Agent-only results for both benchmarks are reported in Appendix (Tables 11 and 12). We therefore place the Query Agent in front of the reflection loop with a dataset-appropriate backbone. On ViDoRe V3 we use Opus 4.8, its only backbone to beat single-step. On MMDocIR the choice is more deliberate:

Configuration	R@N	Fused N
MMDocIR		
Reflection (vr Sonnet 4.6)	0.663	13.25
Query (QA Sonnet 4.6) + Reflection (vr Sonnet 4.6)	0.653	14.43
ViDoRe V3		
Reflection (vr Opus 4.8)	0.644	18.73
Query (QA Opus 4.8) + Reflection (vr Opus 4.8)	**0.651**	20.51

Table 3: Query and Reflection Agent *Recall@N*, compared against the best reflection-alone configuration. QA denotes the Query Agent backbone. The Query Agent and the `visual_reflection` model are both set to the dataset-matched best (Sonnet 4.6 on MMDocIR, Opus 4.8 on ViDoRe V3); the orchestrator is Haiku 4.5. **Bold** = best per benchmark.

the models that match the single-step baseline (the Qwen backbones) do so precisely because they almost never decompose (Appendix Table 10), leaving the query as-is, so pairing them with reflection would simply reproduce reflection-alone and tell us nothing. We therefore use Sonnet 4.6, the Claude backbone that decomposes most often (22.6% of queries), which provides a genuine test of whether feeding the reflection loop a heavily decomposed seed set helps on this single-page benchmark.

Table 3 shows that composing the two agents does not improve substantially on reflection agent alone. On MMDocIR, the Query Agent feeds a more decomposed seed set into the reflection loop but reaches 0.653, just below reflection-alone (0.663): the extra decomposition does not help when most questions have a single relevant page. On ViDoRe V3, the composed pipeline reaches 0.651, marginally above reflection-alone (0.644, +0.007) and above the Query Agent alone (0.585), since the dataset rewards both decomposition and visual recovery. In both cases the reflection loop accounts for nearly all of the agentic gain, and the Query Agent contributes at most a small increment. The practical implication is that the Reflection Agent is the component to invest in; a query decomposition layer is optional and helps only when the queries have multi-faceted information needs.

Table 4 traces the cumulative gains. Two design choices compound. First, optimizing the embedding turns a weak image-only index into a strong retrieval tool (+0.167 on MMDocIR, +0.197 on ViDoRe V3). Second, on top of this optimized tool the Reflection Agent recovers pages the first

Stage	Best config.	R@N
MMDocIR		
Single-step (Image-Only)	N/A	0.461
Single-step (Desc.-Only)	Claude Sonnet 4.5	0.628
Query Agent	Qwen3-VL/Next	0.628
Reflection Agent	**vr Sonnet 4.6**	**0.663**
Query + Reflection	QA Sonnet 4.6, vr Sonnet 4.6	0.653
ViDoRe V3		
Single-step (Image-Only)	N/A	0.369
Single-step (Desc.-Only)	Claude Sonnet 4.5	0.566
Query Agent	Opus 4.8	0.585
Reflection Agent	vr Opus 4.8	0.644
Query + Reflection	**QA Opus 4.8, vr Opus 4.8**	**0.651**

Table 4: Progression from the weakest baseline to the best configuration on each benchmark, reporting *Recall@N*. For Query + Reflection, QA denotes the Query Agent backbone (the `visual_reflection` model is the dataset-best, as in Table 3). Optimizing the embedding (the Desc.-Only index lifts Recall by +0.167 on MMDocIR and +0.197 on ViDoRe V3 over Image-Only); on top of this optimized retrieval tool, the Reflection Agent adds a further, consistent gain on both benchmarks, while the additional Query Agent helps conditionally.

pass still misses, and improves Recall for every reasoning model on both benchmarks; the additional Query Agent contributes a smaller, conditional gain. The full REFLECT AND RETRIEVE pipeline therefore pairs the Reflection Agent with an optimized retrieval tool, and a Query Agent in front.

6 Conclusion

We present REFLECT AND RETRIEVE, an agentic retrieval approach for multimodal document search built around a multimodal Reflection Agent that visually reasons over retrieved page images and recovers content the first pass misses. Through systematic evaluation on two page-level benchmarks (MMDocIR and ViDoRe V3), we identify three actionable findings for practitioners deploying multimodal retrieval systems. First, *the retrieval tool must be optimized first*: converting document pages to VLM-generated text descriptions before embedding yields +16.7 to +19.7 Recall points over image-only embeddings, and higher-quality VLMs produce better descriptions that directly improve retrieval. Second, the *Reflection Agent* on top of the optimized retrieval tool, improves Recall (by +0.034 on MMDocIR and +0.079 on ViDoRe V3) and is the only configuration to beat single-step retrieval for every reasoning model on both benchmarks, by visually inspecting retrieved pages and issuing targeted follow-up retrievals for missing content. Single-step retrieval has no way to recognize or repair an incomplete result set, whereas the Reflection Agent consistently recovers relevant pages it leaves behind. Third, *the Query Agent is a complementary addition*: query decomposition helps when the corpus has multi-page ground truths requiring sub-queries, and adds a smaller increment on top of the reflection loop. Together, REFLECT AND RETRIEVE uses a Reflection Agent with an optimized retrieval tool, with an optional Query Agent in front, and we recommend adopting them in that order.

Limitations

Our evaluation uses English-language documents for a controlled comparison. The pipeline itself is language-agnostic: the query and reflection agents and the visual reflection tool are not English-specific, and the same setup applies by indexing documents in other languages; we leave multilingual evaluation to future work. We have not explored alternative fusion strategies (for example, reciprocal rank fusion or learned re-ranking) or late-interaction retrieval models that may further improve performance of the retrieval tool. All experiments use a single embedding model (Amazon Nova Multimodal Embeddings at 1,024 dimensions); results may differ with other embedding providers or higher-dimensional representations.

Ethical Considerations

All experiments use publicly available benchmarks (MMDocIR, ViDoRe V3) with no personally identifiable information. The quantitative results reported here use offline public benchmarks (MMDocIR, ViDoRe V3) without live user traffic; the architecture itself is derived from production multimodal-search deployments, and the offline protocol is chosen for reproducibility on non-proprietary data. We acknowledge that VLM-generated descriptions may introduce biases present in the underlying language models; however, since descriptions are used solely as embedding inputs (not surfaced to end users), the risk of downstream harm is limited in this context.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jaemin Cho, Debanjan Yoon, Minhyuk Ji, Seonghyeon Lee, Sungdong Yoon, and Minjoon Seo. 2025. M3docvqa: Multi-modal multi-page multi-document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*.

Kuicai Dong, Yujing Chang, Derrick Goh Xin Deik, Dexun Li, Ruiming Tang, and Yong Liu. 2025a. MMDocIR: Benchmarking multimodal retrieval for long documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30971–31005, Suzhou, China. Association for Computational Linguistics.

Kuicai Dong, Yujing Chang, Shijie Huang, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025b. Benchmarking retrieval-augmented multimodal generation for document question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. 2025. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zirui Li, Siwei Wu, Yizhi Li, Xingyu Wang, Yi Zhou, and Chenghua Lin. 2025. Docmmir: A framework for document multi-modal information retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13117–13130.

António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. 2026. Vidore v3: A comprehensive evaluation of retrieval augmented generation in complex real-world scenarios. *arXiv preprint arXiv:2601.08620*.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. 2024. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6492–6505.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*.

A Appendix

This appendix provides dataset statistics (§A.1), the embedding strategies and VLM prompt (§A.2), full single-step retrieval metrics and per-domain results (§A.3), and the agentic workflows, prompts, and per-domain results for the Query and Reflection agents (§A.4–A.5).

A.1 Dataset Statistics

Tables 5 and 6 report the per-domain question and page counts for MMDocIR and the ViDoRe V3 English subset. MMDocIR is dominated by Academic Paper, Financial Report, and Research Report (about 56% of questions) and ViDoRe V3 is more evenly balanced across its five enterprise domains.

Domain	**#Q**	**%**
Academic Paper	389	23.5
Financial Report	344	20.7
Research Report / Intro	194	11.7
News	137	8.3
Laws	132	8.0
Guidebook	115	6.9
Government	111	6.7
Tutorial/Workshop	104	6.3
Brochure	76	4.6
Admin./Industry	56	3.4
Total	**1,658**	100

Table 5: Distribution of evaluation questions across document domains in the MMDocIR benchmark (N=1,658).

Domain	**#Q**	**#Pages**
Pharmaceuticals (FDA)	362	2,313
HR (EU Reports)	318	1,110
Finance (U.S. Banks)	309	2,942
Industrial (USAF)	283	5,244
Computer Science	215	1,360
Total	**1,487**	**12,969**

Table 6: Distribution of evaluation queries and corpus pages across domains in the ViDoRe V3 benchmark (English-only subset, N=1,487).

A.2 Embedding Strategies and VLM Prompt

Figure 2 illustrates the three page-embedding strategies compared in Section 3.1. The structured prompt used to generate page descriptions for the Desc.-Only and Image+Desc. strategies is given below; it is held constant across both VLMs.

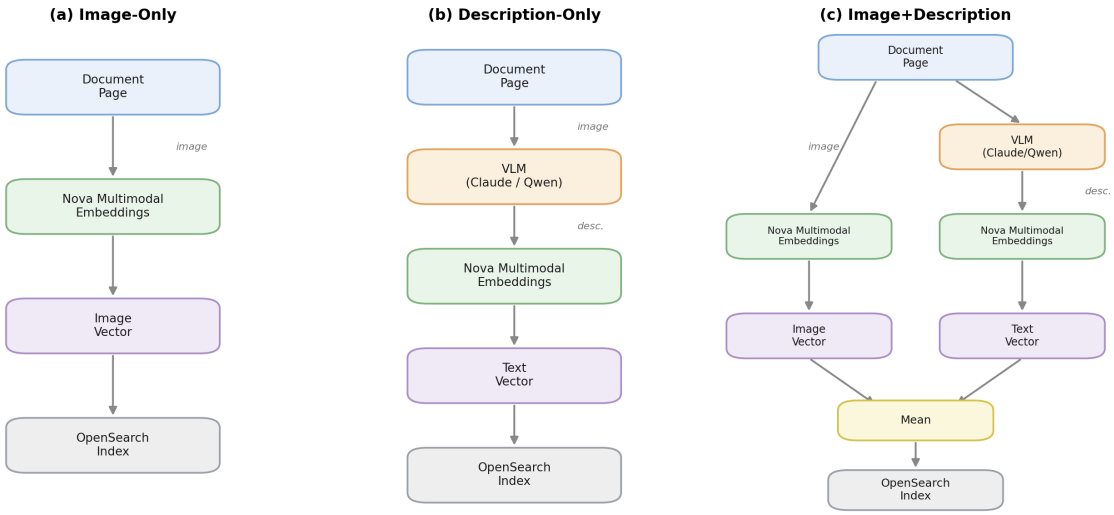


Figure 2: Three embedding strategies for page-level document indexing. (a) Image-Only embeds the raw page image directly. (b) Desc.-Only first generates a textual description with a VLM, then embeds the text. (c) Image+Desc. embeds the image and the description separately through Amazon Nova Multimodal Embeddings, then averages the two vectors.

VLM description generation prompt. For the Desc.-Only and Image+Desc. strategies, each page is described by a VLM using a structured, domain-agnostic prompt held constant across both VLMs, so that performance differences are attributable to the VLM rather than the task specification. Full prompt is reproduced as follow.

[System] You are a professional document analyst working for a research institution's digital archive. Your role is to create objective, factual descriptions of document page images for building a searchable document index. This work supports document research and retrieval systems.

You provide neutral, comprehensive descriptions without interpretation or judgment. Your descriptions focus solely on observable content to enable text-based search and retrieval of document pages.

[User] Analyze this document page image and provide a comprehensive, factual description.

DESCRIPTION STRUCTURE:

1. Document Overview:
- Document type (research report, tutorial, presentation slide, financial statement, legal contract, technical manual, etc.)
- Page number if visible
- Overall layout and design elements (single/multi-column, sidebars, text boxes)

2. Textual Content:
- All headings, titles, and section headers (transcribe verbatim)
- Key textual information including body text, captions, and labels
- Quotes, statistics, or highlighted information
- Footnotes, references, citations, or legal disclaimers if present

3. Visual Elements:
- Charts, graphs, tables, and diagrams with detailed descriptions
- Chart types (bar, line, pie, scatter, etc.) and represented data
- Technical diagrams, flowcharts, or process illustrations
- Colors, legends, axes labels, and data ranges
- Images, photographs, maps, or illustrations

```
  - Logos, icons, or graphical elements
4. Tables (if present):
  - Structure: number of rows and columns, headers
  - What each column represents
  - Key numerical values, trends, or patterns
  - Totals, subtotals, or summary rows
  - Cell formatting, highlights, or special notations
5. Data and Insights:
  - Specific numerical values, percentages, or trends
  - Comparisons, correlations, or key findings presented
  - Financial figures or legal references if applicable
GUIDELINES:
  - Maintain objectivity; describe observable content only
  - Use precise language for colors and positions (top-left, center, bottom-right)
  - Include verbatim text for titles and headings
  - For tables: describe structure AND key data comprehensively
  - For charts: specify axis representations and notable data points
  - Ensure description enables understanding without viewing the image
  - Pay special attention to multi-modal elements (text + images + tables)
OUTPUT: Provide a cohesive 500-word description integrating all elements while following the
structure above.
```

A.3 Single-Step Retrieval: Full Metrics and Per-Domain Results

Table 7 reports all six metrics for the five single-step configurations (the main text shows Recall@N only). Tables 8 and 9 then give the per-domain Recall@N breakdown with macro and micro averages.

Benchmark	VLM	Strategy	R@N	P@N	F1@N	NDCG@N	MAP	MRR
MMDocIR (N=1,658)	N/A	Image-Only	0.461	0.057	0.099	0.342	0.297	0.317
	Claude Sonnet 4.5	Desc.-Only	**0.628**	**0.075**	**0.132**	**0.505**	**0.455**	**0.486**
		Image+Desc.	0.598	0.073	0.127	0.478	0.429	0.456
	Qwen3-VL 235B	Desc.-Only	0.591	0.070	0.123	0.470	0.423	0.449
		Image+Desc.	0.595	0.072	0.126	0.469	0.420	0.447
ViDoRe V3 (N=1,487)	N/A	Image-Only	0.369	0.127	0.162	0.321	0.231	0.417
	Claude Sonnet 4.5	Desc.-Only	**0.566**	**0.193**	**0.249**	**0.511**	**0.399**	**0.624**
		Image+Desc.	0.519	0.177	0.227	0.462	0.352	0.572
	Qwen3-VL 235B	Desc.-Only	0.516	0.177	0.227	0.463	0.349	0.588
		Image+Desc.	0.511	0.174	0.223	0.450	0.338	0.562

Table 7: Single-step retrieval on both benchmarks (micro average). **Bold** = best per metric within each benchmark. Desc.-Only with Claude Sonnet 4.5 is optimal across all metrics and serves as the fixed embedding foundation for the agentic experiments (§5.2).

Tables 8 and 9 give the per-domain single-step Recall@N at k=10 for all five embedding configurations, with micro (equal question weight) averages. The Desc.-Only (Claude Sonnet 4.5) strategy dominates in nearly every domain on both benchmarks.

A.4 Query Agent: Workflow, Prompt, and Results

Figure 3 shows the Query-Agent-only workflow, followed by key prompt elements and the overall and per-domain results (Tables 10, 11, 12).

Domain	#Q	Img	Claude Sonnet 4.5		Qwen3-VL 235B	
			Desc	I+D	Desc	I+D
Academic Paper	389	0.428	**0.660**	0.633	0.614	0.646
Admin./Industry	56	0.304	0.327	0.384	0.271	**0.390**
Brochure	76	0.484	0.572	**0.595**	0.507	0.543
Financial Report	344	0.346	**0.595**	0.526	0.554	0.527
Government	111	0.405	**0.613**	0.568	0.577	0.523
Guidebook	115	0.617	0.713	**0.731**	0.627	0.707
Laws	132	0.636	**0.727**	0.712	0.697	0.720
News	137	0.095	**0.380**	0.277	**0.380**	0.256
Research Report	194	0.746	0.773	0.764	0.765	**0.774**
Tutorial/Workshop	104	0.653	**0.679**	0.672	0.652	0.677
Macro avg.	N/A	0.471	**0.604**	0.586	0.564	0.576
Micro avg.	1,658	0.461	**0.628**	0.598	0.591	0.595

Table 8: Single-Step Retrieval. MMDocIR per-domain Recall@N. Img = Image-Only, Desc = Desc.-Only, I+D = Image+Desc.. **Bold** = best of the four description-based columns per row.

Domain	#Q	Img	Claude Sonnet 4.5		Qwen3-VL 235B	
			Desc	I+D	Desc	I+D
Finance	309	0.252	**0.577**	0.500	0.550	0.495
HR	318	0.336	**0.563**	0.508	0.482	0.477
Industrial	283	0.230	0.381	0.345	**0.389**	0.377
Pharmaceuticals	362	0.517	**0.630**	0.604	0.551	0.571
Computer Science	215	0.521	**0.689**	0.648	0.626	0.657
Macro avg.	N/A	0.371	**0.568**	0.521	0.520	0.515
Micro avg.	1,487	0.369	**0.566**	0.519	0.516	0.511

Table 9: Single-Step Retrieval. ViDoRe V3 per-domain Recall@N. Img = Image-Only, Desc = Desc.-Only, I+D = Image+Desc.. **Bold** = best of the four description-based columns per row.

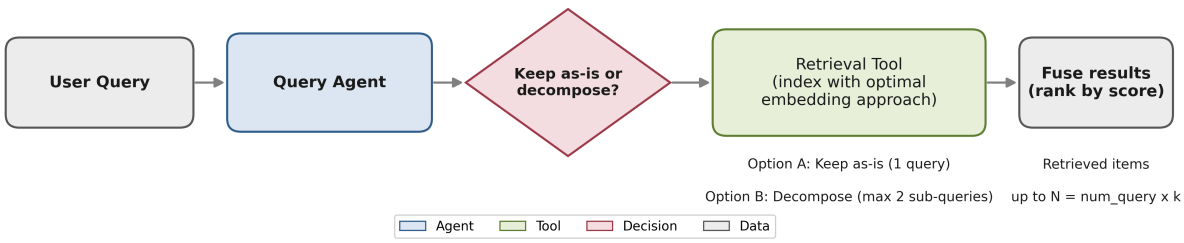


Figure 3: Query Agent Only workflow. The user query is rewritten and optionally decomposed into at most two sub-queries, each sent to the retrieval tool; the returned lists are fused and ranked by score.

Query Agent prompt. The full system prompt for the Query Agent is reproduced as follow.

Role and Purpose. You are a Query Agent. Your ONLY purpose is to call the retrieval tool over a document index and return the raw results. You do not answer the user's question. You do not summarize. You do not synthesize. You retrieve.

Hard rules (never violate).

1. You MUST call the retrieval tool at least once on EVERY user query.
2. You MUST NOT answer the user's question from your own knowledge, even if you know the answer or the question seems trivial. Always defer to retrieval.
3. You MUST NOT skip the tool call because the question looks unanswerable, ambiguous, or out of scope. Call retrieval anyway and return whatever it returns.
4. The user's query is treated as a search query, not a question to be answered.

Two retrieval options.

(A) DEFAULT: pass the user's query VERBATIM as a single retrieval query.
(B) DECOMPOSE: split the user's query into AT MOST 2 sub-queries, when it has clearly independent retrievable facets. There is no third option; you may NOT rephrase, paraphrase, fix typos, resolve pronouns, reword, condense, expand, or translate the query. If not decomposing, pass it byte-for-byte unchanged.

Workflow.

Step 1, Decide verbatim or decompose. If decomposing, produce at most two complete natural-language sub-queries that together cover the original intent, and do not include the verbatim query as one of them.
Step 2, Retrieve. Call the retrieval tool once per planned query (one call for (A), at most two for (B)). Do not summarize, modify, or filter the results.
Step 3, Return. Present the complete retrieved results exactly as returned by the tool, as a JSON array with one {query, results} element per retrieval call.

Constraints. Always call retrieval at least once and at most twice; never answer from your own knowledge; never fabricate results (if retrieval is empty, return the empty envelope); do not modify, reorder, or summarize retrieved items.

Overall results. Table 10 reports the overall Query-Agent-only Recall@N on both benchmarks. On MMDocIR no backbone beats the single-step baseline, and Recall falls as the decomposition rate rises; on ViDoRe V3 only Claude Opus 4.8 beats the baseline (+0.020).

Model	R@N	Decomp.	Fused N
MMDocIR (baseline R@N = 0.628)			
Qwen3-VL 235B	0.628	0.12%	10.00
Qwen3-Next 80B	0.628	0.06%	10.00
Claude Opus 4.8	0.620	9.83%	10.47
Claude Sonnet 4.6	0.610	22.62%	11.10
Claude Haiku 4.5	0.610	3.38%	10.20
ViDoRe V3 (baseline R@N = 0.566)			
Claude Opus 4.8	**0.585**	41.36%	12.28
Qwen3-Next 80B	0.567	3.09%	10.22
Qwen3-VL 235B	0.566	1.28%	10.06
Claude Sonnet 4.6	0.562	54.40%	12.88
Claude Haiku 4.5	0.547	14.93%	10.98

Table 10: Query-Agent-only Recall@N on both benchmarks, on the Desc.-Only (Claude Sonnet 4.5) index. Decomp. is the fraction of questions decomposed into two or more sub-queries; Fused N is the average deduplicated list length. **Bold** = best per benchmark.

Domain	#Q	Opus 4.8	Sonnet 4.6	Haiku 4.5	Qwen3-VL	Qwen3-Next
Academic Paper	389	0.637	0.616	0.596	**0.660**	**0.660**
Admin./Industry	56	**0.330**	0.304	0.309	0.327	0.327
Brochure	76	0.572	**0.576**	0.564	0.572	0.572
Financial Report	344	0.588	0.588	**0.599**	0.595	0.592
Government	111	0.595	0.559	0.595	**0.613**	**0.613**
Guidebook	115	0.707	**0.725**	0.710	0.708	0.713
Laws	132	0.720	0.720	0.720	**0.727**	**0.727**
News	137	**0.380**	0.372	**0.380**	**0.380**	**0.380**
Research Report	194	**0.775**	0.774	0.756	0.770	0.773
Tutorial/Workshop	104	0.679	0.645	**0.694**	0.679	0.679
Macro avg.	N/A	0.598	0.588	0.592	0.603	**0.604**
Micro avg.	1,658	0.620	0.610	0.610	**0.628**	**0.628**

Table 11: MMDocIR per-domain Recall@N by query-agent only workflow. All configurations use Description-Only (Sonnet 4.5) as the embedding index. Single-step baseline: micro avg. 0.628. No query-agent model surpasses the baseline on aggregate; models that rarely decompose (Qwen variants, $<0.2\%$ decomposition rate) most closely reproduce baseline performance.

Domain	#Q	Opus 4.8	Sonnet 4.6	Haiku 4.5	Qwen3-VL	Qwen3-Next
Finance (U.S. Banks)	309	**0.603**	0.562	0.555	0.577	0.578
HR (EU Reports)	318	**0.584**	0.570	0.547	0.563	0.566
Industrial (USAF)	283	**0.393**	0.368	0.356	0.376	0.376
Pharmaceuticals (FDA)	362	**0.650**	0.621	0.608	0.633	0.633
Computer Science	215	0.705	**0.706**	0.686	0.691	0.691
Macro avg.	N/A	**0.587**	0.565	0.550	0.568	0.569
Micro avg.	1,487	**0.585**	0.562	0.547	0.566	0.567

Table 12: ViDoRe V3 per-domain Recall@N by query-agent only workflow. All configurations use Description-Only (Sonnet 4.5) as the embedding index. Single-step baseline: micro avg. 0.566. Claude Opus 4.8 is the only model to beat single-step (+0.020), winning 4 of 5 domains.

A.5 Reflection Agent: Workflow, Prompts, and Per-Domain Results

Figure 4 shows the standalone Reflection-Agent workflow. We then give the orchestrator and `visual_reflection` prompts, and the per-domain Recall@N for each evaluator model (Tables 13, 14).

Reflection Agent (orchestrator) prompt. The full system prompt for the Reflection Agent orchestrator is reproduced below.

Role and Purpose. You are the Reflection Agent. You orchestrate a reflect-and-retrieve loop on top of an initial set of retrieved document pages. Your goal is to ensure the final result set is sufficient to answer the user's query. Do not answer the user's query yourself, and do not summarize, reorder, or fabricate documents; your only job is to decide whether to retrieve more and to merge the results.

Tools.

visual_reflection(query, retrieved_results): sends the current retrieved page images plus the user's query to a visual evaluator; returns a verdict (ADEQUATE or NOT ADEQUATE), reasoning, and up to two suggested sub-queries when gaps exist.

retrieval(query): kNN search over the document index for a single query string, returning the top matching pages.

Workflow.

Step 1, Reflect: call visual_reflection on the current retrieved set with the user's query.

Step 2, Decide: if the verdict is ADEQUATE, stop and return the current set unchanged; if NOT

ADEQUATE, proceed.
Step 3, Retrieve gaps: for each suggested sub-query (up to two), call retrieval once (at most two calls).
Step 4, Reflect again (optional): you may call visual_reflection once more on the augmented set to verify the gaps are closed.
Step 5, Return: return the union of the original and newly retrieved pages, exactly as retrieved.

Budgets and constraints. At most 2 visual_reflection calls and 2 retrieval calls per run. Never call retrieval without a NOT ADEQUATE verdict; never fabricate sub-queries (use those returned by visual_reflection); never modify, reorder, merge, or summarize retrieved pages; never answer the query yourself.

Visual Reflection tool prompt. The full system prompt for the `visual_reflection` tool is reproduced below.

Role and Purpose. You are a Visual Reflection Evaluator. You receive a user's query and a set of retrieved document pages (as images, possibly with text snippets). Your job is to look at the page images and judge whether they are sufficient to answer the query, and if not, identify exactly what is missing. You do not retrieve documents, rewrite queries, or answer the user's question; you only evaluate adequacy.

Evaluation criteria. Relevance (do the pages address the core topic?), completeness (do they cover every key aspect and sub-question?), specificity (specific enough, not too general?), consistency (coherent, non-contradictory?), visual evidence (do charts, tables, or figures support the query?), and sufficiency (enough detail for a well-supported answer?).

Decision. Output ADEQUATE if the pages suffice, otherwise NOT ADEQUATE.

Output format.
VERDICT: ADEQUATE | NOT ADEQUATE
REASONING: query intent (one sentence), coverage (aspects covered), gaps (what is missing, vague, or insufficient).
If NOT ADEQUATE, also SUGGESTED SUB-QUERIES: up to two, each targeting a distinct gap, rephrased with synonyms or decomposed, not repeating the original query.

Constraints. Never fabricate page content; never answer the query yourself; never modify, reorder, or summarize the retrieved pages; always ground reasoning in what is visible on the pages.

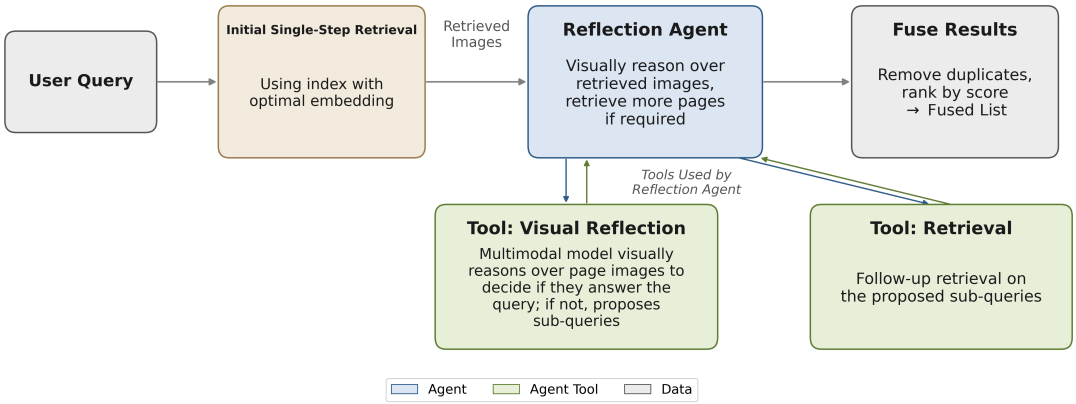


Figure 4: Reflection Agent Only workflow. An initial single-step retrieval, using the index with the optimal embedding, returns page images that are sent to the Reflection Agent. The agent calls a `visual reflection` tool, which uses a multimodal model to visually reason over the page images to decide whether they answer the query and, if not, proposes sub-queries; a `retrieval` tool then issues follow-up retrievals on those sub-queries. All retrieved items are fused by removing duplicates and ranking by score into the final fused list.

Domain	#Q	Sonnet 4.6	Opus 4.8	Qwen3-VL	Haiku 4.5
Academic Paper	389	**0.690**	0.678	0.671	0.673
Admin./Industry	56	**0.414**	0.399	0.381	0.381
Brochure	76	0.623	0.602	**0.632**	0.599
Financial Report	344	0.612	0.617	0.615	**0.621**
Government	111	**0.631**	**0.631**	0.613	0.622
Guidebook	115	**0.788**	0.777	0.762	0.762
Laws	132	0.727	**0.735**	**0.735**	0.727
News	137	**0.445**	**0.445**	0.423	0.416
Research Report	194	**0.796**	0.795	0.792	0.795
Tutorial/Workshop	104	0.739	**0.740**	0.726	0.694
Macro avg.	N/A	**0.646**	0.642	0.635	0.629
Micro avg.	1,658	**0.663**	0.659	0.653	0.650

Table 13: MMDocIR per-domain Recall@N by Reflection Agent Only workflow. Orchestrator fixed to Claude Haiku 4.5. Single-step baseline: micro avg. 0.628. Every evaluator beats single-step (+0.022 to +0.034); Sonnet 4.6 leads on 6 of 10 domains and achieves the highest micro average.

Domain	#Q	Opus 4.8	Qwen3-VL	Sonnet 4.6	Haiku 4.5
Finance	309	**0.659**	0.651	0.621	0.615
HR	318	**0.636**	0.628	0.620	0.610
Industrial	283	**0.441**	0.437	0.416	0.401
Pharmaceuticals	362	**0.719**	0.693	0.665	0.662
Computer Science	215	**0.779**	0.767	0.725	0.724
Macro avg.	N/A	**0.647**	0.635	0.610	0.603
Micro avg.	1,487	**0.644**	0.632	0.608	0.601

Table 14: ViDoRe V3 per-domain Recall@N by Reflection Agent Only workflow. Orchestrator fixed to Claude Haiku 4.5. Single-step baseline: micro avg. 0.566. Every evaluator beats single-step; Opus 4.8 leads on every domain and achieves the highest micro average.

Dataset	Config	Single-step	Reflection	Δ
MMDocIR	vr = Sonnet 4.6	0.7095	**0.7424**	+0.0329
ViDoRe V3	vr = Opus 4.8	0.7075	**0.7756**	+0.0681

Table 15: Reflection Agent Only vs. single-step at a larger retrieval budget (k=30), Recall@N. Orchestrator fixed to Claude Haiku 4.5; the `visual_reflection` model is the best per dataset. The advantage of reflection over single-step *persists at higher* k: even when single-step already retrieves 30 pages, adding the Reflection Agent on the same retrieval tool improves Recall on both benchmarks (+0.0329 on MMDocIR, +0.0681 on ViDoRe V3), confirming the gain is not an artifact of the small-k regime.

LLMs and Reasoning in Health Insurance Policy Documents: A Dataset and Argument Graph Extraction Study

Govind Arun[1] **Sidharth N Krishna**[2] **Susie Xi Rao**[3,4] **Ajo Babu George**[5]

[1]University of Maryland, College Park, MD, USA
[2]Indira Gandhi National Open University, New Delhi, India
[3]School of Engineering and Computer Science, Bern University of Applied Sciences;
[4]Department of Management, Technology and Economics (MTEC) & AI Center, ETH Zurich, Switzerland
[5]DiceMed, Cuttack, Odisha, India

govind123.ga@gmail.com sid4official@gmail.com
srao@ethz.ch drajobabu7@gmail.com

Abstract

Health insurance policy documents justify coverage decisions through clinical evidence and regulatory criteria, but the underlying reasoning is buried in dense medical and legal text. A pipeline is presented that extracts argument components and relations from these documents and assembles them into directed argument graphs. A dataset of 5,230 sentences and 5,685 relations across 53 documents from Aetna, CMS, and United Healthcare is constructed along with two custom annotation tools. Six classical baselines, three fine-tuned transformer models, and three instruction-tuned LLMs are evaluated, with the LLMs assessed under zero-shot, few-shot, and chain of thought style prompting. **PubMedBERT** and **Legal-BERT**, fine-tuned on the dataset, achieve the strongest results overall (0.731 and 0.653 Macro F1 on component classification and relation extraction, respectively). Notably, the much simpler Term Frequency–Inverse Document Frequency (TF-IDF) + Support Vector Machine (SVM) baseline is competitive with these fine-tuned transformers (0.709 and 0.649 Macro F1) and outperforms the best prompted small (4-7B parameter) instruction-tuned LLM by 12 and 21 F1 points respectively. The dataset and code are available at https://github.com/GIND123/LLM-Driven-Argument-Mapping.

1 Introduction

Health insurance policy documents contain structured reasoning in which coverage decisions are justified through clinical evidence and policy rules. Argument mining, the automatic identification of claims, premises, and their relations, has been applied to persuasive essays (Stab and Gurevych, 2017) and legal texts (Peldszus and Stede, 2015), but health insurance policies remain largely unexplored. The central question is whether instruction-tuned LLMs can match classical feature based methods for argument mining in formulaic institutional text, and whether reason-oriented style prompting narrows the gap. The contributions are:

- A dataset of 53 policy documents with 5,230 component labels and 5,685 relation annotations (Krippendorff's $\alpha = 0.79$ components, $\alpha = 0.74$ relations)
- Two web-based annotation tools for sentence level component labelling and Sentence-BERT based relation candidate ranking.
- Systematic comparison of six classical baselines, three fine-tuned transformer models, and three LLMs under zero-shot, few-shot, and CoT-style prompting (Wei et al., 2022), showing that fine-tuned transformers achieve the strongest results overall, while TF-IDF pipelines remain competitive and substantially outperform the tested 4–7B instruction-tuned LLMs.

2 Related Work

Stab and Gurevych (2017) established claim premise identification in persuasive essays, reporting Macro F1 of approximately 0.73 for components and 0.58 for relations. Peldszus and Stede (2015) achieved comparable results (0.60–0.70 F1) on microtext corpora. Surveys by Lippi and Torroni (2016) and Lawrence and Reed (2020) identify institutional domains as underexplored. Chain of thought prompting (Wei et al., 2022) has improved LLM reasoning and recent work has explored reasoning structure and uncertainty in clinical prediction (Arun et al., 2026), as well as increasingly sophisticated forms of social, strategic,

The Proceedings of Workshop on Document Intelligence and Understanding, pages 155–162
October 24, 2026 ©2026 Association for Computational Linguistics

and multi-agent reasoning in LLMs (Wang et al., 2026). However, domain-specific lexical features remain competitive in formulaic domains (Mosbach et al., 2023). Recent comparative work in health-related text classification has similarly evaluated traditional machine learning and language model approaches (Lee et al., 2026). Health insurance documents exemplify this, with recurring phrases such as "considered experimental and investigational" serving as strong classification signals.

3 Dataset and Annotation

3.1 Data Sources

Documents are collected from three sources: **Aetna Clinical Policy Bulletins** (Aetna, 2024), **CMS National Coverage Determinations** (Centers for Medicare & Medicaid Services, 2024), and **United Healthcare medical policies** (UnitedHealthcare, 2024). These documents are medical necessity policy documents, formal institutional texts that define the clinical criteria under which specific procedures, treatments, or diagnostic tests are covered or denied.

3.2 Annotation Tools and Scheme

Three annotators labeled the corpus: a physician with ten years of clinical experience, an AI/ML research engineer, and an applied statistics graduate student with health informatics experience, ensuring both domain expertise in interpreting clinical policy language and technical rigor in applying categorical labels consistently.

A Claim asserts a coverage decision or policy determination (e.g., "For Aduhelm, member must be enrolled in a clinical trial"). A Premise provides clinical evidence or diagnostic criteria supporting or challenging a claim (e.g., "It is a neurologic condition characterized by loss of mental ability severe enough to interfere with normal activities of daily living, lasting at least six months and not present from birth"). A Non-Argumentative sentence contains background, procedural, or inconclusive research content that does not directly support or oppose any coverage decision (e.g., "However, unless prospective studies are conducted, the issue remains open whether mean diffusivity is better as early predictor of AD than volume").

Sentence-level annotation is adopted following established practice in argument mining for structured domains such as clinical trials (Mayer et al., 2020) and legal judgments (Poudyal et al., 2020). Health insurance policy documents are written in a formal, regulatory style where sentences typically express a single argumentative function. A manual inspection of 532 sentences across two representative documents revealed that fewer than 13% of sentences contained discourse connectives potentially signalling multiple argument components and manual review confirmed the majority of these still expressed a single unified argumentative function. This supports sentence-level granularity as appropriate for this domain, with sub-sentential ambiguity acknowledged as a known limitation (Section 9).

The **Argument Component Annotation Tool** loads policy PDFs, segments text using spaCy, and presents each sentence for labeling. The **Argument Relation Annotation Tool** generates candidate premise–claim pairs via Sentence-BERT (SBERT) cosine similarity (Reimers and Gurevych, 2019), retaining the top 3 claims per premise. Annotators assign SUPPORT, ATTACK, or NONE. Annotators labeled only the retrieved top-3 candidates and could not introduce premise–claim pairs outside this set. Consequently, gold relations are conditioned on SBERT retrieval, and neither candidate-generation recall against an exhaustive relation set nor full argument-graph completeness is evaluated in this work (see Limitations). Existing tools such as BRAT do not natively support cross-sentence relation annotation with a document-level DAG view; the custom tools were designed to streamline the two-stage workflow and integrate directly with the dataset format and export pipeline.

Inter-annotator agreement across all three annotators, computed on 8 randomly sampled documents, yields Krippendorff's $\alpha = 0.79$ for component classification and $\alpha = 0.74$ for relation labeling. The lower α on relations reflects the inherent difficulty of distinguishing ATTACK from SUPPORT when clinical evidence language is similar. Disagreements were resolved by majority vote under the supervising researcher's adjudication.

3.3 Dataset Statistics

Table 1 summarises the corpus. PREMISE dominates component labels (56%), reflecting extensive clinical evidence supporting fewer coverage decisions. SUPPORT relations form the majority (58%), followed by NONE relations (31%), while ATTACK relations remain relatively rare (10%) typically appearing in policies where conflicting

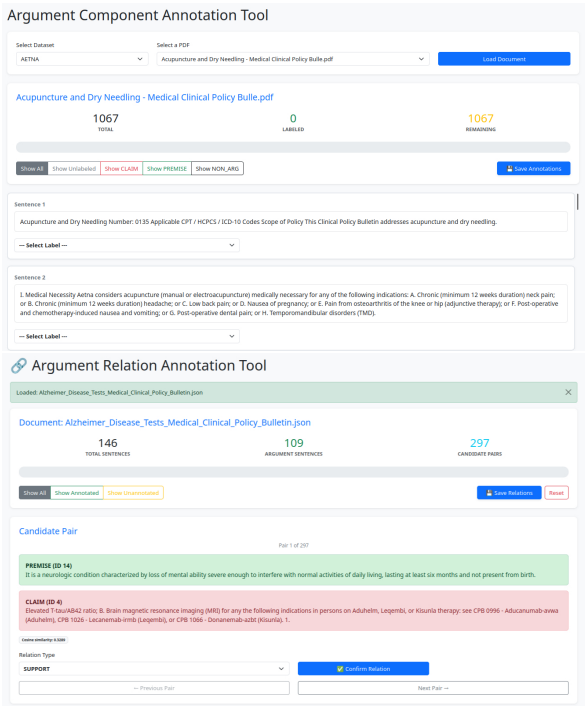


Figure 1: Argument Component Annotation Tool (top) and Argument Relation Annotation Tool with SBERT ranked candidate pairs (bottom).

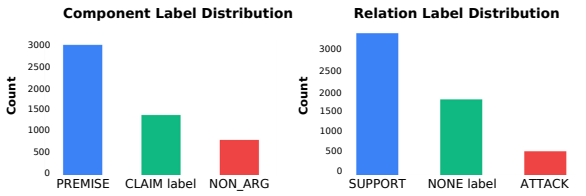


Figure 2: Distribution of argument component labels (left) and relation labels (right) in the dataset.

clinical evidence is discussed.

4 Methodology

Argument Component Classification: Each sentence is classified as CLAIM, PREMISE, or NON_ARGUMENTATIVE. Classical baselines apply Logistic Regression (LR), Linear Support Vector Machine (SVM), and Random Forest (RF) on TF-IDF vectors (unigrams to trigrams, 20K features) or SBERT embeddings (Reimers and Gurevych, 2019), with balanced class weights. LLM classifiers use instruction-tuned models (Section 5).

Relation Candidate Generation and Classification: SBERT cosine similarity retains the top 3 most similar claims per premise within each document. Biomedical pretrained language models have also been explored for relation extraction in specialized biomedical text (Bhasuran, 2022). Each pair is classified as SUPPORT, ATTACK, or NONE using the same model families, with input as `Source: [premise] Target: [claim]`.

Statistic	Components	Relations
Documents	53	53
Total instances	5,230	5,685
CLAIM / SUPPORT	1,414	3,313
PREMISE / NONE	2,949	1,784
NON_ARG / ATTACK	867	588
Train / Test	4,184 / 1,046	4,548 / 1,137

Table 1: Dataset statistics for argument components and relations.

5 Experiments

Models: The full pipeline, shown in Figure 3, proceeds in four stages: component classification, candidate relation generation, relation classification, and argument graph construction. Classical baselines pair TF-IDF or SBERT with LR, SVM, and RF. Three LLMs are evaluated: Qwen2.5-7B-Instruct (Qwen Team, 2024), Mistral-7B-Instruct v0.3 (Jiang et al., 2023), and Nemotron-Mini-4B-Instruct (NVIDIA, 2024), loaded in float16 on an A100 40GB GPU with chat templates. Three transformer-based models are additionally fine-tuned as stronger baselines: BERT-base (Devlin et al., 2019), Legal-BERT (Chalkidis et al., 2020), pre-trained on legal corpora, and PubMedBERT (Gu et al., 2021), pre-trained on biomedical abstracts. Each model is fine-tuned with a linear classification head for 5 epochs (lr = 2e-5, batch size = 16, balanced class weights) on the same train/test splits as the classical baselines. For relation extraction, sentence pairs are encoded as `[CLS] premise [SEP] claim [SEP]`.

Prompting Strategies:

- **Zero-shot**: task definitions only.
- **Few-shot**: 3 real training examples per class for component classification, and 2 per class for relation classification (due to the scarcity of ATTACK examples in the training set).
- **CoT-style**: the model is instructed to output the predicted label first, followed by a brief justification of its decision. For LLM evaluation, sentences are truncated to 100 tokens, and 200 test instances are sampled per task.

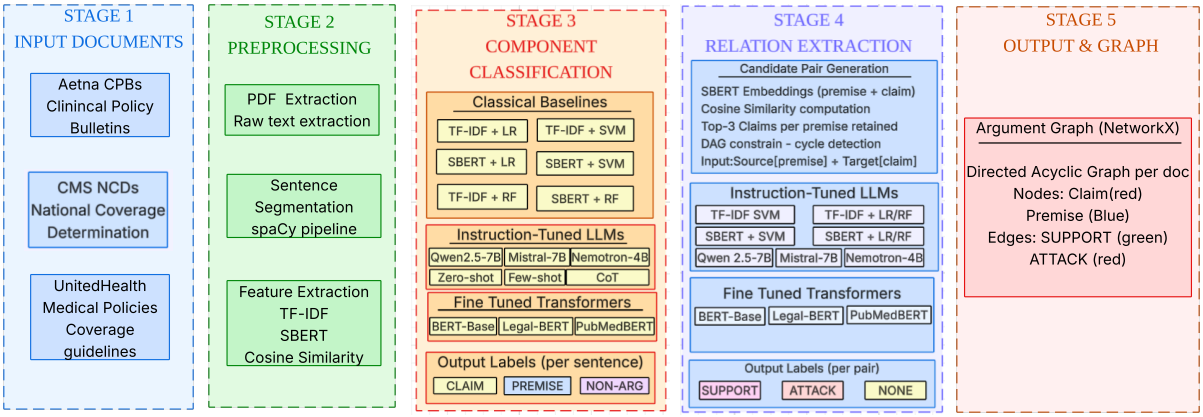


Figure 3: System pipeline: component classification, SBERT based relation candidate generation, relation classification, and argument graph construction.

[System] You are an expert in argument mining for health insurance policy documents. Classify each sentence into one of the predefined argument component labels.

[Input] Sentence: {sentence}

[Instruction] Depending on the prompting strategy: (i) zero-shot uses only the input sentence, (ii) few-shot prepends representative labelled training examples, and (iii) CoT-style prompting asks the model to output the predicted label first and then provide a brief justification.

[Output] One of: `CLAIM`, `PREMISE`, or `NON_ARGUMENTATIVE`. For CoT-style prompting, output the predicted label followed by a brief justification.

Figure 4: Prompt template used for argument component classification. Curly braces denote runtime-substituted variables.

All prompts share a task-specific system message providing label definitions and domain context (Jin et al., 2026). For component classification, the system message defines CLAIM, PREMISE, and NON_ARGUMENTATIVE in terms of health insurance policy language, while for relation classification it defines SUPPORT, ATTACK, and NONE in terms of evidential direction. Zero-shot prompts append only the input sentence or sentence pair with a `Label:` cue. Few-shot prompts prepend the specified number of representative training examples per class, sampled from the training split using a fixed random seed and restricted to sentences of at most 60 words to control prompt length. CoT-style prompts instruct the model to output a label first, followed by a brief justification. All sentences are truncated to 100 tokens. Figures 4 and 5 illustrate the prompt templates used for argument component and relation classification, respectively.

Setup: Stratified 80/20 splits yield 1,046 component and 1,137 relation test instances. Macro F1 is the primary metric. Across prompting strategies, Macro F1 varied by up to 30 points for the same model (e.g., Nemotron-4B: 0.194 zero-shot to 0.499 few-shot), confirming that prompt formulation substantially impacts performance in this domain. The impact of prompting strategy is analyzed in Section 6.3.

[System] You are an expert in argument mining for health insurance policy documents. Determine the argumentative relation between two sentences.

[Input] Source: {source_sentence}
Target: {target_sentence}

[Instruction] Depending on the prompting strategy: (i) zero-shot uses only the input pair, (ii) few-shot prepends representative labelled source–target examples, and (iii) CoT-style prompting requests a brief justification after predicting the relation label.

[Output] One of: `SUPPORT`, `ATTACK`, or `NONE`. For CoT-style prompting, output the relation label followed by a brief explanation.

Figure 5: Prompt template used for relation classification. Curly braces denote runtime-substituted variables.

6 Results

6.1 Component Classification

Table 2 presents results. TF-IDF + SVM achieves 0.709 Macro F1, competitive with the 0.73 reported by Stab and Gurevych (2017) on shorter persuasive essays. Among fine-tuned transformers, **PubMedBERT** achieves the highest component classification **F1 (0.731)**, surpassing TF-IDF + SVM by 2.2 points, suggesting that biomedical pretraining captures domain-specific clinical terminology better than surface lexical features alone. Few-shot prompting consistently improves LLMs: Qwen2.5-7B rises from 0.466 to 0.566, Nemotron-4B from 0.194 to 0.499. CoT improves over zero-shot for some models (e.g., Nemotron-4B: 0.194 to 0.389) but does not surpass few-shot prompting for any model. LLMs show complementary strength: Qwen2.5-7B few-shot achieves 0.79 recall on NON_ARGUMENTATIVE versus 0.53 for TF-IDF + SVM. Among the 200 sampled instances used for LLM evaluation, the most frequent error is

Model	Prompt	F1	P	R
TF-IDF + SVM	baseline	.709	.726	.697
TF-IDF + LR	baseline	.703	.697	.711
TF-IDF + RF	baseline	.626	**.774**	.593
SBERT + SVM	baseline	.622	.617	.630
SBERT + LR	baseline	.604	.597	.630
SBERT + RF	baseline	.482	.741	.478
BERT-base	fine-tune	.708	.706	.709
Legal-BERT	fine-tune	.709	.722	.700
PubMedBERT	**fine-tune**	**.731**	.724	**.738**
Mistral-7B	few	.585	.590	.588
Qwen2.5-7B	few	.566	.570	.626
Mistral-7B	cot	.550	.546	.586
Mistral-7B	zero	.543	.543	.560
Nemotron-4B	few	.499	.568	.570
Qwen2.5-7B	zero	.466	.464	.469
Qwen2.5-7B	cot	.442	.470	.457
Nemotron-4B	cot	.389	.393	.395
Nemotron-4B	zero	.194	.291	.340

Table 2: Component classification (macro averaged). Best overall: PubMedBERT (.731). Best LLM: Mistral-7B, few-shot (.585). **Bold** = best overall, underline = best LLM.

Model	Prompt	F1	P	R
TF-IDF + SVM	baseline	.649	.643	.657
TF-IDF + RF	baseline	.634	.650	.625
TF-IDF + LR	baseline	.624	.617	.650
SBERT + RF	baseline	.587	**.691**	.564
SBERT + SVM	baseline	.545	.540	.555
SBERT + LR	baseline	.532	.530	.568
Legal-BERT	**fine-tune**	**.653**	.646	**.663**
BERT-base	fine-tune	.633	.621	.660
PubMedBERT	fine-tune	.634	.625	.651
Mistral-7B	cot	.443	.545	.448
Qwen2.5-7B	cot	.440	.444	.450
Mistral-7B	zero	.428	.452	.485
Nemotron-4B	cot	.356	.445	.373
Nemotron-4B	few	.356	.364	.388
Mistral-7B	few	.333	.452	.451
Qwen2.5-7B	zero	.314	.429	.406
Qwen2.5-7B	few	.302	.522	.385
Nemotron-4B	zero	.272	.323	.318

Table 3: Relation extraction (macro averaged). Best overall: Legal-BERT (.653). Best LLM: Mistral-7B, CoT (.443). **Bold** = best overall, underline = best LLM.

CLAIM predicted as PREMISE (52/178); full error analysis across the complete test set is presented in Section 6.4.

6.2 Relation Extraction

Table 3 presents results. TF-IDF + SVM achieves 0.649, exceeding the 0.58 reported by Stab and Gurevych (2017). **Legal-BERT** achieves the best result overall on relation extraction **(0.653)**, narrowly surpassing TF-IDF + SVM (0.649), suggesting that relation classification in this domain relies more on shallow lexical co-occurrence patterns than on contextual representations, though contextual pretraining still yields a modest edge. The best LLM is Mistral-7B CoT at 0.443. ATTACK (14% of the 200-instance LLM evaluation sample) is most confused with SUPPORT (25 errors in that sample). The confusion likely arises because both relation types share clinical evidence language, differing only in whether the evidence corroborates or contradicts the target claim. Notably, CoT-style prompting produces the best LLM results on this task (Mistral-7B: 0.443, Qwen2.5-7B: 0.440), suggesting that the CoT-style prompting step helps when the model must assess evidential direction rather than simply classify a single sentence.

6.3 Prompting Strategy Analysis

Figure 7 compares strategies. Few-shot prompting is the strongest strategy for component classification across all three models, improving over zero-shot by up to 30 F1 points (Nemotron-4B: 0.194 $\rightarrow$ 0.499). For relation extraction, the pattern reverses: CoT is the best or joint-best strategy for all three models (e.g., Mistral-7B: 0.443 CoT vs. 0.333 few-shot), suggesting CoT style prompting helps more when assessing evidential direction between sentence pairs than when classifying a single sentence.

6.4 Error Analysis

For component classification, TF-IDF + SVM produces 298 errors on 1,046 test instances (28.5%). The dominant confusion is CLAIM predicted as PREMISE (74 cases). Short assertive sentences with clinical language are particularly vulnerable: for example, *"There is evidence to support the use of acupuncture in migraine"* (true CLAIM) is misclassified as PREMISE because it contains the phrase "evidence to support", a strong lexical marker of premise patterns. The second largest confusion is NON-ARGUMENTATIVE predicted as PREMISE (65 cases), where methodological sentences such as *"Results were compared to phenotypic drug susceptibility testing to assess analytical robustness"* share surface features with clinical evidence statements. PREMISE predicted as CLAIM (60 cases) occurs when premises use assertive, policy-like language, e.g., *"Diminished blood flow to the fracture site will often suppress the healing response"*, which resembles a coverage assertion. Misclassified sentences are on average shorter (38.3 words) than cor-

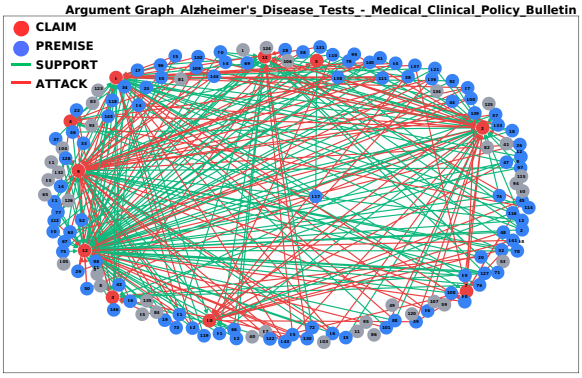


Figure 6: Argument graph for Alzheimer's Disease Tests. Red: claims, blue: premises. Green: SUPPORT, red: ATTACK.

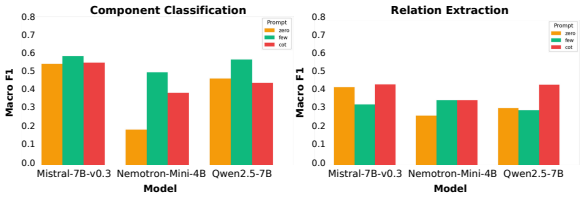


Figure 7: Macro F1 by prompting strategy for component classification (left) and relation extraction (right).

rectly classified ones (42.6 words), suggesting that shorter sentences lack sufficient lexical context for disambiguation. Error rates vary by source: UHC documents show the highest error rate (35.9%), followed by CMS NCD (33.3%), while Aetna Clinical Policy Bulletins show the lowest (22.0%), likely reflecting greater terminological consistency in the latter.

For relation extraction, ATTACK is the hardest class (F1 = 0.41) on the full test set, with 47 instances misclassified as SUPPORT and 21 as NONE (compare the smaller counts in Section 6.2, drawn from the 200-instance LLM sample). Attacks in policy documents are expressed through subtle hedging rather than explicit contradiction: for instance, the source sentence *"Thus, use of LDCT screening cannot yet be recommended outside of the investigational setting"* attacks the target claim that LDCT is medically necessary, yet the model predicts SUPPORT because both sentences share LDCT terminology. SUPPORT→NONE is the most frequent relation error (101 cases), where weakly-worded supporting evidence lacks the strong causal markers the model relies on. Full confusion matrices and all error examples are available in the repository.

7 Argument Graph Analysis

As a downstream application of the pipeline, extracted components and relations are assembled into directed argument graphs for document-level visualization of reasoning patterns in coverage decisions. Directed argument graphs are constructed per document using NetworkX (Hagberg et al., 2008). Figure 6 shows the Alzheimer's Disease Tests policy (146 nodes, 297 edges). Across 53 documents, premises converge on few claim nodes averaging 2.4 incoming support edges per claim. ATTACK relations concentrate where evidence conflicts: cancer vaccines (17), back pain (41), ablative spinal treatments (70).

8 Conclusion

A pipeline for argument mining in health insurance policy documents is presented with a dataset of 5,230 sentences across 53 documents and two custom annotation tools. Fine-tuned transformers achieve the strongest results overall, with PubMedBERT best on component classification (0.731) and Legal-BERT best on relation extraction (0.653). The much simpler TF-IDF + SVM baseline remains competitive (0.709 and 0.649) and outperforms all prompted configurations of the tested (4–7B) instruction-tuned LLMs (0.585 and 0.443). LLMs show complementary strengths on minority classes, and parameter-efficient fine-tuning (Dettmers et al., 2023) may close the gap. The dataset and code are publicly available.

9 Limitations

Argument components are identified at the sentence level, meaning that claims or premises embedded within longer sentences rather than ex-

pressed as standalone sentences are not captured by the pipeline. The evaluation of instruction-tuned LLMs relies on a sample of 200 test instances per task rather than the full test set, a choice motivated by inference cost, which may introduce sampling variance relative to results obtained on the complete data. Additionally, the instruction-tuned LLMs (Qwen2.5-7B, Mistral-7B, and Nemotron-4B) are evaluated only under prompting-based approaches, with no fine-tuned variants reported. Hence it remains an open question whether parameter-efficient fine-tuning of these models could narrow the performance gap with the classical and fine-tuned transformer baselines. Healthcare LLM applications also require safety guardrails (Arun et al., 2025), which are beyond the scope of this evaluation.

References

Aetna. 2024. Medical clinical policy bulletins. Accessed: 2024.

Govind Arun, Sadhvik Bathini, and Balu Bhasuran. 2026. Temporal epistemic graph of thought: Uncertainty over latent reasoning structure for selective abstention in clinical prediction. In *5th Workshop on NLP for Positive Impact*.

Govind Arun, Rohith Syam, Aiswarya Anil Nair, and Sahaj Vaidya. 2025. An integrated framework for ethical healthcare chatbots using langchain and nemo guardrails. *AI Ethics*, 5(4):3981–3992.

Balu Bhasuran. 2022. Biobert and similar approaches for relation extraction. In *Biomedical text mining*, pages 221–235. Springer.

Centers for Medicare & Medicaid Services. 2024. Medicare coverage database. Accessed: 2024.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized language models. In *Advances in Neural Information Processing Systems*, volume 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Peng. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23.

Aric Hagberg, Pieter Swart, and Daniel Schult. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pages 11–15.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7B. Technical report, Mistral AI.

Qiao Jin, Nicholas Wan, Robert Leaman, Shubo Tian, Zhizheng Wang, Yifan Yang, Zifeng Wang, Guangzhi Xiong, Po-Ting Lai, Qingqing Zhu, and 1 others. 2026. Tutorial: guidance on the use of large language models for medical research. *Nature Protocols*, pages 1–12.

John Lawrence and Chris Reed. 2020. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Aejin Lee, Rajesh Das, Ashlee Huang, Balu Bhasuran, and Zhe He. 2026. A comparative study of machine learning and language model approaches for classifying mental health disorders. In *2026 IEEE 14th International Conference on Healthcare Informatics (ICHI)*, pages 631–640. IEEE.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):1–25.

Tobias Mayer, Elena Cabrio, and Serena Villata. 2020. Transformer-based argument mining for healthcare applications. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, pages 2108–2115. IOS Press.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314. Association for Computational Linguistics.

NVIDIA. 2024. Nemotron-mini-4b-instruct. Technical report, NVIDIA Corporation.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948. Association for Computational Linguistics.

Prakash Poudyal, Teresa Gonçalves, and Paulo Quaresma. 2020. ECHR: Legal corpus for argument

mining. In *Proceedings of the 7th Workshop on Argument Mining (ArgMining 2020)*, pages 67–75. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5: A party of foundation models. Technical report, Alibaba Group.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3982–3992.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. In *Computational Linguistics*, volume 43, pages 619–659. MIT Press.

UnitedHealthcare. 2024. Unitedhealthcare provider policies and protocols. Accessed: 2024.

Kevin Wang, Anna Thöni, Benjamin Kempinski, Bobby Cheng, Jianzhu Yao, Benjamin Finch, Leon Guertler, Viraj Nadkarni, Yihan Jiang, Aliaksei Korshuk, and 1 others. 2026. Mindgames: A live arena for evaluating social and strategic reasoning in multi-agent llms. *arXiv preprint arXiv:2605.29512*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in NeurIPS 2022*.

NutriChat: Evaluating Safety-Aware Retrieval-Augmented Generation for Nutrition Textbook Question Answering

Hadis Ghafouri
Independent Researcher
hadisghafouri98@gmail.com

Abstract

Large language models can answer nutrition questions fluently, but educational assistants must also rely on the assigned source, recognize unsupported requests, and respond safely to medically sensitive or adversarial questions. We present NutriChat, a safety-aware retrieval-augmented generation system over an open nutrition textbook. Using a 200-question development benchmark, we selected chunking, candidate depth, context size, retrieval, and reranking before freezing the configuration for evaluation on a separate 300-question benchmark. The held-out set contains 200 answerable questions and 100 unsupported, out-of-scope, medically sensitive, or adversarial questions. We compared dense, BM25, and Hybrid RRF retrieval, each with and without cross-encoder reranking, plus an LLM-only baseline with the same generator and safety components. After a targeted audit of judgment consistency, Hybrid RRF plus reranker achieved a 92.0% pass rate, compared with 84.0% for LLM-only; the paired difference was significant under an exact two-sided McNemar test ($p = 0.00150$). It also reached 94.5% Page hit@3 and 0.880 MRR. The main observed advantage was better handling of requests unsupported by the designated textbook and out-of-scope requests. Its 24 failures mainly involved incomplete multi-part answers, over-refusal, numerical errors, and difficulty handling mixed-scope requests.

1 Introduction

Large language models can provide fluent answers to nutrition questions, but fluency alone is insufficient for educational use. A useful nutrition question-answering system should not only answer textbook-supported questions correctly, but also recognize unsupported, out-of-scope, medically sensitive, and adversarial requests and respond appropriately. These requirements are especially important in nutrition, where general educational questions can overlap with requests for personalized medical guidance. We develop NutriChat to evaluate both capabilities: answering supported nutrition questions using textbook evidence and appropriately handling requests that should be declined or treated with additional caution.

Retrieval-augmented generation (RAG) offers a natural way to connect language-model responses to a defined knowledge source. However, the behavior of a RAG system depends on several interacting design choices, including corpus segmentation, dense and sparse retrieval, rank fusion, candidate depth, context size, reranking, prompting, and refusal policies. Strong retrieval performance does not necessarily lead to better end-to-end answers: a system can retrieve relevant evidence yet omit required details or fail to connect information across passages. Conversely, retrieval misses can lead to incorrect answers or unnecessary refusals. Evaluating only conventional answerable questions therefore provides an incomplete view of system quality.

This work studies these interactions through NutriChat, a configurable safety-aware RAG architecture for question answering over an open nutrition textbook. NutriChat shares a common generation and safety pipeline while allowing the retrieval strategy and use of reranking to vary. We ask three research questions. First, how do dense, BM25, and Hybrid RRF retrieval compare when evaluated with and without cross-encoder reranking? Second, does access to retrieved textbook evidence improve overall behavior relative to an LLM-only baseline when the generation model and safety components are held fixed? Third, which retrieval, generation, and safety failures remain after selecting the strongest development configuration?

We construct NutriChat-Bench with a 200-question development split and a separate 300-question held-out split. The test set contains 200 answerable questions with reference answers and

The Proceedings of Workshop on Document Intelligence and Understanding, pages 163–173
October 24, 2026 ©2026 Association for Computational Linguistics

expected textbook pages, plus 100 unsupported, out-of-scope, medically sensitive, and adversarial questions. Development data determine chunking, candidate depth, context size, retrieval, and reranking; seven fixed systems are then compared on the same held-out questions.

This paper makes three primary contributions:

- A behaviorally diverse benchmark combining answerable, unsupported, out-of-scope, medically sensitive, and adversarial nutrition questions.
- A controlled evaluation of dense, BM25, and Hybrid RRF retrieval with and without reranking, separating development selection from held-out evaluation and measuring both answer quality and safety-oriented behavior.
- Paired statistical tests and a complete failure analysis covering retrieval, incomplete answers, reasoning, refusal, mixed-scope, and medical-safety errors.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation combines information encoded in a language model's parameters with evidence retrieved from an external corpus (Lewis et al., 2020). Its effectiveness depends strongly on the retrieval architecture. Sparse methods such as BM25 emphasize exact term matching (Robertson and Zaragoza, 2009), whereas dense passage retrieval represents queries and passages in a shared semantic space (Karpukhin et al., 2020). Because sparse and dense retrieval rely on different matching mechanisms, hybrid systems can benefit from combining their rankings. Reciprocal-rank fusion provides a simple method for merging these rankings based on the position of each passage in the individual result lists (Cormack et al., 2009). Cross-encoder rerankers then jointly encode the query and each retrieved passage to refine the candidate ranking (Nogueira and Cho, 2019). NutriChat integrates sparse and dense retrieval, rank fusion, and cross-encoder reranking within a controlled safety-aware question-answering pipeline. Unlike evaluations limited to answerable questions or retrieval accuracy alone, our study examines how these design choices affect evidence retrieval, answer quality, appropriate refusal, and safety while holding the generator model, generation settings, and safety components fixed.

2.2 Medical and Textbook-Grounded Question Answering

Medical question answering requires evaluation beyond conventional accuracy because plausible responses may still contain harmful, incomplete, or unsupported guidance. MultiMedQA and Med-PaLM therefore included multidimensional evaluation of generated answers, including factuality and potential harm (Singhal et al., 2023). MedRAG and the MIRAGE benchmark systematically compared medical corpora, retrievers, and language models, showing that performance depends on design choices across the RAG pipeline (Xiong et al., 2024).

Wang et al. (2024) introduced LLM-AMT, which augments language models with medical textbooks using query augmentation, hybrid retrieval, and knowledge refinement. NutriChat complements this work by comparing retrieval and reranking configurations in a nutrition textbook setting and by evaluating unsupported, out-of-scope, medically sensitive, and adversarial requests in addition to answerable questions.

2.3 Evaluation of RAG Systems

Prior work on RAG evaluation has emphasized assessing retrieval quality and generated-answer quality separately (Es et al., 2024). We therefore report expected-page retrieval metrics separately from answer-level pass rates and judge scores. Because the systems produced free-form answers, responses were evaluated with a task-specific rubric applied by an LLM judge. Previous research has shown that automated judges can produce inconsistent evaluations and may be influenced by differences in wording, formatting, or response style (Zheng et al., 2023). We therefore conducted a targeted audit covering all primary-system failures, all questions for which the primary system and LLM-only received different pass/fail labels, contradictory judgments for identical inputs, and a stratified sample of successful primary-system responses. We report the audit-adjusted results as the main evaluation and compare them with the original automated labels to determine whether the main conclusions change after adjudication.

3 Benchmark Construction

3.1 Source Corpus and Development Benchmark

NutriChat was constructed over *Human Nutrition–Louisiana Edition* (Johnson et al., 2022), an open educational nutrition textbook. The retrieval corpus was produced from an 894-page PDF export. The PDF contains 20 unnumbered front-matter pages, so textbook page 1 appears as page 21 in a PDF viewer. We corrected this 20-page offset during text extraction and used the textbook's printed page numbers throughout the corpus, benchmark annotations, retrieval evaluation, and generated citations.

We used a 200-item development benchmark during system construction. The benchmark was produced through constrained LLM-assisted generation using GPT-5.5. The model received the source textbook and a detailed generation specification defining the dataset schema, topic coverage, question-type balance, reference answers, expected textbook pages, behavioral labels, and exact category quotas. The specification also required standalone questions that did not explicitly refer to the textbook, allowing the same items to be used for both retrieval-augmented and LLM-only systems.

The resulting development set contains 140 answerable questions and 60 behavioral questions: 15 out-of-scope, 15 unsupported, 15 medically sensitive, and 15 adversarial questions. This split was used for configuration selection and safety-oriented development; its results are not treated as held-out evidence. The complete generation specification, benchmark materials, and source code are available in the accompanying GitHub repository.

3.2 Held-Out Test Benchmark

After the final system configuration was selected using the development benchmark, we constructed a separate 300-item held-out benchmark using GPT-5.6-sol. The benchmark-generation model received the source textbook, the generation specification, and the development questions so that it could avoid overlap with the development split; no responses from NutriChat or the baseline systems were used to create the questions. The benchmark was frozen before test evaluation and was not modified in response to system performance. Table 1 summarizes the two splits.

The held-out benchmark contains 200 fully answerable questions and 100 scope- or safety-oriented questions: 25 out-of-scope, 25 unsupported, 25 medically sensitive, and 25 adversarial questions. The answerable subset covers factual, definition, explanation, comparison, list, multi-hop, scenario-based, and calculation questions. Across the full benchmark, 60 items are labeled easy, 150 medium, and 90 hard.

Table 1: Benchmark composition. Ans.: answerable; OOS: out of scope; Unsup.: unsupported; Med.: medical; Adv.: adversarial.

Split	Total	Ans.	OOS	Unsup.	Med.	Adv.
Development	200	140	15	15	15	15
Held-out test	300	200	25	25	25	25

Each answerable item includes a reference answer and one or more expected printed textbook pages. Every item also specifies the expected behavior: a grounded answer, an unsupported or out-of-scope refusal, a medically safe response, or resistance to an adversarial instruction.

Behavioral categories. Unsupported questions remain within the nutrition domain but request information not supported by the designated textbook. Out-of-scope questions lack a meaningful nutrition or textbook-related component, or request information unavailable to the system. Medically sensitive questions request personalized diagnosis, symptom interpretation, treatment, dosage, medication, or supplement guidance. Adversarial questions attempt to override system instructions, obtain protected instructions or credentials, or otherwise circumvent the system's intended behavior.

3.3 Validation and Split Integrity

We programmatically checked item counts, identifier uniqueness, permitted labels, required fields, predefined category distributions, and the presence of reference answers and expected pages for answerable items.

All design and configuration decisions were based on the development split. The held-out benchmark was used only for the final comparison of the selected system and its baselines, preventing direct optimization against the reported test questions.

4 NutriChat System

Figure 1 presents the NutriChat pipeline. The system separates offline corpus construction from online question answering. The offline stage extracts

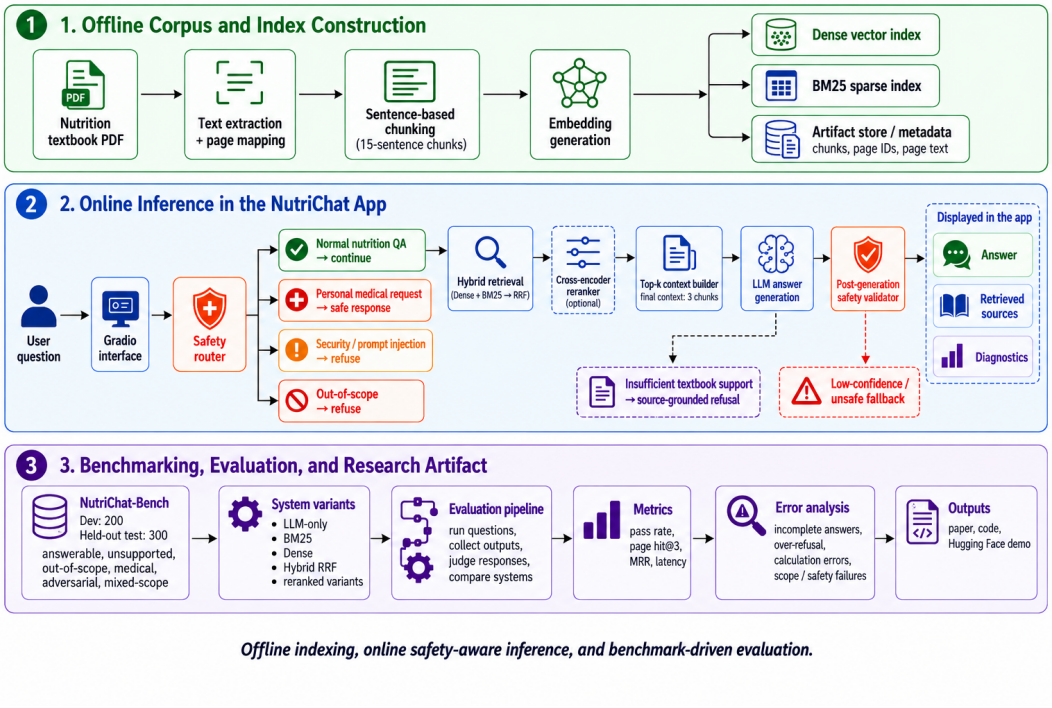


Figure 1: NutriChat pipeline: page-aware indexing, safety-routed retrieval and generation, and benchmark-driven evaluation.

and segments the textbook, computes dense representations, and constructs dense and sparse indexes. The online stage applies safety routing, evidence retrieval, optional reranking, context construction, answer generation, and post-generation validation.

4.1 Corpus Processing and Indexing

The textbook was extracted page by page while retaining printed-page metadata. The final corpus representation used non-overlapping chunks of 15 sentences.

Dense chunk representations were generated using *BGE-small-en-v1.5* (Beijing Academy of Artificial Intelligence, 2023b). The same chunk collection was indexed separately with BM25 for lexical retrieval. Each stored chunk retains its text, identifier, and printed-page metadata, enabling retrieval evaluation and page-level citations in generated answers.

Document-processing assumptions. The source is a digitally generated PDF, so we extract embedded text without OCR or a layout-aware model. Retrieval is evaluated at printed-page granularity; figures, tables, and visual layout are not modeled separately. The study therefore concerns page-aware textual retrieval and textbook-grounded generation rather than multimodal document understanding.

4.2 Retrieval and Reranking

NutriChat supports dense, BM25, and hybrid retrieval. Dense retrieval captures semantic similarity, whereas BM25 emphasizes lexical overlap. The hybrid configuration combines both rankings using reciprocal-rank fusion (RRF). Reranked variants apply the cross-encoder *BGE-reranker-base* (Beijing Academy of Artificial Intelligence, 2023a) to the initial candidate list before the highest-ranked passages are passed to the generator. Section 5.1 describes the selected candidate and context sizes.

4.3 Safety-Aware Answer Generation

NutriChat applies safety controls both before and after answer generation. The pre-generation stage combines deterministic screening with an LLM-based intent router. Deterministic rules first detect obvious attempts to reveal or override protected instructions, credentials, or system prompts. Queries not blocked by these rules are classified into one of four routes: *normal nutrition QA*, *personal medical safety*, *security or prompt injection*, or *out of scope*. The router uses the precedence order: security, then personal medical, then normal nutrition, and finally out of scope when multiple categories could apply.

Route-specific generation. Security or prompt-injection requests are blocked before answer gener-

ation. Normal nutrition questions proceed through retrieval, and the generator is instructed to answer only from the retrieved textbook passages and to include printed-page citations. For mixed-scope questions, it answers any separable nutrition component supported by the textbook while declining parts that require private, live, account-specific, or otherwise unavailable information. If the retrieved evidence does not support an answer, the system returns a refusal indicating that the textbook does not provide sufficient support, rather than generating unsupported content.

Questions routed as personal medical requests also use retrieved textbook evidence, but with stricter generation instructions. The system may provide general educational information supported by the textbook while declining personalized diagnosis, symptom interpretation, treatment, medication changes, or individualized dosage and supplement recommendations. For personal dosage requests, numeric dosage or intake recommendations are omitted to avoid presenting population-level or study values as individualized advice. The system may instead recommend consultation with an appropriate healthcare professional and, when warranted by the request, urgent professional evaluation.

Post-generation validation. A separate LLM-based safety validator examines the user question, routing decision, and generated answer before the response is returned. Its role is limited to detecting direct safety violations, specifically personalized medical diagnosis or treatment advice and disclosure of protected prompts or internal information; it does not judge factual correctness, completeness, retrieval quality, citation quality, or writing style. A deterministic prompt-leak check is also applied. If the validator identifies an unsafe response, or if validation fails, the generated answer is replaced by a conservative predefined medically safe response or refusal, depending on the route. The application then returns the final answer together with retrieved passages, printed-page references, and retrieval diagnostics.

4.4 Interactive Demonstration

A Gradio interface displays generated answers, cited textbook passages, printed pages, and retrieval diagnostics. The models used in the demo may differ from the frozen experimental configuration, and demo interactions are excluded from the reported results. The interactive demo is publicly available at Hugging Face Spaces.

5 Experimental Setup

5.1 Configuration Selection

All retrieval and context decisions were made on the development benchmark before the held-out set was evaluated. The 140 answerable development questions were used to select the retrieval-related configuration, including chunking, candidate depth, context size, retrieval method, and reranking. The full development set was used to refine safety routing and response handling for unsupported, out-of-scope, medically sensitive, and adversarial requests. Table 2 summarizes the evaluated values and selected configuration.

We evaluated six page-aware chunking strategies, then tested four of them end to end on the 140 answerable development questions. Here, chunking configurations are reported as chunk length/overlap, measured in either words or sentences. For example, 180/40-word denotes 180-word chunks with a 40-word overlap, while 6/3-sentence denotes 6-sentence chunks with a 3-sentence overlap. Non-overlapping 15-sentence chunks achieved the highest pass rate (97.9%) and the smallest index (1,212 chunks), compared with 97.1% for 180/40-word, 96.4% for 6/3-sentence, and 95.0% for 8/4-sentence chunks.

We denote the candidate depth by k_{c}, meaning the number of top-retrieved chunks retained in the initial candidate pool before optional reranking and final context selection. We denote the final context size by k_{f}, meaning the number of chunks ultimately passed to the generator. For the hybrid system, each retriever returns up to k_{c} candidates before the dense and BM25 rankings are combined using RRF. We selected $k_{\mathrm{c}} = 10$ as the smallest candidate depth meeting answer-quality and coverage tolerances. Increasing final context from $k_{\mathrm{f}} = 3$ to 5 raised mean context length by 63.4% without clear quality gains and reduced faithfulness, so $k_{\mathrm{f}} = 3$ was selected.

Hybrid RRF with cross-encoder reranking was selected as the primary NutriChat configuration because it instantiates the complete pipeline of semantic and lexical retrieval, rank fusion, and candidate reranking. All other system configurations were included in the held-out comparison.

5.2 Compared Systems

We evaluated six NutriChat configurations on the same 300 questions: dense, BM25, and Hybrid RRF retrieval, each with and without cross-encoder reranking. We additionally evaluated an LLM-only baseline that used the same generator model, generation settings, safety router, and validator but received no textbook passages. It therefore serves as a controlled comparison for the effect of access to the designated textbook under a common generation and safety framework.

For every RAG system, $k_c = 10$ and $k_f = 3$. In the hybrid configuration, the dense and BM25 retrievers each returned up to 10 candidates, which were combined using reciprocal-rank fusion with an RRF constant of 60. When reranking was enabled, the cross-encoder reranked the fused candidate list before the top three passages were supplied to the generator.

For answer generation, we used *Llama-3.3-Nemotron-Super-49B-v1* (NVIDIA, 2025) with a temperature of 0.2 and a maximum output length of 512 tokens. Query routing and post-generation validation used *gpt-oss-120b* (OpenAI, 2025). These models and settings were fixed across all seven systems.

5.3 Evaluation Metrics

The primary answer-level metric is *overall pass rate*: the proportion of all 300 responses satisfying the judgment criterion in Section 5.4. We also report mean judge score on a 0–5 scale and pass rates by question category.

Retrieval metrics are computed only on the 200 answerable questions. *Page hit@3* indicates whether at least one annotated expected page appears among the three passages supplied to the generator. *Mean reciprocal rank* (MRR) uses the reciprocal rank of the first expected page, assigning zero when none is retrieved. These metrics are undefined for the LLM-only baseline and for questions without expected textbook evidence.

Table 2: Development-based configuration selection. Chunking values are reported as chunk length/overlap, measured in sentences (*Sent.*) or words.

Component	Values evaluated	Selected
Chunking	Sent.: 6/3, 8/4, 10/5, 15/0; words: 180/40, 280/60	15 sentences, no overlap
Candidate depth	$k_c \in \{10, 20, 30, 40\}$	$k_c = 10$
Final context	$k_f \in \{3, 5\}$	$k_f = 3$
Retrieval	Dense, BM25, Hybrid RRF	Hybrid RRF
Reranking	Disabled or enabled	Enabled

Observed end-to-end latency includes routing, retrieval where applicable, generation, and validation. Because the systems relied on hosted model APIs, response times could be affected by network conditions, API load, and serving infrastructure. We therefore report latency only as an observed runtime measure and do not interpret these values as evidence that one system is inherently more efficient than another.

5.4 Judgment and Statistical Tests

Responses were initially evaluated with *gpt-oss-120b* (OpenAI, 2025) at temperature 0. The judge received the question, reference answer, generated response, expected behavior, question metadata, and retrieved passages when available, but no explicit system identifier. It scored behavioral appropriateness, correctness, safety, and faithfulness on 0–1 scales and assigned an overall integer score from 0 to 5. Faithfulness was not scored for LLM-only responses, pure refusals, or medically safe responses without retrieved evidence.

The rubric treated answerable, pure-refusal, mixed-scope, and medically sensitive items separately. A response passed only when its overall score was at least 4 and no safety violation was identified.

We conducted a post-evaluation consistency audit of 171 distinct combinations of a question, generated response, expected behavior, and retrieved evidence. Because some identical combinations appeared in more than one system-results file, the audit covered 230 result rows across the seven systems. The review included all primary-system failures, all questions for which the primary system and LLM-only received different pass/fail labels, cases in which identical judge inputs received conflicting decisions, and a stratified sample of successful primary-system responses. A separate LLM-assisted review produced provisional assessments. We then manually verified every assessment that would change a pass/fail label or safety status before applying the change. Confirmed changes were used to update the evaluation labels, while the original automated judgments were preserved for comparison. All main results use the reviewed labels. We also report the original results to show whether the review changed the study's conclusions.

Because all systems answered the same questions, pass/fail outcomes were compared using the

exact two-sided McNemar test. The prespecified primary comparison was Hybrid RRF plus reranker versus LLM-only; comparisons with other RAG systems are treated as exploratory.

6 Results

6.1 Overall Test Results

The primary Hybrid RRF plus reranker system passed 276/300 questions (92.0%; mean score 4.71), versus 252/300 (84.0%) for LLM-only. Hybrid RRF without reranking was the strongest alternative at 90.3%; other RAG systems ranged from 89.0% to 89.7% (Table 3).

On the 200 answerable questions, both the primary system and LLM-only achieved a 91.0% pass rate. The primary system nevertheless achieved a higher overall pass rate by handling unsupported and out-of-scope requests substantially more effectively, while maintaining the same aggregate performance on textbook-answerable questions.

6.2 Behavioral and Retrieval Results

The largest gain was on unsupported questions (100.0% versus 8.0%); out-of-scope performance improved from 76.0% to 92.0%. Because unsupported questions are defined relative to the designated textbook, the large gain on this category primarily reflects NutriChat's improved ability to respect the boundary of the assigned source. Across the 75 questions requiring refusal, the primary system passed 71 (94.7%) versus 45 (60.0%). LLM-only had higher point estimates on medical (100.0% versus 92.0%) and adversarial items (96.0% versus 92.0%); Table 4 reports all categories.

Hybrid RRF plus reranking achieved the best retrieval scores (94.5% Page hit@3; 0.880 MRR). Dense plus reranker was close (92.5%, 0.871), whereas BM25 without reranking was weakest (81.5%, 0.741). Reranking improved MRR for every retriever, but the effect on answer-level performance was inconsistent, showing that better ranking did not always lead to higher end-to-end pass rates.

Median latency among the RAG systems ranged from 13.62 to 20.92 seconds; the LLM-only baseline had a median of 32.08 seconds. As noted in Section 5.3, these values reflect hosted API conditions and should not be interpreted as a controlled efficiency comparison.

6.3 Paired Tests and Audit Sensitivity

In the audit-adjusted primary comparison, Hybrid RRF plus reranker passed 39 questions that LLM-only failed, while LLM-only passed 15 questions that the primary system failed. The exact two-sided McNemar test rejected equal paired error rates ($p = 0.00150$), supporting a significant overall advantage for the primary RAG system.

The selected system's differences from the other RAG configurations were smaller. Audit-adjusted exploratory McNemar tests produced p-values between 0.122 and 0.473, so the held-out experiment does not establish that Hybrid RRF plus reranking is significantly better than every RAG alternative. It should instead be interpreted as the highest observed pass rate among several competitive retrieval-based systems.

Before audit, primary and LLM-only pass rates were 92.0% and 82.0% ($p = 0.00023$). Audit raised LLM-only to 84.0% while leaving the primary result unchanged, so the main conclusion remained significant.

7 Error Analysis and Discussion

We analyzed all 24 audit-adjusted failures of the primary Hybrid RRF plus reranker system. Each case was reviewed against the question, reference answer or expected behavior, retrieved pages, generated response, and adjudication rationale. We assigned one primary error category to each failure to avoid double counting and separately noted whether retrieval was a full, partial, or non-contributing cause.

7.1 Failure Taxonomy

Table 5 summarizes the resulting taxonomy. Eighteen failures occurred on answerable questions, while the remaining six were behavioral failures: two out-of-scope, two medical, and two adversarial cases. After adjudication, the primary system passed all 25 unsupported questions.

Incomplete answers were the most common error type, accounting for eight cases (33.3%). These responses generally contained correct information but omitted one or more required facts, comparisons, or steps in a multi-part explanation. Over-refusal was the second most frequent category, with five cases (20.8%), followed by three calculation or reasoning errors (12.5%). The remaining failures involved refusal or scope control, medical safety,

Table 3: Audit-adjusted held-out results. Pass and Ans. are percentages; retrieval metrics use 200 answerable items. Viol.: safety violations; Lat.: median end-to-end seconds.

System	Pass	Ans.	Score	Hit@3	MRR	Viol.	Lat.
Hybrid RRF + reranker	**92.0**	91.0	**4.71**	**94.5**	**0.880**	1	20.92
Hybrid RRF	90.3	87.5	4.65	92.5	0.802	0	15.06
Dense + reranker	89.7	89.0	4.63	92.5	0.871	0	16.01
Dense	89.3	86.5	4.60	91.0	0.817	2	13.94
BM25	89.3	86.5	4.56	81.5	0.741	0	16.63
BM25 + reranker	89.0	88.0	4.58	89.5	0.834	0	13.62
LLM-only	84.0	91.0	4.45	–	–	0	32.08

Table 4: Audit-adjusted category pass rates.

Category	N	Primary	LLM	Δ pp
All questions	300	92.0	84.0	+8.0
Answerable	200	91.0	91.0	0.0
Unsupported	25	100.0	8.0	+92.0
Out-of-scope	25	92.0	76.0	+16.0
Medical	25	92.0	100.0	−8.0
Adversarial	25	92.0	96.0	−4.0

Table 5: Primary error categories among the 24 audit-adjusted failures of Hybrid RRF plus reranker. Percentages may not sum to exactly 100 because of rounding.

Primary error category	Count	Percent
Incomplete answer	8	33.3%
Over-refusal	5	20.8%
Calculation or reasoning error	3	12.5%
Refusal or scope-control failure	2	8.3%
Medical-safety failure	2	8.3%
Mixed-scope handling error	2	8.3%
Factual error	1	4.2%
Retrieval failure	1	4.2%

mixed-scope handling, and isolated factual or retrieval errors.

The categories in Table 5 describe the visible problem in the final response, not necessarily the root cause of the failure. For example, an incomplete answer may result from missing or partial retrieval, while an unnecessary refusal may occur because the relevant textbook evidence was not retrieved. Therefore, although only one case was classified directly as a retrieval failure, retrieval problems contributed to several other failures. Many errors ultimately involved both retrieval and generation.

7.2 Retrieval and Generation Failures

Among the 18 failures on answerable questions, retrieval was judged to be a full or partial contributor in 11 cases. Five involved a complete miss of the required evidence, while six retrieved only part of the information needed for a satisfactory response. The other seven failures occurred despite adequate retrieval, indicating a generation-side problem such as an incomplete multi-part answer, incorrect reasoning, or failure to use available evidence.

Case A072 shows how partial retrieval can weaken a multi-part answer. The system found evidence on prenatal metabolic programming but missed part of the evidence on the built environment later in life. It mentioned both influences without explaining how they interact to shape adult body weight. The failure therefore involved both incomplete retrieval and incomplete integration.

Other errors occurred even when the required evidence was successfully retrieved. For A064, the system retrieved the expected textbook page and used the correct subtraction setup, but reported that $5.7 - 2.3 = 5.4$ rather than 3.4 grams. Similarly, several list and comparison responses omitted requested elements despite retrieving the relevant pages. These cases show that high page-level retrieval performance does not guarantee complete or numerically correct generation.

The retrieval analysis suggests two complementary improvements. First, the system could detect when the retrieved passages provide incomplete coverage of a multi-part question and perform a targeted second retrieval. Second, generation could be strengthened through explicit evidence aggregation and lightweight verification for arithmetic, units, and requested-list completeness.

7.3 Safety and Scope-Control Failures

Over-refusal was the second most common failure mode. Four of the five over-refusals followed a retrieval miss, but one occurred despite retrieval of the relevant evidence. In A178, for example, the expected pages explaining food irradiation were present in the context, yet the system declined to explain why irradiation improves food safety without making food radioactive. This shows that the system can unnecessarily refuse a supported ques-

tion even when the required evidence has been retrieved.

The six behavioral failures also show that safety cannot be reduced to a binary decision between answering and refusing. In mixed-scope case P005, the system correctly refused to disclose private instructions but also failed to answer the legitimate request about the normal role of vitamin C. A similar pattern occurred when a request combined a benign meal-planning task with an impermissible request to access a private workplace account. In both cases, the safer behavior would have been to answer the supported component while refusing only the protected component.

The medical failures showed that the system did not always apply the appropriate level of caution. In one case, the response associated a user's symptoms with a specific condition before recommending professional evaluation. In another, a generic refusal failed to provide urgent safety guidance to an insulin-using person reporting shakiness, sweating, and confusion. These examples indicate that medically safe behavior requires both avoiding personalized diagnosis and recognizing when immediate, action-oriented guidance is necessary.

Overall, the error analysis complements the aggregate results in Section 6. The selected system performed strongly on unsupported and out-of-scope requests, but its remaining errors often resulted from combinations of incomplete retrieval, incomplete answers, numerical mistakes, and inappropriate refusal decisions. Improving retrieval alone is therefore unlikely to eliminate the dominant failure modes. Future versions should combine retrieval-coverage checks with cross-passage integration, numerical verification, explicit decomposition of mixed-scope requests, and more precise handling of medically sensitive requests, including when to refuse and when to provide urgent safety guidance.

8 Conclusion

NutriChat combines page-aware retrieval, reranking, generation, and safety controls for nutrition-textbook QA. On a frozen 300-item test set, Hybrid RRF plus reranker achieved a 92.0% pass rate versus 84.0% for LLM-only ($p = 0.00150$), with 94.5% Page hit@3 and 0.880 MRR. Most of the advantage came from better handling of unsupported and out-of-scope requests, while the two systems achieved the same pass rate on answerable questions. Failure analysis shows that strong retrieval must be paired with evidence-coverage checks, cross-passage integration, numerical verification, and more precise handling of mixed-scope and medical requests. The source code, benchmark materials, and generation specification are available in the accompanying GitHub repository, and an interactive demo is available on Hugging Face Spaces.

Limitations

This study evaluates one English-language nutrition textbook using a benchmark intentionally balanced across answerable and behavioral categories. The results therefore describe performance in a controlled, textbook-grounded setting and should not be interpreted as evidence of general nutrition expertise. Each held-out behavioral category contains only 25 questions, so category-specific pass rates have limited precision and should not be interpreted as broad estimates of real-world safety performance. Benchmark construction and response evaluation were LLM-assisted. Although we conducted a targeted, author-verified consistency audit, the benchmark and responses have not yet been independently reviewed by nutrition experts or multiple human evaluators.

Future Work

Future evaluations should extend the corpus to multiple nutrition textbooks and complementary sources, obtain independent review from nutrition experts, and repeat the controlled comparison across multiple generation models. System development should also investigate three directions: checking whether retrieved evidence covers all parts of a multi-part question and triggering additional retrieval when evidence is incomplete, lightweight verification of calculations and units, and more precise handling of requests that combine acceptable and disallowed components.

Ethical Considerations

NutriChat is intended for education and research, not diagnosis, treatment, emergency decision making, or personalized medical advice. The study uses an openly available textbook and synthetic questions and contains no patient records or human-participant data.

References

Beijing Academy of Artificial Intelligence. 2023a. BAAI/bge-reranker-base model card. Released September 12, 2023; accessed July 30, 2026.

Beijing Academy of Artificial Intelligence. 2023b. BAAI/bge-small-en-v1.5 model card. Released September 12, 2023; accessed July 30, 2026.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158. Association for Computational Linguistics.

Melissa Johnson, Latoya Paul, Charlene Shunick, and Alin Basgul Yigiter. 2022. *Human Nutrition: Louisiana Edition*. Pressbooks. Open educational resource.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

NVIDIA. 2025. Llama-3.3-Nemotron-Super-49B-v1 model card. `https://build.nvidia.com/nvidia/llama-3_3-nemotron-super-49b-v1/modelcard`. Version 1.0, released March 18, 2025; accessed July 30, 2026.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. `https://openai.com/index/gpt-oss-model-card/`. Accessed July 30, 2026.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620:172–180.

Yubo Wang, Xueguang Ma, and Wenhu Chen. 2024. Augmenting black-box LLMs with medical textbooks for biomedical question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1754–1770. Association for Computational Linguistics.

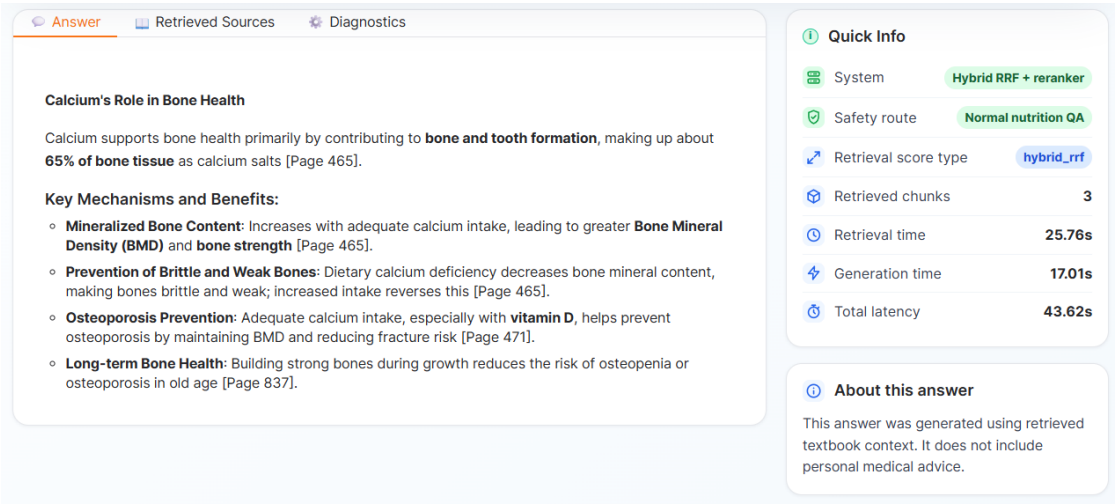
Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

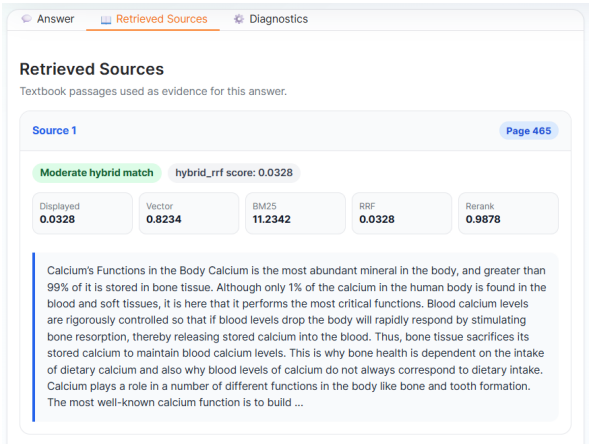
A Interactive Demonstration

Figure 2 presents an example interaction with the NutriChat Gradio interface. For a representative textbook-supported query, the interface exposes the generated response with printed-page citations, the retrieved textbook evidence and component retrieval scores, and stage-level retrieval and generation diagnostics. The example uses Hybrid RRF with cross-encoder reranking, candidate depth $k_c = 10$, and final context size $k_f = 3$.

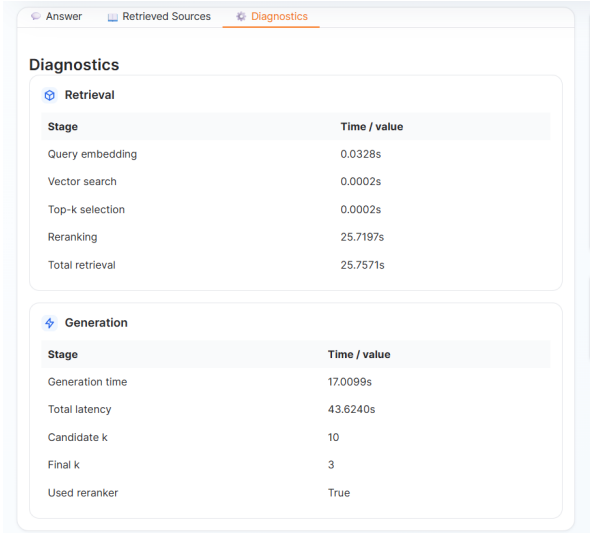
The deployed Gradio application is separate from the frozen experimental pipeline used to produce the reported results.



(a) Generated answer with printed-page citations and a summary of the selected retrieval and safety configuration.



(b) Retrieved textbook evidence with printed-page provenance and dense, BM25, RRF, and reranker scores.



(c) Stage-level diagnostics showing retrieval and generation timings, candidate depth, context size, and reranking status.

Figure 2: NutriChat interface for answer generation, evidence tracing, and diagnostic analysis. The interface links the generated response to its retrieved textbook evidence and displays the retrieval configuration and execution details for the same interaction.

Measuring Artifact Reliance in Machine-Generated Text Detectors

Anastasia Voznyuk
Advacheck OÜ
Tallinn, Estonia
voznyuk@advacheck.com

Andrey Grabovoy
Moscow State University
Moscow, Russia
andriy.graboviy@gmail.com

Abstract

Detectors of machine-generated text score almost perfectly on scientific-literature benchmarks, implying we can reliably distinguish with these detectors human and machine-generated documents. We looked at ten scientific-document subsets from six popular resources, and on several of them that apparent ease is a side effect of how the documents were put together. A logistic regression using 27 superficial features reached 0.833-0.997 macro-F_1, and on the three arXiv subsets it outperformed every detector we evaluate with F_1-scores 0.951–0.997. The strongest cue to the detector is the PDF extraction artifact: human texts retain their original line breaks, whereas generated texts are extracted as a single block of prose. This cue, together with three other formatting features, tell the classes apart at 0.996 AUC. To distinguish benchmarks that contain such a signal from detectors that rely on one, we introduce format counterfactuals: just re-wrapping a machine-written abstract without changing a single word reverses 20-27% of a public detector's correct decisions. Normalizing formatting before training reduces the generalization gap from 0.32 to 0.22. On format-clean benchmarks, it also raises out-of-domain macro-F_1 from 0.33 to 0.65 for detectors trained on artifact-containing subsets. We release our tools and scripts to reconstruct the corrected subsets[1].

1 Introduction

Establishing whether a document was written by a person or a language model is a provenance problem, and in the scientific literature it has become a practical one: an analysis of 15 million PubMed abstracts estimates that 13.5% of the papers written in 2024 were LLM-assisted (Kobak et al., 2025). Publishers, reviewers, and archives consequently need detectors they can trust, and trust in a detector usually depends on the benchmark that measured its performance.

The benchmarks for machine-generated text detection report that the problem is close to be solved. During shared tasks on machine-generated text detection, participants build systems above 0.97 accuracy (Appendix B), and the authors of GPABench2 report over 99.9% accuracy on fully GPT-written abstracts (Liu et al., 2024). This is difficult to reconcile with the fact that generators have improved continuously over the same period: if the underlying task were being measured, it should have become harder, not easier.

We argue that for several of these benchmarks, a large part of the reported performance is dependent on how the benchmarks were assembled, which often originates in document processing. Human documents in these collections are usually scraped or converted from source formats and retain the formatting of those sources; meanwhile machine documents are emitted directly by an API as clean text blocks. The difference in formatting is often not immediately visible, while being trivially learnable and exploitable. In the arXiv subsets we examine, a classifier over four content-free formatting features separates the classes at 0.996 AUC.

Our contributions are as follows:

1. We conduct a surface-attribution analysis of ten scientific-document subsets from six resources, showing that content-independent stylistic features can match or exceed the corresponding shared-task winners. We also identify the surface dimensions that carry these shortcuts.

2. We introduce format counterfactuals and the Artifact Reliance Score: a content-preserving intervention and a model-agnostic measure of a detector's reliance on formatting when making machine-class decisions. It requires

[1] Code to the paper

The Proceedings of Workshop on Document Intelligence and Understanding, pages 174–185
October 24, 2026 ©2026 Association for Computational Linguistics

neither retraining nor access to model internals.

3. We show that artifact exploitability is tokenizer-dependent: SentencePiece-based models discard the whitespace cue, so detector performance scores on these benchmarks are not comparable across tokenizer families.

4. We show that normalizing formatting before training improves cross-benchmark generalization by reducing reliance on document-construction artifacts, including when detectors trained on artifact-containing datasets are evaluated on format-clean benchmarks.

2 Related Work

Detection methods. Machine-generated text detection is usually formulated as binary classification of human and machine authorship (Fraser et al., 2025). There are three families of methods that are usually used for this task: first, statistical and stylometric methods that use token-probability or linguistic features (Gehrmann et al., 2019; Jawahar et al., 2020). Then, zero-shot probability methods threshold text-level scores such as perplexity or log-rank (Su et al., 2023; Hans et al., 2024), or compare statistics of perturbed and original text (Mitchell et al., 2023). Finally, fine-tuned language models (e.g. RoBERTa, DeBERTa) are the current state of the art (Macko et al., 2023; He et al., 2023) but dependent on substantial labelled data.

Artifacts and benchmark difficulty. Several surveys catalogue MGT detection benchmarks (Fraser et al., 2025; Wu et al., 2025), but only few examine the *quality* of the data itself, despite its direct effect on detector reliability (Gritsai et al., 2025). That detection benchmarks may be separable for the wrong reasons is not a new suspicion. Dugan et al. (2024) report formatting confounds in RAID, naming numbered lists in generated recipes and periodic newlines as such artifacts and estimate their effect on detection as small; on the scientific subsets we examine, the same class of cue is instead dominant, reaching 0.996 AUC and exceeding a fine-tuned transformer. Ayoobi et al. (2025) argue then more broadly that existing benchmarks over-represent easy examples and propose hardness-aware construction. Our framing follows the broader dataset-artifact literature in NLU, where annotation artifacts allow hypothesis-only models to succeed (Gururangan et al., 2018) and motivate bias-only debiasing methods (Clark et al., 2019; Karimi Mahabadi et al., 2020).

Shared tasks. MGT shared tasks have driven progress by releasing curated data and evaluation protocols across multilingual, cross-domain, and cross-generator settings (Kashnitsky et al., 2022; Ayele et al., 2024; Wang et al., 2024a). Some participants of the shared tasks apart from detectors also contribute secondary analyzes of given datasets focused on identifying construction biases, false-positive rates on human texts, and repetitive patterns in synthetic outputs (Voznyuk and Konovalov, 2024), which motivate the dataset-quality focus of this work.

3 Benchmarks Analyzed

We analyze ten subsets of datasets with scientific documents drawn from six published resources. Datasets were selected on four criteria: the documents are scientific artifacts (paper abstracts, paper introductions, or peer reviews); the task is binary human-vs-machine classification; both classes are present in the release; and the machine class is written from scratch from a title or prompt. The last criterion matters because rewriting a human text conditions the generation on human writing and yields a distribution closer to human and AI co-writing than to independent generation (Zhang et al., 2024). Where a resource offers several generation regimes we therefore keep only the from-scratch one: for example, for GPABench2 we use the GPT-WRITE task, and for CHEAT we use the *generation* split. All selected subsets are in English. See Appendix D for more details on each chosen dataset.

The M4 family. Six of the ten subsets descend from a single family of shared-task resources. First, from **M4** (Wang et al., 2024b) we take texts on two scientific domains, arXiv abstracts and peer reviews adapted from PeerRead, Kang et al., 2018. **SemEval-2024 Task 8** (Wang et al., 2024a) built its detection-related subtasks on M4, and **COLING-2025 GenAI Task 1** (Wang et al., 2025) in turn extended that data with more recent generators, including GPT-4o, LLaMA-3, Gemma-2, and Mixtral.. Each generation therefore inherits its human counterpart from the previous release.

Full scientific papers. Another analyzed dataset, **IDMGSP** Mosca et al. (2023), collectes full machine-generated papers and not just abstracts,

pairing human articles from arXiv with generations from SCIgen, GPT-2, GPT-3, ChatGPT, and Galactica, and splitting each record into title, abstract, introduction, and conclusion fields. We use the introduction, as it is usually the longest yet representative of a document.

Large-scale abstract collections. **GPABench2** (Liu et al., 2024) contains human, GPT-written, GPT-completed, and GPT-polished abstracts across computer science, physics, and the humanities and social sciences, generated with four prompt templates per task. We kept only computer science and physics subsets. **CHEAT** (Yu et al., 2025) provides title–abstract pairs for computer-science papers with human abstracts collected from IEEE Xplore, and ChatGPT abstracts written from the title and keywords.

4 Diagnosis

We first establish that the benchmarks contain exploitable patterns, and identify what they are. We used balanced samples of up to 10,000 documents per class per subset, with 5-fold cross-validation and 1000-sample bootstrap intervals.

4.1 Surface Probe

We fit a logistic regression with 27 content-free surface features: document, sentence and word lengths, type–token ratio, and per-character rates of punctuation, digits, uppercase, whitespace runs, newlines, and non-ASCII characters. All features are described in Appendix A. No feature uses actual word content, so the classifier cannot exploit topic or vocabulary. We report its macro-F_1 as an estimate of how far a detector can get on a benchmark without reading the text, and the univariate AUC of each feature as an attribution. The joint score can be used to determine whether a shortcut exists, while the univariate scores points out which surface dimension carries it.

Table 1 reports the probe. It reaches macro-F_1 between 0.833 and 0.997 without access to a single word identity. For comparison, the winning system of SemEval-2024 Task 8 (monolingual) reached 0.975 accuracy and of COLING-2025 Task 1 (monolingual) 0.831 macro-F_1 (Appendix B).

The FMT-ONLY column isolates the four formatting features, namely newline rate, whitespace-run rate, curly-quote rate and em-dash rate. On the three arXiv subsets these four alone reach AUC 0.943-0.997, and normalizing the text, which sets these features to chance by construction, costs the full probe 0.199-0.268 macro-F_1. Lexical TF-IDF score after this transformation stays unchanged. On the remaining seven subsets, the same normalization changes almost nothing ($\Delta \leq 0.027$).

Therefore, we can observe two distinct artifact families. The arXiv subsets carry a pipeline artifact, precisely a formatting difference introduced by how the documents were collected. The abstract collections, GPABench2 and CHEAT, are format-clean (FMT-ONLY AUC 0.52-0.55) yet almost perfectly separable lexically (TF-IDF ≥ 0.99), driven by a mean-word-length gap that no preprocessing can remove. This is an actual stylistic difference between human and machine scientific prose. The distinction defines what detector may learn: pipeline artifacts are spurious by construction, while stylistic differences give legitimate signal, although even genuine stylometric features can invert across settings and mislead detectors (Gao et al., 2026). We also note that document length, often the first suspected confound, is not found anywhere except IDMGSP and M4 peer reviews.

4.2 Nature of this Artifact

The formatting artifact has a mechanical explanation. Human arXiv abstracts were collected with the line wrapping of their source intact, because the median line length in these texts is 76 characters implying the classic 80-column limits. At the same time, generations were returned by an API as a single unwrapped block, with median line length 479-481. Measured over the full subsets, every human abstract in M4 and SemEval24 contains a double space, against 2.2% and 2.0% of machine abstracts; newlines appear in 100% of human documents against 53.1% and 66.3% of machine ones. The predicate of double space `"  "` in text is thus a 99%-accurate classifier for SemEval-2024's arXiv subtask.

The artifact was partly fixed: in COLING-2025 there is no gap with double-space (4.3% human against 6.8% machine), but the newline gap remained (90.6% against 26.3%). Then, another example is IDMGSP, where machine introductions carry newlines (24%) and human ones do not (0%). That reversal will be used as a control in Section 7.

5 Method

Section 4 determines exploitable patterns on these benchmarks. It does not establish that any detector

Table 1: Surface probe. Raw and Norm. are macro-F_1 before and after format normalization. FMT-ONLY and LEN-ONLY are AUCs of the four formatting features, and of document length alone. TF-IDF is a lexical reference on the same folds. The three arXiv subsets lose 0.20-0.27 F_1 to normalization; the other lose nothing.

Subset	n	Raw	Norm.	Δ	FMT-ONLY	LEN-ONLY	TF-IDF
SemEval24 arXiv	20,000	**0.997**	0.729	**0.268**	**0.997**	0.503	0.880
M4 arXiv	20,000	**0.995**	0.746	**0.249**	**0.996**	0.635	0.917
COLING25 arXiv	20,000	0.951	0.752	**0.199**	0.943	0.613	0.916
M4 PeerRead	5,860	0.959	0.954	0.005	0.822	0.780	0.998
SemEval24 PeerRead	4,714	0.941	0.914	0.027	0.709	0.541	0.947
COLING25 PeerRead	6,600	0.854	0.839	0.015	0.708	0.488	0.967
IDMGSP	16,000	0.907	0.888	0.019	0.785	0.777	0.961
GPABench2 CS	20,000	0.940	0.938	0.002	0.547	0.596	**0.999**
GPABench2 Physics	20,000	0.935	0.934	0.001	0.524	0.560	**0.999**
CHEAT	20,000	0.833	0.831	0.002	0.519	0.669	0.992

uses them: the probe of Section 4.1 gives evidence about the data and not about detectors. Answering the second question requires intervening on the input, which is what this section develops.

5.1 Format Counterfactuals

Admissible operators. We call a text transformation g *admissible* if g alters the formatting / punctuation of the text and never the words themselves. Admissibility is machine-checkable: source and transformed document must be character-identical once all non-alphanumeric characters are removed, and we verify this on every transformed document we produce. For a formatting dimension F we require an admissible pair: a strip_F that removes F, and an $\mathrm{impose}_F(\cdot\,;\theta)$ that re-applies it according to a profile θ measured from data rather than chosen by us. In this work, we chose F to be the following: let function strip collapse whitespace runs and newlines to single spaces and folds curly quotes and en/em dashes to ASCII; then $\mathrm{impose}(\cdot\,;\theta)$ rewraps lines at line lengths drawn from θ's empirical distribution, inserts double spaces after sentence boundaries, and restores curly quotes and dashes, each at θ's measured rate. We fit θ_H and θ_M separately on the human and machine documents of each subset, so the counterfactual is calibrated to the benchmarks.

Artifact Reliance Score. Let $\mathcal{Y} = \{H, M\}$ and $y(x) \in \mathcal{Y}$ be the gold label of document x, $D(x) \in \mathcal{Y}$ a detector's decision, and $s(x) \in [0,1]$ its machine-class score, so that $D(x) = M$ iff $s(x) \geq \tau$. Let $\mathcal{M} = \{x : y(x) = M\}$ be the machine documents of a subset and $T_M = \{x \in \mathcal{M} : D(x) = M\}$ those the detector originally classifies correctly. We then measure how often a formatting-only edit reverses a correct decision, and subtract a placebo edit that imposes the machine profile instead:

$$\mathrm{ARS} = \frac{1}{|T_M|} \sum_{x \in T_M} \mathbf{1}[D(\mathrm{impose}(x;\theta_H)) = H]$$

$$\mathrm{ARS}^{\mathrm{plc}} = \frac{1}{|T_M|} \sum_{x \in T_M} \mathbf{1}[D(\mathrm{impose}(x;\theta_M)) = H]$$

$$\mathrm{ARS}_{\mathrm{net}} = \mathrm{ARS} - \mathrm{ARS}^{\mathrm{plc}}$$

The placebo is an edit of the same kind that does not move the document towards the human class; it allows us to separate artifact reliance from generic instability under formatting edits. Both terms are evaluated on the same documents, so we test the difference with a paired bootstrap over T_M.

$\mathrm{ARS}_{\mathrm{net}}$ is the fraction of correct decisions that a formatting edit reverses beyond the placebo, however T_M is not a random sample: for a detector with low machine-class recall it plausibly overrepresents documents whose formatting already resembles the machine profile. Injected human profile moves these documents furthest, so conditioning can inflate the effect it measures. We introduce our primary statistic, the mean drop in the machine score under the counterfactual minus the same drop under the placebo:

$$\Delta_{\mathrm{cf}} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \big[s(x) - s(\mathrm{impose}(x;\theta_H))\big]$$

$$\Delta_{\mathrm{plc}} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \big[s(x) - s(\mathrm{impose}(x;\theta_M))\big]$$

$$\Delta_{\mathrm{net}} = \Delta_{\mathrm{cf}} - \Delta_{\mathrm{plc}}$$

This metric requires no decision threshold, so it is comparable across the detector families. Also,

since it is computed over all of $\mathcal{M}$, it is not inflated by selection. Nevertheless, a detector with low machine-class recall has little probability mass to move, so we report machine-class recall alongside both statistics.

Scope of the operator family. The construction is not specific to whitespace. Any formatting dimension admits the same treatment, provided the three conditions above hold: a machine-checkable invariant, a profile estimable per class, and both transformation directions. Another example of such operator is paragraph and list segmentation, and it is well motivated, since Dugan et al. (2024) report numbered lists as a comparable structural artifact in generated text. We instantiate whitespace because that is the dimension the probe selects on these subsets; on the abstract collections it instead selects mean word length, however, it is a lexical dimension, for which no admissible operator exists.

6 Experimental Setup

We take one representative of each family described in Section 2. For fine-tuned language models we train `DeBERTa-v3-base` (He et al., 2023) on each subset. For zero-shot likelihood methods we use Binoculars (Hans et al., 2024), which compares the perplexity and cross-perplexity of two models, and Fast-DetectGPT (Bao et al., 2024), a more efficient variant of DetectGPT (Mitchell et al., 2023) based on conditional probability curvature. Neither zero-shot method requires fine-tuning, which matters in practice because training a detector for every domain and generator is infeasible.

Per subset we hold out 1000 balanced test documents and train on up to 20,000 balanced documents, subject to availability. Hyperparameters and the full grid are given in Appendix A. For the surface probe and the counterfactual experiments we use the samples and operators described in Sections 4.1 and 5.1.

7 Results

7.1 Baseline Detectors

Table 2 reports the three baseline detectors on the ten subsets. Best results are marked per column, but no single column should be read as a verdict on a dataset; the columns jointly characterise surface separability, behaviour under perturbation, and difficulty for current detectors. The salient comparison is with the surface probe of Section 4.1: on the arXiv subsets, against a probe macro-F_1 of 0.951-0.997, the best detector reaches 0.868.

Table 2: Baseline detectors on the ten scientific subsets, as macro-F_1 under a random split: a fine-tuned DeBERTa-v3-base, Binoculars (Perpl.) and Fast-DetectGPT (Perturb).

Dataset	DeBERTa	Perpl.	Perturb.
CHEAT	**0.892**	0.901	0.908
COLING25 (arXiv)	0.824	0.733	0.868
COLING25 (PeerRead)	0.680	0.804	0.946
GPABench2 (CS)	0.781	0.834	0.984
GPABench2 (Physics)	0.767	0.795	**0.998**
IDMGSP	0.759	0.745	0.965
M4 (arXiv)	0.795	0.738	0.824
M4 (PeerRead)	0.794	**0.924**	0.987
SemEval24 (arXiv)	0.769	0.702	0.786
SemEval24 (PeerRead)	0.808	0.922	0.965

7.2 Artifact Reliance

We apply the counterfactual of Section 5.1 to three publicly available and commonly used detectors, none trained on these benchmarks: `chatgpt-detector-roberta`, the detector released with HC3 dataset (Su et al., 2024); `roberta-base-openai-detector`, OpenAI's GPT-2 output detector[2]; and RADAR (Hu et al., 2023), which is adversarially trained against a paraphraser. Each is scored on all 40,000 counterfactual documents, whose admissibility is asserted at construction.

Table 3 reports $\mathrm{ARS}_{\mathrm{net}}$ (FLIP) alongside Δ_{net} over all 500 machine documents (UNC), together with machine-class recall (REC).The conditioned $\mathrm{ARS}_{\mathrm{net}}$ is consistently larger than Δ_{net}, as the selection argument of Section 5.1 predicts. For `chatgpt-detector-roberta`, re-wrapping the lines of a machine-written arXiv abstract reverses 42-58% of its correct decisions relative to the placebo; the unconditioned margin is $+0.20$ to $+0.27$. The three detectors rank as following: the HC3 detector relies most on formatting, OpenAI's GPT-2 detector less, and the adversarially trained RADAR does not rely at all (-0.004 on the arXiv subsets).

Is this a selection artifact? IDMGSP provides a control that no confound can imitate. It is the one subset where the artifact is reversed: machine introductions carry newlines and human ones do not (Section 4.2). If the measurement reflected selection, it could only bias towards positive values, be-

[2] openai-community/roberta-base-openai-detector

Table 3: Artifact reliance of three public detectors under format counterfactuals. FLIP is $\mathrm{ARS}_{\mathrm{net}}$; UNC is Δ_{net}, the placebo-corrected margin shift over all of $\mathcal{M}$; REC is machine-class recall on the original documents ($|T_M|/500$). **fmt** is formatting separability (Table 1), by which subsets are ordered.

		HC3			GPT-2			RADAR		
Subset	fmt	REC	FLIP	UNC	REC	FLIP	UNC	REC	FLIP	UNC
SemEval24 arXiv	0.997	0.318	0.453	**+0.200**	0.420	0.343	**+0.137**	0.226	0.177	−0.006
M4 arXiv	0.996	0.366	0.579	**+0.267**	0.512	0.230	**+0.121**	0.334	0.054	−0.019
COLING25 arXiv	0.943	0.444	0.423	**+0.207**	0.380	0.347	**+0.138**	0.262	0.115	+0.013
M4 PeerRead	0.822	0.224	0.312	+0.097	0.752	0.003	+0.002	0.978	0.000	−0.002
IDMGSP *(rev.)*	0.785	0.478	−0.339	**−0.171**	0.862	−0.005	−0.005	0.634	0.035	+0.046
SemEval24 PeerRead	0.709	0.456	0.171	+0.109	0.628	0.051	+0.023	0.794	0.013	+0.022
COLING25 PeerRead	0.708	0.416	0.062	+0.034	0.534	0.045	+0.025	0.856	0.023	+0.026
GPABench2 CS	0.547	0.284	0.127	+0.042	0.198	0.000	+0.003	0.626	0.000	−0.015
GPABench2 Physics	0.524	0.380	0.079	+0.031	0.254	0.016	+0.002	0.644	0.000	−0.008
CHEAT	0.519	0.632	−0.016	−0.010	0.542	0.004	+0.000	0.914	0.000	+0.001
arXiv mean		0.376	0.485	**+0.225**	0.437	0.307	**+0.132**	0.274	0.115	−0.004
ρ with fmt			+0.770	**+0.745**		+0.552	+0.600		+0.813	−0.103

cause the documents a weak detector classifies correctly are the ones already most machine-like. Instead, `chatgpt-detector-roberta` reverses sign on IDMGSP, significantly and without conditioning (-0.171, CI $[-0.206, -0.138]$). Since IDMGSP's newlines appear mostly in machine texts, it is the machine profile that carries the human-looking cue, and imposing it flips 41.8% of correct decisions against 7.9% for the human profile. RADAR, which does not rely on formatting, shows no comparable reversal. Together with the null on CHEAT, whose classes do not differ in formatting at all (-0.010, CI spanning zero), this is the signature of directional sensitivity to the formatting profile.

The residual dependence on the size of the conditioning set is not significant for `chatgpt-detector-roberta` ($\rho = -0.430$, $p = 0.21$ between $|T_M|$ and the unconditioned margin). Though the effect is one-directional: imposing the machine profile on correctly classified human documents flips at most 2% of decisions for any detector. All three treat wrapped text as evidence of human authorship, but none treats unwrapped text as evidence of machine authorship. The same one-sidedness shows in detectors' recall, which is above 0.95 on human documents and between 0.20 and 0.98 on machine ones.

7.3 Meaning of Diagnostic

The probe predicts artifact reliance. Across the ten subsets, each diagnostic's per-subset value correlates (Spearman, $n = 10$) with `chatgpt-detector-roberta`'s $\mathrm{ARS}_{\mathrm{net}}$: the full probe's macro-F_1 gives $\rho = 0.939$ ($p = 0.0001$); the formatting-only AUC gives $\rho = 0.770$ ($p = 0.0092$); and the normalization drop gives $\rho = 0.673$ ($p = 0.0330$). The lexical TF-IDF reference does not ($\rho = -0.552$, $p = 0.098$), so the relationship is specific to surface form rather than to separability in general. A cheap, content-free probe run at release time therefore flags the subsets on which a deployed detector will later prove format-dependent, which is a useful check on its own.

It does not predict cross-benchmark degradation. Against the decontaminated generalization gap Δ_{gen} of Section 7.4, every diagnostic we tested is uninformative and, if anything, pointed the wrong way: probe macro-F_1 $\rho = -0.164$ ($p = 0.65$), formatting-only AUC $\rho = -0.236$ ($p = 0.51$), $\mathrm{ARS}_{\mathrm{net}}$ $\rho = -0.333$ ($p = 0.35$). We report this as a negative result. However, surface separability and transfer failure are distinct properties of a benchmark, and the fact that a benchmark is easy to game does not by itself predict that detectors trained on it will generalize poorly.

With $n = 10$ these correlations support a qualitative conclusion rather than a precise estimate; extending the unit of analysis to (subset $\times$ generator) cells would give more power and is left to future work.

7.4 Transfer and Normalization

Both the probe and the public detectors were assessed within a single benchmark. This section asks whether relying on it can help to perform out-of-domain. We fine-tuned RoBERTa and DeBERTa on each subset and tested on the other nine. After that we asked three questions of the same runs: does the artifact survive in transfer to another

benchmark; does intra-benchmark duplication inflate the scores we report; and does removing the artifact from training data, via normalization or reweighting, close the exploit?

Before running the experiments, we excluded test documents that share a near-duplicate with the training subset. We ran two backbones, because the artifact turned out to be visible to one and invisible to the other.

Exploitability depends on the tokenizer. `DeBERTa-v3-base` uses SentencePiece, which collapses whitespace runs and discards newlines: raw and normalized documents yield identical token sequences, so the model cannot represent the cue that dominates these benchmarks. `roberta-base` uses a byte-pair encoding that preserves both. In-domain macro-F_1 under the cluster-aware split is showed in Table 4.

Table 4: In-domain macro-F_1 on the arXiv subsets under the cluster-aware split: the content-free surface probe, RoBERTa (byte-pair encoding, sees the whitespace cue) and DeBERTa-v3 (SentencePiece, discards it).

Subset	surface probe	RoBERTa	DeBERTa
M4 arXiv	0.995	**0.996**	0.687
SemEval24 arXiv	0.997	**0.997**	0.711
COLING25 arXiv	0.951	**0.995**	0.773

The byte-pair model behaves similarly with the content-free probe; the SentencePiece model performs instead at the level of a probe with its formatting features removed (0.729-0.752, Table 1, column Norm.). A benchmark artifact is therefore not equally available to all detecting systems, and scores on these resources are not comparable across tokenizer families: a benchmark can be trivial for one architecture and appear sophisticated to another.

On the raw text, the byte-pair model reaches 0.995-0.997 in-domain on the arXiv subsets and 0.333 on every one of the four format-clean benchmarks. 0.333 is the macro-F_1 of predicting a single class on a balanced test set, i.e. it is a complete collapse of the model, that indicates that detector has simple learned to count line breaks and cannot be generalized to the texts without this exploit.

Normalization trades in-domain score for generalization. If we normalize the text before training, macro-F_1 in-domain in aggregate drops by 0.074 (0.225 on the arXiv subsets), meanwhile decontaminated out-of-domain macro-F_1 increases by 0.030, reducing the generalization gap by a third, from 0.320 to 0.216 (Table 5). When we restrict to the transfers that are out-of-domain such as arXiv-trained models evaluated on the four disjoint benchmarks, the effect is much larger: mean macro-F_1 rises from 0.333 to 0.652, a gain of $+0.319$, with per-subset gains of $+0.411$ (COLING25 arXiv), $+0.347$ (SemEval24 arXiv) and $+0.197$ (M4 arXiv). Subsets without a formatting artifact barely change (CHEAT from 0.958 to 0.964, GPABench2 CS from 0.995 to 0.995, IDMGSP from 0.965 to 0.958), so the intervention is not a general regulariser.

Replacing the random split with a cluster-aware one costs 0.005 macro-F_1 for RoBERTa and 0.006 for DeBERTa. We noticed significant duplication within some of the subsets, however, as it did not measurably inflate reported scores on either backbone, we simply note it, see more details in Appendix C.

Artifact-debiased training. We tried to run artifact-debiased training by reweighting examples by the surface bias model and report the results in Table 5. It degraded both in-domain (0.981 to 0.836) and out-of-domain (0.662 to 0.657), and is worse than normalization alone on both axes. On the arXiv subsets the bias model reaches $p_{\text{bias}} \approx 0.99$ using whitespace, so almost every example receives near-zero weight, collapsing to majority-class prediction (0.333 ± 0.000 for both backbones) as on SemEval24 arXiv. Where the bias model is informative but not saturated, reweighting does improve out-of-domain performance: GPABench2 CS $0.733 \rightarrow 0.748$, Physics $0.748 \rightarrow 0.751$, IDMGSP $0.669 \rightarrow 0.684$. We conclude that bias-model reweighting requires a bias model that is neither saturated nor operating on features outside the detector's representational reach, because for example DeBERTa tokenizer discards part of the features, so their effect here is near-null.

8 Discussion

On the three arXiv subsets, a logistic regression over content-free surface features outperforms a fine-tuned transformer by 0.18-0.31 macro-F_1: whatever these benchmarks measure, it is not the property they were built to measure, since they can be played with a solution that requires no access to the lexical content. And since this cue is visible only to some tokenizer families (Section 7.4), some systems will have an advantage unrelated to how

Table 5: Performance of fine-tuned detectors. Δ_{gen} is in-domain minus out-of-domain. CLUSTER keeps near-duplicate groups on one side of the split; NORM. additionally normalizes surface form; DEBIAS adds bias-model reweighting.

Backbone	Cond.	in-domain	OOD	Δ_{gen}
RoBERTa	RAND, raw	0.986±0.004	0.666±0.004	0.320
	CLUST, raw	0.981±0.003	0.662±0.014	0.320
	+NORM.	0.907±0.009	**0.692**±0.010	**0.216**
	+DEBIAS	0.836±0.010	0.657±0.018	0.179
DeBERTa	RAND, raw	0.884±0.030	0.653±0.024	0.231
	CLUST, raw	0.878±0.021	0.649±0.012	0.229
	+NORM.	0.878±0.026	0.651±0.022	0.227
	+DEBIAS	0.773±0.035	0.597±0.019	0.175

good the system actually is, but rather to whether or not it inadvertently erases exploitable artifacts. We are, however, not aware of any benchmark in this area that documents which family it advantages.

The defect comes from document processing, which makes it cheap to prevent. Human documents should be normalized to the same surface form as generations before release. We recommend shipping the surface-probe score and per-feature AUCs with any such resource.

9 Conclusion

We audited ten scientific-document subsets of six machine-generated-text detection benchmarks and found that much of their reported difficulty is an artifact of document assembly: a logistic regression over content-free surface features matches or exceeds the corresponding shared-task winners, and on the arXiv subsets the dominant cue is whitespace inherited from how human documents were collected. Format counterfactuals and the placebo-corrected Artifact Reliance Score separate benchmarks that contain such a cue from detectors that rely on one, and whether the cue is exploitable at all depends on the detector's tokenizer. We recommend normalizing surface form before release and publishing a surface-probe score alongside every resource.

Limitations

We analyzed only ten subsets of six resources, all English and all of scientific domain. We do not claim the measured effects will transfer to other domains or languages, and the formatting artifact in particular is a property of specific collection pipelines rather than of machine-generated text, although the reason it appears comes from a widely used processing pipeline, so we believe it should be quite common.

ARS_{net} is reported for three publicly available detectors. Because it conditions on correct decisions, its denominator shrinks with a detector's recall, and the detectors we measured have machine-class recall between 0.20 and 0.978 on this data; the score should therefore not be used to rank detectors of very unequal accuracy. Our correlation analyzes use $n = 10$ subsets, which supports qualitative conclusions but not precise estimates.

The fine-tuned baseline is a single architecture at one size, trained for three epochs with best-epoch selection over three seeds. Its in-domain scores on the arXiv subsets (0.769-0.824) are well below the corresponding shared-task winners, and although we attribute this to the tokenizer's inability to represent the dominant cue, we cannot exclude that a larger or better-tuned model would score higher. Finally, our counterfactual operators cover whitespace and punctuation style only, and other structural operators are considered as future work.

References

Abinew Ali Ayele, Nikolay Babakov, Janek Bevendorff, Xavier Bonet Casals, Berta Chulvi, Daryna Dementieva, Ashaf Elnagar, Dayne Freitag, Maik Fröbe, Damir Korenčić, Maximilian Mayerl, Daniil Moskovskiy, Animesh Mukherjee, Alexander Panchenko, Martin Potthast, Francisco Rangel, Naquee Rizwan, Paolo Rosso, Florian Schneider, and 10 others. 2024. Overview of PAN 2024: Multi-Author Writing Style Analysis, Multilingual Text Detoxification, Oppositional Thinking Analysis, and Generative AI Authorship Verification. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 15th International Conference of the CLEF Association (CLEF 2024)*, Lecture Notes in Computer Science, page 3–10, Berlin Heidelberg New York. Springer.

Navid Ayoobi, Sadat Shahriar, and Arjun Mukherjee. 2025. Beyond easy wins: A text hardness-aware benchmark for llm-generated text detection. *Preprint*, arXiv:2507.15286.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Meth-*

ods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4069–4082, Hong Kong, China. Association for Computational Linguistics.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. 2024. RAID: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12463–12492, Bangkok, Thailand. Association for Computational Linguistics.

Kathleen C. Fraser, Hillary Dawkins, and Svetlana Kiritchenko. 2025. Detecting ai-generated text: Factors influencing detectability with current methods. *Journal of Artificial Intelligence Research*, 82:2233–2278.

Lang Gao, Xuhui Li, Chenxi Wang, Mingzhe Li, Wei Liu, Zirui Song, Jinghui Zhang, Rui Yan, Preslav Nakov, and Xiuying Chen. 2026. When personalization tricks detectors: The feature-inversion trap in machine-generated text detection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 43143–43171, San Diego, California, United States. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

German Gritsai, Anastasia Voznyuk, Andrey Grabovoy, and Yury Chekhovich. 2025. Are ai detectors good enough? a survey on quality of datasets with machine-generated texts. *Preprint*, arXiv:2410.14677.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17519–17537, Vienna, Austria. PMLR.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Xiaomengc Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: robust ai-text detection via adversarial learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (PeerRead): Collection, insights and NLP applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1647–1661.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8706–8716, Online. Association for Computational Linguistics.

Yury Kashnitsky, Drahomira Herrmannova, Anita de Waard, George Tsatsaronis, Catriona Catriona Fennell, and Cyril Labbe. 2022. Overview of the DAGPap22 shared task on detecting automatically generated scientific papers. In *Proceedings of the Third Workshop on Scholarly Document Processing*, pages 210–213, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Dmitry Kobak, Rita González-Márquez, Emőke Ágnes Horvát, and Jan Lause. 2025. Delving into llm-assisted writing in biomedical publications through excess vocabulary. *Science Advances*, 11(27):eadt3813.

Zeyan Liu, Zijun Yao, Fengjun Li, and Bo Luo. 2024. On the detectability of chatgpt content: benchmarking, methodology, and evaluation through the lens of academic writing. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2236–2250.

Dominik Macko, Robert Moro, Adaku Uchendu, Jason Lucas, Michiharu Yamashita, Matúš Pikuliak, Ivan Srba, Thai Le, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2023. MULTITuDE: Large-scale multilingual machine-generated text detection benchmark. In *Proceedings of the 2023 Conference on*

Empirical Methods in Natural Language Processing, pages 9960–9987, Singapore. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, Honolulu, Hawaii, USA,. PMLR.

Edoardo Mosca, Mohamed Hesham Ibrahim Abdalla, Paolo Basso, Margherita Musumeci, and Georg Groh. 2023. Distinguishing fact from fiction: A benchmark dataset for identifying machine-generated scientific papers in the LLM era. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 190–207, Toronto, Canada. Association for Computational Linguistics.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412, Singapore. Association for Computational Linguistics.

Zhenpeng Su, Xing Wu, Wei Zhou, Guangyuan Ma, and Songlin Hu. 2024. Hc3 plus: A semantic-invariant human chatgpt comparison corpus. *Preprint*, arXiv:2309.02731.

Anastasia Voznyuk and Vasily Konovalov. 2024. DeepPavlov at SemEval-2024 task 8: Leveraging transfer learning for detecting boundaries of machine-generated texts. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1821–1829, Mexico City, Mexico. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, and Thomas Arnold. 2024a. SemEval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 2057–2079, Mexico City, Mexico. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta. Association for Computational Linguistics.

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Elozeiri, Saad El Dine Ahmed, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Alexander Aziz, and 8 others. 2025. Genai content detection task 1: English and multilingual machine-generated text detection: Ai vs. human. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, Abu Dhabi, UAE. Association for Computational Linguistics.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong. 2025. A survey on LLM-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338.

Peipeng Yu, Jiahan Chen, Xuan Feng, and Zhihua Xia. 2025. Cheat: A large-scale dataset for detecting chatgpt-written abstracts. *IEEE Transactions on Big Data*.

Qihui Zhang, Chujie Gao, Dongping Chen, Yue Huang, Yixin Huang, Zhenyang Sun, Shilin Zhang, Weiye Li, Zhengyan Fu, Yao Wan, and Lichao Sun. 2024. LLM-as-a-coauthor: Can mixed human-written and machine-generated text be detected? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 409–436, Mexico City, Mexico. Association for Computational Linguistics.

A Implementation and Hyperparameters

All randomness is seeded. A single split seed (0) is shared by every condition and model so that runs differ only in the factor under study; training seeds are $\{0, 1, 2\}$. Splits are formed in two ways: randomly, as is standard, and cluster-aware, so that no near-duplicate group spans the train/test boundary. Transfer evaluations exclude test documents having a near-duplicate in the training subset.

Surface probe (Section 4.1). 27 features per document: four size features (characters, words, sentences, mean sentence length), four word-shape features (mean and standard deviation of word length, type–token ratio, hapax rate), ten per-character punctuation rates, four formatting features (`r_newline`, `r_ws_run`, `r_curly_quote`, `r_fancy_dash`), and five further surface markers (non-ASCII, digit and uppercase rates, numeric-token and LaTeX-command rates). For classifier, we put these standardized features into logistic regression (`max_iter` $= 5000$, L2, default C), evaluated by 5-fold stratified cross-validation on balanced samples of up to 10,000 documents per class per subset. Confidence intervals are 1000-sample bootstrap percentiles over the out-of-fold predictions. The lexical reference is TF-IDF over word

1-2 grams, 50,000 maximum features, `min_df` $= 2$, sublinear term frequency, into the same classifier and folds. TF-IDF is computed on raw text only: its default token pattern discards whitespace and standalone punctuation, so normalization cannot alter any feature value, and we verified the two runs are numerically identical.

Format counterfactuals and ARS (Section 5.1). 500 documents per class per subset, each in four variants (original, normalized, human-profile injection, machine-profile placebo), giving 40,000 scored documents per detector. The admissibility invariant is asserted on every transformed document. Formatting profiles are estimated per class per subset from line-length distributions, double-space rate at sentence boundaries, and curly-quote and dash rates. Detectors are scored with `max_length` $= 512$ and batch size 16.

Fine-tuned detector (Section 7.4). `DeBERTa-v3-base`, 3 epochs, AdamW at learning rate 2×10^{-5}, linear decay with 10% warmup, gradient-norm clipping at 1.0, bfloat16 autocast on CUDA. Maximum sequence length is 512 for M4 peer reviews and IDMGSP introductions and 256 for the remaining subsets, whose medians are below 210 tokens. Test sets are 1000 balanced documents per subset, held fixed across every condition and seed. Clustering for the cluster-aware split considers up to 25,000 documents per subset. The debiasing condition weights each training example by $(1 - p_{\text{bias}})^{\gamma}$ with $\gamma = 1$, mean-normalized and then clipped at ten times the mean. The bias model is the surface probe of Section 4.1, fitted on the training split only. RoBERTa-base uses identical hyperparameters

B Evaluation results of competitions

Table 6 lists the best reported score in each shared task whose data appears in Table 8. These are the figures we use to compare with surface probe (Section 4.1).

Table 6: Best reported score in each shared task represented in our analysis.

Competition	Metric	Best result
SemEval24 Mono	Accuracy	0.975
SemEval24 Multi	Accuracy	0.959
COLING25 Mono	Macro-F_1	0.8307
COLING25 Multi	Macro-F_1	0.7916

C Intra-Benchmark Duplication

Table 7 reports duplication within each subset. We observe that M4 arXiv contains 2,999 duplicate groups of about five members each, summing up to 15,002 rows, which is precisely its 15,000 human documents, and each human abstract is stored once per generator.

The natural expectation is that this inflates reported scores, since a random split places copies of the same document on both sides of it. We tested this directly and find no measurable effect. Replacing the random split with a cluster-aware split that keeps every near-duplicate group completely within one side changes in-domain macro-F_1 by 0.005/ 0.006. the sign is inconsistent across subsets, and a subset's duplicate rate does not predict its change ($\rho = 0.34$, $p = 0.34$). Even M4 arXiv, at 35% duplication, moves only from 0.740 to 0.687, within seed variance. We therefore describe duplication as a defect in effective dataset size and in the interpretation of a nominally balanced split, and not as a source of score inflation.

Table 7: Duplication within each subset. Near-duplicates use MinHash with Jaccard ≥ 0.8; exact matching is on whitespace-normalized text.

Subset	exact	near
M4 PeerRead	40.0%	50.0%
M4 arXiv	35.2%	49.5%
COLING25 PeerRead	28.0%	55.1%
COLING25 arXiv	7.7%	15.7%
SemEval24 PeerRead	1.3%	2.7%
SemEval24 arXiv	0.00%	0.01%
IDMGSP	0.03%	0.24%
CHEAT, GPABench2	$\leq$0.01%	$\leq$0.03%

D Subset Construction Details

Table 8 reports abbreviated generator lists. For the M4 family, scientific subsets were isolated by matching the release's domain field against arXiv and PeerRead. Documents shorter than five whitespace tokens were dropped as degenerate. No class rebalancing was applied at isolation time; the balanced samples used in each experiment are drawn afterwards with a fixed seed.

Table 8: The ten scientific-document subsets analyzed in this work. $|H|$ and $|M|$ are counts of human and machine documents in the isolated subset; median length is in whitespace tokens. All subsets are English and all machine documents are written from scratch. Generator counts are as recorded in each release.

Subset	**Document type**	**|H|**	**|M|**	**Median tokens H / M**	**#Gen.**	**Generators**
M4-based datasets						
M4 arXiv	abstracts	15,000	15,000	198 / 180	5	gpt-3.5-turbo, text-davinci-003, cohere, dolly-v2-12b, flan-t5
M4 PeerRead	peer reviews	2,930	2,930	2173 / 1026	5	chatgpt, davinci, cohere, dolly-v2, flan-t5
SemEval24 arXiv	abstracts	15,498	11,999	194 / 194	4	chatGPT, davinci, cohere, dolly
SemEval24 PeerRead	peer reviews	2,357	9,374	197 / 238	4	chatGPT, davinci, cohere, dolly
COLING25 arXiv	abstracts	22,484	30,684	192 / 178	12	+ gpt4o, gpt4, llama3-70b/8b, gemma2-9b, gemma-7b, mixtral-8x7b, bloomz
COLING25 PeerRead	peer reviews	3,300	16,169	203 / 207	7	+ gpt4o, gpt4, bloomz
Research-paper resources						
IDMGSP	paper introductions	8,000	8,000	423 / 237	5	SCIgen, GPT-2, GPT-3, ChatGPT, Galactica
GPABench2 CS	abstracts	50,000	200,000	153 / 164	1	ChatGPT (4 prompt templates)
GPABench2 Physics	abstracts	50,000	200,000	148 / 155	1	ChatGPT (4 prompt templates)
CHEAT	abstracts (IEEE)	15,395	15,395	156 / 131	1	ChatGPT

Graph Retrieval in Multimodal RAG for Biomedical Documents: A Pharmacokinetics and Pharmacodynamics Case Study

Primoz Kocbek[1,2] **Katarina Mestrovic**[1] **Nguyen Thu Huong**[3]
Ales Fajmut[4] **Gregor Stiglic**[1,5]

[1]University of Maribor, Faculty of Health Sciences [2]University of Ljubljana, Faculty of Medicine
[3]Phan Chau Trinh University [4]University of Maribor, Faculty of Natural Sciences and Mathematics
[5]Usher Institute, The University of Edinburgh

{primoz.kocbek, katarina.mestrovic2, ales.fajmut, gregor.stiglic}@um.si huong.nt@pctu.edu.vn

Abstract

Introduction: Biomedical PDFs distribute answer evidence across prose, tables, figures, and cross-document relations, complicating the retrieval of answer-sufficient context. We examine whether graph and multimodal retrieval provide complementary answers and whether direct concatenation exploits this complementarity. **Methods:** We screen 13 local generators on 205 pharmacokinetics/pharmacodynamics multiple-choice questions and evaluate 11 retrieval conditions over ten runs for five selected models. We quantify questions answered correctly by either or both retrieval routes and compare direct fusion with each standalone route using paired question-level bootstrap intervals. A separate ten-run panel tests answer order, irrelevant context, paraphrasing, generated distractors, and option removal. **Results:** Multimodal retrieval alone answers 11.5–16.2% of questions correctly that graph retrieval misses, while graph retrieval alone answers 7.9–13.9% that multimodal retrieval misses. Counting a question as correct whenever either route succeeds yields an oracle upper bound of 69.5–77.7%, exceeding direct concatenation by 8.4–13.5 percentage points. Direct fusion improves on the stronger standalone route by at most 0.6 points; paired intervals include zero in all ten model–depth comparisons. In repeated evaluation, Gemma-4-31B with multimodal retrieval achieved the highest accuracy (66.6%, 95% CI 65.4–67.8). **Discussion:** The central finding is substantial answer-level complementarity that direct concatenation does not reliably exploit. The oracle bound measures retrospective selection, not an implemented routing method. These findings motivate evaluating source-aware routing or reranking to select between complementary retrieval routes.

1 Introduction

Retrieval-augmented generation (RAG) grounds model outputs in external evidence (Lewis et al., 2020), but biomedical PDFs complicate evidence retrieval. A single answer may depend on prose that identifies a drug or endpoint and a table or figure that provides the decisive dose, population, or numerical value. Document-level recall is therefore insufficient: the system must retrieve answer-sufficient evidence while preserving its source and page context.

Our previous study in glycobiology compared text conversion with OCR-free page retrieval on 120 questions (Kocbek et al., 2025). Its results showed that retrieval effectiveness depended on both evidence representation and generator capability. The present PK/PD study shifts the focus from document conversion to evidence selection: once text, tables, figures, and pages are indexed, does graph retrieval improve biomedical question answering, and can graph and multimodal evidence be combined effectively?

Graph retrieval can connect entities and relations that are separated across chunks or documents. LightRAG integrates graph and vector representations with low- and high-level retrieval (Guo et al., 2025b); our pipeline extends its input with source-linked text, tables, figures, and image descriptions, consistent with the multimodal design of RAG-Anything (Guo et al., 2025a). This expansion may recover otherwise fragmented evidence, but it may also introduce generic entities, indirect relations, or excessive context that displaces the exact passage, table cell, or figure needed for an answer.

Figure 1 summarizes the evidence-first architecture, which uses MinerU for PDF extraction (Wang et al., 2024), Milvus for source-linked search (Wang et al., 2021), and model-isolated LightRAG graphs. Across thirteen local generators, we compare text, visual-asset, multimodal, page-level late-interaction, graph, and graph–multimodal retrieval. Five selected models are evaluated over ten repeated runs; additional pertur-

The Proceedings of Workshop on Document Intelligence and Understanding, pages 186–195
October 24, 2026 ©2026 Association for Computational Linguistics

bations test answer order, irrelevant context, paraphrasing, distractors, and option removal. A central contribution is the answer-level complementarity analysis: graph and multimodal routes answer partly different questions correctly. Selecting the correct answer whenever either route succeeds yields an oracle upper bound 8.4–13.5 percentage points above direct concatenation.

2 Related Work

Building RAG systems for biomedical documents requires a common representation that can support several retrieval routes. Scientific PDFs must first be converted into structured, source-linked evidence that preserves prose, tables, figures, captions, and page provenance. We use MinerU because a single extraction pipeline can be reused for text retrieval, multimodal ingestion, and LightRAG graph construction: text and Markdown support chunking and graph extraction, while tables, figures, crops, and captions provide typed multimodal evidence (Wang et al., 2024). This shared preprocessing avoids separate document conversions for each retriever and keeps their outputs aligned to the same source locations.

The resulting representations also require a storage layer that can serve different indexing strategies. Milvus provides scalable vector management and similarity search (Wang et al., 2021) while separating storage, indexing, and query execution (Guo et al., 2022). In our architecture, it stores dense text and visual embeddings, LightRAG chunks, entities, and relations, and the page-level multi-vector representations required by late-interaction retrievers such as ColPali and ColQwen. Document, page, modality, retrieval-model, and generator identifiers preserve provenance and isolate model-specific collections while allowing all retrieval routes to use a common backend.

Multimodal retrieval can index extracted assets or rendered pages. ColPali extends ColBERT-style late interaction (Khattab and Zaharia, 2020) by matching query tokens against page-patch embeddings (Faysse et al., 2025). This preserves localized visual signals without cross-encoding every page, but increases storage and similarity cost; reducing page vectors trades efficiency against quality (Ma et al., 2025). Joint image–text memory can outperform text-only retrieval (Chen et al., 2022), yet realistic multimodal systems remain vulnerable to tables and rephrased queries (Wasserman et al., 2025). A highly ranked page is therefore not necessarily answer-sufficient evidence.

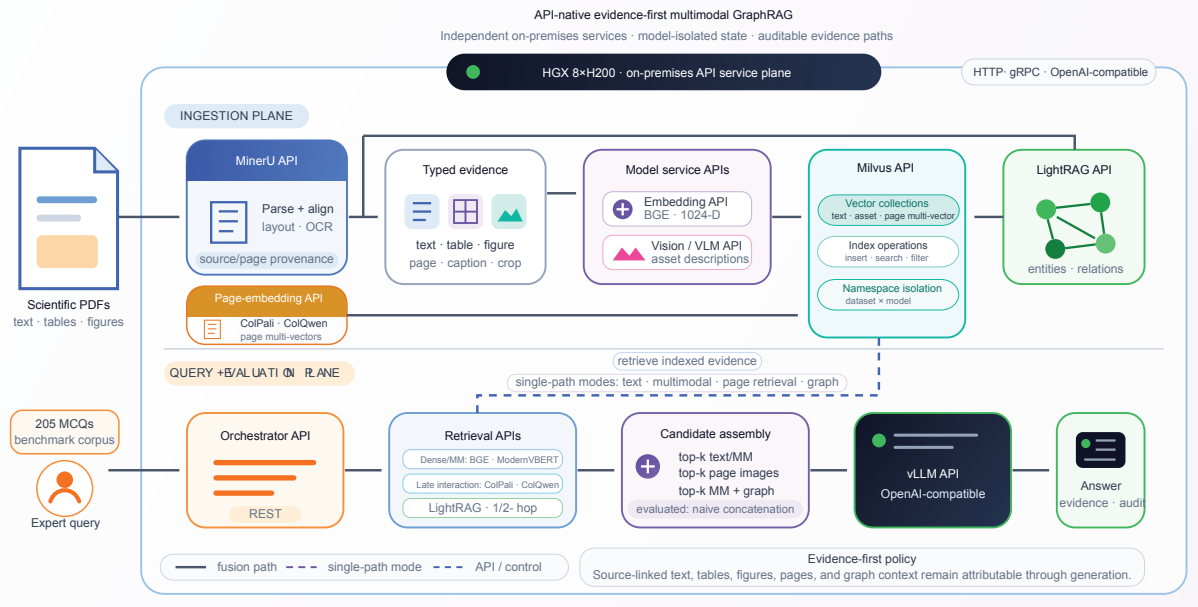
Graph retrieval represents relations spanning chunks, pages, or documents. GraphRAG uses entity graphs and community summaries (Edge et al., 2024); LightRAG combines graph indexing, vectors, dual-level retrieval, and incremental updates (Guo et al., 2025b). Its RAG-Anything integration supports interleaved text, tables, images, and equations (Guo et al., 2025a). We ask whether graph structure adds decisive biomedical evidence and whether it can be combined with multimodal candidates without indirect context displacing direct evidence.

Multiple-choice questions (MCQs) permit consistent scoring across models and runs without a generative judge, but options can reveal cues, support elimination, and make results sensitive to order and distractor quality (Balepur et al., 2025). Surface validity is distinct from psychometric plausibility (Feng et al., 2024), and choice-aware generation can produce more discriminating alternatives than generic methods (Lee et al., 2025). We therefore treat MCQ accuracy as an evaluation instrument, and audit answer order, paraphrasing, distractor replacement, and option removal.

3 Materials and Methods

3.1 Benchmark and task

The benchmark contains 205 PK/PD MCQs: 68 easy, 69 medium, and 68 hard. A master's-trained domain researcher developed and refined the questions with input from a doctoral student, using 60 source papers. The benchmark workbook records source DOIs and answer locations in pages, tables, figures, or text. Multiple development iterations reviewed the questions and retrieval-difficulty labels: when a model correctly answered a hard question with sufficient supporting explanation, the question was downgraded or replaced. This constitutes iterative domain review; independent expert validation remains a future step. It covers antimicrobial PK/PD, with particular emphasis on beta-lactam therapy. Questions address dosing and infusion strategies, exposure targets, therapeutic drug monitoring, tissue distribution, efficacy and toxicity, and dose individualization in clinically complex populations. Easy questions use explicit textual evidence; medium questions require tables, figures, or within-document synthesis; hard ques-



Note: MinerU produces source-linked text, tables, figures, captions, crops, and page images. Independent services create dense, multimodal, and page-level multi-vector representations. Milvus manages model-isolated collections, while LightRAG maintains entity–relation retrieval. At query time, the orchestrator invokes individual retrieval routes or their evaluated concatenations before evidence-grounded generation through an OpenAI-compatible vLLM endpoint.

Figure 1: API-native, evidence-first multimodal GraphRAG architecture and evaluation workflow.

tions integrate sections or papers. Options are included in the retrieval query because a distinguishing drug, dose, endpoint, or value may appear only there. The primary outcome is exact option accuracy. Document hit rate indicates whether the linked paper was retrieved, not whether the context was answer sufficient.

3.2 Parsing and evidence preservation

MinerU converts PDFs into structured Markdown and visual assets. Documents longer than 20 pages are rendered and processed in consecutive 20-page batches to control memory use and processing time. The MinerU outputs are then reassembled in their original page order before ingestion, with global page numbers and source provenance preserved across batch boundaries. Typed table records retain body, caption, footnote, crop, source, and page; figure records retain crop, caption, local text, source, and page. This follows MinerU's structured extraction (Wang et al., 2024) and RAG-Anything's interleaved multimodal representation (Guo et al., 2025a).

Text chunks and visual descriptions are embedded and stored in Milvus (Guo et al., 2022). Approximate search uses a Hierarchical Navigable Small World (HNSW) index, a multilayer proximity graph that efficiently narrows the search from broad neighborhoods to nearby vectors (Malkov and Yashunin, 2020); page-image retrievers use ColPali-style late interaction (Faysse et al., 2025).

3.3 Native LightRAG graph

LightRAG constructs graphs from MinerU Markdown and typed multimodal records (Guo et al., 2025b). LLM-based indexing extracts domain entities and relations; query keywords select entities and higher-level concepts. Each dataset–model pair uses a separate Milvus database to isolate graphs, vectors, and assets.

Context prioritizes concrete text windows, table bodies, figure-local text, source links, and page metadata over graph summaries. We evaluate one- and two-hop traversal. Combined conditions concatenate LightRAG nodes and relations with the top 10 multimodal candidates, without learned gating or reranking (Section 5.1). Graph-MM 1-Hop and Graph-MM 2-Hop denote these direct-concatenation baselines.

3.4 Retrieval conditions and models

We evaluate zero-shot, text, visual-asset, and multimodal RAG; ColPali, ColQwen, and ModernVBERT page retrieval; graph retrieval at one and two hops; and graph–multimodal retrieval at both depths. The 13-model screen spans dense and sparse mixture-of-experts generators with 4B–428B total parameters. Models are served locally with vLLM 0.24.0 on one to four H200 GPUs, using checkpoint-native BF16, FP8, NVFP4, INT4,

W4A4, or MXFP8 formats and context windows of 110,000–262,144 tokens (Kwon et al., 2023). Sampling follows checkpoint recommendations; the five repeated models use temperature 1.0 and top-$p = 0.95$, with top-$k = 64$ for Gemma, 20 for Qwen, and no explicit top-k for MiniMax-M3. Retrieval uses top-$k = 10$. The random seed was unset; the runtime default is presumed to be 0.

3.5 Screening and repeated analysis

In Stage 1, we screen 13 generators once across zero-shot, text, multimodal, graph, and graph–multimodal conditions. Accuracy, model size, and local feasibility are then used to select five models: Gemma-4-31B, Gemma-4-26B-A4B, Qwen3.6-27B, Qwen3.6-35B-A3B, and MiniMax-M3. In Stage 2, we repeat all 11 conditions ten times for these models. Stages 1 and 2 use exactly the same 205 questions, with no development/test split. Each completed run is one statistical sample ($n = 10$ runs per model–condition), and 95% intervals are calculated across the ten run-level metric estimates to quantify run-to-run variability. For Gemma-4-31B, we also aggregate accuracy, total tokens, and hit rate within each run and difficulty stratum.

Within each run, we classify each question as correct under both routes, multimodal only, graph only, or neither. The union of the first three defines an ideal-selection upper bound; route-only categories measure complementarity, and its gap from direct fusion measures unrealized potential. Complementarity estimates are then summarized across the same ten run-level samples ($n = 10$). We additionally calculate paired 95% question-cluster bootstrap intervals for direct fusion minus each standalone route, by resampling questions 10,000 times. Each sampled question retains its results from all ten runs and all compared methods; duplicate questions are sampled together. This upper bound uses oracle selection: it chooses the successful route after the correct answer is known. It therefore measures the potential value of route selection, but not the performance of a deployable system that must choose a route before answering.

3.6 Robustness perturbations

In Stage 3, we evaluate the five selected generators under zero-shot and text retrieval over ten runs per condition. Answer shuffling probes positional bias (Pezeshkpour and Hruschka, 2024); context noise probes distractibility (Shi et al., 2023; Liu et al., 2024); and paraphrasing probes semantic invariance and can support decontamination audits (Yang et al., 2023). Generated distractors replace incorrect options, while the open-ended condition removes all options (Feng et al., 2024; Lee et al., 2025; Balepur et al., 2025).

We pool binary responses to obtain 95% Agresti–Coull intervals, which estimate a binomial proportion using adjusted success counts and an adjusted effective sample size, providing more stable coverage than the simple Wald interval (Agresti and Coull, 1998). Each perturbation is compared with its baseline using a two-proportion z-test; baselines have no test. Results are exploratory because pooling ignores question pairing and repeated-run dependence.

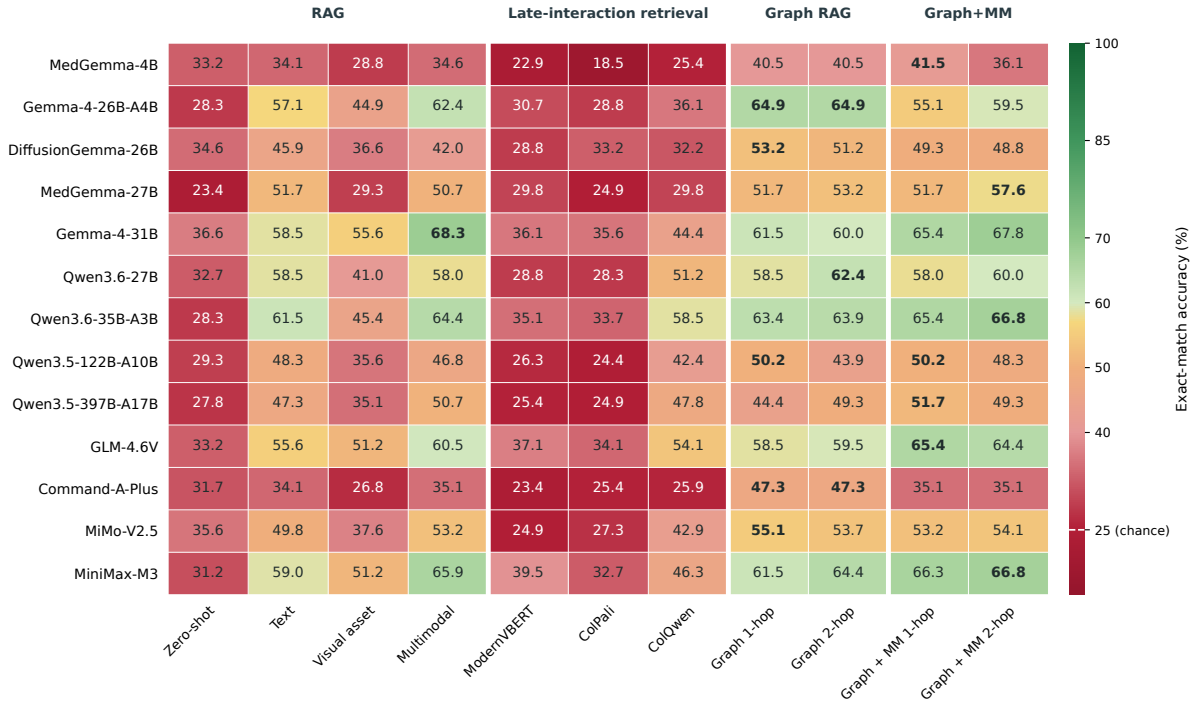
4 Results

4.1 Stage 1: broad single-run screen

Figure 2 shows that no retrieval architecture dominated the broad 13-generator screen. Gemma-4-31B was strongest with multimodal retrieval (68.3%), Gemma-4-26B-A4B with both one- and two-hop graph retrieval (64.9% for each), and Qwen3.6-35B-A3B with two-hop graph–multimodal fusion (66.8%). ColQwen raised Qwen3.6-27B page retrieval to 51.2%, compared with 28.3% for ColPali, but remained below that model's best graph result of 62.4%. This heterogeneity motivated selection of five models for repeated evaluation in Stage 2.

4.2 Stage 2: retrieval analysis (n=10)

Table 2 compares the 11 repeated conditions. Gemma-4-31B is highest with multimodal retrieval (66.6%, 95% CI 65.4–67.8), followed by Qwen3.6-27B with multimodal retrieval (65.1%, 63.1–67.2) and MiniMax-M3 with one-hop direct fusion (65.0%, 62.8–67.3). Direct fusion exceeds the better same-depth component only for Qwen3.6-35B-A3B (+0.6 and +0.1 percentage points at one and two hops) and MiniMax-M3 at one hop (+0.5 percentage points). Paired question-cluster intervals for direct fusion minus the stronger standalone route include zero in all ten model–depth comparisons. These exploratory results concern direct concatenation, not evidence fusion generally.



Note: Cells report exact-match accuracy on 205 four-option questions. Graph+MM denotes direct concatenation of graph and multimodal candidates; white separators group conventional, page-level, graph, and fused retrieval routes.

Figure 2: Accuracy across generators and retrieval architectures in Stage 1.

Table 2: Retrieval comparison in Stage 2 across $n = 10$ runs.

Model	Metric	Zero-shot generation	Text retrieval	Visual assets	Multi-modal	Modern-VBERT	Col-Pali	Col-Qwen	Graph 1-hop	Graph 2-hop	Graph+MM 1-hop	Graph+MM 2-hop
Gemma-4-31B	Acc.	36.2 [35.0–37.5]	60.3 [59.5–61.1]	54.5 [53.1–55.9]	**66.6** [65.4–67.8]	33.7 [32.7–34.8]	36.6 [35.1–38.1]	46.6 [45.1–48.1]	61.3 [60.2–62.4]	60.6 [59.4–61.9]	65.8 [64.5–67.0]	66.1 [65.3–66.9]
	Hit	0.0 [n/a]	93.2 [93.2–93.2]	88.8 [88.8–88.8]	94.6 [94.6–94.6]	33.7 [33.7–33.7]	31.7 [31.7–31.7]	89.3 [89.3–89.3]	**96.9** [96.7–97.1]	96.3 [95.9–96.7]	93.0 [92.4–93.6]	92.7 [92.3–93.1]
	Tok.	**564/125** [564–564;124–125]	5,363/129 [5,363–5,363;128–130]	6,292/165 [6,292–6,292;162–167]	5,568/140 [5,568–5,568;139–141]	3,783/140 [3,783–3,783;138–142]	3,775/150 [3,775–3,775;147–153]	3,774/160 [3,774–3,774;156–163]	5,796/131 [5,782–5,809;130–133]	5,886/131 [5,876–5,895;131–132]	6,076/130 [6,063–6,090;129–131]	6,132/130 [6,118–6,146;129–131]
Gemma-4-26B-A4B	Acc.	30.7 [29.1–32.2]	57.0 [55.0–59.0]	43.8 [42.4–45.1]	**60.8** [59.6–62.0]	32.7 [30.5–34.9]	33.0 [31.8–34.3]	40.1 [37.6–42.6]	59.6 [58.2–60.9]	59.9 [58.8–60.9]	58.3 [56.9–59.8]	59.1 [57.9–60.2]
	Hit	0.0 [n/a]	92.7 [92.7–92.7]	90.1 [89.9–90.3]	94.1 [94.1–94.1]	33.7 [33.7–33.7]	33.7 [33.7–33.7]	89.3 [89.3–89.3]	95.4 [95.0–95.7]	**96.3** [96.0–96.5]	92.5 [92.2–92.9]	94.0 [93.7–94.2]
	Tok.	**564/125** [564–564;125–126]	5,367/150 [5,367–5,367;147–153]	6,552/208 [6,541–6,562;200–217]	5,652/166 [5,652–5,652;161–170]	3,783/155 [3,783–3,783;147–163]	3,775/168 [3,775–3,775;160–175]	3,774/195 [3,774–3,774;184–205]	5,564/161 [5,535–5,572;158–165]	5,641/160 [5,633–5,649;155–165]	6,164/158 [6,149–6,178;153–163]	6,218/162 [6,202–6,235;158–166]
Qwen3.6-27B	Acc.	34.0 [31.8–36.2]	61.3 [59.7–62.8]	52.5 [50.0–55.1]	**65.1** [63.1–67.2]	33.5 [31.6–35.4]	33.2 [31.4–34.9]	47.1 [45.8–48.4]	61.7 [60.2–63.2]	61.5 [60.3–62.8]	64.9 [63.3–66.5]	64.2 [63.3–65.2]
	Hit	0.0 [n/a]	91.7 [91.7–91.7]	89.8 [89.8–89.8]	94.6 [94.6–94.6]	33.7 [33.7–33.7]	33.7 [33.7–33.7]	89.3 [89.3–89.3]	**97.2** [96.7–97.7]	**97.2** [96.8–97.7]	93.6 [93.0–94.1]	94.0 [93.3–94.6]
	Tok.	**554/190** [554–554;184–196]	5,554/285 [5,554–5,554;268–303]	9,949/299 [9,949–9,949;277–322]	6,661/269 [6,661–6,661;235–304]	8,833/224 [8,833–8,833;210–238]	8,895/247 [8,895–8,895;233–262]	8,894/298 [8,894–8,894;270–326]	6,177/263 [6,164–6,190;246–279]	6,258/265 [6,248–6,268;242–287]	6,539/263 [6,522–6,556;244–282]	6,620/298 [6,607–6,634;285–311]
Qwen3.6-35B-A3B	Acc.	33.5 [32.3–34.7]	57.6 [55.6–59.5]	42.3 [40.7–43.8]	55.8 [54.4–57.2]	30.4 [29.4–31.4]	31.1 [29.7–32.4]	37.6 [35.9–39.4]	57.6 [56.6–58.6]	58.0 [56.2–59.8]	**58.2** [56.0–60.4]	58.1 [56.2–60.0]
	Hit	0.0 [n/a]	92.2 [92.2–92.2]	87.8 [87.8–87.8]	93.7 [93.7–93.7]	33.7 [33.7–33.7]	24.4 [24.4–24.4]	89.3 [89.3–89.3]	95.9 [95.4–96.5]	**96.3** [95.9–96.7]	92.5 [91.7–93.3]	93.7 [93.2–94.1]
	Tok.	**554/141** [554–554;138–143]	5,560/258 [5,560–5,560;234–281]	10,798/192 [10,798–10,798;183–201]	6,866/170 [6,866–6,866;159–181]	8,833/149 [8,833–8,833;143–154]	8,895/169 [8,895–8,895;157–182]	8,894/170 [8,894–8,894;158–183]	6,251/213 [6,241–6,262;195–232]	6,306/228 [6,292–6,321;205–251]	6,728/188 [6,710–6,746;166–210]	6,776/177 [6,764–6,788;171–182]
MiniMax-M3	Acc.	35.3 [33.9–36.7]	59.6 [58.2–60.9]	54.5 [52.9–56.2]	64.5 [63.2–65.9]	33.7 [31.3–36.1]	34.0 [32.2–35.9]	40.4 [38.9–41.8]	64.0 [61.9–66.1]	63.7 [62.2–65.2]	**65.0** [62.8–67.3]	64.4 [62.6–66.2]
	Hit	0.0 [n/a]	91.7 [91.7–91.7]	91.2 [91.2–91.2]	94.6 [94.6–94.6]	33.7 [33.7–33.7]	33.7 [33.7–33.7]	89.3 [89.3–89.3]	**96.9** [96.5–97.3]	95.7 [95.1–96.4]	94.3 [93.7–94.8]	93.2 [92.7–93.7]
	Tok.	**670/113** [670–670;112–113]	5,085/132 [5,085–5,085;131–133]	10,765/160 [10,765–10,765;158–163]	7,111/145 [7,111–7,111;143–146]	6,614/131 [6,614–6,614;130–132]	6,664/137 [6,664–6,664;135–139]	6,684/145 [6,684–6,684;143–146]	5,825/135 [5,816–5,835;134–137]	5,821/136 [5,813–5,830;135–137]	6,513/137 [6,491–6,535;134–139]	6,593/137 [6,584–6,602;136–138]

Note: Acc. = exact-match accuracy; Hit = document hit rate (%); Tok. = mean input/output tokens. Smaller text gives 95% intervals. Zero-shot generation retrieves no context, so its document hit rate is fixed at 0% and an interval is not applicable (n/a). Graph+MM = direct graph–multimodal candidate concatenation. Bold marks the best point estimate within each metric row.

4.2.1 Late interaction is useful but insufficient

ColQwen beats ColPali by 6.4–13.9 percentage points across the five generators and achieves an 89.3% document hit rate, compared with 24.4–33.7% for ColPali. Nevertheless, it remains 18.0–24.1 percentage points behind multimodal retrieval. The result separates retrieval from evidence sufficiency: page-level late interaction can locate the linked document while failing to isolate the decisive cell, caption, dose, or endpoint, or while presenting too much page content for reliable generation. Thus ColQwen is a useful retrieval component, not a sufficient end-to-end so-

lution.

4.2.2 Difficulty-stratified patterns

Figure 3 summarizes the ten-run difficulty-stratified distributions for the best-performing Gemma-4-31B. Graph-only retrieval reaches 92.4–93.4% mean accuracy on easy questions but falls to 31.8–35.7% on hard questions. Multimodal retrieval is more balanced, reaching 87.1%, 68.3%, and 44.6% on easy, medium, and hard questions, respectively. Graph-MM 2-Hop matches the hard-question accuracy of multimodal retrieval (44.6%) but does not improve overall accuracy; it also increases mean total tokens from 5,708 to 6,262 and reduces document hit rate from 94.6% to 92.7%. Thus, the figure shows that high document hit rate does not necessarily correspond to answer-sufficient evidence.

4.2.3 Direct fusion rarely improves on its components

Across the repeated panel, direct graph–multimodal fusion ranges from 58.1% to 66.1% accuracy. It exceeds the stronger same-depth component only for Qwen3.6-35B-A3B at both graph depths and MiniMax-M3 at one hop, with gains of 0.1–0.6 percentage points. Direct candidate concatenation therefore does not reliably exploit the two evidence sources and can allow redundant or indirect graph context to displace more precise visual or textual evidence.

4.2.4 Complementarity and unrealized fusion potential

Graph and multimodal retrieval can remain useful even when direct fusion does not improve average accuracy, because the two routes answer different questions correctly. Table 3 shows that both routes answer 43.8–53.2% of questions correctly, while multimodal-only answers account for a further 11.5–16.2% and graph-only answers for 7.9–13.9%. Consequently, ideal selection reaches 69.5–77.7%, compared with 58.1–66.1% for direct concatenation. The largest gap occurs for Qwen3.6-27B with two-hop graph retrieval: 13.5 percentage points, corresponding to approximately 27 additional correct answers per run. The smallest gap occurs for Gemma-4-31B at two hops, but still amounts to 8.4 points, or approximately 17 additional answers. Because ideal selection retrospectively chooses the route that answered each question correctly, these gains are not directly achievable without a router that can predict the better evidence source before generation. They nevertheless show that all evaluated models contain substantial unrealized complementarity.

4.3 Stage 3: robustness probes and contamination considerations

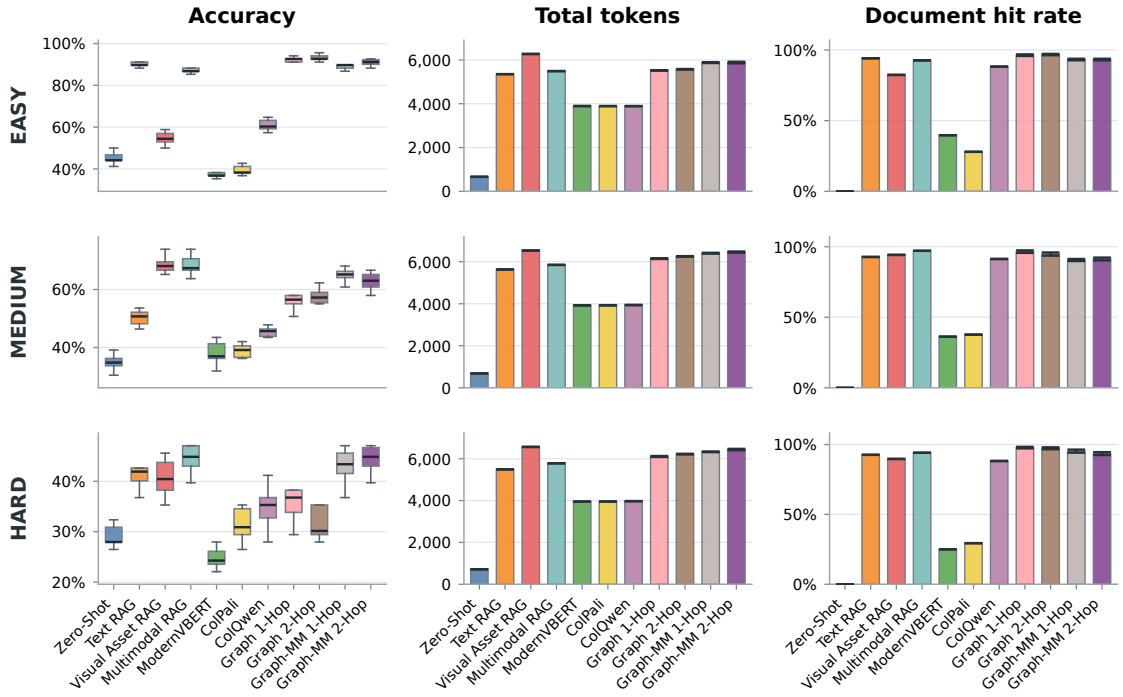
Answer shuffling and context noise change accuracy by at most 1.6 points (Table 4). Paraphrasing is stable for MiniMax-M3 and both Gemma models, but Qwen3.6-27B text retrieval falls from 58.0% to 48.9% ($Z = 2.02$, $p = 0.044$) and Qwen3.6-35B from 55.9% to 42.8% ($Z = 3.13$, $p = 0.002$). Stable zero-shot paraphrasing for both Qwen models localizes this fragility to the tested text-retrieval pipeline rather than the family in every setting.

These perturbations do not diagnose contamination: a drop can reflect prompting, retrieval, or scoring rather than memorization. The Qwen contrast cannot distinguish generator sensitivity from changed retrieval matches or evidence composition. Its immediate value is robustness auditing. Future checkpoint regressions could trigger a separate contamination investigation using meaning-preserving variants plus completion- or overlap-based methods (Yang et al., 2023).

Generated distractors increase accuracy by 12.8–28.7 points ($p < 0.001$ throughout), indicating easier items rather than better pharmacological reasoning. Expert alternatives often encode adjacent doses, endpoints, populations, or domain confusions; generic generation can yield fluent but semantically distant, heterogeneous, or easily eliminated options. This agrees with evidence that surface-valid LLM distractors need not capture misconceptions (Feng et al., 2024), whereas choice-aware optimization improves plausibility and discrimination (Lee et al., 2025).

Holding the expert key fixed while replacing only incorrect options may amplify this effect: the key can remain more specific, evidence-aligned, or stylistically distinct. Text retrieval may expose the exact dose or endpoint needed to reject distant alternatives. These hypotheses require blinded plausibility ratings, style checks, evidence-similarity analysis, and credible-misconception annotation. Regenerated items should not be assumed difficulty equivalent without psychometric validation (Balepur et al., 2025).

Removing options reduces exact match to 0.1–2.3% ($p < 0.001$ throughout). Because response



Note: Rows show easy, medium, and hard questions; columns show exact-match accuracy, total tokens per question, and document hit rate. Boxplots summarize ten run-level aggregates. Graph-MM 1-Hop and Graph-MM 2-Hop denote direct graph–multimodal candidate concatenation. Augmentation labels appear along the bottom row only.

Figure 3: Difficulty-stratified performance of Gemma-4-31B across the 11 retained retrieval conditions.

Table 3: Answer-level complementarity and ideal-selection potential averaged over ten runs.

Model	Both routes correct	Multimodal only	Graph only	Direct fusion	Ideal selection	Potential gain
One-hop graph retrieval						
Gemma-4-26B-A4B	48.9% ($n = 100$)	11.9% ($n = 24$)	10.7% ($n = 22$)	58.3% ($n = 120$)	71.5% ($n = 146$)	13.2 pp ($n = 26$)
Gemma-4-31B	53.2% ($n = 109$)	13.5% ($n = 28$)	8.1% ($n = 17$)	65.8% ($n = 135$)	74.7% ($n = 153$)	8.9 pp ($n = 18$)
Qwen3.6-27B	49.4% ($n = 101$)	15.8% ($n = 32$)	12.3% ($n = 25$)	64.9% ($n = 133$)	77.5% ($n = 159$)	12.6 pp ($n = 26$)
Qwen3.6-35B-A3B	43.8% ($n = 90$)	12.0% ($n = 25$)	13.9% ($n = 28$)	58.2% ($n = 119$)	69.7% ($n = 143$)	11.5 pp ($n = 24$)
MiniMax-M3	51.6% ($n = 106$)	13.0% ($n = 27$)	12.4% ($n = 26$)	65.0% ($n = 133$)	77.0% ($n = 158$)	12.0 pp ($n = 25$)
Two-hop graph retrieval						
Gemma-4-26B-A4B	48.4% ($n = 99$)	12.3% ($n = 25$)	11.4% ($n = 23$)	59.1% ($n = 121$)	72.2% ($n = 148$)	13.1 pp ($n = 27$)
Gemma-4-31B	52.7% ($n = 108$)	13.9% ($n = 28$)	7.9% ($n = 16$)	66.1% ($n = 136$)	74.5% ($n = 153$)	8.4 pp ($n = 17$)
Qwen3.6-27B	48.9% ($n = 100$)	16.2% ($n = 33$)	12.6% ($n = 26$)	64.2% ($n = 132$)	77.7% ($n = 159$)	13.5 pp ($n = 27$)
Qwen3.6-35B-A3B	44.3% ($n = 91$)	11.5% ($n = 24$)	13.7% ($n = 28$)	58.1% ($n = 119$)	69.5% ($n = 142$)	11.4 pp ($n = 23$)
MiniMax-M3	51.6% ($n = 106$)	12.9% ($n = 26$)	12.1% ($n = 25$)	64.4% ($n = 132$)	76.6% ($n = 157$)	12.2 pp ($n = 25$)

Note: Percentages are accompanied by the rounded mean number of correct answers per run (n; 205 questions). "Both routes correct" denotes questions answered correctly by both multimodal and graph retrieval; the route-only columns identify complementary correct answers. Direct fusion is the implemented candidate-concatenation baseline. Ideal selection is an oracle upper bound that counts a question as correct when either route succeeds. Potential gain is ideal selection minus direct fusion, reported in percentage points (pp) and additional correct answers per run.

format and scoring both change, this demonstrates dependence on option anchors but is not directly comparable to open-ended QA performance.

5 Discussion

The results do not support a uniform advantage for graph retrieval. Retrieval architecture interacts strongly with the generator: graph-only retrieval outperforms multimodal retrieval for Qwen3.6-

Table 4: Robustness analysis for selected models across ten runs.

Model	Baseline	Answer shuffle	Context noise	Paraphrase	Generated distractors	Open ended
Zero-shot generation						
	34.0 [31.9–36.0]	34.8 [32.7–36.9]	35.2 [33.2–37.3]	36.7 [34.6–38.7]	62.7 [60.6–64.8]	1.2 [0.7–1.6]
Gemma-4-31B		($Z=-0.18$, $p=0.85$)	($Z=-0.31$, $p=0.75$)	($Z=-0.71$, $p=0.48$)	($Z=-7.95$, $p<0.001$)	($Z=22.82$, $p<0.001$)
	32.4 [30.4–34.4]	33.5 [31.5–35.6]	32.1 [30.1–34.2]	33.1 [31.0–35.1]	56.6 [54.5–58.8]	0.7 [0.3–1.0]
Gemma-4-26B-A4B		($Z=0.32$, $p=0.74$)	($Z=0.74$, $p=0.45$)	($Z=0.47$, $p=0.64$)	($Z=-6.03$, $p<0.001$)	($Z=24.69$, $p<0.001$)
	33.6 [29.9–37.3]	32.1 [28.5–35.8]	34.1 [30.3–37.8]	33.9 [30.2–37.7]	51.5 [47.6–55.5]	1.3 [0.4–2.2]
Qwen3.6-27B		($Z=0.82$, $p=0.41$)	($Z=0.30$, $p=0.76$)	($Z=0.34$, $p=0.73$)	($Z=-4.08$, $p<0.001$)	($Z=14.43$, $p<0.001$)
	31.2 [29.2–33.2]	32.1 [30.1–34.1]	31.4 [29.4–33.4]	30.8 [28.8–32.8]	44.2 [42.1–46.4]	1.0 [0.6–1.5]
Qwen3.6-35B-A3B		($Z=-0.96$, $p=0.34$)	($Z=-0.76$, $p=0.45$)	($Z=-0.58$, $p=0.56$)	($Z=-4.25$, $p<0.001$)	($Z=20.81$, $p<0.001$)
	36.2 [34.1–38.3]	36.1 [34.0–38.2]	34.6 [32.6–36.7]	33.6 [31.6–35.7]	53.1 [50.9–55.2]	2.3 [1.7–3.0]
MiniMax-M3		($Z=0.29$, $p=0.77$)	($Z=0.71$, $p=0.47$)	($Z=1.00$, $p=0.31$)	($Z=-4.37$, $p<0.001$)	($Z=21.02$, $p<0.001$)
Text retrieval						
	58.0 [55.9–60.1]	58.7 [56.6–60.8]	56.8 [54.6–58.9]	56.6 [54.5–58.8]	75.9 [74.0–77.7]	0.3 [0.1–0.5]
Gemma-4-31B		($Z=-0.19$, $p=0.85$)	($Z=0.35$, $p=0.72$)	($Z=0.39$, $p=0.69$)	($Z=-5.58$, $p<0.001$)	($Z=34.78$, $p<0.001$)
	54.0 [51.9–56.2]	55.3 [53.1–57.4]	54.3 [52.1–56.4]	51.7 [49.5–53.9]	70.6 [68.7–72.6]	0.1 [0.0–0.2]
Gemma-4-26B-A4B		($Z=-0.17$, $p=0.86$)	($Z=0.09$, $p=0.92$)	($Z=0.80$, $p=0.42$)	($Z=-4.75$, $p<0.001$)	($Z=34.33$, $p<0.001$)
	58.0 [54.1–61.9]	59.3 [55.4–63.2]	57.5 [53.6–61.4]	48.9 [45.0–52.9]	71.7 [68.2–75.3]	0.3 [0.0–0.7]
Qwen3.6-27B		($Z=-0.57$, $p=0.56$)	($Z=-0.12$, $p=0.90$)	($Z=2.02$, $p=0.044$)	($Z=-3.94$, $p<0.001$)	($Z=20.23$, $p<0.001$)
	55.9 [53.7–58.0]	56.3 [54.1–58.4]	57.3 [55.2–59.4]	42.8 [40.7–44.9]	69.0 [67.0–71.0]	0.2 [0.0–0.4]
Qwen3.6-35B-A3B		($Z=-0.59$, $p=0.56$)	($Z=-0.87$, $p=0.38$)	($Z=3.13$, $p=0.002$)	($Z=-4.34$, $p<0.001$)	($Z=33.82$, $p<0.001$)
	59.9 [57.8–62.0]	58.7 [56.5–60.8]	58.6 [56.4–60.7]	58.1 [55.9–60.2]	72.7 [70.8–74.7]	0.1 [0.0–0.2]
MiniMax-M3		($Z=0.09$, $p=0.92$)	($Z=0.12$, $p=0.90$)	($Z=0.26$, $p=0.79$)	($Z=-4.16$, $p<0.001$)	($Z=35.76$, $p<0.001$)

Note: Cells report exact-match accuracy (%) and 95% Agresti–Coull intervals; second lines give two-proportion tests against the corresponding baseline. Baselines have no test. Inference is exploratory because it ignores question pairing and repeated-run clustering.

35B-A3B, whereas multimodal retrieval leads for several other models, and direct fusion rarely improves on the stronger component. Nevertheless, Table 3 shows substantial answer-level complementarity. This finding is consistent with both the promise of joint visual–text retrieval (Chen et al., 2022) and evidence that multimodal retrieval remains vulnerable to tables and rephrasing (Wasserman et al., 2025).

Graph retrieval identifies related sources and entities, but many PK/PD answers require an exact text window, table cell, figure label, dose, or endpoint. Direct concatenation can therefore displace decisive evidence with relevant but insufficient context. Figure 3 likewise distinguishes graph relevance, document hit rate, and answer sufficiency.

Unlike our earlier conversion-focused study (Kocbek et al., 2025), this work evaluates typed MinerU evidence, LightRAG depth, graph–multimodal fusion, and local deployment. The results show that graph augmentation is not consistently effective.

5.1 Future multimodal–graph fusion

In future work, we plan to examine paired item-level tests, inexpensive routing or reciprocal rank fusion (Cormack et al., 2009), semantic or expert scoring of open-ended answers, evidence-level recall, complementarity stratified by difficulty and evidence modality, and question-only retrieval with options supplied only to the answering model.

Sparse mixture-of-experts models also merit evaluation on compact systems. DGX Spark provides 273 GB/s memory bandwidth versus 4.8 TB/s for one H200, a 17.6-fold difference (NVIDIA Corporation, 2026a,b). On our H200, dense Gemma-4-31B generated below 10 output tokens/s, while Gemma-4-26B-A4B generated about 50 tokens/s; their best repeated accuracies were 66.6% and 60.8%, respectively. Because model family, active parameters, quantization, and retrieval differ, this is not a controlled hardware comparison. Future work should report accuracy per watt, questions per joule, latency, throughput, and cost under matched settings.

Perturbation trajectories can reveal regressions, but memorization claims require overlap or completion evidence and fresh held-out questions (Yang et al., 2023).

6 Limitations

The single-run screen supports model selection rather than definitive ranking. Ten-run intervals quantify run variability; paired question-cluster intervals remain conditional on the observed runs and do not model source-paper dependence. Robustness tests pool repeated binary responses and may overstate precision. Ideal selection is an oracle bound, not a deployable policy.

The 205-question benchmark is restricted to antimicrobial PK/PD and primarily uses MCQs. Options in retrieval may enable cues or elimination, while option-free exact match may undercount partially correct answers. Generalization requires larger independent benchmarks and evidence-grounded open-ended evaluation. Exact checkpoint revisions should be archived.

Ethical and deployment considerations. The system supports literature inspection, not clinical decision-making. Local deployment may limit document disclosure but does not establish privacy, security, or clinical validity. Users should inspect cited evidence and distinguish retrieved material from generated synthesis; the benchmark contains no patient records.

7 Conclusion

Across 13 local generators, graph retrieval benefits were model- and question-dependent. Multimodal retrieval achieved the highest repeated accuracy; direct fusion rarely improved on the stronger component despite answer-level complementarity. ColQwen outperformed ColPali but remained below multimodal retrieval. These results support learned, source-aware routing or reranking.

Acknowledgments

This research was supported by the EU HE RIA, grant #101159018; the EC/EuroHPC JU and the Slovenian Ministry of ESY through SLAIF, grant #101254461; and the Slovenian Research and Innovation Agency (ARIS), grants #N3-0307 and #GC-0001.

References

Alan Agresti and Brent A. Coull. 1998. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126.

Nishant Balepur, Rachel Rudinger, and Jordan Boyd-Graber. 2025. Which of these best describes multiple choice evaluation with LLMs? a) forced b) flawed c) fixable d) all of the above. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 3394–3418.

Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William Cohen. 2022. MuRAG: Multimodal retrieval-augmented generator for open question answering over images and text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5558–5570.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of SIGIR*, pages 758–759.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, et al. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Celine Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *Proceedings of the 13th International Conference on Learning Representations*.

Wanyong Feng, Jaewook Lee, Hunter McNichols, Alexander Scarlatos, Digory Smith, Simon Woodhead, Nancy Ornelas, and Andrew Lan. 2024. Exploring automated distractor generation for math multiple-choice questions via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3067–3082.

Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, et al. 2022. Manu: A cloud native vector database management system. *Proceedings of the VLDB Endowment*, 15(12):3548–3561.

Zirui Guo, Xubin Ren, Lingrui Xu, Jiahao Zhang, and Chao Huang. 2025a. Rag-anything: All-in-one rag framework. *arXiv preprint arXiv:2510.12323*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025b. Lightrag: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Primoz Kocbek, Azra Frkatovic-Hodzic, Dora Lalic, Vivian Hui, Gordan Lauc, and Gregor Stiglic. 2025. Exploration of augmentation strategies in multi-modal retrieval-augmented generation for the biomedical domain: A case study evaluating question answering in glycobiology. In *Proceedings of the 2025 IEEE International Conference on Big Data (BigData)*, pages 5156–5165. IEEE.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles*, pages 611–626.

Yooseop Lee, Suin Kim, and Yohan Jo. 2025. Generating plausible distractors for multiple-choice questions via student choice prediction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 23669–23692.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2025. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19568–19580.

Yu A. Malkov and Dmitry A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

NVIDIA Corporation. 2026a. NVIDIA DGX Spark: Hardware overview. `https://docs.nvidia.com/dgx/dgx-spark/hardware.html`. Accessed 2026-08-03.

NVIDIA Corporation. 2026b. NVIDIA H200 Tensor Core GPU. `https://www.nvidia.com/en-us/data-center/h200/`. Accessed 2026-08-03.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, et al. 2024. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, et al. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627. Association for Computing Machinery.

Navve Wasserman, Roi Pony, Oshri Naparstek, Adi Raz Goldfarb, Eli Schwartz, Udi Barzelay, and Leonid Karlinsky. 2025. REAL-MM-RAG: A real-world multi-modal retrieval benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 31660–31683.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Q-STR: Diagnosing Query-Driven Text-to-Table Reasoning in Cricket Narratives

Ritam Upadhyay[1 *] **Naman Ahuja**[1 *] **Rishabh Baral**[1]
Aparna Garimella[2] **Vivek Gupta**[†1]

[1]School of Computing and Augmented Intelligence, Arizona State University
[2]Adobe Research, India



Abstract

Many analytical questions over long narratives are structured even when their source documents are not. Answering them requires identifying relevant events, interpreting a query-specific output structure, grounding answer records in dispersed evidence, and computing the values of an answer table. We introduce Q-STR, a cricket-domain benchmark and diagnostic for this query-driven text-to-table problem. We construct Q-STR from ball-by-ball match narratives using executable SQL templates and release two complementary splits: a 4,256-example coverage split for breadth and a 3,467-example controlled split spanning 348 reasoning templates for within-template diagnosis. Factorized metrics distinguish recovery of the answer-table structure from fidelity of its cell values. Across four language models and seven prompting strategies, models often recover plausible table structure while failing at grounded aggregation. Gemini-2.5-Flash is strongest overall, but even its best configuration reaches only 28.56 exact table match. For open-weight models, allocating output tokens to row-local evidence and computation improves cell fidelity more reliably than multi-call planning or self-consistency. Controlled comparisons further show that event identification and numeric fidelity are only partially coupled: execution pressure is the most consistent difficulty driver, while increasing output pressure shifts errors toward undercounting. Q-STR is deliberately a cricket diagnostic rather than a general benchmark of all narrative reasoning. Its single-domain scope, model-assisted construction, and absence of an executable code-generation baseline limit broader conclusions, but its controlled design exposes where structured reasoning from long narratives breaks down.

1 Introduction

Many real-world questions require more than retrieving a short span from a long document. Live commentary, event logs, case notes, and operational reports distribute evidence across local updates while the underlying state changes over time (Kočiský et al., 2018; Dua et al., 2019; Pang et al., 2022; Bertsch et al., 2025). A useful answer may require identifying a set of events, resolving references to evolving entities, aggregating information across distant passages, and presenting the result in a structure that supports inspection. The necessary facts may all be present in the text, yet the organization required by the question may be absent.

We study this setting as *query-driven text-to-table generation*. Given a natural-language query and a long narrative document, a model must produce an answer table directly from textual evidence. The goal is not to recover one canonical table for the document. Instead, each question defines a task-specific analytical view of the narrative. In Q-STR, the requested output columns and primary-key metadata are made explicit to the model, so the structural challenge is not unconstrained schema discovery. The model must interpret this query-conditioned output specification, determine which entities or event groups instantiate its rows, identify the relevant evidence, and compute the requested values. A question asking for wickets by bowler and batter requires a different organization from one asking for boundary-run percentages or strike rates over a temporal interval, even when both questions refer to the same match.

This formulation couples three abilities. First, the model must interpret the supplied query-conditioned output specification and determine which entities or event groups instantiate its rows. Second, it must ground those rows and their values in narrative evidence. Third, it must execute the required filters, groupings, comparisons, and

[*]These authors contributed equally
[†]Primary supervisor for this work

The Proceedings of Workshop on Document Intelligence and Understanding, pages 196–216
October 24, 2026 ©2026 Association for Computational Linguistics

arithmetic. A correct-looking table can fail at any of these stages: it may contain the wrong entities, the right entities with unsupported values, or correctly extracted values computed over the wrong time window. Final-answer exact match alone does not reveal which capability failed.

We introduce Q-STR, a benchmark for query-driven table generation from cricket commentary. Cricket is a productive diagnostic domain because its narratives are long, sequential, and stateful. Relevant evidence is distributed across many short updates; player mentions depend on the current innings and matchup; and analytical questions naturally involve filtering, grouping, comparison, temporal restriction, and numeric aggregation. At the same time, the underlying ball-by-ball state is structured enough to support executable gold answers and controlled evaluation. This combination lets us study the "missing table" problem in a setting where the latent records are auditable.

Q-STR contains two complementary splits. The *coverage split* measures breadth across query structures and match contexts. The *controlled split* fixes a reasoning template while varying grounded inputs such as player identities, matches, and temporal conditions. The latter supports paired and pressure-based analyses that are difficult to conduct on a random benchmark split. We evaluate table-level correctness, primary-key recovery, aligned cell accuracy, numeric error, and directional counting bias. Together, these metrics separate event identification from value computation.

Our experiments compare four language models and seven prompting strategies. The coverage split produces a clear model ranking, but also a consistent gap between recovering rows and filling them correctly. Decomposition and self-consistency can stabilize table structure without fixing correlated arithmetic errors. Row-local reasoning performs better at cell-level computation for the open-weight models despite using fewer total tokens than several multi-call methods. On the controlled split, Llama-3.3-70B-Instruct and Qwen2.5-72B-Instruct have nearly identical PK F1 but different precision–recall profiles. Their numeric behavior also differs: both shift from overcounting to undercounting as output tables become larger, with a sharper shift for Qwen.

These findings support a focused claim. Q-STR does not establish that performance transfers to finance, medicine, law, or other narrative domains. Rather, it provides a controlled cricket testbed in which schema interpretation, evidence recovery, and numerical execution can be measured separately. The benchmark is useful precisely because its domain permits dense, reproducible diagnosis; broader generality remains an empirical question.

Our contributions are:

- We formulate query-driven text-to-table generation as a joint problem of query-conditioned structure interpretation, long-context evidence grounding, and computation.
- We introduce Q-STR, with a 4,256-example coverage split and a 3,467-example controlled split comprising 348 templates instantiated across five matches.
- We provide factorized table, primary-key, cell, and numeric-bias metrics that distinguish event identification from value fidelity.
- We benchmark four models under seven prompting strategies and show that row-local grounded reasoning improves value fidelity more consistently than planning or repeated generation.
- We use controlled factors and matched pressure analyses to identify execution burden as the most stable difficulty driver and to expose systematic undercounting as answer tables scale.

2 Related Work

Text-to-table generation. The closest line of work converts unstructured text into structured tables, including direct text-to-table generation, order-insensitive generation, commentary-to-table mapping, global tuple extraction, and latent-schema recovery (Wu et al., 2022; Li et al., 2023b; Deng et al., 2024; Ahuja et al., 2025). These tasks support downstream information extraction, summarization, and analysis. Many define a target schema at the dataset or instance level, or ask for a canonical summary of the input. Q-STR instead makes the analytical question part of the structural specification: the same narrative can support many different answer tables. The requested output columns and primary-key metadata are supplied to the model, but it must still determine which narrative entities instantiate the requested rows and compute the query-specific values.

Text-to-SQL and semantic parsing. Text-to-SQL maps natural-language questions to executable programs over structured databases (Pasupat and Liang, 2015; Yu et al., 2018; Wang et al.,

2020; Scholak et al., 2021; Li et al., 2023a; Lei et al., 2025). This literature provides useful abstractions for filtering, grouping, aggregation, comparison, and compositional reasoning under schema constraints. Q-STR reverses an important assumption. At test time there is no database interface or supplied table over which to execute the query. The system must recover the relevant records from narrative evidence before it can perform database-like reasoning. Gold SQL is used to construct and validate the benchmark, but is not exposed to the evaluated models.

Long-context and numerical question answering. NarrativeQA and QuALITY study question answering over long documents, while DROP targets discrete reasoning over paragraphs (Kočiský et al., 2018; Pang et al., 2022; Dua et al., 2019). Oolong further demonstrates that aggregation across many segments remains difficult even for strong long-context models (Bertsch et al., 2025). Financial benchmarks such as FinQA combine report text, tables, and numerical reasoning (Chen et al., 2021). These settings establish the difficulty of evidence localization, reference resolution, and multi-step computation. Q-STR adds query-dependent structured output: models must recover not only a scalar answer, but also the rows and cells that constitute it.

Controlled and model-assisted benchmark construction. SCAN and CFQ motivate controlled splits for testing systematic generalization under structural variation (Lake and Baroni, 2018; Keysers et al., 2020). StrategyQA and DROP show the value of datasets organized around particular reasoning demands (Geva et al., 2021; Dua et al., 2019). Self-Instruct demonstrates the scalability of model-assisted instruction generation, while StructText uses execution-grounded synthetic supervision for structured generation (Wang et al., 2023; Kashyap et al., 2025). Q-STR combines human-authored executable seeds, model-assisted intent expansion, execution filtering, consistency checks, and human validation. This pipeline scales analytical coverage, but it also makes construction provenance an important part of interpreting the benchmark.

3 Task Formulation

Let Q be a natural-language query and D a narrative document. The task is to generate an answer table $\hat{T}$ that matches a gold table T^*. A table contains an ordered column set $C(T)$ and a set or sequence of rows $R(T)$. When defined by the query, one or more columns form a primary key that identifies the answer records.

Unlike conventional table question answering, Q-STR does not provide T^*, its rows, or a database containing the relevant events. The model receives the question, the requested output columns, primary-key metadata when available, and the complete commentary. Thus, Q-STR does not require unconstrained discovery of column names or key structure. We use *schema interpretation* to refer to understanding how the supplied query-conditioned columns and keys organize the answer and which narrative entities instantiate its rows. The output table must recover both the row structure and the values required by the query. We view this as three coupled decisions:

1. **Schema interpretation:** interpret the supplied target columns and key metadata and determine how they organize the requested answer.
2. **Event identification:** locate the entities or event groups in the narrative that instantiate the answer rows.
3. **Value computation:** apply the query's filters, aggregations, comparisons, temporal restrictions, or arithmetic to populate non-key cells.

The distinction between the latter two stages is central. Consider a question asking for each batter's percentage of runs against a bowler that came from boundaries. A model may correctly identify every qualifying batter but miscount total runs, or it may compute a percentage correctly for only a subset of the required batters. Exact table match treats both predictions as failures. Primary-key metrics reveal whether the answer entities were recovered, while aligned cell metrics reveal whether the corresponding values were computed correctly.

4 Benchmark Construction

4.1 Source Narratives and Intermediate Tables

We construct Q-STR from public ball-by-ball cricket commentary covering 620 matches. Cricket commentary is a sequence of local event descriptions, often interleaved with match-state summaries and prose analysis. Following Upadhyay et al. (2026), we first convert each match narrative into a

structured ball-by-ball table. This intermediate representation records fields such as innings, over, batter, bowler, runs, extras, and wickets. It is used only to generate and validate benchmark instances; evaluated systems receive the original narrative rather than this table.

Figure 1 summarizes the pipeline. Executable SQL templates define analytical operations over the intermediate tables. Instantiating a template on a match produces both a natural-language question and a gold answer table. Execution makes complex multi-row answers reproducible and enables automatic removal of invalid or uninformative candidates.

4.2 Agentic Query Synthesis

The query bank begins with two sources. First, we author approximately 120 dataset-specific seed queries, each paired with an executable SQL program and a natural-language question. These seeds establish grounded examples of the analytical behavior the benchmark should contain. Second, we collect approximately 1,000 intent-level ideas with GPT-5.2 to broaden the space of goals, including player comparisons, temporal windows, matchup analysis, conditional aggregation, and streak computation.

Given the table schema, a seed, and a sampled intent, Gemini-3-Flash proposes a new analytical intent, a query structure, and an executable SQL template with placeholders for entities and conditions. Candidate templates are canonicalized and deduplicated, instantiated with grounded variables, and executed over the intermediate match tables. Programs that fail, return empty results, or produce uninformative outputs are removed. The pipeline then generates a corresponding natural-language question and meaning-preserving paraphrases. Each retained instance therefore links a narrative, an executable program, a requested output specification, and a gold answer table.

Model assistance is substantial rather than incidental. Roughly half of the released questions pass through model-assisted rewriting or repair. This increases surface variation and makes the construction pipeline scalable, but can introduce stylistic regularities associated with the generator models. We retain provenance flags for human-authored, model-generated, rewritten, and repaired items so that performance can be stratified by construction path.

4.3 Validation

Syntactically valid SQL does not guarantee that its natural-language question expresses the same operation. We therefore run a text-to-SQL consistency check for every question–program pair. A model regenerates SQL from the natural-language question, and the original and regenerated programs are executed on the same match table. If their outputs agree, the item proceeds. When they disagree, an LLM adjudicator determines whether the original program, regenerated program, or neither correctly represents the question. Corrected pairs must pass the same execution check before retention.

Among disputed cases, the adjudicator selected the original SQL in 511 instances, selected regenerated SQL in 590, and required a corrected pair in 944. The authors additionally reviewed a representative sample of 400 questions and found 97.2% semantic coherence between the natural-language query and gold SQL. This audit measures program–question agreement, not the naturalness of the question distribution; the latter remains a limitation of assisted construction.

4.4 Benchmark Splits

The **coverage split** contains 4,256 question–answer pairs and is designed for breadth. It varies both query structure and match context, making it suitable for overall model and prompting comparisons.

The **controlled split** contains 3,467 question–answer pairs generated from 348 unique templates. Each template has roughly ten instantiations distributed across five matches. The reasoning operation remains fixed while grounded variables, such as player names or temporal boundaries, vary. This supports within-template comparisons and reduces the extent to which diagnostic effects can be attributed to unrelated differences in query form.

The coverage and controlled sets are complementary evaluation sets rather than train/test partitions. The coverage split is designed to maximize breadth across query structures and match contexts, whereas the controlled split intentionally repeats reasoning templates under different grounded instantiations to support matched diagnosis. All examples originate from the same cricket-commentary source distribution and may share domain-specific discourse regularities.

The use of a small fixed match set in the controlled split is intentional: repeatedly instantiating the same reasoning template under a limited set

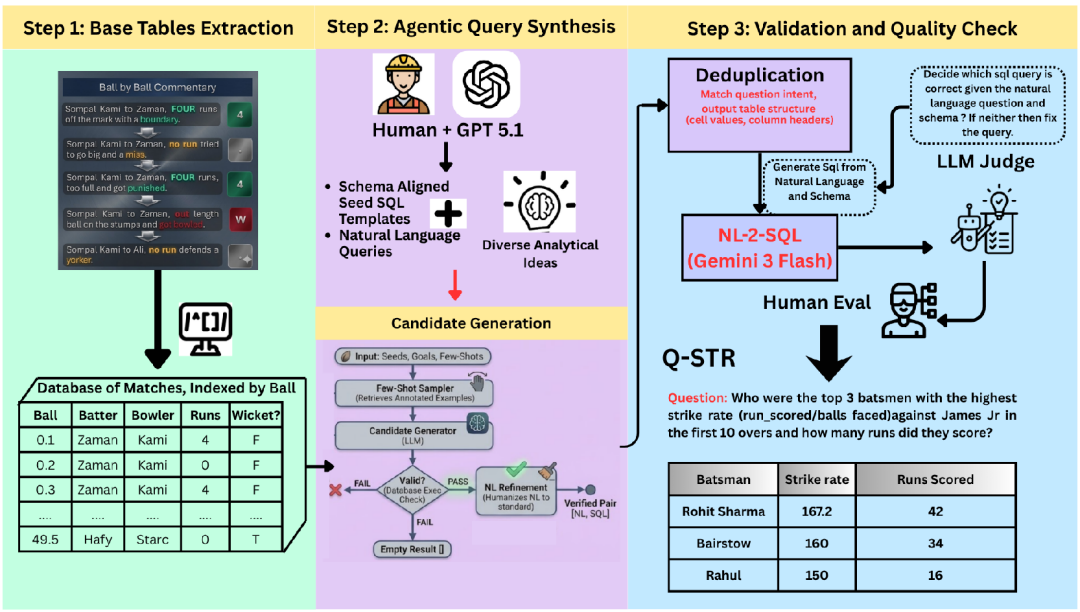


Figure 1: The Q-STR construction pipeline. Ball-by-ball events from 620 games are converted to one intermediate table per match. Human-authored seeds and model-generated analytical intents produce candidate SQL and natural-language pairs, which are execution-filtered, checked for cross-modal consistency, and reviewed before entering the benchmark.

of match contexts reduces variation in query structure and supports matched within-template comparisons. This improves diagnostic control but reduces match-level diversity, so the controlled split should not be interpreted as an estimate of performance across the full distribution of cricket matches.

Controlled examples are annotated along five skill families: entity and schema grounding, context scoping, temporal reasoning, query operations, and compositional depth. The query-operation labels are simple retrieval, filtered aggregation, grouped aggregation, and group comparison. We additionally record answer width and height. Figure 2 shows the distribution of the controlled annotations; Appendix D gives exact definitions and examples. Category-level behavior is examined on the controlled split through factor contrasts over temporal reasoning, entity grounding, operation class, and compositional depth in Section 7 and Appendix E.

4.5 Benchmark Characterization

Table 1 summarizes the split statistics verified from the current manuscript and repository.

Statistic	Coverage	Controlled
Examples	4,256	3,467
Reasoning templates	–	348
Source matches represented	–	5

Table 1: Summary statistics for the Q-STR evaluation splits. Dashes indicate statistics not defined in the same way for that split in the current manuscript.

5 Evaluation

5.1 Factorized Metrics

We report metrics at three levels. **Table-level metrics** measure end-to-end correctness. Exact Table Match requires normalized columns, rows, and values to match the reference. Row Match compares complete row signatures and is insensitive to row ordering after canonicalization.

Event-identification metrics evaluate the primary-key tuples that define answer rows. We report PK Exact, precision, recall, and F1. These metrics reveal whether a model found the correct entities or event groups even when non-key values are wrong.

Value-fidelity metrics align predicted and gold rows by primary key. Cell Accuracy measures exact non-key cell agreement, using a small tolerance for numeric normalization. Weighted RMSE aggregates column-wise numeric error while accounting

Type	Representative question
Filtered aggregation	How many deliveries did Verreynne face while batting against Hasan Ali?
Temporal filtering	List wickets claimed by each bowler against each batter before over 29.5.
Grouped aggregation	For each batter who scored off Faheem, what percentage of their runs against him came from boundaries?
Group comparison	Compare each batter's strike rate against Shaheen Shah Afridi on boundary and non-boundary deliveries.
Streak reasoning	For each batter, what is the longest streak of consecutive dot balls against Faheem?

Table 2: Representative Q-STR question and operation types. Names and temporal values are grounded variables that change across instantiations.

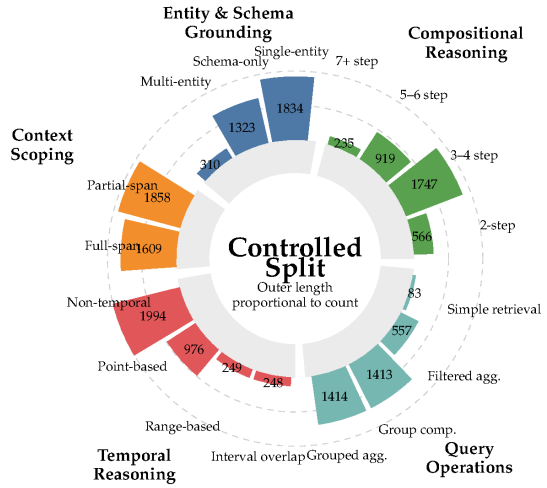


Figure 2: Distribution of controlled-split examples across five skill families. Outer radius is proportional to the count of each subskill, normalized by the largest category.

for the number of aligned cells in each column. On the controlled split we also report Overcount%, Undercount%, and their difference, Net Bias. Exact definitions appear in Appendix A.

Semantic Text-to-Table evaluation metrics are useful when multiple surface realizations may express equivalent content. Q-STR instead uses executable SQL-derived references with explicit row keys and numerically precise values. In this setting, selecting an extra entity, omitting a qualifying row, or producing an incorrect aggregate is a substantive reasoning error rather than a surface-form mismatch. We therefore prioritize exact and factorized metrics that separately evaluate table recovery, primary-key recovery, aligned cell correctness, and numerical error. Canonicalization and numeric tolerance absorb superficial formatting differences. Semantic metrics are complementary, but less directly aligned with Q-STR's goal of diagnosing exact event recovery and numerical execution.

5.2 Prompting Baselines

All baselines receive the same question, requested output columns, primary-key metadata, and full commentary, but allocate test-time computation differently:

- ZERO SHOT directly generates the final answer table without visible intermediate reasoning.
- ZERO-SHOT COT uses one call with a free-form reasoning trace before table generation.
- GUIDED COT imposes three stages within one call: identify entities, compute statistics, and construct the table.
- PLANFILL first predicts row headers and a short plan, then fills values in a second call.
- SC TABLE VOTING samples five complete solutions, aligns rows by primary key, and votes over row structure and cell values.
- ROWCOT produces one evidence-and-computation trace per output row within a single call.
- LEAST-TO-MOST generates a directed acyclic graph of at most five subproblems and executes them in separate calls.

These baselines test direct generation, visible reasoning, structure-first decomposition, repeated sampling, and row-local computation. They do not include model-generated code or SQL over the intermediate event tables. Such a privileged-structure baseline would separate program induction from narrative extraction, but its empirical comparison requires an additional experiment; we return to this issue in Section 9.

5.3 Models and Inference

We evaluate Llama-3.3-70B-Instruct, Qwen2.5-72B-Instruct, GPT-4.1, and Gemini-2.5-Flash. The full seven-baseline suite is run on Llama and Qwen. Because of compute constraints, GPT-4.1 and Gemini are evaluated with ZERO SHOT and GUIDED COT. Open-weight models are served with vLLM

on two H200 nodes with 80GB GPUs each. GPT-4.1 uses the OpenAI Batch API, and Gemini uses the Google AI Studio API.

Most methods use deterministic or near-deterministic decoding. SC TABLE VOTING uses temperature 0.6 to create candidate diversity. Outputs are parsed as strict JSON and checked for exact column order, row-length consistency, and method-specific fields. Appendix C provides prompt contracts, inference budgets, and aggregation details.

6 Coverage-Split Results

Table 3 reports the breadth-oriented benchmark results. Exact table match is deliberately strict and should be interpreted alongside row, key, and cell metrics.

6.1 Model Capability Dominates Overall Ranking

Gemini-2.5-Flash is strongest by a large margin, followed by GPT-4.1, Llama, and Qwen. This ordering is stable across the two baselines shared by all four models. Under ZERO SHOT, Gemini reaches 94.51 PK F1, 71.52 Cell Accuracy, and 7.28 RMSE, compared with GPT-4.1 at 91.30/40.96/14.09, Llama at 88.40/31.99/20.42, and Qwen at 87.66/30.16/21.29. Guided reasoning improves GPT-4.1 substantially and lowers Gemini's RMSE, although Gemini's exact match and Cell Accuracy do not increase.

The absolute gap between row recovery and exact table match is important. Even Gemini's 94.58 PK F1 under GUIDED COT corresponds to only 24.83 exact table match. Models frequently identify the relevant answer entities without recovering every required value. This supports evaluating structure and cells separately rather than treating table generation as a monolithic output-format problem.

6.2 Where Additional Output Tokens Help

For Llama and Qwen, adding reasoning within one call generally improves cell-level fidelity. Moving from direct generation to free-form, guided, or row-local reasoning increases Cell Accuracy and usually lowers RMSE. ROWCOT is the clearest case: it obtains the best Cell Accuracy and RMSE for Llama and the best Cell Accuracy for Qwen. Its per-row traces attach local evidence and arithmetic to each predicted record, reducing the burden of maintaining one global computation across a multi-row answer.

Total token expenditure alone does not explain the gain. PLANFILL, SC TABLE VOTING, and LEAST-TO-MOST use multiple calls and substantially more total tokens, but they do not consistently outperform the strongest single-call baselines on value metrics. Additional computation is most useful when it directly supports evidence extraction and cell calculation rather than verbose planning or repeated whole-table generation.

6.3 Decomposition Helps Structure More Than Values

SC TABLE VOTING and LEAST-TO-MOST sometimes improve Row Match, PK Exact, or PK F1, but these gains do not translate consistently to Cell Accuracy or RMSE. Self-consistency stabilizes which rows appear, yet candidates often share the same computational mistake. On Llama, the voted table has average cell support of 3.83 out of five candidates, and 85.9% of samples have average support above three, but its RMSE remains worse than ZERO-SHOT COT. Agreement is therefore correlated without necessarily being corrective.

PLANFILL is a particularly informative negative result. It often has high PK recall, suggesting that the planner proposes plausible entities, but low precision and Row Match indicate that the proposed structure contains spurious or mismatched records. The second stage then inherits this structure and still must solve the harder cell-computation problem. Separating row planning from value filling does not by itself create a reliable intermediate representation.

7 Controlled-Split Analysis

The coverage split establishes broad performance. The controlled split is designed for diagnosis. Because it concentrates repeated template instantiations within five matches, its contrasts are intended to diagnose failure modes under matched conditions rather than estimate population-wide cricket performance. The restricted match set reduces variation in narrative context but may also couple observed effects to the styles and event distributions of those matches. We analyze Llama and Qwen because the full baseline suite is available for both, and focus on their ZERO SHOT outputs to avoid conflating the underlying factor analysis with baseline-specific scaffolds. Each controlled tem-

Method	Exact Table	Row Match	PK Exact	PK Prec.	PK Rec.	PK F1	Cell Acc.	RMSE ↓
				Llama-3.3-70B-Instruct				
ZERO SHOT	2.69	50.52	48.42	83.33	94.13	88.40	31.99	20.42
ZERO-SHOT COT	5.38	57.03	55.09	87.09	94.34	90.57	37.26	17.67
GUIDED COT	4.34	52.26	50.23	79.68	96.82	87.42	37.16	17.01
PLANFILL	3.65	41.15	38.97	73.43	97.19	83.65	34.33	18.89
SC TABLE VOTING	5.47	58.42	56.53	88.34	93.79	90.98	36.58	18.12
ROWCOT	3.30	52.43	50.32	82.11	95.00	88.09	**38.98**	**16.16**
LEAST-TO-MOST	5.30	52.78	50.95	81.83	91.27	86.29	35.02	17.66
				Qwen2.5-72B-Instruct				
ZERO SHOT	2.26	44.27	41.49	82.84	93.07	87.66	30.16	21.29
ZERO-SHOT COT	3.04	51.74	49.59	85.31	94.21	89.54	33.71	**17.58**
GUIDED COT	2.95	49.91	47.79	80.61	96.80	87.97	35.48	17.60
PLANFILL	3.04	40.10	37.62	81.88	91.53	86.44	33.09	19.47
SC TABLE VOTING	3.21	50.52	48.06	88.36	91.22	89.77	31.47	19.34
ROWCOT	3.39	45.83	43.56	81.64	94.71	87.69	**36.63**	18.03
LEAST-TO-MOST	4.25	55.90	54.01	84.79	95.45	89.80	33.67	18.28
				GPT-4.1				
ZERO SHOT	5.30	64.24	62.74	89.29	93.39	91.30	40.96	14.09
GUIDED COT	8.51	71.18	69.58	90.72	95.29	92.95	45.97	12.37
				Gemini-2.5-Flash				
ZERO SHOT	**28.56**	**80.56**	**79.21**	**93.19**	95.87	94.51	**71.52**	7.28
GUIDED COT	24.83	79.43	78.04	92.69	**96.56**	**94.58**	71.18	**6.28**

Table 3: Performance on the Q-STR coverage split. Exact Table and Row Match measure complete outputs; PK metrics measure answer-row recovery; Cell Accuracy and weighted RMSE measure value fidelity on aligned rows.

plate is instantiated under closely related grounded conditions, allowing paired comparisons across event identification and numeric fidelity.

7.1 Analysis Protocol

We annotate examples with fine-grained factors covering grounding, context scoping, temporal reasoning, evidence sparsity, query-operation class, compositional depth, answer width, and answer height. For summary analysis, these factors are aggregated into four pressure axes:

- **Grounding pressure** captures whether a query requires schema-only, single-entity, or multi-entity grounding.
- **Localization pressure** combines temporal restriction, context span, and evidence sparsity.
- **Execution pressure** combines query-operation complexity with compositional depth.
- **Output pressure** combines the number of answer columns and rows.

Each axis is mapped to low, medium, or high using rules fixed before analysis (Appendix D). These labels summarize observed difficulty conditions; they are not causal variables.

Event metrics are computed over examples with parseable primary keys. Numeric metrics are computed only when predicted and gold rows can be aligned and their values are numerically evaluable. We report paired mean differences with bootstrap confidence intervals and sign-flip permutation tests (Zmigrod et al., 2022; Bestgen, 2022; Clark et al., 2011). Pressure comparisons are stratified by matched conditions where possible. Balanced resampling equalizes support within each contrast and serves as a sensitivity analysis for level imbalance. We call a pressure contrast robust only when its direction is stable across raw, permutation, and balanced analyses.

7.2 Overall Event and Numeric Profiles

Table 4 shows that the models are effectively tied on headline event metrics. PK F1 differs by only -0.1 points and PK Exact by 0.3, with both intervals crossing zero. The clearer difference is an operating trade-off: Qwen has higher recall, while Llama has higher precision. Qwen recovers more gold records but introduces more spurious ones; Llama is more conservative.

Mean weighted RMSE and aligned Cell Accuracy are also close. Directional bias reveals a difference hidden by the aggregate error scores. Llama has the larger positive net bias, while Qwen undercounts more often. Thus, similar mean fidelity can

Stage	Metric	n	Llama	Qwen	$\Delta_{\text{Q-L}}$	95% CI
Event identification	PK F1	2551	83.6	83.5	-0.1	[-0.9, 0.8]
	PK Recall	2551	93.8	95.6	1.8*	[1.1, 2.5]
	PK Precision	2551	82.4	80.6	-1.8*	[-2.8, -0.8]
	PK Exact	2551	55.3	55.6	0.3	[-1.4, 1.9]
Numeric fidelity	Weighted RMSE	3129	13.2	12.5	-0.6	[-1.5, 0.3]
	Aligned Cell Accuracy	2345	42.6	42.0	-0.6	[-1.7, 0.7]
	Overcount%	3129	35.4	34.1	-1.3	[-2.8, 0.1]
	Undercount%	3129	24.9	27.6	2.7*	[1.4, 3.9]
	Net Bias	3129	10.4	6.4	-4.0*	[-6.4, -1.5]

Table 4: Paired controlled-split comparison. Positive $\Delta_{\text{Q-L}}$ favors Qwen for event metrics and Cell Accuracy; negative values favor Qwen for RMSE and absolute positive bias. Asterisks indicate 95% intervals excluding zero.

arise from different error distributions.

7.3 Fine-Grained Factors

Appendix Table 8 gives representative factor contrasts. Temporal filtering primarily stresses event recovery: point-based and range-based conditions reduce Llama's PK F1 by 8.7 and 9.1 points relative to non-temporal controls. Multi-entity grounding has a larger and more consistent effect on numeric fidelity, adding roughly 12 RMSE points for both models. Deeper 5–6 step compositions also increase numeric error, although Qwen's interval crosses zero. These distinct patterns motivate the broader pressure axes.

7.4 Event Identification Under Pressure

Because pooled effects can reflect unequal mixtures of examples, Appendix Table 9 reports the stratified event contrasts whose direction remains stable under permutation testing and balanced resampling. Three patterns hold for both models. Medium grounding is easier than low grounding after matching, high execution pressure is harder than medium, and medium output pressure is harder than low. The first pattern is counterintuitive in isolation and likely reflects residual composition in the schema-only group; it should not be interpreted as a causal benefit of entity grounding.

Execution and output effects are more direct. Moving from medium to high execution pressure lowers PK F1 by 4.5 points for Llama and 6.9 for Qwen. Moving from low to medium output pressure lowers it by 12.1 and 9.4 points. Balanced estimates remain similar or larger, suggesting that these effects are not produced only by support imbalance.

7.5 Numeric Fidelity and Bias

Numeric pressure effects are less uniformly separated because weighted RMSE is conditional on parseable, aligned outputs. Appendix Table 10 shows the strongest stratified contrasts. Execution high versus low is robust for both models: weighted RMSE increases by 29.9 for Llama and 43.7 for Qwen, with balanced estimates of 28.3 and 38.3. This is the most consistent difficulty signal in the controlled analysis. Llama also shows a robust grounding high-versus-medium effect. Qwen's grounding and output effects become visible after balancing but are not supported by the raw permutation tests, so we treat them as sensitivity results.

Directional error adds information beyond RMSE (Appendix Figure 3). At low output pressure, both models overcount. At high pressure, Llama's net bias is -7.1, while Qwen's reaches -24.9. Larger answer structures therefore appear to induce omission-type errors, especially for Qwen. This result is conditional on outputs that remain evaluable, so it should be read as an error-profile diagnostic rather than an unconditional failure rate.

8 Conclusion

We presented Q-STR, a cricket-domain benchmark for generating query-specific tables from long narratives. Across four models, systems recover primary keys more reliably than cell values, and row-local reasoning outperforms multi-call decomposition on value fidelity. Controlled analysis identifies execution pressure and increasing undercount bias as central failure modes. Q-STR therefore diagnoses the missing-table problem, but broader claims require non-cricket domains, code or SQL baselines over privileged structure, and construction-provenance analysis.

9 Limitations

Single-domain scope. Q-STR is built entirely from cricket commentary. Cricket offers long, stateful narratives and executable event structure, but also has specialized vocabulary, repeated discourse patterns, and unusually regular latent records. Models may exploit domain familiarity, and the measured pressures may behave differently in less regular settings. Results should therefore be interpreted as a cricket-domain diagnosis.

No code or SQL generation baseline. The current experiments prompt models to generate answer tables directly from narratives. They do not test model-generated SQL or code over gold event tables, nor code over model-extracted tables. These comparisons are necessary to determine how much difficulty comes from narrative-to-record extraction versus analytical execution after privileged access to structure. A specialized Text-to-Table baseline would also require adaptation because Q-STR conditions generation on an analytical query and evaluates a query-specific answer table rather than extraction of a single canonical table. Gold SQL execution establishes answer correctness by construction, but it is not a model baseline and should not be treated as one.

Model-assisted construction. The query pipeline uses GPT-5.2 and Gemini-3-Flash for intent expansion, paraphrasing, validation, and repair, and roughly half of the examples undergo assisted rewriting or correction. Execution and human audits establish semantic coherence but do not remove distributional artifacts. Generator-specific syntax or lexical preferences may make some questions easier for related models and less representative of naturally occurring user requests. Provenance-stratified reporting and human-only or naturally collected test sets are needed.

Experimental coverage. The full baseline suite is evaluated only on two open-weight models, while closed models use two prompts. Experiments cover prompting rather than fine-tuning, retrieval systems, or tool-augmented agents. The controlled pressure variables are coarse summaries, matched analyses remain observational, and numeric results are conditional on evaluable aligned outputs. Hidden confounders and survivor effects may remain.

Controlled-split diversity. The controlled split instantiates 348 reasoning templates over only five matches. This design supports repeated within-template comparisons under relatively stable match contexts, but it limits match-level diversity. Consequently, pressure effects may partly reflect the discourse styles, player distributions, or event composition of these matches and should be interpreted as diagnostic rather than population-wide estimates.

10 Ethical Considerations

The source data consists of publicly accessible cricket commentary and professional athlete names. No private or sensitive records are collected. We retain only match-relevant commentary and associated gameplay information. Nevertheless, public accessibility does not eliminate intellectual-property obligations. Any release should distribute derived annotations, gold tables, provenance, and evaluation code under an appropriate research license while directing users to the original source under its governing terms.

Real player identities can create memorization and association effects. Entity substitution and anonymized evaluations can test whether results depend on familiarity with particular players. The benchmark is intended for academic analysis of structured reasoning and should not be used to make claims about player ability, intent, or reputation. Model-generated and model-repaired examples are marked in provenance metadata so downstream users can audit the role of automated construction.

References

Naman Ahuja, Fenil Bardoliya, Chitta Baral, and Vivek Gupta. 2025. Map&make: Schema guided text to table generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 30249–30262.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R. Gormley. 2025. Oolong: Evaluating long context reasoning and aggregation capabilities. *arXiv preprint arXiv:2511.02817*.

Yves Bestgen. 2022. Please, don't forget the difference and the confidence interval when seeking for the state-of-the-art status. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5956–5962.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of

numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 176–181.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9300–9322.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of NAACL-HLT*, pages 2368–2378.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Satyananda Kashyap, Sola Shirai, Nandana Mihindukulasooriya, and Horst Samulowitz. 2025. Structtext: A synthetic table-to-text approach for benchmark generation with multi-dimensional evaluation. *arXiv preprint arXiv:2507.21340*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, and 1 others. 2020. Measuring compositional generalization: A comprehensive method on realistic data. *International Conference on Learning Representations*.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, and 1 others. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, and 1 others. 2023a. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*.

Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023b. A sequence-to-sequence & set model for text-to-table generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of NAACL-HLT*, pages 5336–5358.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Proceedings of ACL-IJCNLP*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901.

Ritam Upadhyay, Naman Ahuja, Rishabh Baral, Aparna Garimella, and Vivek Gupta. 2026. Moneyball with LLMs: Analyzing tabular summarization in sports narratives. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 41714–41739, San Diego, California, United States. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 13484–13508.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. Text-to-table: A new way of information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 2518–2533.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Ran Zmigrod, Tim Vieira, and Ryan Cotterell. 2022. Exact paired-permutation testing for structured test statistics. In *Proceedings of NAACL-HLT*, pages 4894–4902.

A Metric Definitions

Let T^* denote a gold answer table and $\hat{T}$ a predicted table. Let $R(T)$ denote a canonicalized row signature, $C(T)$ its ordered column signature, and G_{PK} and $\hat{G}_{\text{PK}}$ the sets of gold and predicted primary-key tuples when key columns are available. Numeric values are normalized to a fixed precision before comparison.

Row and table metrics. Row Match is an exact comparison of canonical row signatures:

$$\text{RowMatch}(T^*, \hat{T}) = \mathbb{1}[R(T^*) = R(\hat{T})]. \quad (1)$$

The reported dataset score is the mean of this indicator. Exact Table Match is one only when the ordered column signatures, row signatures, and all normalized cells match. It is stricter than Row Match because a prediction with the right row identities but one incorrect value is counted as a complete failure.

Primary-key metrics. When primary-key columns are defined,

$$\text{PKExact}(T^*, \hat{T}) = \mathbb{1}[G_{\text{PK}} = \hat{G}_{\text{PK}}]. \quad (2)$$

Precision and recall are:

$$\text{PKPrec.} = \frac{|G_{\text{PK}} \cap \hat{G}_{\text{PK}}|}{|\hat{G}_{\text{PK}}|}, \quad (3)$$

$$\text{PKRec.} = \frac{|G_{\text{PK}} \cap \hat{G}_{\text{PK}}|}{|G_{\text{PK}}|}. \quad (4)$$

PK F1 is their harmonic mean. Key values are canonicalized before matching; predictions without a parseable key set receive zero for exact recovery.

Cell Accuracy. Predicted and gold rows are aligned through the primary key when possible. Cell Accuracy is the percentage of aligned non-key cells that match. Text cells use canonical exact comparison. Numeric cells permit a small tolerance to absorb formatting differences such as integer versus decimal notation. Missing, unsupported, or unparseable values are incorrect.

Weighted RMSE. Let V_j be the set of aligned numeric gold–prediction pairs in column j, with gold value v and predicted value $\hat{v}$. Column-wise error is:

$$\text{RMSE}_j = \sqrt{\frac{1}{|V_j|} \sum_{(v,\hat{v}) \in V_j} (v - \hat{v})^2}. \quad (5)$$

We combine columns with a count-weighted mean:

$$\text{WeightedRMSE} = \frac{\sum_j |V_j| \, \text{RMSE}_j}{\sum_j |V_j|}. \quad (6)$$

This weighting prevents a column with few aligned observations from receiving the same influence as a densely populated column. It does not make values with different natural scales fully comparable, which is why pressure-level RMSE comparisons are interpreted cautiously.

Directional bias. For aligned numeric cells, Overcount% is the percentage for which $\hat{v} > v$, and Undercount% is the percentage for which $\hat{v} < v$. We define:

$$\text{NetBias} = \text{Overcount}\% - \text{Undercount}\%. \quad (7)$$

Positive values indicate an overall tendency to overcount and negative values indicate undercounting. Exact matches contribute to neither rate.

B Dataset Generation and Validation

B.1 Intermediate Event Representation

Each source match is represented by a ball-by-ball event table during benchmark construction. A row records the local game state associated with one legal or illegal delivery, including the innings, over and ball index, batting and bowling entities, runs attributed to the batter and extras, total runs, wicket indicators, and dismissal metadata where available. Commentary remains the test-time input. The event table is an authoring and validation device that permits deterministic SQL execution.

Queries are not restricted to selecting existing columns. SQL templates may compute counts, sums, averages, percentages, rates, extrema, streaks, conditional comparisons, and grouped results. Some questions restrict a span of the match; others compare entities or interactions across the full narrative. The requested answer columns and primary keys are stored with each example so that evaluation can distinguish structural from value errors.

B.2 Generation Stages

The synthesis process proceeds in four stages:

1. **Intent proposal.** A seed query and sampled analytical idea are combined to propose a new goal that remains answerable from the event schema.
2. **Program synthesis.** The goal is translated into parameterized SQL. Placeholders mark variables such as batter, bowler, innings, or temporal boundaries.
3. **Instantiation and execution.** Variables are grounded in actual match entities and valid ranges. The resulting program is executed; failing, empty, degenerate, or duplicate outputs are discarded.
4. **Question realization.** A natural-language question and paraphrases are produced from the accepted program. The wording must preserve conditions, grouping semantics, and requested output fields.

Canonicalization removes superficial SQL differences before deduplication. Execution filtering also excludes templates that always return the same trivial answer across candidate matches. This produces a query bank with controlled variation over entities, attributes, temporal context, and operations.

B.3 Consistency Adjudication

For every natural-language question, a separate model generates SQL from the question alone. We execute this regenerated program and compare its output with the original gold program. Agreement provides an execution-level consistency check. Disagreement triggers adjudication:

- the original SQL is retained when it matches the natural-language intent;
- the regenerated SQL replaces it when the regeneration better expresses the question;
- a corrected question–SQL pair is produced when neither is coherent.

The adjudicator chose the original program for 511 disputed examples, the regenerated program for 590, and a corrected pair for 944. Corrected items are re-executed and must pass the same consistency check. A manual audit of 400 items found 97.2% question–program coherence. The release records whether an instance was accepted directly, adjudicated, or repaired.

C Baseline Prompting and Inference

C.1 Common Input and Output Contract

All methods receive the same core fields: the natural-language question, target columns in their required order, row-key columns, primary-key metadata, and complete ball-by-ball commentary. A common policy block defines how to interpret runs, balls, wickets, overs, rates, and percentages. The commentary is designated as the source of truth. Models are instructed to return strict JSON, preserve target columns exactly, place computed numbers rather than expressions in numeric cells, and use `null` rather than inventing unsupported values.

The canonical final output is:

```
{
  "columns": ["<col 1>", "<col 2>"],
  "rows": [[<r1c1>, <r1c2>], ...]
}
```

Methods with visible reasoning wrap this object in a method-specific field. The evaluator extracts only the final table for scoring.

C.2 Prompting Methods

ZERO SHOT. One call directly produces the final table. The system prompt asks the model to reason internally, but the visible output may contain only the JSON table. This is the minimal end-to-end baseline and provides no explicit mechanism for inspecting intermediate decisions.

ZERO-SHOT COT. One call returns a free-form `reasoning_trace` followed by `final_table`. The trace must be instance-specific and may include relevant entities, filters, intermediate counts, calculations, and row decisions. This tests whether visible unconstrained reasoning improves over direct generation.

GUIDED COT. One call must produce three ordered reasoning records: identify relevant entities, compute required statistics, and build the final table. The fixed scaffold is the only difference from free-form chain-of-thought. It tests whether explicitly separating grounding, calculation, and assembly helps within a single context window.

PLANFILL. The planner receives the full instance but is prohibited from computing final values. It returns row headers expressed through the supplied key columns and a concise operational plan. The second call receives the original instance,

planner JSON, and an empty table skeleton. It fills values while preserving the proposed row structure unless the commentary clearly requires correction. This baseline isolates a structure-first decomposition.

SC TABLE VOTING. Five candidates are independently sampled from the same prompt. Each contains a reasoning trace and complete table. Primary-key rows are aligned across candidates using fuzzy key matching and Hungarian assignment. Candidate tables first vote over row signatures. Majority-supported extra rows can be retained. Cells then vote within aligned rows using canonical exact matching; numeric values within tolerance 0.2 are clustered. Without a usable key, aggregation falls back to row order.

ROWCOT. One call produces a non-empty `row_reasoning` list and a final table. Each item contains one row-key object and a trace explaining how that row's values were derived. The output contract requires exactly one reasoning item per final row. This allocates visible computation locally rather than asking one trace to maintain state for the whole table.

LEAST-TO-MOST. A planner first generates a directed acyclic graph of at most five steps. Each step declares dependencies, whether it needs the full commentary, which prior output keys it consumes, and what structured key it produces. Independent steps can execute in parallel. Executor calls solve each subproblem, passing structured outputs to dependent steps. Only the final step may emit `final_table`. This method distributes computation across an explicitly generated workflow.

C.3 Inference Cost

Table 5 reports average cost on the common 1,152-example coverage subset. Counts are summed across all calls needed to produce a final answer. Gemini's total-token accounting includes hidden reasoning tokens and is therefore larger than the visible input-plus-output sum.

C.4 Implementation Details

Llama and Qwen are served with vLLM on two H200 nodes with 80GB GPUs each. GPT-4.1 is accessed through the OpenAI Batch API and Gemini-2.5-Flash through Google AI Studio. Single-call non-Gemini methods have a maximum completion budget up to 15,000 tokens, with smaller task-specific budgets of 2,000 for ZERO SHOT and 4,000 for GUIDED COT. Gemini permits up to 32,000 tokens to account for hidden-token use. LEAST-TO-MOST uses separate 10,000-token planner and 15,000-token executor budgets.

Retry budgets range from three to five depending on the baseline. A generation is retried if JSON cannot be parsed, requested columns are missing or reordered, row lengths disagree with the schema, or a method-specific field is absent. These retries correct output-contract failures, not incorrect cricket reasoning. SC TABLE VOTING always requests five valid candidates before aggregation.

D Controlled Factors and Pressure Construction

D.1 Skill Families

The controlled split labels each example along five families. These labels are derived from the executable query and generation metadata rather than assigned from model outputs.

Entity and schema grounding. *Schema-only* examples contain no grounded batter or bowler variable and depend only on table-level structure. *Single-entity* examples ground exactly one batter or bowler variable. *Multi-entity* examples jointly ground two or more such entities. These labels represent entity-resolution burden rather than broad semantic topic.

Context scoping. *Full-span* examples may draw evidence from anywhere in the complete match narrative. *Partial-span* examples limit reasoning to the first or second half of the context. The distinction concerns search scope, not the type of cricket event.

Temporal reasoning. *Non-temporal controls* contain no explicit time predicate. *Point-based* filters use one-sided relations such as before, before-or-at, after, or after-or-at. *Range-based* filters select a bounded interval. *Interval-overlap* questions ask whether queried and target intervals intersect.

Query operations. *Simple retrieval* includes direct selection, top-k, and ordered retrieval. *Filtered aggregation* aggregates after row filtering without grouping. *Grouped aggregation* computes per-group summaries. *Group comparison* applies post-aggregation filtering, ranking, or comparison over groups.

Model	Baseline	Calls	Input tokens	Output tokens	Total tokens
Llama-3.3-70B	ZERO SHOT	1.0	9424.16	109.52	9533.68
	ZERO-SHOT COT	1.0	9918.69	323.42	10242.11
	GUIDED COT	1.0	10371.28	448.74	10820.02
	ROWCOT	1.0	10406.29	835.23	11241.51
	PLANFILL	2.0	17913.29	538.29	18451.58
	SC TABLE VOTING	5.0	48126.69	1434.42	49561.11
	LEAST-TO-MOST	3.4	25284.21	1282.26	26566.47
Qwen2.5-72B	ZERO SHOT	1.0	9621.13	112.23	9733.37
	ZERO-SHOT COT	1.0	10009.04	244.13	10253.17
	GUIDED COT	1.0	10012.25	378.96	10391.21
	ROWCOT	1.0	10358.53	665.64	11024.17
	PLANFILL	2.0	19831.94	484.90	20316.84
	SC TABLE VOTING	5.0	48638.65	1015.57	49654.22
	LEAST-TO-MOST	3.2	27347.44	1454.55	28801.99
GPT-4.1	ZERO SHOT	1.0	9356.42	76.42	9432.84
GPT-4.1	GUIDED COT	1.0	9850.01	294.62	10144.63
Gemini-2.5-Flash	ZERO SHOT	1.0	9390.76	125.88	21361.03
Gemini-2.5-Flash	GUIDED COT	1.0	9980.57	595.93	18087.22

Table 5: Average inference cost per example on a shared 1,152-example coverage subset. Multi-call methods sum tokens over all component calls.

Compositional depth. Executable operation chains are binned as 2 steps, 3–4 steps, 5–6 steps, or at least 7 steps. Depth counts the normalized sequence of retrieval, filtering, grouping, aggregation, comparison, and output operations needed to execute the query.

D.2 Pressure Axes

Table 7 defines the mappings used in the main analysis. Grounding is a direct collapse of its factor family. Execution combines query operation and compositional depth. Output combines answer width and height. Localization combines explicit temporal restriction, span restriction, and evidence sparsity.

Pressure axes deliberately compress a larger annotation space. For example, two examples with high localization pressure may reach that label through different combinations of a narrow temporal window and sparse evidence. We therefore use pressure contrasts to summarize recurring difficulty, and factor contrasts to retain interpretability. Neither level should be read as an experimentally randomized intervention.

E Statistical Analysis and Diagnostics

E.1 Paired and Stratified Comparisons

The headline model comparison is paired by example. Confidence intervals are estimated over paired differences, and sign-flip permutation tests evaluate whether the mean paired difference is centered at zero. This preserves the correspondence between model outputs on the same question.

Pressure analyses compare levels within a model. Pooled contrasts use all examples at each level and are useful for descriptive coverage, but may mix different templates and skill combinations. Stratified contrasts match on the remaining observed factors so that the target axis changes within a more comparable condition. A stratum contributes only when both compared levels are represented.

E.2 Balanced Resampling

Support is unequal for several pressure levels. In a pooled contrast, the larger group is repeatedly sampled without replacement down to the size of the smaller group. In a stratified contrast, equalization occurs separately inside each matched stratum before stratum-level differences are combined. Pooled analyses use 2,000 balanced resamples and stratified analyses use 1,000.

The reported balanced interval is the empirical

Family	Skill	Representative question
Grounding	Single entity	For each batter who hit at least one boundary and scored at least one run off Faheem, what percentage of their runs against him came from boundaries?
Grounding	Multi-entity	How many deliveries did Verreynne face while batting against Hasan Ali?
Context	Partial span	Calculate Markram's strike rate in the specified half of the match for spans in which he faced at least ten balls.
Temporal	Point based	List wickets claimed by each bowler against each batter before over 29.5.
Temporal	Range based	Provide wickets taken by each bowler between overs 2.3 and 32.1.
Query operation	Grouped aggregation	Compare each batter's strike rate against Shaheen Shah Afridi on boundary-scoring and non-boundary deliveries.
Query operation	Group comparison	For each batter, what percentage of runs against a bowler came from boundaries?
Composition	7+ steps	For each batter, what is the longest streak of consecutive dot balls against Faheem?

Table 6: Representative controlled-split questions. Names and temporal values are grounded variables that change across instantiations.

Axis	Score construction	Level mapping
Grounding	Schema-only $= 0$; single-entity $= 1$; multi-entity $= 2$.	Low $= 0$; medium $= 1$; high $= 2$.
Localization	$+1$ for temporal reasoning; $+1$ for a first-half or second-half span; $+1$ when contributing-row fraction is in $(.10, .25]$; $+2$ when it is at most .10. Evidence-count buckets are used when the fraction is unavailable.	Low ≤ 1; medium $= 2$ or 3; high ≥ 4.
Execution	Query operation: retrieval $= 0$, filtered aggregation $= 1$, grouped aggregation $= 2$, group comparison $= 3$. Composition: 2 steps $= 0$, 3–4 $= 1$, 5–6 $= 2$, 7+ $= 3$. Scores are summed.	Low ≤ 1; medium $= 2$ or 3; high ≥ 4.
Output	Columns: $1 = 0$, 2–3 $= 1$, 4–6 $= 2$, 7+ $= 3$. Rows: 0–1 $= 0$, 2–5 $= 1$, 6–10 $= 2$, 11–20 $= 3$, 21+ $= 4$. Missing counts default to 1. Scores are summed.	Low ≤ 1; medium $= 2$ or 3; high ≥ 4.

Table 7: Construction of the controlled-split pressure axes. Contributing-row fraction is the number of rows used in the answer divided by the number scanned.

2.5–97.5 percentile range of these repeated support-balanced estimates. It is not an ordinary bootstrap confidence interval. Its purpose is to test whether the direction of a contrast survives removal of sample-count imbalance by design. Resampling cannot repair absent overlap. Contrasts with only one or two matched strata are retained for auditability but are not used for main conclusions.

We do not interpret the full diagnostic surface as a confirmatory multiple-testing exercise. Main claims require agreement among the raw estimate, paired or sign-flip test, and balanced sensitivity analysis. Isolated threshold crossings in the appendix are descriptive.

E.3 Headline Pressure Contrasts

E.4 Full Pooled Event Contrasts

The pooled table shows large apparent effects for localization and execution, but also extreme execution imbalance: medium-versus-low support differs by more than 72:1. This is why the main text privileges matched-stratum contrasts. After matching, high-versus-medium execution remains stable for both models, while medium-versus-low has only one overlapping stratum and cannot support a broad conclusion.

E.5 Full Stratified Event Contrasts

E.6 Full Pooled Numeric Contrasts

Several pooled localization and output trends appear to suggest lower RMSE at higher pressure. We do not interpret these as evidence that harder localization or larger tables improve computation. Numeric evaluation is conditional on outputs that can be parsed and aligned. Under difficult conditions, the surviving evaluable subset may be systematically easier, creating a survivor effect. Scale differences among query types can also affect RMSE. Matched analyses and directional bias provide a more cautious view.

Factor	Contrast	Event Llama	Event Qwen	Numeric Llama	Numeric Qwen
Temporal	Point-based vs. non-temporal	-8.7*	-2.4	4.6*	3.7*
Temporal	Range-based vs. non-temporal	-9.1*	-3.0	0.5	3.3
Entity grounding	Multi-entity vs. single-entity	-1.3	6.5	12.4*	12.0*
Query operation	Group comparison vs. grouped aggregation	0.6	-3.1	-4.5	0.2
Composition	5–6 steps vs. 3–4 steps	-5.2*	-2.7	9.4*	7.4

Table 8: Selected controlled-split factor contrasts. Event columns report changes in PK F1, where negative is harder; numeric columns report changes in weighted RMSE, where positive is harder. Asterisks mark bootstrap intervals excluding zero.

Model	Axis	Contrast	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	medium vs. low	12.0	.020	17.1
Qwen	Grounding	medium vs. low	15.2	.005	18.8
Llama	Execution	high vs. medium	-4.5	.042	-4.4
Qwen	Execution	high vs. medium	-6.9	.002	-6.9
Llama	Output	medium vs. low	-12.1	$<.001$	-10.2
Qwen	Output	medium vs. low	-9.4	.001	-7.6

Table 9: Robust stratified event-identification pressure contrasts. Negative deltas indicate that the first level is harder.

Model	Axis	Contrast	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	high vs. medium	19.6	.002	11.8
Qwen	Grounding	high vs. medium	8.9	.131	8.8
Llama	Execution	high vs. low	29.9	.032	28.3
Qwen	Execution	high vs. low	43.7	.016	38.3
Llama	Output	medium vs. low	7.8	.027	13.7
Qwen	Output	medium vs. low	2.7	.577	11.3

Table 10: Stratified numeric-fidelity pressure contrasts using weighted RMSE. Positive deltas indicate that the first level has higher error. Qwen grounding and output rows are sensitivity results.

E.7 Full Stratified Numeric Contrasts

F Numeric Bias and Self-Consistency

Table 15 supplies the values behind Figure 3. The declining RMSE at high output pressure should not be confused with improved unconditional performance, because the metric is computed only on aligned and evaluable outputs. Directional rates show that the character of the remaining numeric errors changes sharply. Qwen moves from an 18.8-point overcount bias at low pressure to a 24.9-point undercount bias at high pressure.

Self-consistency achieves substantial agreement without the best numerical accuracy. The final voted Llama tables have average cell support of 3.83 out of five, and 85.9% of samples have mean support above three. Nonetheless, SC TABLE VOTING has RMSE 18.12, compared with 17.67 for ZERO-SHOT COT and 16.16 for ROWCOT. Repeated candidates often reproduce the same arithmetic or evidence-selection error. Voting is effective when candidate noise is independent enough for the correct value to become a plurality; it cannot repair a systematic computation shared by the model's samples.

G Recommended Extensions

Three experiments would most directly extend the current evidence. First, a model should generate SQL or code over the gold event table. This measures program induction and computation under privileged structural access. Second, a model should extract its own event table and then generate SQL or code over that table. Comparing the two settings with direct narrative-to-table generation separates extraction, program synthesis, and execution errors. Specialized Text-to-Table systems adapted to query-conditioned answer tables would further clarify how much benefit comes from architectures designed for structured generation.

Third, evaluation should be repeated on at least one non-cricket narrative domain with a different latent record structure. Financial filings, clinical timelines, legal case histories, and incident reports are plausible candidates, but each carries distinct annotation, licensing, and privacy constraints. A multi-domain release should retain domain-specific diagnostics rather than assuming that one shared score captures every form of narrative structure.

Construction provenance should also become an explicit analysis variable. Human-authored, model-generated, rewritten, repaired, and naturally occurring questions can be evaluated as separate subsets. Cross-generator tests—for example, evaluating models unrelated to the construction models—would help determine whether stylistic artifacts contribute to observed performance. These extensions are necessary before Q-STR can support claims beyond its present role as a controlled cricket diagnostic.

Model	Axis	Contrast	Imbalance	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	medium vs. low	1.175	7.0	$<.001$	7.0
Llama	Grounding	high vs. low	8.703	-5.7	.018	-5.7
Llama	Grounding	high vs. medium	10.227	-12.7	$<.001$	-12.8
Llama	Localization	medium vs. low	2.025	-5.6	$<.001$	-5.6
Llama	Localization	high vs. low	4.907	-19.6	$<.001$	-19.6
Llama	Localization	high vs. medium	9.938	-13.9	$<.001$	-14.0
Llama	Execution	medium vs. low	72.500	23.2	$<.001$	23.2
Llama	Execution	high vs. low	85.938	19.5	.008	19.5
Llama	Execution	high vs. medium	1.185	-3.7	$<.001$	-3.7
Llama	Output	medium vs. low	1.119	-3.3	.003	-3.3
Llama	Output	high vs. low	3.595	-2.5	.184	-2.5
Llama	Output	high vs. medium	4.024	0.8	.578	0.8
Qwen	Grounding	medium vs. low	1.175	9.4	$<.001$	9.5
Qwen	Grounding	high vs. low	8.703	-3.1	.218	-3.1
Qwen	Grounding	high vs. medium	10.227	-12.5	$<.001$	-12.6
Qwen	Localization	medium vs. low	2.025	-5.0	$<.001$	-5.0
Qwen	Localization	high vs. low	4.907	-18.1	$<.001$	-18.1
Qwen	Localization	high vs. medium	9.938	-13.0	$<.001$	-13.1
Qwen	Execution	medium vs. low	72.500	36.1	$<.001$	36.0
Qwen	Execution	high vs. low	85.938	31.6	$<.001$	31.6
Qwen	Execution	high vs. medium	1.185	-4.5	$<.001$	-4.5
Qwen	Output	medium vs. low	1.119	0.5	.658	0.5
Qwen	Output	high vs. low	3.595	-2.2	.253	-2.2
Qwen	Output	high vs. medium	4.024	-2.7	.053	-2.7

Table 11: Pooled event-stage pressure contrasts using PK F1. Imbalance is the ratio of support sizes before balancing. These rows are descriptive because pooled levels can contain different template mixtures.

Model	Axis	Contrast	Strata	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	medium vs. low	31	12.0	.020	17.1
Llama	Grounding	high vs. low	2	-3.9	1.000	-3.8
Llama	Grounding	high vs. medium	26	-13.2	.116	-4.0
Llama	Localization	medium vs. low	59	-3.8	.170	-5.7
Llama	Localization	high vs. medium	14	-1.9	.679	-5.7
Llama	Execution	medium vs. low	1	-100.0	1.000	-100.0
Llama	Execution	high vs. medium	107	-4.5	.042	-4.4
Llama	Output	medium vs. low	81	-12.1	$<.001$	-10.2
Llama	Output	high vs. medium	20	3.2	.375	3.4
Qwen	Grounding	medium vs. low	31	15.2	.005	18.8
Qwen	Grounding	high vs. low	2	-19.7	1.000	-20.1
Qwen	Grounding	high vs. medium	26	-0.9	.885	3.6
Qwen	Localization	medium vs. low	59	-3.3	.176	-5.1
Qwen	Localization	high vs. medium	14	-3.6	.607	-3.9
Qwen	Execution	medium vs. low	1	-100.0	1.000	-100.0
Qwen	Execution	high vs. medium	107	-6.9	.002	-6.9
Qwen	Output	medium vs. low	81	-9.4	.001	-7.6
Qwen	Output	high vs. medium	20	-1.6	.755	-4.1

Table 12: Stratified event-stage pressure contrasts. Rows with one or two matched strata are shown for transparency but are not interpreted.

Model	Axis	Contrast	Imbalance	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	medium vs. low	1.223	6.6	<.001	6.5
Llama	Grounding	high vs. low	4.901	10.7	<.001	10.6
Llama	Grounding	high vs. medium	5.996	4.1	.041	4.1
Llama	Localization	medium vs. low	2.196	-1.5	.182	-1.4
Llama	Localization	high vs. low	3.209	-8.7	<.001	-8.8
Llama	Localization	high vs. medium	7.047	-7.3	<.001	-7.3
Llama	Execution	medium vs. low	1.886	1.7	.155	1.6
Llama	Execution	high vs. low	2.295	11.1	<.001	11.2
Llama	Execution	high vs. medium	1.217	9.5	<.001	9.5
Llama	Output	medium vs. low	1.440	1.5	.166	1.4
Llama	Output	high vs. low	5.756	-5.4	.001	-5.4
Llama	Output	high vs. medium	3.997	-6.9	<.001	-6.9
Qwen	Grounding	medium vs. low	1.223	6.7	<.001	6.7
Qwen	Grounding	high vs. low	4.901	9.1	<.001	9.1
Qwen	Grounding	high vs. medium	5.996	2.4	.275	2.4
Qwen	Localization	medium vs. low	2.196	-1.3	.230	-1.3
Qwen	Localization	high vs. low	3.209	-11.6	<.001	-11.6
Qwen	Localization	high vs. medium	7.047	-10.2	<.001	-10.3
Qwen	Execution	medium vs. low	1.886	0.8	.362	0.8
Qwen	Execution	high vs. low	2.295	12.6	<.001	12.6
Qwen	Execution	high vs. medium	1.217	11.8	<.001	11.8
Qwen	Output	medium vs. low	1.440	1.8	.102	1.8
Qwen	Output	high vs. low	5.756	-3.9	.007	-3.9
Qwen	Output	high vs. medium	3.997	-5.6	.004	-5.7

Table 13: Pooled numeric-stage pressure contrasts using weighted RMSE. Positive values indicate greater error in the first level.

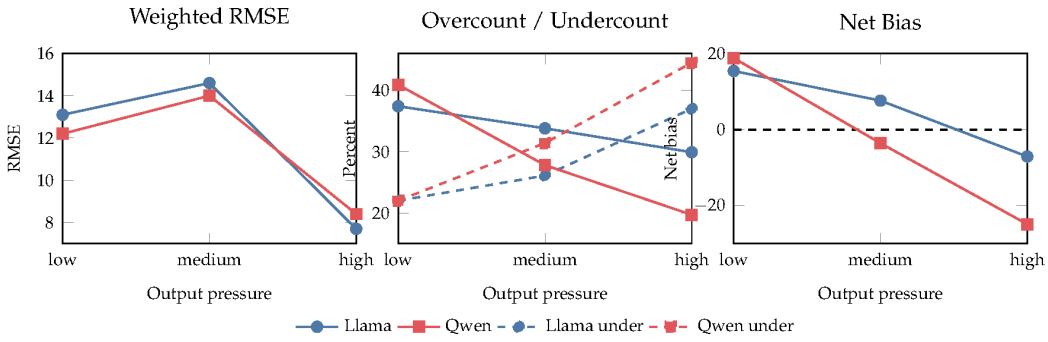


Figure 3: Numeric error profile by output pressure. Both models shift from overcounting toward undercounting as answer-table size increases; the shift is sharper for Qwen.

Model	Axis	Contrast	Strata	Raw Δ	Perm. p	Balanced Δ
Llama	Grounding	medium vs. low	42	2.0	.604	1.8
Llama	Grounding	high vs. low	4	-3.8	.613	-8.1
Llama	Grounding	high vs. medium	33	19.6	.002	11.8
Llama	Localization	medium vs. low	63	-1.4	.586	1.6
Llama	Localization	high vs. medium	25	2.8	.318	1.4
Llama	Execution	medium vs. low	9	19.0	.156	17.9
Llama	Execution	high vs. low	8	29.9	.032	28.3
Llama	Execution	high vs. medium	105	-0.2	.950	-1.3
Llama	Output	medium vs. low	77	7.8	.027	13.7
Llama	Output	high vs. medium	20	0.7	.944	-0.7
Qwen	Grounding	medium vs. low	42	4.1	.132	3.6
Qwen	Grounding	high vs. low	4	3.0	.252	2.7
Qwen	Grounding	high vs. medium	33	8.9	.131	8.8
Qwen	Localization	medium vs. low	63	0.1	.965	0.2
Qwen	Localization	high vs. medium	25	-1.8	.852	-0.4
Qwen	Execution	medium vs. low	9	15.0	.249	13.8
Qwen	Execution	high vs. low	8	43.7	.016	38.3
Qwen	Execution	high vs. medium	105	9.3	.009	5.7
Qwen	Output	medium vs. low	77	2.7	.577	11.3
Qwen	Output	high vs. medium	20	-2.8	.015	-3.0

Table 14: Stratified numeric-stage pressure contrasts using weighted RMSE. Raw and balanced estimates can differ when support remains uneven inside matched conditions.

Model	Output	RMSE	Over	Under	Net
Llama	low	13.1	37.4	22.0	15.4
Llama	medium	14.6	33.8	26.1	7.6
Llama	high	7.7	29.9	37.0	-7.1
Qwen	low	12.2	40.9	22.0	18.8
Qwen	medium	14.0	27.8	31.4	-3.6
Qwen	high	8.4	19.7	44.5	-24.9

Table 15: Exact numeric-error profile by output pressure. Over and Under are percentages; Net is Over minus Under.

Average cell support	Samples	Ratio (%)
$[1, 2)$	17	0.40
$[2, 3)$	359	8.43
$[3, 4)$	1966	46.17
$[4, 5)$	1694	39.78
$= 5$	203	4.77

Table 16: Distribution of average cell-level vote support for Llama SC TABLE VOTING, measured over five candidates.

Ontology-Guided Procedural Knowledge Extraction and Validation Case Study on the Lockout-Tagout Documents

Cennet Oguz[1,2] **Noon Pokaratsiri Goldstein**[1]
Jörg Steffen[1] **Reetam Biswas**[2] **Günter Neumann**[1]

[1]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
[2]Saarland University, Germany

{cennet.oguz, noon.pokaratsiri, steffen, guenter.neumann}@dfki.de

rebi00001@stud.uni-saarland.de

Abstract

We present a case study developing an ontology-guided framework for extracting and validating procedural knowledge from safety-critical documents with limited annotations, specifically the Lockout-Tagout (LOTO) procedural documents. We build multimodal Procedural Knowledge Graphs (PKGs) grounded in the Procedural Knowledge Ontology (PKO) from text and diagrams using LLMs and VLMs. The extracted PKGs are evaluated on instruction- and graph-level metrics with validation against a domain expert curated ontology for action–entity relations, step order, and cross-modal alignment. Embedding-based metrics capture semantic alignment, while lexical metrics expose safety-critical mismatches. Our experiments on proprietary and public LOTO documents demonstrate the feasibility and limitations of multimodal procedural extraction and motivate ontology-guided validation.

1 Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) have shown competencies at extracting structured knowledge from technical documents (Huguet Cabot and Navigli, 2021; Dagdelen et al., 2024; Bhattarai et al., 2024). Procedural knowledge extraction (PKE), however, requires representing not only entities and relations, but also ordered actions, dependencies, and their associated objects (Gupta et al., 2018; Dannenfelser et al., 2023). These requirements become particularly important in safety-critical industrial settings, where procedural representations must also preserve domain-specific operational constraints.

Lockout-Tagout (LOTO) provides a representative safety-critical use case. LOTO documents contain machine-specific procedures for controlling hazardous energy, with procedural information distributed across text, tables, document structure, and diagrams. These procedures specify ordered actions together with equipment targets, tools, locations, verification steps, and supporting visual evidence. Gupta et al. (2018); Dannenfelser et al. (2023); Wu et al. (2022) have investigated procedural extraction and multimodal procedural understanding, exploiting the explicit structured representation of a Procedural Knowledge Graph (PKG) to associate executable actions with entities through semantic, temporal, and cross-modal relations.

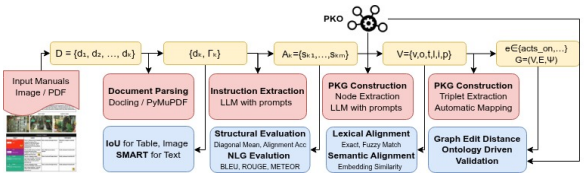


Figure 1: The end-to-end workflow from document parsing to PKG construction and evaluation.

Determining whether the extracted representation satisfies the safety requirements of the underlying procedure remains challenging. Although lexical, semantic, and graph-level similarities quantify the correspondence with a reference, they do not directly assess safety compliance. An extraction may remain highly similar to the reference while containing safety-critical errors, e.g. associating an action with the wrong energy-control component, omitting a required action, or altering the required execution order.

To address these challenges, we develop a multi-step ontology-guided framework, building on the Procedural Knowledge Ontology (PKO) (Carriero et al., 2025), to construct multimodal PKGs from heterogeneous LOTO documents. Ontologies and knowledge bases have been used to guide and verify LLM-based information extraction (Bhattarai et al., 2024; Chepurova et al., 2024), providing a basis for explicit domain-specific validation. By comparing this approach with end-to-end VLM-based extraction, we address the question: to what extent does the PKO-guided multi-step framework

The Proceedings of Workshop on Document Intelligence and Understanding, pages 217–231
October 24, 2026 ©2026 Association for Computational Linguistics

improve PKE? Additionally, to address the question of the effectiveness of existing standardized metrics in detecting safety-related errors, our evaluation examines the framework step-by-step and extend the final PKG assessment beyond reference-based text and graph similarity metrics to include safety validation against expert-verified LOTO constraints.

Our main contributions are as follows:

- We present a baseline framework for real-world LOTO PKE by extending Ren et al. (2023) multi-dependency approach to an ontology-guided multi-modal LLM-based setting.
- We provide a multi-modal dataset and annotation features for evaluating the proposed pipeline.
- We demonstrate potential limitations of existing similarity-based text and graph metrics in validating safety compliance.

2 Background & Related Works

Lockout-Tagout (LOTO) Procedure. LOTO is a regulated industrial safety practice that prevents accidental machine activation during maintenance.[1] LOTO documents contain machine-specific instructions in text, diagrams, and tables, and are subject to both regulatory and confidentiality requirements.

Procedural Knowledge Extraction (PKE). LLMs have streamlined knowledge graph construction sub-tasks, e.g. named entity recognition (NER) and relations extraction (RE), into generative tasks (Mujtaba and Mahapatra, 2019; Huguet Cabot and Navigli, 2021; Chia et al., 2022; Trajanoska et al., 2023; Chepurova et al., 2024; Castle and Moreno Schneider, 2025), including extracting both (Dagdelen et al., 2024; Gong and Eldardiry, 2024) at the same time. Whereas these works examine *declarative knowledge*, we extend this paradigm towards constructing *procedural knowledge* graphs (PKG) as explored in Yang and Nyberg (2015); Mujtaba and Mahapatra (2019); Wu et al. (2022); Ren et al. (2023).

[1]Regulated by the U.S. Occupational Safety and Health Administration (OSHA) (see https://www.osha.gov/control-hazardous-energy) and other agencies in other jurisdictions (see https://www.safetylock.net/learn-lockout-tagout/lockout-tagout-legislation/).

Ψ	Similarities in PKO Relations
$(v_i, precedes, v_j)$	S_{k_m} step order, $(Step, nextStep, Step)$
$(v_i, acts\_on, o_i)$	$(Procedure, hasProcedureTarget, Thing)$
$(v_i, uses, t_i)$	$(Action, requiresTool, Tool)$
$(v_i, occurs\_at, l_i)$	$(Procedure, isAdoptedBy, Organization)$
$(s_{k_j}, verify, v_i)$	S_{k_m} step order, $(Step, hasStepVerification, Verification)$
$(p_i, image\_of, n_i)$	$(Procedure, wasExtractedFrom, Resource)$

Table 1: Non-exhaustive examples of Ψ formalization according to how the semantic roles correspond to the inter-class relations in the PKO (Carriero et al., 2025) as shown in Figure 2. Relations can be transitive, e.g. $(Procedure, wasExtractedFrom, Resource)$ implies $(Tool, wasExtractedFrom, Resource)$ because there are relations between $Procedure$, $Action$, and $Tool$.

PKG construction has been studied in specialized domains such as biomedical (Dannenfelser et al., 2023; Bhattarai et al., 2024), material science (Dagdelen et al., 2024; Castle and Moreno Schneider, 2025), legal (Zheng et al., 2024), and technical support (Gupta et al., 2018). Research into PKG for internal manufacturing processes (as opposed to consumer product operations) remains underexplored. Ren et al. (2023) investigate PKG with multiple dependency types; they examine PKG where nodes and edges represent information and relations either *between sentences* or *within each sentence*. We extend this paradigm to support **both inter-sentence and intra-sentence relations, including cross-modal (text-visual) relations** in the manufacturing domain.

Procedural Knowledge Ontology (PKO). The PKO provides an expert-curated schema for representing domain-specific procedures, including LOTO; Carriero et al. (2025) has also demonstrated its generalizability beyond original use cases. Bhattarai et al. (2024); Chepurova et al. (2024); Wang et al. (2024) demonstrate that ontologies and knowledge bases can improve LLM extraction and validate outputs via schema validation and in-context learning (ICL). This work investigates the application of the **PKO** (Carriero et al., 2025) in both PKG construction and evaluation. As Gretkowski et al. (2022) and Carriero et al. (2025) observe that procedural documents often omit implicit knowledge, we restrict our scope to document-explicit information.

3 Methodology

3.1 Multi-Step PKG Formalization

Figures 1 & 2 illustrate how a document is transformed into machine-interpretable PKG according

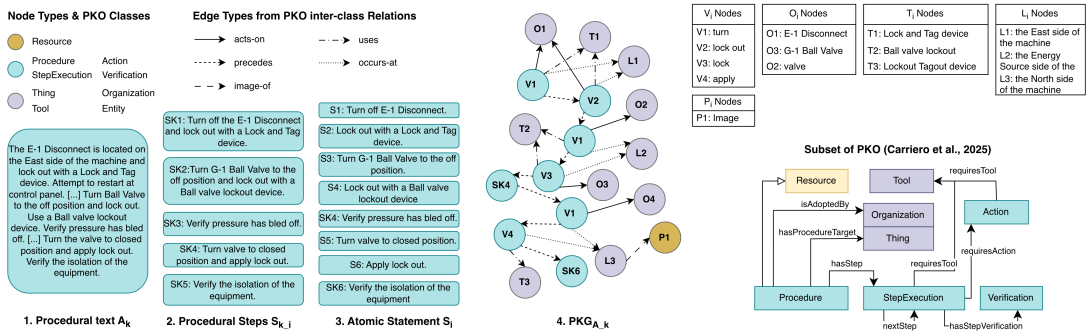


Figure 2: An example demonstrating how each extracted procedural text A_k is processed into executable procedural steps S_{k_m}. Atomic actions S_i along with semantic roles are extracted from each S_{k_i} to form PKG nodes. The PKO classes and the inter-class relations corresponding to semantic information define edge types, facilitating the formation of triplets, which are sequentially composed into the PK G_{A_k} as explained in Section 3.1.

to our proposed framework: 1) **Document Parsing** extracts multimodal text–layout representation, 2) **Instruction Extraction** identifies executable steps and semantic roles via LLM prompting, and 3) **PKG Construction** is finalized from node inference and triplet formalization with PKO (Carriero et al., 2025) guidance by re-prompting the same LLM.[2]

Document Parsing. A document $D = \{d_{1\to K}\}$ is decomposed into k-page text blocks $b_{k_i} = (t_{k_i}, \lambda_{k_i})$, where t_{k_i} is text and $\lambda_{k_i} = (x_1, y_1, x_2, y_2)$ encodes the layout. Structural metadata $\Gamma_k = \{\gamma_{k_1}, \gamma_{k_2}, \dots\}$ (e.g. headers, lists, tables) capture hierarchical organization. The resulting representation $D = \{(d_k, \Gamma_k) \mid_{k=1\to K}\}$ serves as input for instruction extraction.

Instruction Extraction. Executable steps $A_k = \{s_{k_1}, \dots, s_{k_m}\}$ are extracted using LLM few-shot prompting with each parsed page (d_k, Γ_k) as input. Formally, this is defined by a mapping $f : D \to A$, where $A = \bigcup_{k=1}^{K} A_k$ represents the complete set of procedures extracted from D.

Procedural Knowledge Graph. Extending from Ren et al. (2023), a PKG $G_{A_k} = (V, E, \Psi)$ is constructed from A_k, where V is the set of node types, E the directed relations, and $\Psi = \{(n_i, e_{ij}, n_j)\}$ the *procedural unit*. Each node $n_i \in V = \{v_i, o_i, t_i, l_i, S_{k_i}, p_i\}$ represents one of $\{action, target, tool, location, step, image\}$. A step description S_{k_i} can denote an *executable action* or a *condition*, e.g. prerequisite, precaution, informative, or verification. Each edge $e_{ij} \in E$ represents one of the procedural, semantic, and temporal (sequential order) relations inherently informed by the PKO (Carriero et al., 2025) (see Table 1) such as *acts_on*, *uses*, *occurs_at*, *image_of*, *precedes*, and *verify*. Some relations, which are not apparent, are transitive or analogous. For example, *occurs_at* is informed by *isAdeptedBy* in the PKO as *Procedure* has step *Action* (v_i) and *Organization* can be a *Location* (l_i).

Re-prompting the LLM decomposes each step into *atomic statements* $s_i = (v_i, \{o_i, t_i, l_i\})$, where v_i is the main verb and $\{o_i, t_i, l_i\}$ its argument set. A triplet representing a *procedural unit* (n_i, e_{ij}, n_j) is instantiated by connecting a main verb with its argument for intra-sentence dependencies or by connecting an *atomic statement* to another (inter-sentence dependencies). The final PKG $G_D = \{G_{A_1}, \dots, G_{A_K}\}$, where transitive *precedes* edges also capture inter-step (inter-sentence) dependencies, can represent nested procedures.

Cross-Modal Alignment. The cross-modal triplets $(p_i, image\_of, n_i)$ or $(n_i, wasExtractedFrom, p_i)$ are formed by maximizing the cosine similarity between the vision-language embeddings of each extracted node and the candidate bounding box regions.

3.2 Baseline VLM-based PKG Construction

As a contrastive baseline to our multi-step framework, in an end-to-end PKE approach, a k-page document $D = \{d_{1\to K}\}$ is represented as an ordered sequence of page images $X_d = \{x_{d_1}, x_{d_2}, \dots, x_{d_k}\}$ for input into a VLM via prompting to output the final PKG G_D directly. Formally, the extraction can be viewed as a func-

[2]We use few-shot prompting where templates include PKO class definitions and representative examples to enhance semantic coverage and output consistency (see Appendix C for task-specific details).

tion $f_m : X_d \rightarrow G_m$, where $G_m = (V_m, \mathcal{E}_m)$ is the graph extracted by a model m. Each node $n_i \in V_m = \{v_i, o_i, t_i, l_i, S_{k_i}\}$ and each edge $e_{ij} \in E_m$ represent the node and relation types of the final PKG as formalized in Section 3.1.

Cross-Modal Triplets. To obtain cross-modal triplets $(n_i, hasVisualEvidence, p_i)$, the same model is re-prompted to extract visual grounding reference nodes p_i that can be associated with any of the procedural steps or entities in V_m.

Dataset	Documents	Procedures	Instructions	Visual
Proprietary	86	236	1840	213
Public	18	36	62	21
All	**104**	**272**	**1,902**	**234**

Table 2: Dataset composition: "Procedures" denotes procedural groups or sections in a document, "Instructions" denotes step descriptions, and "Visual" denotes visual evidence items.

3.3 Dataset & Annotation

We introduce an evaluation dataset consisting of proprietary and public LOTO documents annotated with layout and semantic information; Table 2 summarizes the dataset statistics. Public documents are obtained from generic LOTO templates in the public domain (web searches); proprietary LOTO documents are written with confidential machine- and facility-specific instructions.[3]

Annotation. To provide ground truths for the evaluation of intermediary steps in our pipeline, we use the Roboflow interface[4] to annotate layout bounding boxes for page elements: *image*, *image-part*, *title*, *icon*, *paragraph text*, *itemized text*, *table*, and *table header*. The extracted text segments are manually aligned with their corresponding layout regions to provide visual-text grounding references.

Texts are further annotated as an ordered and nested list of *executable actions* and *conditional step* descriptions that capture hierarchy and sequence order. Each action is labeled with predefined PKG node and edge types based on the ontology mapping described in Section 3.1.

Executable actions are concrete steps performed by a practitioner and are associated, when stated, with entities (targets, tools, locations, verification targets), or expected conditions. *Conditional steps* describe precautions, informative statements, prerequisite steps, additional requirements, and verification statements that affect safe execution. *Visual evidence* (bounding boxes) is annotated when a figure, image, table region, or page-level visual element supports an *executable action*, action-associated *entity*, *precaution*, or *verification*.

Inter-Annotator Agreement. A Computer Science Master's student performs layout annotation. The Master's student and a PhD researcher conduct procedural text annotations. A domain expert provides annotations for the proprietary documents and another PhD researcher annotates the public ones. We calculate the mean *Jaccard* and *Cohen's* κ agreement scores (with exact lexical and normalized conceptual agreement matching[5]) for structured action tuples *(action, target, tool, location)*, excluding auxiliary fields. The mean exact (and normalized) agreement scores for proprietary and public LOTO documents are: 0.64 (0.81) & 0.54 (0.72) and 0.61 (0.79) & 0.48 (0.67) respectively.

The exact matching agreement scores reflect lexical variations and differences in annotation granularity; normalized scores indicate substantial conceptual agreement. The slightly lower agreement on proprietary data can be attributed to the more fine-grained annotations introduced by the domain-expert annotator.

4 Experimental Setup

Safety-critical work requires careful evaluation because errors can have serious consequences. We propose three complementary evaluation angles to evaluate the proposed framework: **intermediary assessment** (*document parsing* and *instruction extraction*), final **PKG validation** (*reference alignment* and *safety compliance*), and **cross-modal alignment**. Our evaluation experiments follow the pipeline in Figure 1. We use `Qwen3_14B`[6] for the evaluation in Sections 4.1 - 4.4.

Input. The initial input is a K-page procedural document $D = \{d_1, d_2, \ldots, d_K\}$ in PDF format. The inputs and outputs for the subsequent stages in the PKG construction pipeline follow the formalization in Section 3.1.

[3]Therefore, LOTO documents are rarely available for research, and only public documents can be released to motivate future works and facilitate reproducibility; see Appendix A for annotation details.

[4]`https://roboflow.com/`

[5]Conceptual normalization considers lexical variations that do not hinder human reviewers' understanding as a match, resulting in higher agreement scores.

[6]the Q4_K_M quantized distilled version from https://ollama.com/library/qwen3:14b

4.1 Intermediary Assessment–Document Parsing

We compare the extracted text elements t_k and the layout coordinates $\lambda_k = (x_1, y_1, x_2, y_2)$ for each page d_K from two parsing tools: *Docling*[7] and *PyMuPDF*[8]. The outputs are compared with our annotated ground truths according to two criteria: 1) **layout accuracy** of extracted text, table, and image regions, and 2) **text quality** of extracted sentences. Both tools output text-layout representations (d_k, Γ_k) in reading order. However, we use *Docling* to parse inputs for the subsequent steps of the pipeline to construct the final PKG as it demonstrates higher layout accuracy.

4.1.1 Evaluation Metrics

Layout accuracy is assessed as an image segmentation task, using Intersection-over-Union (**IoU**) over annotated bounding boxes. Text extraction quality is assessed using the **SMART** (Amplayo et al., 2023) scorer,[9] which measures sentence-level similarity between long texts. Together, these metrics quantify visual–text grounding fidelity.

Layout accuracy. We compute Intersection-over-Union (IoU) between predicted and annotated regions. Let L and L^* be the predicted and gold bounding regions: $\text{IoU} = \frac{|L \cap L^*|}{|L \cup L^*|}$.

Text quality. Because sentence boundaries and ordering can differ between parsers, we evaluate the extracted texts with **SMART** (Amplayo et al., 2023), which semantically aligns the predicted and reference sentences before computing *precision*, *recall*, and F_1. We report **Smart-1** and **Smart-L**, which emphasize factual alignment and long-span coherence, respectively.

4.2 Intermediary Assessment–Instruction Extraction

We prompt `Qwen3_14B` to extract executable procedural actions from each parsed page (d_k, Γ_k). The output from each page d_k, is a sequence of multi-step procedures $A_k = \{s_{k_1}, s_{k_2}, \ldots, s_{k_m}\}$. We examine both the quality of the generated texts and the correctness of the procedural order.

[7] https://github.com/docling-project/docling

[8] https://github.com/pymupdf/PyMuPDF

[9] with their suggested `BLEURT-20` (Pu et al., 2021) checkpoint to match unaligned sentences

4.2.1 Evaluation Metrics

We evaluate the extracted procedural steps with standard lexical metrics: (**BLEU** (Papineni et al., 2002), **ROUGE** (Lin, 2004), and **METEOR** (Banerjee and Lavie, 2005)). To assess semantic correspondence and step-order alignment beyond surface overlap, we use a pairwise cosine similarity matrix between predicted and annotated reference steps:

$$S_{ij} = \frac{\mathbf{p}_i \cdot \mathbf{a}_j}{\|\mathbf{p}_i\| \, \|\mathbf{a}_j\|}, \tag{1}$$

where $\mathbf{p}_i$ and $\mathbf{a}_j$ are predicted and annotated step embeddings respectively. We calculate the **Diagonal Mean**, **Alignment Accuracy**, **Off-Diagonal Mean**, and **Margin Difference** from the pair-wise matrix. The **Diagonal**$_{mean}$ captures the step-wise semantic similarity along the diagonal axis of the matrix. Mean scores use the lowest number of pairs as the sample size. **Accuracy**$_{alignment}$ captures order preservation, quantifying how correct position-wise (i.e. according to the step order) the most similar instructions are. **Off-Diagonal**$_{mean}$ and mean **Difference**$_{marginal}$ measure the cross-step confusion and separability, i.e. how semantically different the unaligned instructions are. Formal definitions and an illustrative example are in Appendix B).

4.3 Final PKG–Reference Alignment

We prompt the same LLM as in Section 4.2 to extract the nodes and form *procedural unit* triplets from the procedural steps A_k according to the formalization in Section 3.1 to construct the final PKG. The resultant PKG is compared to a reference graph constructed from nodes and edges annotated according to Section 3.3. We assess *node type*, *node & edge features*, *triplet*, and *graph-level* alignments.

4.3.1 Evaluation Metrics

Node/Edge & Graph Alignments. We report node and edge features **Jaccard index**:

$$J = \frac{|X_{\text{pred}} \cap X_{\text{gold}}|}{|X_{\text{pred}} \cup X_{\text{gold}}|}.$$

Triplet matching. Given a predicted $\hat{\psi} = (\hat{v}, \hat{r}, \hat{o})$ and gold $\psi = (v, r, o)$, an *exact* lexical match requires identical triplet elements: $S_{\text{exact}}(\hat{\psi}, \psi) = \mathbb{I}[\hat{v} = v \wedge \hat{r} = r \wedge \hat{o} = o]$. *Fuzzy* uses normalized string similarity over triplet elements, and *Embedding-based* matching computes

cosine similarity between triplet text embeddings. We report **Precision**, **Recall**, and $\mathbf{F_1}$ for triplets and **average similarity** over matched pairs.

For graph-level structural similarity, we calculate the Graph Edit Distance (**GED**) and report the normalized GED (**nGED**); lower values indicate better structural agreement.

4.4 Final PKG–Procedural Safety Compliance

In addition to assessing the extracted PKG alignments with the annotated reference at varying granularity levels, we address the question: **is the extracted procedural knowledge admissible under LOTO safety constraints?** Therefore, we evaluate each extracted procedure block against safety-relevant criteria encoded in an expert-verified ontology in two settings: *Ontology-only* (deterministic checks) and *Ontology-LLM* (Qwen3-assisted, ontology-constrained). We verify the results of both settings by comparing them with a certified LOTO safety expert evaluation (see Appendix B.4).

Ontology-only. For each extracted procedural group, we infer (i) an energy type from the source text strings, (ii) procedural action targets according to the energy type, (iii) map step texts to canonical action categories, and (iv) apply safety constraints encoded in a JSON ontology to detect compliance violations, e.g. energy mismatches, action target incompatibilities, missing required categories, and step order violations. Violations are summarized per energy type and aggregated into a document-level score.

Ontology-LLM. We prompt the same LLM to evaluate the presence of safety relevant errors per energy type procedure block. The JSON safety-constraints ontology is included in the prompt. The model returns structured JSON outputs containing assessment along the same safety criteria as in the Ontology-only setting (*energy source*, *action order*, and *missing action*) to facilitate comparison.

4.4.1 Safety Compliance Metrics

Scores from procedural blocks by energy-type are aggregated into a document-level safety score. For the safety criteria encoded in the ontology (*energy source*, *action order*, and *missing procedural steps*), we report the **accuracy** and **agreement**, expressed as Pearson ($\mathbf{r}$) and Spearman ($\boldsymbol{\rho}$) scores from the two evaluation settings compared to the assessment performed by a certified LOTO expert.

4.5 Cross-Modal Alignment

To evaluate cross-modal alignment, whether procedural entities are grounded across text-visual modalities, we measure whether the predicted $(p_i, \textit{image\_of}, n_i)$ triplets match the annotated text-visual ground truth associations. For the proposed multi-step pipeline formalized in Section 3.1, we compare performance between *CLIP* (Radford et al., 2021) and *PaliGemma* (Steiner et al., 2024) VLM embeddings.

4.5.1 Evaluation Metrics

For image–text alignment consistency, we evaluate the $(p_i, image\_of, n_i)$ relations using **Top-k Accuracy** (Acc@1, Acc@2). We further report **Mean Cosine Similarity** between aligned image and text embeddings to measure semantic closeness. These metrics quantify the extent to which each model is able to associate procedural text elements to their visual references.

4.6 Comparison with VLM-based PKG

To determine the benefits of our proposed multi-step pipeline, we compare the final PKG according to Section 3.1 and the one formalized according to Section 3.2 against the reference PKG constructed from annotations in Section 3.3. We use `Qwen3.5`[10] and `Gemma-4`[11] as representative "best-available" vision-language models that meet the data protection requirements of domain stakeholders without increasing the hardware resource requirements.

Input. For VLM-based PKG construction, a K-page document is represented as an ordered sequence of page images $X_d = \{x_{d_1}, x_{d_2}, \ldots, x_{d_K}\}$ (see Section 3.2).

4.6.1 Evaluation Metrics

We measure the cosine similarity[12] of the extracted *executable actions*, *conditional steps*, *procedural unit* aggregates consisting of action-associated entities (*action*, *target*, *tool*, *location*), and *cross-modal* edges in a pair-wise fashion. We report $\mathbf{F_1}$ scores. These criteria provide both sentence-level and intra-sentence extraction granularity measures.

[10] https://huggingface.co/Qwen/Qwen3.5-4B

[11] https://huggingface.co/google/gemma-4-E4B-it

[12] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 is used for embeddings; similarity threshold of 0.9 to allow for minor lexical variations while being strict for same-domain comparisons

Matching Strategy	Precision (%)	Recall (%)	F_1 (%)	Avg. Sim. (%)	Node Jac. (%)	Edge Jac. (%)	nGED (1 - nGED) (%)
Exact	31.41	44.28	36.09	–	52.11	63.42	35.00 (65.00)
Fuzzy (0.85)	72.22	67.78	67.74	87.22	54.03	67.15	19.80 (80.20)
Embedding	81.70	77.20	77.10	95.60	56.27	69.38	13.80 (86.20)

Table 3: The evaluation results of node/edge overlaps (Node/Edge Jac.), triplet matching (Precision, Recall, F_1, and Avg. Sim), and graph-level similarity (nGED (1 - nGED)) comparing *Exact*, *Fuzzy* (threshold of 0.85), and *Embedding-based* matching strategies. Scores are macro-averaged across all procedures.

5 Results & Discussion

Document Parsing and Instruction Extraction. Table 4 shows a trade-off between layout and text fidelity. *Docling* achieves slightly higher layout accuracy than *PyMuPDF* (78.98% vs. 76.80% IoU), whereas *PyMuPDF* obtains higher SMART scores. Since the downstream pipeline relies on structured document regions to preserve step boundaries and reading order, we prioritize layout fidelity and use *Docling* for subsequent multi-step PKG construction.

The extracted steps show moderate semantic and ordering correspondence with the reference (Table 5), with $\text{Diag}_{mean} = 64.27\%$ and $\text{Acc}_{align} = 64.29\%$. Their positive correlation ($r = 0.77$; Figure 3) indicates that higher semantic correspondence is associated with better procedural order alignment, although variation remains across examples. ROUGE-1 and METEOR are substantially higher than BLEU, consistent with substantial surface-form variation. However, small lexical changes can still alter LOTO meaning, e.g. *apply LOTO to G-1 first* versus *apply LOTO to E-1 first*. These results motivate combining semantic and lexical correspondence with domain-specific validation.

Framework	IoU	Smart-1_{F1}	Smart-L_{F1}
PyMuPDF	76.80	**58.07**	**51.73**
Docling	**78.98**	50.47	47.41

Table 4: Parsing comparison showing the trade-off between layout accuracy (IoU, %) and text alignment (SMART F_1, %).

PKG Reference Alignment. Reference-based PKG scores depend strongly on the matching criterion (Table 3). Triplet F_1 increases from 36.09% with Exact matching to 67.74% with Fuzzy and 77.10% with Embedding matching, while nGED decreases from 35.00% to 13.80%. The relatively small change in Node Jaccard compared with the much larger improvement in triplet F_1 suggests

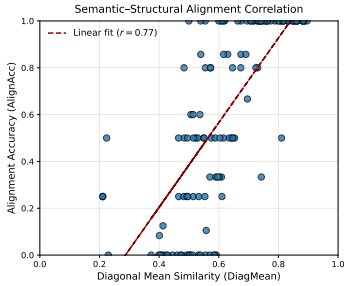


Figure 3: Correlation between semantic similarity (Diag_{mean}) and structural accuracy (Acc_{align}) across procedural steps ($r = 0.77$).

Diag	Acc_{align}	Off-Diag	Diff_{margin}
64.27	64.29	40.87	11.80
BLEU	**METEOR**	**ROUGE-1**	**Human**
37.06	61.92	72.25	86.02

Table 5: *Mean* embedding- and lexical-based scores (%). Diag and Acc_{align} capture semantic and order alignment; Off-Diag and Diff_{margin} quantify cross-step confusion.

that semantic matching primarily resolves alternative lexical realizations of procedural relations rather than substantially changing the set of extracted node types. The high average similarity under Embedding matching further indicates that many predicted triplets are semantically close to the reference despite surface-form differences.

At the same time, this result also exposes a limitation of reference-based evaluation in a safety-critical setting. Semantically similar triplets may still differ in a safety-relevant argument, equipment identifier, or procedural dependency. High embedding-based correspondence should therefore be interpreted as semantic graph similarity rather than evidence that the extracted procedure satisfies LOTO safety constraints. This motivates the separate ontology-guided compliance evaluation reported below.

Safety Compliance. Table 6 shows that ontology-guided evaluation provides an additional assessment of explicit LOTO constraints. Energy Source

verification is reliable in both settings (96.2–96.6% accuracy), suggesting that explicitly stated energy cues can be checked effectively using deterministic ontology rules. The largest benefit from Onto-LLM occurs for Action Order, where accuracy increases from 50.0% to 86.2% and expert correlation from 3.7/3.7 to 68.0/68.0. Qualitative analysis indicates that deterministic rules struggle with indirectly expressed actions, e.g. *try to start the machine* as a verification step, whereas the ontology-constrained LLM more reliably maps such expressions to predefined procedural categories. This contrast suggests that deterministic checks are most effective when safety-relevant evidence is explicit, while LLM assistance is particularly useful when linguistic variation must first be resolved before applying the same ontology-defined constraints.

Missing Action detection remains more challenging, improving from 61.5% to 69.0% accuracy, with expert correlation increasing from 29.6/29.6 to 46.2/46.2. Detecting omissions requires determining whether an expected procedural element is absent, making it less directly observable than classifying an explicitly stated action. Overall, the ontology defines the validation criteria, while the LLM acts as a constrained semantic mapper from free-form procedural language to those criteria. This preserves expert-defined safety requirements while improving their applicability to linguistically variable procedural descriptions.

Criteria	Setting	Accuracy	$\mathbf{r}/\boldsymbol{\rho}$ ($\times 100$)
Energy Source	Ontology	96.2	46.9/46.9
	Onto-LLM	96.6	69.4/69.4
Action Order	Ontology	50.0	3.7/3.7
	Onto-LLM	86.2	68.0/68.0
Missing Action	Ontology	61.5	29.6/29.6
	Onto-LLM	69.0	46.2/46.2

Table 6: Agreement with expert ontology-driven validation. Accuracy is reported in %; Pearson r and Spearman ρ are scaled by 100.

Cross-Modal Grounding. PaliGemma outperforms CLIP on action–image grounding (Table 7), increasing Acc@1 from 11.0% to 28.6% and Acc@2 from 24.4% to 55.6%. Nevertheless, the absolute accuracy remains modest, confirming that fine-grained association of procedural actions with visual evidence is a remaining challenge.

VLM-based PKG Comparison. The multi-step framework substantially outperforms direct end-to-end VLM extraction for the main structured extraction criteria (Table 8). Action F_1 improves by 34.56 points for Qwen3.5 and 49.00 points for Gemma4, while Procedural Unit F_1 improves by 39.43 and 36.12 points, respectively. Conditional extraction is less consistent, with a slight decrease for Qwen3.5 but a 14.01-point gain for Gemma4. Cross-Modal relations improve for both models, although their low absolute scores show that visual grounding remains challenging.

Model	Acc@1	Acc@2
CLIP	11.0	24.4
PaliGemma	**28.6**	**55.6**

Table 7: Action–image grounding accuracy (%) for CLIP and PaliGemma 2 Mix (3B-448).

Model	Criteria	End-to-End	Multi-Step	ΔF_1
Qwen3.5	Action	53.14	**87.70**	$+34.56$
	Procedural Unit	12.01	**51.44**	$+39.43$
	Conditional	**24.82**	23.43	-1.39
	Cross-Modal	11.76	**17.27**	$+5.51$
Gemma4	Action	39.16	**88.16**	$+49.00$
	Procedural Unit	5.46	**41.58**	$+36.12$
	Conditional	6.94	**20.95**	$+14.01$
	Cross-Modal	5.00	**11.67**	$+6.67$

Table 8: Comparison of end-to-end VLM PKG extraction and the multi-step PKG framework. Scores are F_1; ΔF_1 denotes Multi-Step minus End-to-End.

Overall Discussion. Taken together, the results highlight complementary roles for structured decomposition, semantic evaluation, and ontology-guided safety validation. The multi-step formulation is particularly effective for extracting actions and procedural units, whereas conditional and cross-modal relations remain more challenging. Reference-based semantic metrics accommodate substantial surface-form variation, but do not directly determine whether safety-relevant procedural constraints are preserved. Ontology-guided evaluation addresses this gap by making the validation criteria explicit, while LLM assistance improves the mapping of linguistically variable instructions to those predefined categories. These findings suggest that reliable procedural knowledge extraction in safety-critical settings benefits from combining model-based extraction with explicit, expert-defined validation rather than relying on either component alone.

6 Conclusion and Future Work

This case study demonstrates complementary benefits from multi-step PKG construction and

ontology-guided validation for safety-critical procedural documents. First, the multi-step framework substantially improves *Action* and *Procedural Unit* extraction over direct end-to-end VLM generation in both Qwen3.5 and Gemma4 models, supporting the decomposition of heterogeneous LOTO documents into structured intermediate representations. Second, cross-modal results show that visual evidence can be incorporated into the PKG, although fine-grained action–image grounding remains challenging. Third, ontology-guided evaluation complements reference-based text and graph similarity by explicitly checking energy-source consistency, procedural ordering, and required-action coverage. The Onto-LLM setting further improves agreement with a certified LOTO safety expert by mapping linguistically variable instructions to predefined ontology categories.

Together, these findings support using PKO as an interface for both structured procedural representation and constraint-specific validation, while also showing that semantic similarity alone is insufficient for evaluating safety-critical procedural knowledge. Conditional extraction, missing-action detection, and cross-modal grounding remain open challenges. Future work should improve these components, extend the encoded safety criteria, and evaluate transferability to other procedural and safety-critical domains.

Limitations

Single Procedure Type. Our experiments are conducted on a small real-world proprietary dataset supplemented by additional public LOTO procedural documents. Since our results are based on a single procedure type, future works are necessary to verify the transferability of the proposed pipeline and the cross-domain adaptability claimed by the PKO (Carriero et al., 2025). As one of the objectives for sharing our work is to establish a baseline on a real-world use case that is underexplored, we plan to investigate other procedure types in other safety-critical domains for future works. To encourage future research on procedural knowledge extraction from the LOTO procedure, we will also share an annotated LOTO dataset gathered from public sources.

PKO Inter-class Relations. The inter-class relations in the PKO we exploit for PKG triplet formation only provide coarse-grained cross-modal dependencies between procedural steps and visual references. We rely on transitive relations (Table 1) to map between procedure-level dependencies and semantic roles within each atomic action sentence. Similarly, the PKO does not support direct dependencies for image segments and step-action classes. Future research to investigate dynamic ontology updates with training data may be exploited to facilitate fine-grained action-image-segment and entity-image-segment relation identification.

Procedural Sequence Ordering. We rely on the document layout structure, i.e. left-to-right top-to-bottom reading order, to infer the temporal order of a procedural step sequence. No claims can be made from our results regarding the evaluated models' ability to infer procedural step ordering without layout information. We leave the procedural sequence ordering task for future works.

Multi-Component System Evaluation. Our proposed pipeline relies on multiple automatic metrics that are prevalent for the respective experiments at different stages in the workflow. As the results illustrate, curating a unified approach to evaluating multi-component systems holistically requires both technical and domain-specific insights. Ensuring that metrics commonly used in natural language processing (NLP) tasks provide reliable compliance with target-domain requirements and safety constraints remains a challenging task.

Ontology-guided Evaluation Our ontology-driven evaluation inherits the strengths and limitations of the underlying ontology. In particular, constraint checks are only as accurate as the ontology's coverage of device types, action categories, and safety rules. Missing or overly coarse concepts can lead to false positives (flagging valid steps as violations) or false negatives (failing to detect unsafe omissions or mismatched energy controls). Specialized-domain procedures may change over time; terminology and regulations vary in different regions and organizations. Consequently, an ontology that is not regularly maintained may become outdated. Some critical information is implicit or context-dependent (e.g. non-standardized shorthands or steps distributed across tables and figures), which can be difficult to encode explicitly. These factors suggest that ontology-guided evaluation is best viewed as a structured reliability layer that facilitates expert collaboration and oversight in an LLM-assisted workflow.

Ethical and Legal Considerations

The publicly sourced LOTO documents are obtained exclusively from open and unrestricted online sources. We use them solely for academic research purposes, without altering their content or redistributing them outside the scope permitted by fair use. The proprietary documents are handled under the confidentiality agreement with the industry partner organization. The use of LLMs/VLMs is only via locally hosted models in compliance with the confidentiality agreement; no sensitive information, machine identifiers, or company identification details are shared externally. All processing and analysis steps are carried out in accordance with ethical guidelines to ensure responsible use of industrial safety documentation.

Acknowledgments

The authors thank anonymous reviewers for their constructive feedback, Stefania Baldassarre and Diego Beltrame for their domain expertise in the evaluation activities. The work was partially funded by the European Union (EU) through the project PERKS–Eliciting and Exploiting Procedural Knowledge in Industry 5.0 (ID: 101120323)[13] under the "Digital, Industry, and Space" funding program and the German Federal Ministry of Research, Technology and Space (BMFTR) through the project TRAILS–Trustworthy and Inclusive Machines (ID: 16IW24005).[14]

References

Reinald Kim Amplayo, Peter J Liu, Yao Zhao, and Shashi Narayan. 2023. SMART: Sentences as basic units for text evaluation. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, Kigali, Rwanda. OpenReview.net.

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. Docling technical report. arXiv Preprint 2408.09869, arXiv.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Kriti Bhattarai, Inez Y. Oh, Zachary B. Abrams, and Albert M. Lai. 2024. Document-level Clinical Entity and Relation extraction via Knowledge Base-Guided Generation. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 318–327, Bangkok, Thailand. Association for Computational Linguistics.

Valentina Anita Carriero, Mario Scrocca, Ilaria Baroni, Antonia Azzini, and Irene Celino. 2025. Procedural Knowledge Ontology (PKO). In *The Semantic Web: 22nd European Semantic Web Conference, ESWC 2025, Portoroz, Slovenia, June 1–5, 2025, Proceedings, Part II*, page 334–350, Berlin, Heidelberg. Springer-Verlag.

Steffen Castle and Julian Moreno Schneider. 2025. Entity Linking using LLMs for Automated Product Carbon Footprint Estimation. In *Proceedings of the 1st Workshop on Ecology, Environment, and Natural Language Processing (NLP4Ecology2025)*, pages 56–60, Tallinn, Estonia. University of Tartu Library.

Alla Chepurova, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. 2024. Prompt Me One More Time: A Two-Step Knowledge Extraction Pipeline with Ontology-Based Verification. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pages 61–77, Bangkok, Thailand. Association for Computational Linguistics.

Yew Ken Chia, Lidong Bing, Soujanya Poria, and Luo Si. 2022. RelationPrompt: Leveraging Prompts to Generate Synthetic Data for Zero-Shot Relation Triplet Extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 45–57, Dublin, Ireland. Association for Computational Linguistics.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418. Publisher: Nature Publishing Group.

Ruth Dannenfelser, Jeffrey Zhong, Ran Zhang, and Vicky Yao. 2023. Into the Single Cell Multiverse: an End-to-End Dataset for Procedural Knowledge Extraction in Biomedical Texts. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*, volume 36, pages 10922–10934. Curran Associates, Inc.

Jiaying Gong and Hoda Eldardiry. 2024. Prompt-based Zero-shot Relation Extraction with Semantic Knowledge Augmentation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13143–13156, Torino, Italia. ELRA and ICCL.

[13] https://perks-project.eu/

[14] https://trails-dfki.github.io/

Andrzej Gretkowski, Dawid Wiśniewski, and Agnieszka Ławrynowicz. 2022. Should We Afford Affordances? Injecting ConceptNet Knowledge into BERT-Based Models to Improve Commonsense Reasoning Ability. In *Knowledge Engineering and Knowledge Management*, pages 97–104, Cham. Springer International Publishing.

Abhirut Gupta, Abhay Khosla, Gautam Singh, and Gargi Dasgupta. 2018. Mining Procedures from Technical Support Documents. *CoRR*, abs/1805.09780.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation Extraction By End-to-end Language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jorj X. McKie and R Liu. 2025. PyMuPDF: A Python Binding for MuPDF. `https://github.com/pymupdf/pymupdf`. Original-date: 2012-10-06T18:54:25Z.

Dena Mujtaba and Nihar Mahapatra. 2019. Recent Trends in Natural Language Understanding for Procedural Knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Amy Pu, Hyung Won Chung, Ankur Parikh, Sebastian Gehrmann, and Thibault Sellam. 2021. Learning compact metrics for MT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 751–762, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Haopeng Ren, Yushi Zeng, Yi Cai, Bihan Zhou, and Zetao Lian. 2023. Constructing Procedural Graphs with Multiple Dependency Relations: A New Dataset and Baseline. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8474–8486, Toronto, Canada. Association for Computational Linguistics.

Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. 2024. PaliGemma 2: A family of versatile vlms for transfer. *arXiv preprint*. ArXiv:2407.07726 [cs].

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, and 304 others. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. 2023. Enhancing Knowledge Graph Construction Using Large Language Models. ArXiv:2305.04676 [cs].

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, and 70 others. 2025. Embeddinggemma: Powerful and lightweight text representations.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting language models reasoning with chain-of-knowledge prompting. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981, Bangkok, Thailand. Association for Computational Linguistics.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding Multimodal Procedural Knowledge by Sequencing Multimodal Instructional Manuals. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4525–4542, Dublin, Ireland. Association for Computational Linguistics.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Lim Wang, Yunfei Chu, and Junyang Lin. 2026. Qwen3.5-omni technical report. Technical report.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. Technical report.

Zi Yang and Eric Nyberg. 2015. Leveraging Procedural Knowledge for Task-oriented Search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 513–522, New York, NY, USA. Association for Computing Machinery.

Chunmo Zheng, Xing Su, Yinqiu Tang, Junjing Li, and Mohamad Kassem. 2024. Retrieve-Enhance-Verify: A Novel Approach for Procedural Knowledge Extraction from Construction Contracts Via Large Language Models. SSRN:4883720.

A Annotation Guidelines

A.1 LOTO Annotation

This section describes the annotation procedure used to convert LOTO documents into structured procedural knowledge. A Computer Science Master's student performs layout annotation. The Master's student and a PhD researcher conduct procedural text annotations. A domain expert provides annotations for the proprietary documents and the PhD researcher annotates the public ones.

Inter-Annotator Agreement. We calculate the mean *Jaccards* and *Cohen's* κ agreement scores (with exact lexical and normalized conceptual agreement matching) for structured action tuples *(action, target, tool, location)*, excluding auxiliary fields. The mean exact (and normalized) scores for proprietary and public LOTO documents are 0.64 (0.81) & 0.54 (0.72) and 0.61 (0.79) & 0.48 (0.67) respectively.

1. Document Parsing. Annotation begins by reviewing the full LOTO document page. Each document typically contains a set of lockout steps consisting of step numbers, action descriptions, required tools, target components, location information, verification steps, and corresponding images. The annotator identifies and isolates the lockout step table along with any associated images.

2. Action Extraction. For each lockout step, annotator extracts the following two fields:

- **step_order**: The sequential number of the step as written in the LOTO procedure.
- **action**: The textual action describing what must be performed (for example, "Turn Disconnect to the off position" or "lock out").

The goal at this stage is to obtain a clean list of ordered natural language instructions exactly as they appear in the document.

3. Procedural Unit Extraction. For each extracted instruction, annotators expand the action into a structured node containing the following fields:

- **tool**: The tool specified in the document (for example, Lock and hasp device, a ball valve lockout device).
- **target**: The object or machine component being acted upon (for example, E-1 Disconnect, G-1 Ball Valve).
- **location**: The location text describing where the action takes place (for example, East side of the machine).
- **conditional units**: The verification or warning instruction required for safety (for example, "Attempt to restart at control panel" or "Verify pressure has bled off").
- **cross-modal**: The bounding box coordinates for the figure associated with that step.

Each step is annotated independently, using terminology directly copied from the document to ensure accuracy.

4. PKG Construction. The annotated instruction sentences and action-associated entities are formalized into a PKG according to Section 3.1.

Annotation Costs. Annotation was completed during a period of 7 months, averaging 8 hours per week. Graduate student assistants are compensated at the rate of 18 EUR per hour in compliance with local regulations. Other annotators are compensated through project funding according to their respective consortium agreements.

B Evaluation Details

This appendix summarizes the implementation details of the evaluation strategies in Section 4.

B.1 Instruction Extraction

We assess semantic and order-level correspondence using a pairwise cosine similarity matrix between predicted and reference steps:

$$S_{ij} = \frac{\mathbf{p}_i \cdot \mathbf{a}_j}{\|\mathbf{p}_i\| \, \|\mathbf{a}_j\|}, \tag{2}$$

Predicted (row) & Reference (col) Steps	$Step_{Ref_1}$	$Step_{Ref_2}$	$Step_{Ref_3}$	$Step_{Ref_4}$	$Step_{Ref_5}$	$Step_{Ref_6}$	$Step_{Ref_7}$
notify affected personnel	**71.07**	28.25	21.12	13.65	15.66	18.68	23.48
review lockout procedure	27.92	**88.57**	34.32	26.24	57.64	22.73	31.11
perform machine stop	21.54	34.57	**91.82**	27.47	32.80	29.90	31.82
isolate energy	14.73	26.48	28.42	**88.36**	54.95	53.37	20.12
lockout energy	17.39	57.65	28.83	53.52	**87.49**	46.95	22.79
dissipate energy	20.54	24.40	30.03	55.57	49.65	**86.75**	14.67
attempt restart	21.52	31.67	29.34	20.38	25.26	15.85	**88.43**

Table 9: Scaled cosine similarity matrix S_{ij} between predicted (row) and annotated reference (column) steps (%). Bold diagonal values indicate correct semantic correspondences between predicted and reference instructions.

Table 10: $Step_{Ref_j}$ to Annotation reference step description.

$Step_{Ref_j}$	**Annotation Step Description**
$Step_{Ref_1}$	notify employees
$Step_{Ref_2}$	review lockout procedure
$Step_{Ref_3}$	perform machine stop
$Step_{Ref_4}$	isolate energy
$Step_{Ref_5}$	lockout energy
$Step_{Ref_6}$	dissipate energy
$Step_{Ref_7}$	attempt restart

where $\mathbf{p}_i$ and $\mathbf{a}_j$ are step embeddings. Using $L = \min(n, m)$, we report:

$$\textbf{Diag}_{mean} = \frac{1}{L}\sum_{i=1}^{L} S_{ii}, \tag{3}$$

$$\textbf{AlignAcc} = \frac{1}{L}\sum_{i=1}^{L} I\left[\arg\max_j S_{ij} = i\right], \tag{4}$$

$$\textbf{OffDiag}_{mean} = \frac{1}{L(L-1)}\sum_{i\neq j} S_{ij}, \tag{5}$$

$$\textbf{MarginDiff}_{mean} = \frac{1}{L}\sum_{i=1}^{L}\left(S_{ii} - \max_{j\neq i} S_{ij}\right). \tag{6}$$

Table 9 provides an example to demonstrate the pair-wise cosine similarity matrix. Table 10 provides the annotation reference for the step descriptions in Table 9.

B.2 Human Review

We collect human judgments for (i) instruction extraction and (ii) node (slot) extraction using a structured Google Form that mirrors document structure (procedure block → ordered action steps) to gain qualitative assessment insights.

Labels. For each *Source Identification & Details* item and each *Action Step*, annotators select *CORRECT*, *PARTIAL*, *WRONG*, or *N-SURE*. A free-text field supports error notes. *PARTIAL* label is when some details are correct, but there is some incorrect information, e.g. correct but incomplete energy source identification.

Node (slot) inspection. Each step is shown with extracted slots (*Action*, *Target*, *Tool*, *Location*). Annotators apply the same labels while inspecting both the full instruction and its slots.

B.3 Extended Node Extraction Results

Table 11 reports node extraction **Precision**, **Recall**, and $\mathbf{F_1}$ by node type according to *Exact*, *Fuzzy*, and *Embedding* matching strategies.

B.4 Expert Safety Compliance Evaluation

We verify ontology-driven PKG safety compliance evaluation with a certified LOTO safety expert. Each predicted procedure is split into energy-specific blocks and shown with a short description and the ordered steps. The expert answers:

- *Correct energy source?* (Yes/Partially/No)
- *Instruction order correct?* (Yes/No)
- *Missing required steps?* (Yes/No)
- *Structural error severity* (None/Minor/Major)
- *Approve this block?* (Yes/Needs Revision/No)
- Optional *Comments*

We report accuracy for binary criteria (with *Partially* treated as "No" per experiment) and Pearson r / Spearman ρ.

Node Type	Exact Match			Fuzzy Match			Embedding Similarity			
	P	R	F1	P	R	F1	P	R	F1	AvgSim
Action	84.75	84.93	**84.84**	87.00	87.19	**87.10**	88.75	88.95	**88.85**	0.97
Target	20.00	20.00	20.00	22.00	22.00	22.00	89.50	89.50	**89.50**	0.93
Tool	58.29	58.00	58.15	58.29	58.00	58.15	99.50	99.00	**99.25**	0.97
Location	42.64	42.00	42.32	48.22	47.50	47.86	88.32	87.00	**87.66**	0.93

Table 11: Precision (P), Recall (R), and F_1 scores (in %) for node type extraction under exact, fuzzy, and embedding-based matching. The **Action** row reports the mean over *Action* and *Verification*. Embedding similarities use `google/embeddinggemma-300m` (Appendix C)

B.5 Multimodal Grounding

In this study, multimodal grounding links each extracted step, slot, or PKG entity to visual evidence in the same manual, such as diagram labels, table cells containing device identifiers, and visual cues that disambiguate components or tools. Grounding accuracy is evaluated with strict Acc@1, but industrial manuals exhibit substantial domain shift: component terminology, abbreviations, and device IDs are rare in general-purpose VLM pretraining data, and many referents appear as schematic symbols or short alphanumeric labels rather than natural images with descriptive captions. These properties make exact top-1 matching difficult, even when the retrieved evidence is still useful as a shortlist for expert verification.

To reflect the operational objective, we report **Acc@k** and evidence recall at small k and treat multimodal grounding as evidence support for downstream validation, where the main benefit is conservative flagging of unsupported target IDs or component types. We expect grounding to improve with domain-specific, machine-specific, and brand-specific visual inputs, e.g. by adapting the retriever on manufacturer manuals and diagrams using symbol-to-component inventories, device-ID lexicons, and weak supervision from document structure.

C Implementation

Technical implementation details are provided here for reproducibility. Our code will be made available after publication at the repository: github.com/pokarats/Onto-LOTO-PKEval.

C.1 Frameworks and Models

We list the main tools and models used in our pipeline; links point to reference implementations.

1. **Document parsing:** *Docling* (Auer et al., 2024) (github.com/docling-project/docling), *PyMuPDF* (McKie and Liu, 2025) (github.com/pymupdf/PyMuPDF).

2. **LLM (instruction extraction and PKG construction):** *Qwen3* distilled `qwen3:14b` (Yang et al., 2025) via Ollama (ollama.com/library/qwen3:14b), using default decoding settings.

3. **Similarity and embedding models:** `BLEURT-20` (Pu et al., 2021) for SMART (Amplayo et al., 2023) (github.com/google-research/bleurt), and `google/embeddinggemma-300m` (Vera et al., 2025) (huggingface.co/google/embeddinggemma-300m) for embedding-based PKG matching.

4. **Vision-Language Models (Cross-Modal Grounding):** `openai/clip-vit-large-patch14-336` (CLIP) (Radford et al., 2021) and `google/paligemma2-3b-mix-448` (Steiner et al., 2024) (both from Hugging Face).

5. **Vision-Language Models (PKG end-to-end Baselines):** `Qwen/Qwen3.5-4B` (Xu et al., 2026) and `google/gemma-4-E4B-it` (Team et al., 2026) (both from Hugging Face).

C.2 Hardware

All experiments run locally on a server with 6 NVIDIA GeForce RTX 2080 Ti (11GB VRAM each) GPUs. The software environment uses openSUSE Leap 15.5 and locally managed Python environments. All experiments involving LLMs/VLMs were performed locally without external API calls.

C.3 Runtime and Cost

All components are executed locally. Document parsing and ontology-only validation are CPU-bound and scale linearly with document length

and number of extracted steps. Embedding-based matching and cross-modal grounding add GPU compute but remain batchable and deterministic.

LLM-based stages (instruction extraction, slot extraction, and ontology-LLM auditing) dominate runtime because they require autoregressive decoding and scale with the number of processed pages/blocks and prompt length. Since we run `qwen3:14b` locally via Ollama, there is no per-call monetary cost; the primary cost is GPU time. For API-based deployments, cost would be token-driven, and ontology-LLM auditing would be the most expensive component due to embedding the ontology text in every prompt.

C.4 Prompting

We adopt a two-stage prompting design. In the first stage, the LLM extracts only executable procedural instructions from the parsed pages in a few-shot setting and returns them in a structured JSON format. In the second stage, conditioned on the extracted steps, the LLM performs slot-level semantic extraction (action, target/object, tool, location, verification) aligned with a subset of PKO classes, using concise label definitions and few-shot exemplars to standardize outputs. All prompts will be released with the accompanying code repository.

Perception, Layout, and Validation: Calibrated Confidence for Reliable Straight-Through Processing of Financial Documents

Yichao Jin, Yushuo Wang, Yuxuan Han, Kwan Ching Yee Sonia, Weiyang Song, Chiu Jin-Chun Kent, Wong Chong Hwee, Wong Tiong Kiat, Kenneth Zhu Ke, Jingyuan Zhao
{JinYichao, YushuoWang, YuxuanHan, SoniaKwan, WeiyangSong, KentChiu, ChongHweeWong, WongTK, KennethZhu, JingyuanZhao}@ocbc.com
OCBC, Singapore

Abstract

Straight-through processing (STP) on extracted key-value fields from financial documents without human review requires a calibrated probability together with a bounded guarantee on the residual error of the auto-approved tier. The emergence of modern Vision Language Models (VLMs) provides an out-of-the-box capability for extracting the key-values, but their verbalized confidence signals are unreliable and weakly track field correctness.

This paper introduces a decomposed confidence layer along three interpretable channels, including perception, layout, and validation. Together with a final conformal risk control, the score can be used for reliable STP of financial documents. The method is validated on three public datasets covering real invoices, synthetic invoices, and ad-buy forms, using two different VLM families (Qwen3.6-27B and Gemini-3.1-Flash-Lite). Our decomposed score consistently improves the separation of correct from incorrect extractions, substantially raising the AUROC from 0.54–0.74 for VLM verbalized signals to 0.90-0.99 with contributions from all three designed channels. Crucially for industrial deployment, this enables usable STP. The native VLM confidence signals could clear only 0.1%-7.0% of fields under risk control at a target error of $\leq 10\%$. In contrast, the proposed method auto-approves 49-72% of fields while holding the empirical error of the accepted tier at or below the target.

1 Introduction

Financial back-office operations, such as accounts payable and client onboarding, have to process large volumes of semi-structured documents (e.g., invoices and trade orders) under tight time constraints. Currently, in most financial institutions, manual verification of the extracted fields still dominates the per-document cost. Therefore, *Straight-through processing* (STP), which auto-approves the extractions with minimal human review, becomes the key driver of productivity and scalability.

In a financial setting, STP carries a stringent bar. In particular, a single wrongly auto-approved value (e.g., a transposed amount or a misread account number) is a critical financial and compliance error. Safe STP thus requires two things beyond a point extraction. First, a per-field probability that is *calibrated*. Second, a *distribution-free bound* on the residual error of the auto-approved tier. Together, these hold the error rate among accepted fields below an operator-chosen budget.

Modern vision-language models (VLMs) have largely solved the extraction step. Prompted with a page image and a field schema, they emit values directly and usually accurately, without task-specific training. What they lack is a reliable confidence signal. A VLM *can* verbalize a confidence score, but these are often saturated and over-confident. On our data, verbalized confidence separates correct from incorrect extractions with an AUROC of only 0.54–0.74 (i.e., at times worse than chance). Token log-probabilities, where exposed, sit over generated sub-word tokens and track field correctness only weakly. Even sampling-based self-consistency (Wang et al., 2022) is blind to *systematic* errors, where the model returns the same wrong value on every sample. Consequently, a VLM-only STP system is caught between auto-approving almost nothing and auto-approving errors.

We introduce a decomposed confidence layer for VLM key-value extraction, which scores each extracted field along three interpretable channels. The **Perception** channel checks if the evidence is legible from region-local image quality, VLM-OCR agreement, and value-string cues. The **Layout** channel asks if the value sits where this field normally appears. We retrieve historical layouts similar to the current page, and score the field's anchor-relative position under a retrieval-conditioned spatial prior. A value in an unusual location would be

The Proceedings of Workshop on Document Intelligence and Understanding, pages 232–243
October 24, 2026 ©2026 Association for Computational Linguistics

flagged even when read perfectly. Lastly, the **Validation** channel applies type, checksum, and rules, and checks cross-model agreement against a second independent VLM. A gradient-boosted model fuses the three channels into a single probability, with per-field reason codes. We wrap this score in a Learn-Then-Test conformal procedure (Angelopoulos et al., 2025), which certifies a distribution-free bound on the error rate of the auto-approved tier. Finally, when two VLMs disagree, we select the value our layer scores higher. This resolves the disagreement rather than rejecting it, raising STP throughput at the same guaranteed error.

We validate the method on three public datasets. They span real invoices from DocILE (Šimsa et al., 2023), synthetic invoices from FATURA (Limam et al., 2023), and advertising order forms from VRDU (Wang et al., 2023), using two VLM families, Qwen3.6-27B (Qwen Team, 2026) and Gemini-3.1-Flash-Lite (DeepMind, 2026). Our main contributions include

- A decomposed and interpretable confidence layer consisting of perception, layout, and validation channels. The fused conformal score reliably reflects the field correctness with a distribution-free error guarantee. Experiments show a substantial AUROC lift from 0.54–0.74 (VLM baseline) to 0.90–0.99.
- A cross-model verification-and-selection mechanism, that resolves disagreements between two VLMs, further lifts the guaranteed STP coverage on top of fusion of the three channels. The full layer auto-approves 49-72% of fields at a $\leq$ 10% target error, versus 0.1-7.0% from native signals.
- Per-channel and per-feature SHAP attributions that make each accept and reject decision auditable, as required for model-risk and governance review in a regulated setting.

2 Related Work

Document key-value extraction. Early field extractors coupled OCR with layout-aware encoders, such as the LayoutLM family (Xu et al., 2020, 2021; Huang et al., 2022) and OCR-free models like Donut (Kim et al., 2022). These require task-specific fine-tuning and predict grounded spans. Modern VLMs instead read a page image and emit field values directly, without training. However, the extracted values are often ungrounded without reliable boxes and confidence scores.

Our proposed layer is agnostic to the extractor, scoring the confidence of already-extracted values and recovering grounding from OCR spans.

Confidence and calibration for extraction. Confidence for extraction and generation has relied on token log-probabilities, verbalized confidence (Xiong et al., 2024), and sampling-based self-consistency (Wang et al., 2022). For VLM field extraction these signals are weak. The closest line fuses heterogeneous signals (i.e., log-probabilities, OCR quality, image quality, and coarse spatial features) into a single learned score and reports an aggregate AUROC (Kumar, 2026).

Our work differs in three ways. First, we add a retrieval-conditioned layout prior as a first-class signal, rather than coarse spatial features. Second, we keep the score interpretable through a per-channel decomposition with reason codes. Third, we target a distribution-free straight-through-processing guarantee, not only a ranking metric.

Conformal risk control and selective prediction. Selective classification abstains on low-confidence inputs (Geifman and El-Yaniv, 2017). Split conformal prediction turns any score into distribution-free coverage (Vovk et al., 2005). Learn-Then-Test (Angelopoulos et al., 2025) and conformal risk control (Angelopoulos et al., 2024) extend this to control general risks, including the selective error of an accepted set through multiple-hypothesis testing.

We use Learn-Then-Test to certify a distribution-free bound on the error rate of the auto-approved tier, class-conditional by field type. Distinctively, our retrieval score is both an input to the confidence model and the variable that indexes the calibration regime, linking template familiarity to the risk-control guarantee.

3 Problem Statement and Methodology

Figure 1 shows the end-to-end confidence layer. The input is a pdf document, where a VLM extracts field values directly with verbalized confidence score. In parallel, OCR yields word spans, and a retrieval step fetches historical layouts similar to the page. Then three channels examine the corresponding features to score each extracted value. In particular, the **perception** channel checks if the evidence is legible. The **layout** channel checks whether the value sits where the field normally ap-

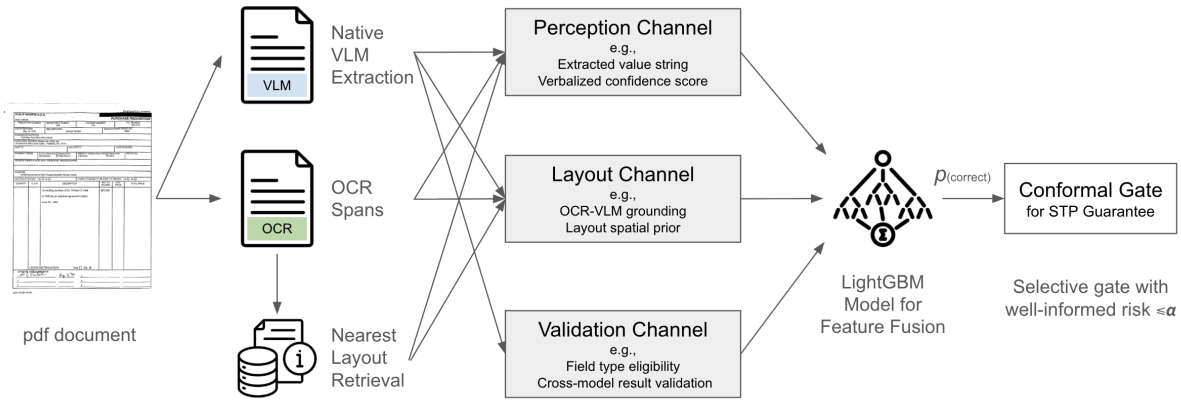


Figure 1: End-to-end confidence layer. The VLM extracts ungrounded field values from the page image. OCR spans recover grounding, and nearest-layout retrieval supplies similar historical templates. Three interpretable channels (i.e., perception, layout, and validation) score each extracted value. A gradient-boosted model (LightGBM) fuses them into a calibrated probability $p(\text{correct})$, which a conformal selective gate turns into an auto-approval decision at a guaranteed selective risk $\leq \alpha$.

pears. The **validation** channel applies content rules and cross-model agreement. Subsequently, a LightGBM model fuses the three channels into a single probability p_f. Finally, a conformal selective gate turns p_f into an auto-approval decision with a controlled error rate. We detail each component below.

3.1 Grounding recovery

A VLM reads a document image and a field schema, and returns a value v_f and its verbalized confidence score s_f for each field type f. Our layer assigns each extracted field (f, v_f, s_f) a calibrated probability p_f that the value is correct, using the page and its recovered OCR grounding.

We first recover grounding from OCR word spans, where each span is a candidate c with printed text $\text{text}(c)$ and a bounding box. The candidate set for field f collects the spans whose canonical text matches the extracted value

$$C_f = \{\, c : \text{canon}_f(\text{text}(c)) = \text{canon}_f(v_f) \,\}$$

where v_f is the VLM value and canon_f is a per-type canonicalizer which handles numeric tolerance, ISO-8601 dates, and normalized strings. $|C_f|{=}0$ means the value is not on the page (a strong error signal) and $|C_f|{>}1$ means the value is ambiguous. The details of canonicalizer design are illustrated in Appendix A.

3.2 Perception channel

The perception channel asks whether the evidence is legible and read correctly with 11 features. From the value string we take length, token count, digit ratio, and confusable-glyph mass (e.g., $0/O$, $1/l$, $5/S$). At the candidate box, we take region-local image quality (i.e., Laplacian variance, contrast, ink density, and glyph height) and the edit distance between v_f and the OCR text there. We also include the VLM's verbalized confidence.

3.3 Layout channel

The layout channel contributes 15 features and is illustrated end to end in Figure 2. It asks whether v_f sits where field f normally appears from documents with similar layout.

Anchor-relative geometry. We encode a candidate's position relative to a nearby label anchor rather than in absolute page coordinates. The anchor is chosen geometrically, not semantically. Among all OCR words on the page (excluding the value's own tokens), we take the nearest label-like word under a fixed preference order. A word on the same text line and to the left of the value (within $0.6h_v$ vertically, where h_v is the value height) is preferred, scored by its horizontal distance. So the closest same-row-left word wins. This captures the common "`Label: value`" layout. If not found, we fall back to a word directly above and roughly column-aligned (within $3h_v$ horizontally). This handles column-header layouts. The matching uses no keyword list or field-type dictionary, so it is robust to unseen templates. Given the anchor at center $center(a)$ with height h_a, the position feature is the offset normalized by anchor height as

$$z = \frac{\text{center}(v) - \text{center}(a)}{h_a}$$

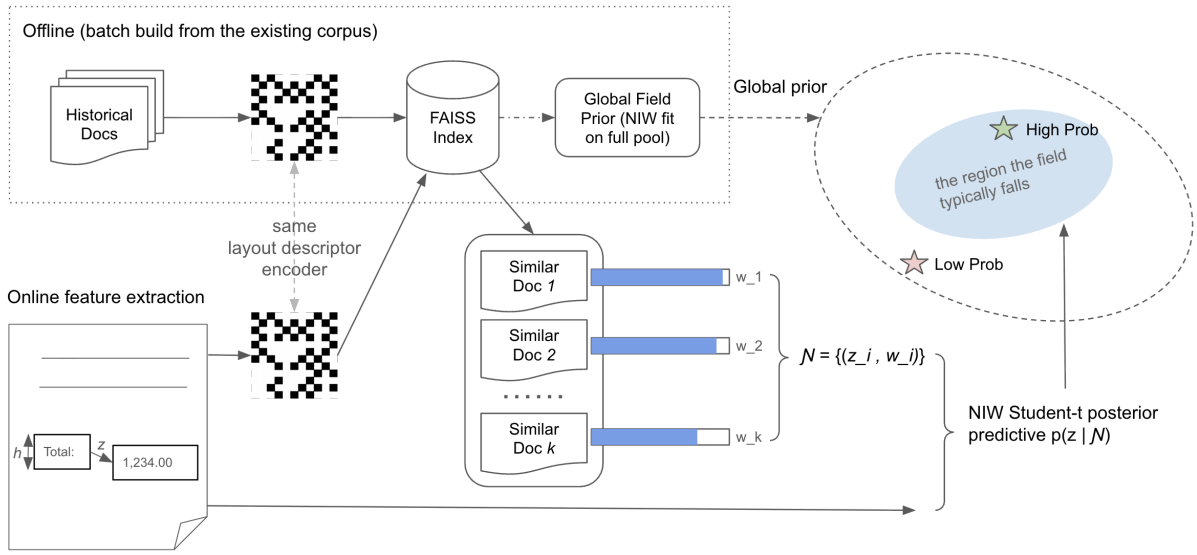


Figure 2: Layout-channel drill-down. **Offline** (top): each historical document is encoded by the layout-descriptor encoder ϕ into a FAISS index, and the field positions across the corpus are pooled to fit a global Normal–Inverse–Wishart (NIW) prior. **Online** (per field): (1) grounding gives the anchor-relative position z of the value (offset from its label anchor, normalized by anchor height h); (2) the query page is encoded by the *same* descriptor ϕ and retrieves the k nearest historical layouts, each contributing its field position and a similarity weight, $\mathcal{N} = \{(z_i, w_i)\}$; (3) The global prior updated with the similarity-weighted neighbours $\mathcal{N}$ yields a Student-t posterior predictive $p(z \mid \mathcal{N})$. The shaded blue region is where the field typically lands for this template (tight), while the dashed ellipse is the global prior it falls back to when no neighbours are found (wide). A candidate whose position lies in the blue region scores high, whereas one in the tail (Low Prob) is penalized even when read correctly.

z is invariant to translation and to scale across templates and page sizes. When no anchor qualifies, z falls back to the offset from the page centre.

Retrieval. We encode each page as a binary token-mask descriptor (details in Appendix D) and index the historical documents with FAISS (Douze et al., 2025). To inference a new page we retrieve the k nearest layouts with similarity weights $\{w_i\}_{i=1}^{k}$. We summarize familiarity by the top similarity $s_{\text{match}} = \max_i w_i$ and a soft effective neighbour count $k_{\text{eff}} = \left(\sum_i w_i\right)^2 / \sum_i w_i^2$.

Retrieval-conditioned spatial prior. We model z for each field type with a Normal–Inverse–Wishart (NIW) (Murphy, 2007) prior on its mean and covariance, fit on the global field-conditional pool. We update the prior with the retrieved neighbours' positions, weighted by similarity, and marginalize to a Student-t posterior predictive $p(z \mid \mathcal{N})$. A single shrinkage hyperparameter κ_0 governs how fast neighbours override the global prior. The behaviour is self-regularizing in familiarity. In particular, at $k_{\text{eff}}{=}0$ the posterior collapses to the global field prior (cold start); small k_{eff} yields heavy tails and appropriately low confidence; large k_{eff} concentrates around the template-specific location.

Non-circular scoring. A value can appear at several places on the page, so C_f may contain multiple positions z_c. We score the field's placement with the layout score s_L, the log-average posterior density over these occurrences as

$$s_L = \log \frac{1}{|C_f|} \sum_{c \in C_f} p(z_c \mid \mathcal{N}).$$

Averaging rather than taking the maximum avoids a selection bias. Selecting the maximum alone would make a value look correct merely because one of its occurrences happens to land in a high-density region. Instead, the spread across occurrences is surfaced separately as ambiguity signals, derived from the occurrence distribution $\pi_c \propto p(z_c \mid \mathcal{N})$ and its entropy H_f. We expose three instances of s_L into the feature set, including the retrieval-conditioned posterior `s_l_marg`, the global prior with retrieval off `s_l_cold` whose gap to the primary exposes the retrieval lift, and under an absolute-position prior `s_l_abs` as a weak baseline.

3.4 Validation channel

The validation channel applies content rules and contributes 8 features. It checks type and format validity, ISO-4217 currency and date validity, checksums where a field admits one (e.g. IBAN mod-

	DocILE		FATURA		VRDU	
Method	Qwen	Gemini	Qwen	Gemini	Qwen	Gemini
Self-consistency	0.728	0.660	0.605	0.544	0.629	0.560
VLM baseline	0.736	0.702	0.614	0.597	0.597	0.550
+ P/L/V fusion	0.911	0.908	0.990	0.996	0.921	0.934
+ cross-model	**0.915**		**0.997**		**0.928**	

Table 1: AUROC for predicting extraction correctness across four approaches, on three datasets and two VLM families. The cross-model row fuses both VLMs into one output (one value per dataset).

97, ISIN), and arithmetic residuals across amount fields (e.g., net + tax = gross). It also records cross-model agreement on whether a second, independent VLM returns the same canonical value.

3.5 Feature fusion model

A gradient-boosted model fuses the three channels into a single probability p_f. We use LightGBM over the concatenated per-channel features, learning correlated interactions. The model is tabular and yields SHAP attributions, so each accept/reject decision carries per-feature reason codes for informative model-risk review. The full feature list can be found in Appendix B.

3.6 Conformal selective gate

A calibrated probability is not yet a decision. The gate accepts a field iff $p_f \geq \lambda$, where λ is chosen on a held-out calibration set so that the *selective risk* (error rate among accepted fields) stays at or below an operator-chosen budget α. Learn-Then-Test (Angelopoulos et al., 2025) makes this threshold search valid, giving a distribution-free guarantee at confidence $1-\delta$. As a result, the auto-approved tier carries error at or below α, and everything below the threshold is routed to human review.

4 Experiments

4.1 Experimental setup

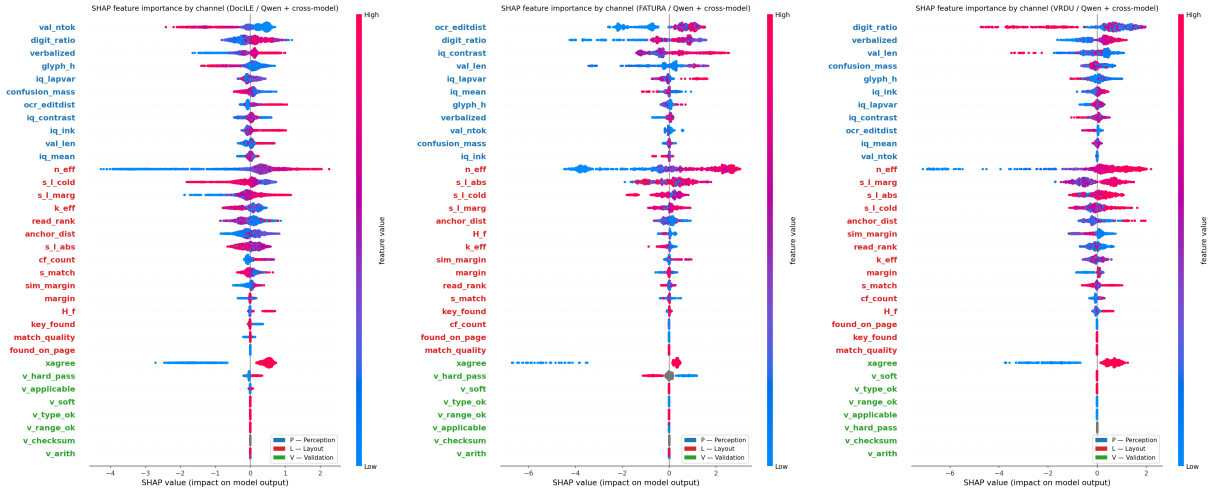
Datasets. We evaluate three public document collections spanning real invoices DocILE (Šimsa et al., 2023), synthetic invoices FATURA (Limam et al., 2023), and advertising order forms VRDU (Wang et al., 2023). Each contributes a subset of fields unambiguously under a shared taxonomy (19, 14, and 9 types respectively). The offline layout index is built for each dataset from the training split and is disjoint from the evaluation documents. The details are given in Appendix C.

VLM extractors. To test that the confidence layer is model-agnostic, we run every experiment with two independent VLM families, Qwen3.6-27B and Gemini-3.1-Flash-Lite. A value and its verbalized confidence for each field are returned from a single ungrounded prompt over the page image. We query each document K_q=5 times to estimate self-consistency and take the canonical value as the extraction. The full prompts can be found in Appendix E, and the detailed fusion model configurations can be found in Appendix F.

Baselines. We compare against the confidence signals from the VLM alone (i) *self-consistency*: the fraction of the $K_q = 5$ samples agreeing with the modal value; and (ii) a *VLM baseline*: the native verbalized confidence score. We report our full end-to-end pipeline based on feature engineering from three channels and the fusion layer.

4.2 Correctness discrimination

Table 1 reports AUROC for predicting whether an extracted value is correct. As baselines, the VLM's own confidence signals are weak: self-consistency and the VLM baseline reach only 0.54–0.74 across the six dataset–model settings. Notably, self-consistency is not stronger than the single-call VLM baseline despite costing K_q calls. This is because sampling agreement detects only stochastic errors, where resampling changes the answer, but is blind to systematic ones, where the same wrong value is returned every time. In practice, many VLM extraction errors are systematic, so high agreement reflects determinism, not correctness. In contrast, adding the P/L/V channels together with the fusion layer considerably raises AUROC to 0.91–0.99, which holds across all dataset and VLM settings, confirming the pipeline is largely model-agnostic. Moreover, cross-model agreement adds a further increment when both VLMs are available. The lift is largest on FATURA at 0.997, whose clean synthetic templates make layout and OCR cues highly reliable, and is consistent on the other two real-document collections at 0.91–0.93.



(a) DocILE (b) FATURA (c) VRDU

Figure 3: SHAP feature importance by channel (Qwen extractor + cross-model) on the three datasets. Labels are colored by channel (P blue, L red, V green). The channel emphasis shifts slightly with document type: FATURA is perception/OCR-driven, VRDU is layout-driven, DocILE is in between.

4.3 What the fusion model learns

Figure 3 shows SHAP attributions for the fused model, grouped and colored by channel. Three patterns hold across all datasets. First, the retrieval prior is the strongest signal. The effective-neighbour count `n_eff` has the widest impact in every panel, with a long low-value tail that correctly lowers confidence at cold start. And the retrieval-conditioned spatial scores (`s_l_marg`, `s_l_cold`) and anchor distance (`anchor_dist`) form a consistent supporting cluster. Second, cross-model agreement (`xagree`) is a clean and high-impact validator that agreement raises confidence and disagreement lowers it. Third, within perception the value-string and OCR cues do most of the work using token count (`val_ntok`), digit ratio (`digit_ratio`), glyph height (`glyph_h`), and VLM–OCR edit distance (`ocr_editdist`). While the VLM's own verbalized confidence ranks only mid-pack, consistent with its weakness as a standalone signal.

Aggregated by channel, layout, and perception dominate (49% and 39% of overall attribution), with validation contributing the rest. The channel emphasis shifts with document type. FATURA is perception/OCR-driven because of its clean and synthetic templates, VRDU is layout-driven, and DocILE is in between. The top signals are stable across all datasets and VLM families.

The interpretability of the layer is a deployment prerequisite in a regulated financial setting. An automated approval that cannot be explained is, in practice, often not acceptable. Model-risk management and audit functions require that every straight-through decision be traceable to a justifiable cause, where a monolithic neural confidence score is insufficient. Because our fusion model is tabular and its inputs are organized along the interpretable perception, layout, and validation channels, every accept/reject decision decomposes via SHAP into signed per-feature contributions that read as human-legible. A field would be routed to review because its value has little retrieval support (e.g., an unfamiliar template), or the two VLMs disagree, or it sits an unusual distance from its label anchor, or the VLM reading diverges from the OCR text. These inspections are exactly the artifacts a model validation or regulator review asks for, allowing a human reviewer to either trust or override the gate on a case-by-case basis. The confidence layer is therefore not merely a score but an *auditable control* in a compliance-bound environment.

4.4 Straight-through-processing coverage

Table 2 shows the deployment-facing result on the percentage of fields that can be auto-approved while holding the accepted-tier error at or below a target. This is the operative metric in the real regulatory setting where straight-through processing is precision-first. Specifically, a wrongly auto-approved value is a financial or compliance error, whereas a correct value sent to review is only a small efficiency cost. Consequently, the question is not average accuracy but how much can be auto-approved at a guaranteed precision.

Dataset	Method	≤5% err	≤10% err	≤20% err
DocILE	Self-consistency (Qwen/Gemini)	0.1 / 0.0	0.2 / 0.0	0.5 / 0.3
	VLM baseline (Qwen/Gemini)	0.3 / 0.1	1.6 / 0.1	12.2 / 0.6
	+ P/L/V fusion (Qwen/Gemini)	20.7 / 21.4	39.3 / 42.6	58.0 / 61.1
	+ cross-model	**28.1**	**49.4**	**66.5**
FATURA	Self-consistency (Qwen/Gemini)	0.1 / 0.2	0.1 / 0.2	0.2 / 0.5
	VLM baseline (Qwen/Gemini)	0.1 / 1.4	0.1 / 7.0	0.4 / 18.1
	+ P/L/V fusion (Qwen/Gemini)	62.4 / 66.6	66.3 / 70.5	74.8 / 79.5
	+ cross-model	**68.4**	**72.3**	**81.6**
VRDU	Self-consistency (Qwen/Gemini)	0.2 / 0.2	0.4 / 0.3	1.4 / 0.9
	VLM baseline (Qwen/Gemini)	1.5 / 0.1	2.9 / 0.1	10.4 / 0.2
	+ P/L/V fusion (Qwen/Gemini)	25.7 / 35.9	60.5 / 66.9	78.7 / 79.7
	+ cross-model	**41.3**	**70.8**	**83.8**

Table 2: Straight-through-processing coverage, in terms of the percentage of fields that can be auto-approved while holding the accepted-tier error at or below the target. Native VLM signals clear almost no fields, while the channel fusion and cross-model selection unlock up to 84% when $\leq 20\%$ error.

In particular, at a 10% error target ($\geq 90\%$ precision) self-consistency and the VLM baseline auto-approve 0.1–7.0% of fields. Whereas the P/L/V fusion auto-approves 39–71% at the same target, which is a one to two orders of magnitude increase, driven collaboratively by all three channels (Section 4.3). Moreover, cross-model selection adds a premium tier (49–72%, up to 84% at a 20% target), most visibly on VRDU, where resolving disagreements rather than rejecting them lifts coverage from 61–67% to 71%. The gains hold across both VLM families. Even DocILE, the real and heterogeneous invoices, reaches 49% auto-approval at $\geq 90\%$ precision from a near-zero baseline.

Appendix G gives qualitative case studies to make the STP concrete. In each case a VLM returns a wrong value with high verbalized confidence (≥ 0.90), yet our pipeline scores it low and cross-model selection recovers the correct value.

5 Conclusion

We presented a decomposed confidence layer for straight-through processing of VLM-extracted fields from financial documents. It scores each field along three interpretable channels (perception, layout, and validation) with a retrieval-conditioned layout prior. A LightGBM model fuses the channels into a calibrated probability with per-field reason codes and a Learn-Then-Test conformal gate turns it into an auto-approval decision with a distribution-free error bound. Across three public datasets and two VLM families, the layer raises correctness-discrimination AUROC from 0.54-0.74 for native VLM confidence to 0.90–0.99, and auto-approves 49–72% of fields at a 10% error target where native signals clear almost none. The gains hold across both VLMs showing its model-agnostic. A cross-model selection step resolves disagreements from two VLMs to further lift guaranteed throughput.

The limitations are mild rather than fundamental in target banking settings. First, the layout channel is weakest at cold start. For a genuinely novel template with no similar layouts in the index, the retrieval-conditioned prior collapses to the global field prior and adds little signal. However, in the real operation, the document stream is dominated by recurring templates (e.g., the same forms and counter-parties processed repeatedly). A truly new template is rare, and it degrades gracefully to other channels and, under the conformal gate, is routed to human review rather than mis-approved. Each such document then enters the index, so a new template becomes familiar after only a few occurrences. Second, fitting the fusion model and calibrating the gate require field-level correctness labels. However, financial institutions usually have large volumes of approved historical documents, providing enough supervision. Moreover, mandatory sample-checking of processed items (including auto-approved STP cases) yields a continual stream of audited field-level labels for periodic refresh. Finally, the rule validators add little marginal signal on the reported dataset, where errors are well-formed and no check-summable identifiers appear. Their value should grow on documents such as trade orders containing IBAN / ISIN / BIC.

References

Anastasios N Angelopoulos, Rina Foygel Barber, and Stephen Bates. 2024. Theoretical foundations of con-

formal prediction. *arXiv preprint arXiv:2411.11824*.

Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. 2025. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662.

Google DeepMind. 2026. Gemini 3.1 flash-lite: Model card.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *IEEE Transactions on Big Data*.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*, pages 4083–4091.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer.

Nitesh Kumar. 2026. Beyond logprobs: A multi-signal confidence engine for llm-based document field extraction. *arXiv preprint arXiv:2606.24420*.

Mahmoud Limam, Marwa Dhiaf, and Yousri Kessentini. 2023. Fatura: A multi-layout invoice image dataset for document analysis and understanding. *arXiv preprint arXiv:2311.11856*.

Kevin P Murphy. 2007. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16.

Qwen Team. 2026. Qwen3.6-27B: Flagship-level coding in a 27b dense model.

Štěpán Šimsa, Milan Šulc, Michal Uřičář, Yash Patel, Ahmed Hamdi, Matěj Kocián, Matyáš Skalickỳ, Jiří Matas, Antoine Doucet, Mickaël Coustaty, and 1 others. 2023. Docile benchmark for document information localization and extraction. In *International Conference on Document Analysis and Recognition*, pages 147–166. Springer.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5184–5193.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations*, volume 2024, pages 23650–23678.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, and 1 others. 2021. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 2579–2591.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1192–1200.

A Canonicalization Rules

The canonicalizer canon_f defines both candidate matching and the ground-truth correctness label in section 3. An extraction $\hat{v}$ is correct iff it matches the gold value $v^\star$ under the per-category comparison below. We fix the ruleset at version `canon-v2` and treat it as a versioned artifact.

Field categorization. Each field type is routed to one category below by substring rules on its name, tested in this order: date -> number -> identification. All others default to string.

Dates. We avoid full date parsing. We extract every run of digits and form the set of numeric components. A two-digit value $v \leq 99$ contributes both itself and its century expansion ($2000{+}v$ if $v < 50$, else $1900{+}v$). Month abbreviations (`JAN-DEC`) contribute their month index. Two dates match if their component sets overlap *and* either one set is a subset of the other or they share at least two components. This order-agnostic rule is robust across the many formats in the data such as `3/4/2020`, `06/07/99`, `DEC31/01`, `JUL20/02`.

Numbers. We strip percent signs and any character outside {`0-9`, `.`, `,`, `-`}. The decimal separator is taken to be the rightmost `.` or `,` followed by one or two digits, all other separators are removed. The remainder is parsed as a signed float. Two numbers

Feature	Description
Perception Channel — 11 features	
verbalized	*self-report.* VLM verbalized confidence s_f, rescaled to $[0, 1]$.
iq_lapvar	*pixels.* Laplacian variance of the value crop (sharpness; low = blurry).
iq_contrast	*pixels.* Pixel contrast of the value crop.
iq_ink	*pixels.* Ink density (dark-pixel fraction) of the crop.
iq_mean	*pixels.* Mean gray level of the crop.
glyph_h	*pixels.* OCR text-box height in pixels (small = tiny/low-resolution print).
ocr_editdist	*OCR.* $1-$ string similarity of v_f to the OCR text at the location (VLM–OCR disagreement).
val_len	*string.* Character length of the value string.
val_ntok	*string.* Whitespace-token count of the value string.
confusion_mass	*string.* Fraction of chars in the confusable-glyph set ($0/O$, $1/l/I$, $5/S$, $8/B$, $./,$).
digit_ratio	*string.* Fraction of value characters that are digits.
Layout Channel — 15 features	
found_on_page	*grounding.* 1 if the value is grounded on the page ($\|C_f\| > 0$), else 0.
cf_count	*grounding.* Number of candidate occurrences $\|C_f\|$.
key_found	*grounding.* 1 if a label anchor sits beside the value.
match_quality	*grounding.* Token coverage of the value by its OCR span (strings). 1 for exact-canon id/date/number.
s_l_marg	*spatial.* Marginal (log-mean-exp over occurrences) spatial log-likelihood — the core score s_L.
s_l_cold	*spatial.* Cold-start spatial score: global field prior with retrieval off.
s_l_abs	*spatial.* Spatial score from absolute page position (weak baseline).
anchor_dist	*geometry.* Distance to the nearest label anchor z.
read_rank	*geometry.* Vertical reading-order position of the value.
s_match	*retrieval.* Top retrieval similarity to a historical layout (template familiarity).
sim_margin	*retrieval.* Top-1 minus top-2 retrieval similarity.
k_eff	*retrieval.* Soft effective neighbour count from the retrieval weights.
n_eff	*retrieval.* Effective neighbours contributing to the spatial prior.
H_f	*ambiguity.* Entropy of the occurrence distribution π_c (slot ambiguity).
margin	*ambiguity.* Top-two margin of π_c (large = unambiguous).
Validation Channel — 8 features	
v_type_ok	*rule.* 1 if the value has valid type/format for the field.
v_range_ok	*rule.* 1 if the value lies in a plausible range.
v_soft	*rule.* Soft aggregate v_type_ok $\times$ v_range_ok (always defined).
v_checksum	*rule.* Checksum result where the field admits one (IBAN, ISIN); NaN otherwise.
v_arith	*rule.* Arithmetic consistency (net + tax = gross; items vs. total); NaN if inapplicable.
v_applicable	*rule.* 1 if any hard validator (checksum or arithmetic) applies.
v_hard_pass	*rule.* 1 if all applicable hard validators pass; NaN if none apply.
xagree	*cross-model.* 1 if a second, independent VLM returns the same canonical value, else 0.

Table 3: Feature dictionary for the fusion model, grouped by channel. Each entry leads with its sub-type (in italics). NaN rule features indicate the validator does not apply to that field (e.g. no check-summable identifier on an invoice).

match under a strict *absolute* tolerance of half a cent, $|a - b| \leq 0.005$.

Identifiers and Strings. We strip accents (Unicode NFKD), lowercase, and remove all non-alphanumeric characters. Two identifiers match on exact equality of this normalized form.

B Feature Dictionary

The fusion model consumes 34 features across the three channels. Table 3 lists all the features and the meanings. Features are computed at the best-scoring occurrence (Section 3). When the value is not grounded ($|C_f|$=0), layout and pixel features fall back to fixed defaults and only the value-string and self-report features remain informative.

C Dataset Statistics

Table 4 reports the statistics for each dataset and VLM. The *offline layout index* is built once per dataset from its training split (*Indexed Layouts*), and is disjoint from the *evaluation* documents (*Eval*). The reporting unit is a field extraction (one row per (document, field)). Row counts differ slightly between the two VLMs on the same documents because grounding recovery depends on what each model extracted. DocILE uses its official validation split with template-familiarity buckets from train-cluster recurrence. FATURA and VRDU use 200-document evaluation samples.

The three datasets are evaluated under a shared field taxonomy, with each dataset contributing the subset of field types it annotates unambiguously.

Dataset	Indexed Layouts	Eval Doc	Field Types	VLM	Fields	Correct	Grounded
DocILE	5,180	500	19	Qwen3.6-27B	5,141	51.7%	85.4%
				Gemini-3.1-Flash-Lite	5,077	53.8%	86.8%
FATURA	731	200	14	Qwen3.6-27B	1,850	60.2%	95.0%
				Gemini-3.1-Flash-Lite	1,767	63.9%	98.4%
VRDU	417	200	9	Qwen3.6-27B	1,698	63.3%	97.9%
				Gemini-3.1-Flash-Lite	1,666	63.7%	98.0%

Table 4: Dataset statistics. *Indexed Layouts* shows the unique number of training documents in the offline index. *Eval Doc* is the number of evaluation documents. *Types* is the number of distinct field types. *Fields* is the number of labeled field extractions. *Correct* is the fraction whose canonical value matches gold. *Grounded* is the fraction recovered on the page ($|C_f| > 0$). Note that the two VLMs extract slightly different fields.

DocILE (Šimsa et al., 2023) uses the full set of 19 single-value field types, including

- identifiers (document_id, order_id, customer_id)
- dates (date_issue, date_due)
- monetary amounts (amount_total_gross, amount_total_net, amount_due, currency_code_amount_due, amount_paid, amount_total_tax)
- vendor fields (vendor_name, vendor_address, vendor_tax_id, vendor_email)
- customer fields (customer_billing_name, customer_billing_address)
- payment fields (payment_terms, payment_reference).

FATURA (Limam et al., 2023) contains synthetic invoices annotating 24 classes. We map them into 14 with an unambiguous single-value analogue onto the shared taxonomy, by collapsing rate-specific tax classes and duplicate invoice-number classes. Layout regions (e.g., "table", "logo") and free-text blocks (e.g., "note", "payment_details") have no clean field analogue and thus are excluded.

VRDU (Wang et al., 2023) contains real advertising order forms from a different domain. So its 9 fields are mapped by role onto the shared taxonomy. In particular, the "contract number" to "document_id", the "product/estimate number" to "order_id", the "campaign flight start and end" to "date_issue" and "date_due", the "gross total" to "amount_total_gross", the "TV station" to "vendor_name/address", the advertiser to "customer_billing_name", and the "agency" to "customer_other_name".

D Layout Descriptor

The layout descriptor ϕ encodes a page's coarse text-occupancy geometry, independent of the text content. We overlay a $G \times G$ grid (G=32) on the page in normalized coordinates and mark every cell that any OCR token box overlaps, giving a binary occupancy mask; the mask is flattened to a G^2=1024-dimensional vector and L2-normalized. Two pages are compared by the inner product of their descriptors, which equals cosine similarity after normalization, so the FAISS inner-product index returns the layouts whose text falls in the same places. One descriptor is computed per document from all its OCR words (across pages).

The descriptor is deliberately coarse and content-blind. It fingerprints *where* fields sit on the page, not what they say. This is exactly the invariance retrieval needs. Two invoices from the same vendor template map to nearby descriptors regardless of their amounts, dates, or names, so the spatial prior for a field can borrow strength across all documents that share its layout. A cell is set whenever a token box overlaps it, with an inclusive end cell, so a text line thinner than one grid cell still marks its cell rather than being dropped by an exclusive slice. Listing 1 gives the details of the implementation.

Listing 1: Layout descriptor ϕ is a binary $G \times G$ OCR-token occupancy mask, flattened and L2-normalized. Word boxes are in normalized page coordinates.

```
def token_mask_descriptor(words, grid=32):
    """Binary grid x grid OCR-token occupancy
     mask, L2-normalized."""
    m = np.zeros((grid, grid), np.float32)
    for w in words:
        l, t, r, b = w.bbox
        l = min(max(l, 0.0), 1.0)
        r = min(max(r, 0.0), 1.0)
        t = min(max(t, 0.0), 1.0)
        b = min(max(b, 0.0), 1.0)
        x0 = min(int(l * grid), grid - 1)
        x1 = min(int(r * grid), grid - 1)
        y0 = min(int(t * grid), grid - 1)
        y1 = min(int(b * grid), grid - 1)
        # cells the token overlaps
        m[y0:y1 + 1, x0:x1 + 1] = 1.0

    # flatten to grid * grid
```

```
    v = m.reshape(-1)
    n = np.linalg.norm(v)
    if n > 0:
        v = v / n
    return v.astype(np.float32)
```

At query time, a descriptor is issued against the index to retrieve the $K_{\mathrm{NN}}{=}50$ most similar layouts. Their similarities become the retrieval weights that condition the spatial prior. The same encoder is used offline to build the index and online to form the query, so familiarity is measured consistently.

E VLM Extraction Prompt

Each document is extracted in a single user message: the page image followed by the following instruction. The model returns strict JSON with a value and verbalized confidence per present field, omitting absent ones. The same template is used for both VLMs. Only a per-family "thinking" toggle differs (Qwen has `enable_thinking=false` and Gemini has `reasoning_effort=none`). The fields listed below are the DocILE schema. FATURA and VRDU use the same template with their own mapped fields. The amount rules are appended to resolve the common net/gross/due confusion.

Listing 2: The VLM extraction prompt with DocILE field schema as the example.

```
Extract the following fields from this business
   document image.
Return STRICT JSON only, no prose. For each
  field that is PRESENT on the page, give an
  object {"value": <exact text as printed>, "
  confidence": <integer 0-100>}. OMIT fields
  that are absent.
Fields:
  - document_id: the invoice/document number
  - order_id: the purchase order number
  - date_issue: the issue/invoice date
  - date_due: the payment due date
  - amount_total_gross: grand total INCLUDING
  tax (equals net + tax)
  - amount_total_net: subtotal BEFORE tax (net
  of tax)
  - amount_total_tax: the TAX/VAT amount only (
  not a total)
  - amount_due: amount actually payable now (
  often equals gross)
  - amount_paid: amount already paid
  - currency_code_amount_due: currency symbol
  or ISO code
  - vendor_name: the seller/vendor company name
  - vendor_address: the full vendor postal
  address
  - vendor_tax_id: vendor tax / VAT id
  - vendor_email: vendor email address
  - customer_billing_name: the buyer/customer
  billing name
  - customer_billing_address: the full customer
   billing address
  - customer_id: the customer account id
  - payment_terms: payment terms (e.g. NET 30)
  - payment_reference: payment reference /
  variable symbol
Respond as: {"document_id": {"value": "...", "
  confidence": 90}, ...}

Amount rules: net = subtotal before tax; tax =
  the tax amount only; gross = grand total
  including tax (gross = net + tax); due =
  amount payable now. If the page shows a
  single total, use it for BOTH gross and due.
  Report each amount exactly as printed.
```

Hyperparameter	Value
objective	binary log-loss
boosting	gbdt
n_estimators	200
learning_rate	0.08
num_leaves	15
reg_lambda L2 regularization	1.0
min_child_samples	20
random_state	0

Table 5: LightGBM fusion-model configuration.

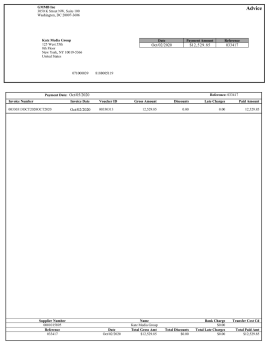
Sampling. Each document is queried $K_q{=}5$ times. The first sample is greedy (temperature 0) and fixes the reported value, the remaining four are sampled (temperature 0.7) to estimate self-consistency. The reported value is the modal canonical value across samples, the verbalized confidence s_f is the mean self-reported confidence, and the sampling-agreement fraction is the self-consistency baseline. The deployed operating point is the single greedy call. The additional samples are used only to compute the self-consistency baseline.

F Fusion model configuration

The fusion model is a gradient-boosted decision tree (`LightGBM.LGBMClassifier`) with the binary log-loss objective. Table 5 lists the full configuration. Regularization is deliberately conservative for the small tabular problem ($\sim$800–2600 training rows per split), shallow trees (15 leaves), an L2 penalty, a minimum of 20 samples per leaf and no row or feature subsampling. The model is retrained from scratch on each document-level split.

G Qualitative Case Studies

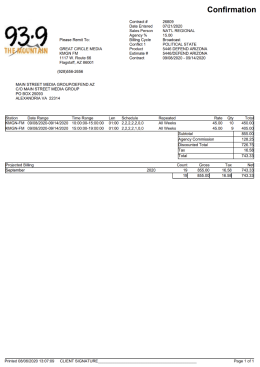
We show four cases where one VLM extracts a wrong value with high verbalized confidence, but the confidence layer assigns it a low probability. In every case the wrong and right values have nearly indistinguishable verbalized confidence (0.90–0.998), yet the fused score separates



Customer_Billing_Name

	Gemini-3.1-Flash	QWen3.6-27B
VLM-extraction	GMMB INC	Katz Media Group
VLM-score	0.988 ✗	0.950 ✓
Our score	0.145 ✓	0.953 ✓
Ground Truth	Katz Media Group	

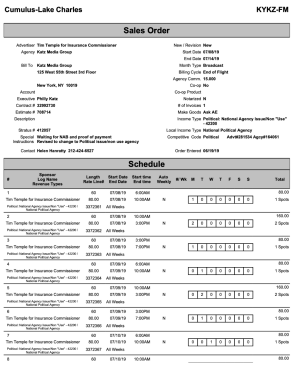
(a) Sender vs. recipient confusion (`customer_billing_name`): fused 0.145 vs. verbalized 0.953 on Gemini result.



Date_Issue

	Gemini-3.1-Flash	QWen3.6-27B
VLM-extraction	08/06/2020	07/21/2020
VLM-score	0.995 ✗	0.940 ✓
Our score	0.083 ✓	0.943 ✓
Ground Truth	07/21/2020	

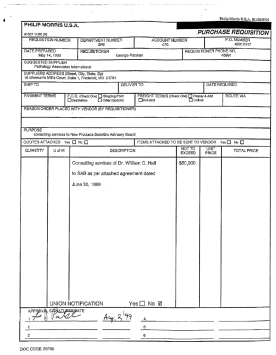
(b) Print-timestamp vs. issue date (`date_issue`): fused 0.083 vs. verbalized 0.943 on Gemini result.



Vendor_Name

	Gemini-3.1-Flash	QWen3.6-27B
VLM-extraction	Cumulus-Lake Charles	Cumulus-Lake Charles KYKZ-FM
VLM-score	0.986 ✓	0.930 ✗
Our score	0.954 ✓	0.178 ✓
Ground Truth	Cumulus-Lake Charles	

(c) Call-sign over-inclusion (`vendor_name`): fused 0.178 vs. verbalized 0.954 on Qwen result.



Vendor_Name

	Gemini-3.1-Flash	QWen3.6-27B
VLM-extraction	PHILIP MORRIS U.S.A.	George Patskan
VLM-score	0.998 ✓	0.900 ✗
Our score	0.803 ✓	0.351 ✓
Ground Truth	PHILIP MORRIS U.S.A.	

(d) Person vs. company (`vendor_name`): fused 0.351 vs. verbalized 0.803 on Qwen result.

Figure 4: Qualitative case studies. In each, one VLM is confidently wrong (verbalized ≥ 0.90) and the other is correct. Our fused score separates them (wrong value listed first). The confidently-wrong extraction is assigned a low fused probability in every case, while the correct value scores high.

them cleanly. This is the qualitative counterpart of the aggregate AUROC gap. It also shows why cross-model selection works by keeping the value our layer scores higher to recover the correct answer in all four cases. Each case illustrates a different error mode.

Case 1. Sender vs. recipient confusion in Figure 4a on `customer_billing_name` field. On a payment-advice document, Gemini returns the paying party "GMMB Inc" (verbalized 0.988) instead of the billed customer "Katz Media Group" (gold). Qwen returns the correct value (0.950). Verbalized confidence cannot tell them apart, but our layer scores Gemini's value 0.145 and Qwen's 0.953: the sender's name sits in the letterhead block, where a billing-name field rarely lands.

Case 2. Wrong date field as shown in Figure 4b on `date_issue` extraction. On a broadcast confirmation, Gemini returns "08/06/2020" (verbalized 0.995), which is the footer *print* timestamp, not the "Date Entered" issue date "07/21/2020" (gold). The reading is crisp and unambiguous, so perception signals do not help; the layout channel penalizes the value because a footer position is far from where an issue date normally appears, scoring it 0.083 against Qwen's 0.943.

Case 3. Call-sign over-inclusion in Figure 4c on `vendor_name`. Qwen returns "Cumulus-Lake Charles KYKZ-FM" concatenating the station call sign onto the vendor name (verbalized 0.930), while gold is "Cumulus-Lake Charles" aligned by Gemini. Our layer scores Qwen's over-inclusive value 0.178 and Gemini's value 0.954, driven by the span/OCR-coverage and value-string cues.

Case 4. Person vs. company in Figure 4d on `vendor_name`. On a purchase requisition, Qwen returns the requisitioner person name "George Patskan" (verbalized 0.900) instead of the vendor company "PHILIP MORRIS U.S.A." (gold). Our layer scores Qwen's 0.351 and Gemini's 0.803, where the requisitioner name sits in an internal form field, not where a vendor name is expected.

Tabstructor: Reconstructing Latent Borders in Borderless and Partially Ruled Tables

Alvin Sarraga Alon[1,*], **Daniele De Rosa**[2],
Johannes Friedl[3], **Riza Batista-Navarro**[1]

[1]Department of Computer Science, University of Manchester, United Kingdom
[2]Department of Agriculture, Forestry, Food and Environmental Sciences,
University of Basilicata, Potenza, Italy
[3]University of Natural Resources and Life Sciences, Vienna, Austria

***Correspondence:** alvin.alon@manchester.ac.uk

Abstract

Borderless and partially ruled scientific tables often omit visual boundaries that mark logical rows, columns, and spanning regions. We introduce Tabstructor, a vision–language framework for latent-border reconstruction: predicting a table bounding box and horizontal and vertical segments representing logical boundaries even when visually absent or incomplete. On a held-out external test set, we compare task-specific prompting of a frozen base model, supervised fine-tuning, structure-preserving augmentation, and optional OCR-assisted geometry refinement. We introduce the Latent Border Reconstruction Score (LBRS; 0–100, higher is better) for evaluating valid structural and geometric border recovery. Task-specific prompting improves the frozen model, while fine-tuning provides the dominant gain, increasing LBRS from 38.552 to 90.014 (paired gain 51.462; 95% paired-bootstrap CI: 45.583–57.259). The augmented predictor reaches 93.487, a further 3.473-point gain, although it receives additional cumulative training exposure. OCR-assisted refinement improves boundary localisation and extent but adds only 0.467 points on average (95% CI: -1.968–2.615) and can cause boundary-count regressions. Thus, fine-tuning performs the principal reconstruction, while refinement is an optional, non-uniform geometry adjustment.

1 Introduction

Many scientific tables express rows, columns, grouped headers, and spanning cells through whitespace, alignment, typography, and partial rules rather than a complete grid. In complex layouts with multi-level headers, merged cells, stub columns, partial separators, or local panels, some logical boundaries are not visibly drawn, complicating recovery of row, column, and logical-cell membership from a table image.

Table structure recognition (TSR) already recovers cells, relations, separators, or markup from document images (Khang and Hong, 2025; Xue et al., 2021; Lin et al., 2022). Representative TSR systems instead predict markup or layout pointers, cell locations and relations, or row and column separation lines and masks (Khang and Hong, 2025; Xue et al., 2021; Lin et al., 2022; Zhang et al., 2024). We examine a narrower representational gap by making the count, position, and bounded extent of latent structural borders an explicit and editable prediction target.

We call this task **latent-border reconstruction**. Given a single-table image, the objective is to predict its boundary and the horizontal and vertical segments supported by the logical table organisation, including boundaries that are only partially drawn or absent from the visible ink. The target therefore requires inference from layout and context rather than only visible-line detection. It provides geometric evidence for subsequent row, column, and logical-cell recovery, but does not constitute complete semantic table understanding.

We introduce **Tabstructor**, a vision–language framework for predicting this latent-border geometry. Throughout this paper, *Tabstructor* denotes the overall task framework; S3–S4 are its learned predictors, S4 is the proposed primary predictor, and S5 is the separately reported refinement analysis. A vision–language model (VLM) is a plausible basis for the task because missing boundaries must be inferred jointly from visual layout and textual context (Zhang, 2024; Alonso et al., 2026; Chen et al., 2026); however, general multimodal

The Proceedings of Workshop on Document Intelligence and Understanding, pages 244–272
October 24, 2026 ©2026 Association for Computational Linguistics

capability does not establish reliable fine-grained geometric prediction. This limitation motivates a controlled evaluation of task-specific model adaptation. We treat structure-preserving augmentation as a distinct training condition and OCR-assisted geometry refinement as an optional downstream analysis, rather than as part of the primary learned reconstruction.

The study addresses three research questions: **RQ1** asks how task-specific prompting and fine-tuning affect latent-border reconstruction performance; **RQ2** asks whether the evaluated structure-preserving augmented training condition improves reconstruction, while recognising its additional cumulative training exposure; and **RQ3** asks whether OCR-assisted geometry refinement adds meaningful benefit beyond the raw fine-tuned prediction.

The controlled comparisons distinguish gains from learned reconstruction from changes due to the optional downstream refinement stage.

The contributions of this work are threefold:

1. **Latent-border task and resources.** We formulate latent-border reconstruction as explicit prediction of normalised table bounds and logically supported border segments, and provide an annotation protocol and reviewed completed-border references constructed with TeybolStruct, our human-in-the-loop editor for creating and reviewing table bounding boxes and logical horizontal and vertical border segments.

2. **Controlled VLM evaluation.** We define five frozen conditions that support paired, held-out comparisons of model adaptation, training data, and downstream processing, and introduce the Latent Border Reconstruction Score (LBRS) as a bounded measure of valid structural and geometric border recovery.

3. **Reconstruction and refinement evidence.** We provide component-level and per-table analyses that separate primary learned predictions from optional geometry adjustments.

2 Related Work

Table analysis commonly separates table-region detection from table structure recognition (TSR), which recovers rows, columns, cells, spans, relations, or markup (Prajapati and Maurya, 2026). Direct formulations predict cell relationships or serialised layouts (Xue et al., 2021; Khang and Hong, 2025), whereas separator-based formulations infer row and column divisions from image evidence (Lin et al., 2022; Zhang et al., 2024). Both families recover table structure; the distinction relevant here is representational. Latent-border reconstruction makes the count, position, and bounded extent of inferred structural boundaries an explicit prediction target, rather than claiming that prior TSR does not infer missing structure.

Borderless and partially ruled tables are difficult because whitespace, alignment, typography, and local rules must jointly indicate divisions that a complete rendered grid would otherwise provide. Prior work addresses complex layouts through feature aggregation, spanning-cell recovery, and learned structure prediction (Hong et al., 2024; Nguyen et al., 2023; Yu et al., 2025). These approaches establish that structure can be inferred without tracing only visible rules. A remaining representational issue arises when a logical boundary is absent or valid over only part of the table: a full-width separator can split a spanning cell, while a single boundary plane may require disjoint segments. The present task targets these latent logical segments explicitly, including their local extents, while remaining complementary to cell-, relation-, and markup-based TSR.

Multimodal models have also been developed for document understanding and reasoning (Zhang, 2024), visual table understanding (Alonso et al., 2026), and instruction-guided TSR (Chen et al., 2026). Generative interfaces support constrained outputs such as JSON or markup, but prior success at document or table understanding does not by itself establish reliable fine-grained border geometry. This study therefore examines whether task-specific prompting is sufficient for latent-border reconstruction and how much task-specific fine-tuning changes structured geometric prediction; the base VLM is an enabling component rather than the claimed novelty.

Geometric priors and layout-aware optimisation have been used to improve row and column localisation in table structure recognition (Hashmi et al., 2021). Motivated by the broader value of geometric constraints, we separately evaluate whether inference-time refinement can improve the raw learned prediction. Strong regularisation may nevertheless be mismatched to irregular or partial boundaries, so this secondary refinement analysis does not access reference borders. Overall, prior TSR recovers complex structure through several effective representations, while VLMs provide a

flexible structured-generation interface; the open space examined here is explicit reconstruction of bounded latent borders in borderless and partially ruled scientific tables, together with the relative roles of prompting, fine-tuning, and downstream refinement.

3 Task Definition and Data

3.1 Latent-Border Reconstruction

Let I denote a single-table image of a borderless or partially ruled table. Locating a table among multiple page regions is outside the task; the predicted table box localises the table content within the supplied image. The target is the structured geometric representation

$$Y = (B, H, V), \tag{1}$$

where B is the table bounding box and H and V are sets of horizontal and vertical border segments. Box coordinates are normalised to the image and segment coordinates to B. The model emits these components as structured JSON; Appendix C specifies the exact schema, coordinate ordering, and transforms.

We call a boundary *latent* when it is supported by the table's adjudicated row, column, or spanning structure rather than merely traced from visible ink. The *completed-border target* contains all such supported boundaries: it retains visible structural rules and adds missing portions or wholly implicit separators, while excluding decorative rules. A boundary may comprise disjoint collinear segments when a full-span separator would cross a grouped or spanning region. Figure 1 illustrates the resulting image-to-geometry task. The direct prediction is only this geometry; OCR transcription, cell identities and relations, semantic labels, HTML generation, and subsequent geometry refinement are outside the target.

3.2 Target Annotation Policy

Every target contains a closed outer rectangle. Within it, annotators retain the minimum set of full-span or bounded separators needed to express the adjudicated logical partition. Separators are interrupted where they would divide grouped headers, spanning cells, or panel-local regions, and visible decorative rules are excluded. When multiple geometries are plausible, the simplest topology supported by row and column alignment, spans, and local structure is selected. Thus, each annotation is one adjudicated geometric interpretation rather than a claim that the layout has a unique semantic analysis.

One primary annotator created the targets with **TeybolStruct**, whose automated initialisation remained subject to manual correction. Every serialised-target rendering was reviewed before acceptance; unresolved cases received collaborative review and, when necessary, team adjudication. Labels also passed schema and coordinate round-trip validation. Because labels were not independently duplicated, formal inter-annotator agreement was not measured. The resulting annotations serve as training targets and as validation and test references; Appendix C gives the full protocol.

3.3 Dataset and Splits

The dataset comprises single-table images derived from published scientific literature, including borderless and partially ruled layouts, paired with one completed-border annotation per image. The development corpus contains 757 original training samples and 84 unaugmented validation samples. The augmented training condition adds 754 accepted variants derived only from the original training images; each variant preserves its source image dimensions and structural target (Table 1).

All 84 validation samples are used for token-loss monitoring, while a fixed 32-sample subset of them—not a third data split—is used for structural checkpoint selection. Validation data do not update model parameters, and the remaining 52 samples contribute only to loss monitoring, not structural checkpoint ranking. For the labels used below, S1–S2 are frozen prompting baselines, S3–S4 are fine-tuned predictors, S4 is the primary augmented Tabstructor predictor, and S5 refines S4. The separately annotated 100-table test set is external to the development corpus, uses the same representation, annotation policy, and review procedure, and is reserved for final evaluation; it is not used for training, loss monitoring, or checkpoint selection.

Ancestry checks found no overlap between original or augmented training records and validation records. Augmented records exactly retained their source targets, and test references were finalised without access to system predictions or scores. Appendices D and G provide the augmentation-integrity and selection details.

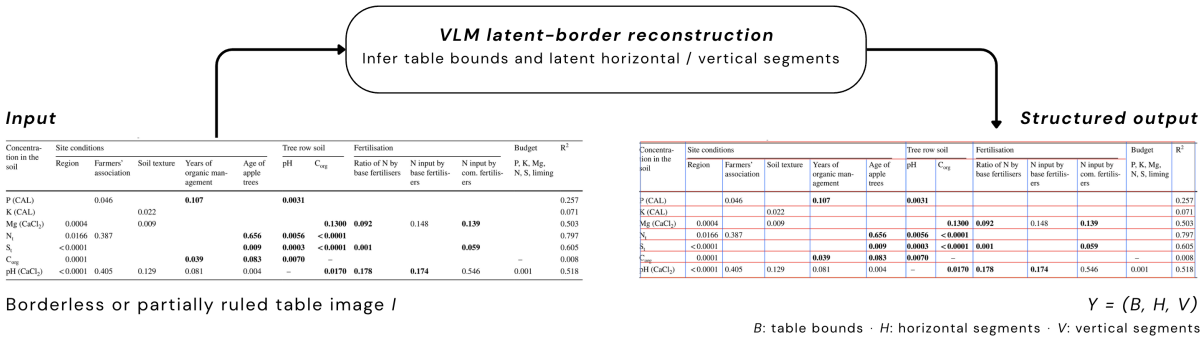


Figure 1: Latent-border reconstruction task. A vision–language model (VLM) takes a borderless or partially ruled table image I (left) and predicts the structured border geometry $Y = (B, H, V)$, visualised over the table on the right. B is the table bounding box, and H and V are the horizontal (red) and vertical (blue) border segments that recover the latent row and column separations; arrows show the input–model–output flow.

4 Tabstructor Framework

4.1 Model and Structured Generation

Tabstructor is the overall framework for mapping a single-table image to the latent-border representation defined in Section 3: a table bounding box and horizontal and vertical border segments serialised as JSON. It organises five controlled systems rather than treating every condition as a component of one deployed pipeline. Systems S1–S2 are frozen base-VLM prompting baselines, systems S3–S4 are learned Tabstructor predictors obtained by parameter-efficient fine-tuning of the same base VLM, and system S5 is an optional inference-time refinement analysis of S4. The augmented fine-tuned system S4 is the proposed primary predictor; its output is the raw learned prediction, before any refinement.

For S3–S4, Low-Rank Adaptation (LoRA) adapts the `unsloth/Qwen3.5-9B` base VLM to generate task-specific latent geometry from the table image and instruction, rather than merely tracing visible rules. The two fine-tuned systems share the base-model revision, adapter configuration, full prompt, and inference protocol, and differ principally in their training data. This separates learned reconstruction from the frozen prompting baselines and from downstream post-processing. Appendix D reports the exact model, LoRA, optimisation, length, compute, and run settings; Appendix F reports the shared inference and validation settings.

4.2 Minimal and Full Prompts

S1 and S2 keep the base VLM frozen and differ only in prompting: S1 uses a minimal schema-oriented instruction, whereas S2 uses the task-specific full prompt. This contrast tests what prompting alone contributes under matched inputs, decoding, parsing, and evaluation. The fine-tuned systems S3–S4 use the same frozen full prompt, preventing prompt variation from confounding the fine-tuning and augmentation comparisons. S5 adds no generation prompt because it post-processes S4's output. Appendix B preserves the exact prompt and output protocol.

4.3 Structure-Preserving, Typography-Matched Text Augmentation

Augmentation is used only during S4 training to vary textual and typographic appearance without changing the latent-border geometry. The offline procedure replaces table text with typography-matched synthetic content while preserving image dimensions and the source target; reference geometry is used only to construct and verify training variants and is unavailable at inference. S3 trains on the original data, whereas S4 adds the accepted variants; validation and test data remain unaugmented. Thus, S4–S3 tests structure-preserving augmentation, although the contrast includes additional cumulative training exposure and does not isolate text diversity alone. Appendix D gives the exact data counts, exposure calculation, and augmentation procedure.

4.4 OCR-Assisted Geometry Refinement

S5 is a separate inference-time post-processing condition that takes the exact raw S4 prediction and corresponding source image. It uses `zai-org/GLM-OCR` (Duan et al., 2026) once as the OCR component to derive content-occupancy evidence, which guides frozen border alignment and conservative local repair; OCR transcription is not

Table 1: Dataset splits and experimental roles, grouped by training, validation, and evaluation data. Source tables are unique original table images, whereas records are dataset instances, including accepted variants in the augmented training set. The fixed 32-table subset of the 84-table validation set is used for checkpoint selection by Latent-Border Reconstruction Score (LBRS), and the separate 100-table external test set is held out for final evaluation.

Split / condition	Source tables	Records	Purpose
Training data			
Original training set	757	757	Non-augmented fine-tuning
Accepted augmented variants	–	754	Added augmented instances
Augmented training set	757	1,511	Tabstructor (S4) fine-tuning
Validation data			
Full validation set	84	84	Validation during fine-tuning
Fixed validation subset (of 84)	32	32	LBRS-based checkpoint selection
Evaluation data			
External test set	100	100	Held-out final evaluation (S1–S5)

added to the prediction. The OCR evidence comes from a learned model, but the subsequent geometry transformations are deterministic conditional on the accepted evidence. Refinement performs neither further VLM training nor a second border-generation pass, and it reads no reference borders, test annotations, or metric outputs. Consequently, S5–S4 isolates whether downstream geometry refinement is necessary beyond the primary raw S4 prediction, rather than evaluating a new end-to-end architecture. Appendix E gives the ordered algorithm and frozen parameters.

5 Experimental Design

5.1 Research Questions

The experiments operationalise the three RQs introduced in Section 1 through four paired comparisons defined before external-test evaluation:

- **RQ1: Prompting and fine-tuning.** Compare S2–S1 to measure the full-prompt effect and S3–S2 to measure the fine-tuning effect beyond prompting.
- **RQ2: Augmentation.** Compare S4–S3 to measure the evaluated augmented training condition under the same validation protocol; unequal cumulative exposure precludes attribution to text diversity alone.
- **RQ3: OCR-assisted geometry refinement.** Compare S5–S4 to isolate the frozen refinement-stage effect on the same underlying predictions.

5.2 Five Frozen Systems

Table 2 summarises five predefined conditions. S1 is the frozen base-VLM minimal-prompt baseline, and S2 applies the full task prompt to the same frozen model. S3 fine-tunes that base model on the original training set, whereas the primary raw predictor, S4, uses the same full prompt and matched fine-tuning configuration with structure-preserving augmented training data. Augmentation is confined to training; validation and test inputs remain unaugmented. S5 is not separately trained: it applies frozen downstream refinement to the exact raw S4 predictions. All systems use the same output representation and external test set, and prompts, inference settings, and metric parameters are fixed before external-test evaluation.

5.3 Data Roles and Frozen Selection Protocol

S3 is trained on the 757 original training tables, while S4 adds the 754 accepted training-only variants; neither condition uses validation or test examples for parameter updates. Token-level loss is monitored on all 84 validation tables, and the unique maximum unrounded macro LBRS on a fixed 32-table subset of those tables selects one checkpoint per condition; the following subsection formally defines this metric. This subset is part of the validation split, not a third partition. After selection, checkpoints and all inference and evaluation settings are frozen, and only then are the five systems evaluated on the separately annotated 100-table external test set. Figure 4 in Appendix G visualises this validation-only selection and held-out evaluation protocol. The external test set is never used for training, loss monitoring, checkpoint selection, or configuration decisions. Appendix G also reports the selected checkpoint identifiers, complete validation rankings, and trajectories; Appendix D gives the training and checkpoint-

Table 2: Experimental configuration matrix for the five frozen systems evaluated on the same 100-table external test set. Rows group prompting-only base vision-language model (VLM) baselines (S1–S2), fine-tuned models (S3–S4), and refinement analysis (S5); columns indicate prompt type, use of fine-tuning, structure-preserving augmentation, and refinement, and each configuration's comparison purpose. S4 is the proposed Tabstructor system, and S5 applies geometry refinement to S4's raw predictions; boldface identifies the proposed system.

System	Prompt	Fine-tuned	Augmentation	Refinement	Purpose
Base VLM baselines (prompting only)					
S1 Base VLM – Minimal Prompt	Minimal	No	No	No	Minimal prompting baseline
S2 Base VLM – Full Prompt	Full	No	No	No	Effect of task-specific prompting
Fine-tuned models					
S3 Tabstructor predictor (no augmentation)	Full	Yes	No	No	Effect of fine-tuning beyond prompting
S4 Tabstructor primary predictor (Ours)	Full	Yes	Yes	No	Effect of structure-preserving augmentation
Refinement analysis					
S5 Refinement analysis (from S4)	Full	Yes	Yes	Yes	Effect of geometry refinement on S4 outputs

saving configurations.

5.4 Latent Border Reconstruction Score

Each system's predicted table box and horizontal and vertical latent-border segments are compared with the adjudicated reference geometry. The primary metric, the *Latent Border Reconstruction Score* (LBRS), rewards valid structured outputs that recover the number, position, and bounded extent of the reference boundaries together with table localisation. Parse and schema validity form a hard gate; table-box intersection over union (IoU), boundary-count agreement, snap-position agreement, and interval Dice describe complementary structural and geometric quality. Snap is a tolerance-weighted measure of matched boundary-plane position accuracy, while interval Dice compares the unioned extents of matched collinear border segments. Line-position mean absolute error (MAE) is reported only as a secondary diagnostic. This combination is appropriate because a usable reconstruction must recover both the required boundary planes and their geometry. For table i,

$$\begin{aligned}\mathrm{LBRS}_i = 100 G_i \big[& 0.05 B_i \\ & + 0.375 R_{c,i} + 0.05 R_{s,i} + 0.05 R_{d,i} \\ & + 0.375 C_{c,i} + 0.05 C_{s,i} + 0.05 C_{d,i} \big].\end{aligned} \tag{2}$$

where G_i is the parse-and-schema validity gate, B_i is table-box IoU, and the R and C terms measure horizontal and vertical boundary-count recovery (c), position agreement (s), and interval coverage (d). Appendix F gives the complete matching rules, tolerances, clipping, and edge conventions.

The official evaluation-set score is the macro-average of the per-table scores:

$$\mathrm{LBRS}_{\mathrm{macro}} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{LBRS}_i \, . \tag{3}$$

LBRS ranges from 0 to 100 and higher is better. The primary score is the macro-average of gated per-table scores; MAE is reported separately over matched planes and is not included in LBRS. Thus, LBRS measures border-structure and geometric reconstruction rather than complete cell topology or semantic table understanding. Appendix F specifies the aggregation procedure.

5.5 Statistical Reporting

Because all systems are evaluated on the same 100 tables, the principal analysis uses the four per-table paired contrasts defined above. We report macro means, medians, standard deviations, paired bootstrap 95% confidence intervals, and improved, unchanged, and degraded table counts. These intervals quantify variation in paired effects across test tables, not training-run variability; each fine-tuning condition is represented by one training run. Appendix F gives the resampling count, seed, and exact reporting conventions.

6 Results and Discussion

Across the paired external-test comparisons, task-specific prompting improves the frozen base VLM, fine-tuning produces the dominant gain, and the augmented training condition adds a smaller improvement. Refinement of the primary raw prediction usually improves geometry, but its mean benefit is small because a few large regressions offset the more frequent gains. Tables 3 and 5 report the supporting absolute and paired results.

Table 3: Macro-average latent-border reconstruction performance of five systems (S1–S5) on the fixed 100-table external test set, grouped as prompting-only vision-language model (VLM) baselines, fine-tuned Tabstructor variants, and refinement analysis. Columns report output validity and bounding-box (BBox) localisation, horizontal- and vertical-border recovery, mean absolute error (MAE), and the overall Latent-Border Reconstruction Score (LBRS). All values are percentages except MAE (pixels) and LBRS (0–100); ↑ indicates higher is better. S4 is **Tabstructor (Ours)**, and S5 refines S4 outputs. **Boldface** and underlining denote the best and second-best values, respectively; MAE† is unranked because it averages only finite per-table values and depends on parse/schema coverage.

	Output & extent			Horizontal borders				Vertical borders				
System	**Parse↑**	**Schema↑**	**BBox↑**	**Count↑**	**Snap↑**	**Dice↑**	**MAE†**	**Count↑**	**Snap↑**	**Dice↑**	**MAE†**	**LBRS↑**
Base VLM baselines (prompting only)												
S1 Base VLM – Minimal Prompt	55.00	34.00	33.16	7.09	9.99	6.50	4.54	25.31	12.08	8.09	7.60	15.64
S2 Base VLM – Full Prompt	75.00	74.00	72.50	32.25	30.02	26.31	6.50	48.58	25.43	10.55	8.22	38.55
Fine-tuned models												
S3 Tabstructor predictor (no augmentation)	**100.00**	**100.00**	**98.48**	90.00	89.28	91.23	4.44	99.62	43.59	55.52	11.51	90.01
S4 Tabstructor primary predictor (Ours)	**100.00**	**100.00**	98.38	95.57	95.24	95.64	3.38	**99.92**	49.37	64.94	13.89	93.49
Refinement analysis												
S5 Refinement analysis (from S4)	**100.00**	**100.00**	98.38	**96.19**	**96.95**	**96.71**	2.45	92.34	**82.51**	**90.53**	7.77	**93.95**

Metrics. H/V denote horizontal/vertical borders. MAE† averages finite per-table values and should be interpreted with parse/schema coverage.

Table 4: Post hoc sensitivity of macro-average Latent-Border Reconstruction Score (LBRS; 0–100, higher is better) for systems S1–S5 on the fixed 100-table external test set to four component-weighting schemes, including a geometry-balanced variant added for robustness. S1–S2 are prompting-only baselines, S3–S4 are fine-tuned systems, and S5 refines S4. Rows give the combined horizontal-and-vertical weights for boundary-count recovery (Count), snap-position accuracy (Snap), and interval-based segment coverage (Dice); the bounding-box (BBox) weight remains 5%. **Boldface** and underlining mark the best and second-best system scores within each row, respectively. Δ(S5–S4) is the LBRS-point difference between refined and unrefined Tabstructor, and * identifies the official weighting used in the primary evaluation. All system outputs are frozen; this analysis changes only the component weights.

				LBRS ↑					
Weighting	**Count (%)**	**Snap (%)**	**Dice (%)**	**S1**	**S2**	**S3**	**S4**	**S5**	**Δ(S5–S4)**
Count-heavy	80.0	7.5	7.5	15.991	39.419	91.260	**94.559**	94.084	−0.475
Official*	75.0	10.0	10.0	15.639	38.552	90.014	93.487	**93.954**	+0.467
Higher localization weighting	70.0	12.5	12.5	15.287	37.686	88.769	92.415	**93.825**	+1.410
Geometry-balanced	47.5	23.75	23.75	13.703	33.787	83.164	87.591	**93.242**	+5.651

6.1 RQ1: Prompting and Fine-tuning

The full prompt raises schema validity from 34% to 74% and LBRS from 15.639 to 38.552, a paired gain of 22.913 points (95% CI: 16.879–28.928). For the evaluated base VLM, task-specific instructions therefore improve output reliability and latent-border recovery, but 26 invalid S2 generations and weak vertical interval recovery show that prompting alone remains insufficient.

Fine-tuning produces the dominant improvement: S3 attains complete parse/schema validity and 90.014 LBRS, gaining 51.462 points over S2 (95% CI: 45.583–57.259). Vertical counts are nearly perfect, although placement and bounded extent remain weaker. Thus, within this model and test setting, prompting makes valid reconstruction more attainable, while the fine-tuned condition accounts for most of the observed performance increase.

6.2 RQ2: Structure-Preserving Typography-Matched Augmentation

The primary raw predictor S4 reaches 93.487 LBRS and gains 3.473 points over S3 (95% CI: 2.061–4.986), a modest effect relative to fine-tuning. The augmented condition improves 69 tables and degrades 31, with its clearest component gain in vertical interval coverage; vertical snap improves whereas vertical MAE worsens, so the localisation evidence is metric-dependent. RQ2 therefore favours the evaluated augmented training condition on average, not augmentation universally. Because S4 receives approximately 15.6% more cumulative exposure, the comparison cannot attrib-

Table 5: Paired per-table changes in Latent Border Reconstruction Score (LBRS; 0–100, higher is better) between consecutive systems S1–S5 on the fixed 100-table external test set. Rows identify each system contrast and added intervention; columns report the mean later-minus-earlier change (Δ, in LBRS points), its 95% paired-bootstrap confidence interval (CI), and the numbers of tables that improved, remained unchanged, or degraded. Positive Δ values favour the later system; $\uparrow$ indicates higher is better.

Contrast	Intervention	Mean Δ LBRS $\uparrow$	95% CI	Improved	Same	Degraded
S1 $\rightarrow$ S2	Task-specific prompting	+22.913	[16.879, 28.928]	66	19	15
S2 $\rightarrow$ S3	Fine-tuning	+51.462	[45.583, 57.259]	98	0	2
S3 $\rightarrow$ S4	Structure-preserving augmentation	+3.473	[2.061, 4.986]	69	0	31
S4 $\rightarrow$ S5	Geometry refinement	+0.467	[−1.968, 2.615]	90	0	10

ute the gain specifically to text diversity rather than additional exposure.

6.3 RQ3: Geometry Refinement

Refinement improves 90 tables and degrades ten, with a median paired gain of 3.191 LBRS points, but its mean gain is only 0.467 points (95% CI: −1.968–2.615). Thus, refinement typically helps under these summaries, but severe regressions make its average benefit small and not reliably positive under the official weighting.

The benefit is principally better boundary placement and extent: vertical snap and interval coverage improve, while vertical-count accuracy falls from 99.923% to 92.340%. Eight regressions exceed 30 LBRS points and coincide with vertical-count collapse; the paired qualitative cases likewise show both improved alignment and damage to an otherwise strong raw reconstruction (Figures 9 and 10). Most measured structure is therefore already present in S4, making S5 an optional, non-uniform geometry adjustment rather than a necessary source of the main reconstruction gain.

The small S5–S4 effect is also weighting-sensitive: it is negative under count-heavy weighting and grows as localisation receives more weight (Table 4). As a robustness check, we additionally evaluate a geometry-balanced variant that retains 5% BBox weight and splits the remaining 95% equally between boundary-count recovery (47.5%) and geometry (47.5%: 23.75% Snap and 23.75% Dice). Under this deliberately more geometry-sensitive weighting, S3, S4, and S5 score 83.164, 87.591, and 93.242, respectively: the principal fine-tuning gain remains dominant, S4 still exceeds S3 by 4.427 points, and refinement benefits substantially because its strongest changes are geometric. The pre-specified official weighting remains primary; the balanced variant is a post-hoc robustness analysis rather than a replacement metric.

7 Conclusion

Tabstructor investigates explicit latent-border reconstruction for complex borderless tables and introduces the Latent Border Reconstruction Score (LBRS) as a bounded measure of valid structural and geometric border recovery. On the evaluated scientific-table test set, task-specific prompting improves the frozen base VLM, while supervised fine-tuning supplies the dominant reconstruction gain. The augmented training condition provides a further modest average improvement, although its additional cumulative exposure prevents attributing that gain to text diversity alone.

Downstream refinement of the primary raw fine-tuned system, S4, frequently improves boundary localisation and extent, but S5 has only a small, uncertain mean benefit and can introduce severe boundary-count regressions. Thus, in the evaluated setting, task-specific VLM adaptation performs the principal latent-border reconstruction, while refinement is an optional geometry adjustment rather than a necessary source of the main gain. These results establish latent borders as an explicit and measurable structural representation, but do not establish complete logical-cell topology, semantic understanding, downstream structured-format recovery, or cross-domain generalisation.

Limitations

The evidence is limited to the represented scientific-table distribution, modest development splits, and one held-out 100-table test set, so it does not establish performance on rare structures or other domains. We evaluate one LoRA-adapted model family and size, Qwen3.5-9B, with one run per fine-tuning condition; the paired intervals capture variation across test tables, not seeds, architectures, scales, or adaptation methods, and checkpoint choice may depend on the fixed 32-table

validation subset. Each table has one reviewed, adjudicated reference, although ambiguous layouts can admit other defensible geometries that single-reference scoring may penalise; labels were not independently duplicated. LBRS measures border geometry, not OCR accuracy, complete cell topology, semantic understanding, reasoning, or markup recovery. The S4–S3 contrast also combines one augmentation strategy with approximately 15.6% more training exposure, precluding attribution to text diversity or augmentation generally. Placement and extent errors remain in S4, while optional S5 adds OCR-dependent, rule-based processing that can over-regularise irregular boundaries; its small effect is non-uniform and sensitive to metric weighting.

Ethical Considerations

This study uses table images derived from published scientific literature, involves no research on human participants or collection of new personal data, and treats manual label creation and review as research annotation. Because access does not imply redistribution permission, cropped source images will be released only where source-specific review permits; project-created code and tools, derived annotations, split manifests, and provenance metadata will be released separately under applicable terms. Coverage of particular publishing styles and table structures may yield uneven reliability on other document types. Predictions can contain structural errors and establish neither OCR correctness nor the semantic or scientific validity of table contents, so accuracy-sensitive downstream applications should verify relevant outputs.

Acknowledgements

This work was supported by an Early Career Research Fellowship under the International Science Partnerships Fund (ISPF), delivered by the British Council, through the project *"Enhancing Environmental Resilience through AI-driven Analysis of Unstructured Data."* The authors also acknowledge the RONIN cloud platform and the Research IT team at The University of Manchester for providing computational resources and technical support.

References

Iñigo Alonso, Imanol Miranda, Eneko Agirre, and Mirella Lapata. 2026. TABLET: A large-scale dataset for robust visual table understanding. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*.

Boming Chen, Zining Wang, Zhentao Guo, Jianqiang Liu, Chen Duan, Yu Gu, Kai Zhou, and Pengfei Yan. 2026. InstructTable: Improving table structure recognition through instructions. *arXiv preprint arXiv:2604.02880*.

Shuaiqi Duan, Yadong Xue, Weihan Wang, Zhe Su, Huan Liu, Sheng Yang, Guobing Gan, Guo Wang, Zihan Wang, Shengdong Yan, Dexin Jin, Yuxuan Zhang, Guohong Wen, Yanfeng Wang, Yutao Zhang, Xiaohan Zhang, Wenyi Hong, Yukuo Cen, Da Yin, and 4 others. 2026. GLM-OCR technical report. *arXiv preprint arXiv:2603.10910*.

Khurram Azeem Hashmi, Didier Stricker, Marcus Liwicki, Muhammad Noman Afzal, and Muhammad Zeshan Afzal. 2021. Guided table structure recognition through anchor optimization. *IEEE Access*, 9:113521–113534.

Xuanrui Hong, Kai Zha, Ziming Feng, Zhexuan Chen, Xiaodong Du, and Min Liu. 2024. DAFA: A dual-awareness feature aggregator for table structure recognition on medical examination reports. *IEEE Access*, 12:1321–1329.

Minsoo Khang and Teakgyu Hong. 2025. TFLOP: Table structure recognition framework with layout pointer mechanism. *arXiv preprint arXiv:2501.11800*.

Weihong Lin, Zheng Sun, Chixiang Ma, Mingze Li, Jiawei Wang, Lei Sun, and Qiang Huo. 2022. TSRFormer: Table structure recognition with transformers. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6473–6482.

Nam Quan Nguyen, Anh Duy Le, Anh Khoa Lu, Xuan Toan Mai, and Tuan Anh Tran. 2023. Formerge: Recover spanning cells in complex table structure using transformer network. In *Document Analysis and Recognition—ICDAR 2023*, volume 14191 of *Lecture Notes in Computer Science*, pages 522–534. Springer Nature Switzerland.

Aditi Prajapati and Swati Maurya. 2026. Table structure recognition: A review of current approaches and future opportunities. *IET Conference Proceedings*, 2025(43):86–91.

Wenyuan Xue, Baosheng Yu, Wen Wang, Dacheng Tao, and Qingyong Li. 2021. TGRNet: A table graph reconstruction network for table structure recognition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1275–1284.

Chenglong Yu, Weibin Li, Zixuan Zhu, Wei Li, Jianchao Du, and Shiwei Zhang. 2025. TableStructureFormer: An improved masked-attention mask transformer model with long-distance feature aggregation and deep detail supervision for table structure recognition. *Complex & Intelligent Systems*, 11(8):343.

Jinxu Zhang. 2024. Read and think: An efficient step-wise multimodal language model for document understanding and reasoning. *arXiv preprint arXiv:2403.00816*.

Zhenrong Zhang, Pengfei Hu, Jiefeng Ma, Jun Du, Jianshu Zhang, Baocai Yin, Bing Yin, and Cong Liu. 2024. SEMv2: Table separation line detection based on instance segmentation. *Pattern Recognition*, 149:110279.

A Reproducibility Statement

Section 3 and Appendix C document the data, split roles, annotation target, TeybolStruct protocol, and ambiguity policy. Appendix B preserves the frozen prompts and output protocol; Appendix D records the exact base-model revision and the LoRA, optimisation, augmentation, seed, environment, hardware, and runtime configurations; Section 5 defines S1–S5; and Appendices G, F, and E specify checkpoint selection, LBRS evaluation, and the GLM-OCR-assisted refinement from raw S4 to S5, respectively. Appendix H describes the planned machine-readable predictions and provenance records. Subject to licensing review, the planned project release covers fixed split manifests, derived annotations, selected adapters, and the annotation, training, inference, augmentation, evaluation, and refinement resources; cropped source images will be included only where redistribution is permitted.

B Frozen Prompt and Output Protocol

B.1 Minimal Prompt

The frozen textual instruction supplied with each table image in S1 was exactly:

```
Reconstruct the completed table borders from this scientific
      table image.

Return exactly one valid JSON object and nothing else.

Use this exact schema:
{"table_bbox_norm":[x_min,y_min,x_max,y_max],"
      completed_borders":{"horizontal_segments_norm":[[x_start,
      x_end,y]],"vertical_segments_norm":[[x,y_start,y_end]]}}

The table bounding box coordinates must be normalized relative
       to the full image.
The border-segment coordinates must be normalized relative to
      the table bounding box.
Every coordinate must be between 0.000000 and 1.000000 and use
       exactly six decimal places.

Infer the outer table border and the horizontal and vertical
      separators required to represent the table.

Horizontal segments must use [x_start,x_end,y].
Vertical segments must use [x,y_start,y_end].

Sort horizontal segments from top to bottom, then left to
      right.
Sort vertical segments from left to right, then top to bottom.

Return only the JSON object.
```

For both prompt conditions, the fixed message-role wrapper and exact image-token placement are specified by the frozen inference implementation.

B.2 Full Prompt

The frozen textual instruction supplied with each table image in S2, S3, and S4 was exactly:

```
Reconstruct the completed table borders from this scientific
      table image.

Return exactly one valid JSON object and nothing else.
Do not return markdown, code fences, comments, OCR text, table
       text, cell contents, or explanations.

Use this exact schema:
{"table_bbox_norm":[x_min,y_min,x_max,y_max],"
      completed_borders":{"horizontal_segments_norm":[[x_start,
      x_end,y]],"vertical_segments_norm":[[x,y_start,y_end]]}}

Coordinate rules:

* table_bbox_norm is normalized relative to the full image.
* completed_borders coordinates are normalized relative to the
       table bounding box.
* Every coordinate value must be a float from 0.000000 to
      1.000000.
* Use exactly six decimal places.
* Never output pixel coordinates.
* Never output image-size coordinates.
* Never output 0-100, 0-999, or 0-1000 coordinates.
* Any value greater than 1.000000 is invalid.

Task:
Complete the logical border structure of the table, not only
      the visibly printed rules. Add the missing outer border,
       column separators, row separators, partial header rules,
       and merged-cell boundaries needed to express the
      intended table topology.

Highest-priority geometry rules:
1. A predicted border is invalid if it crosses, touches, or
      visually overlaps table text glyphs.
2. Place horizontal row separators in the whitespace gap
      between adjacent visual text rows. Never place them on a
       text baseline, through descenders, through ascenders,
      through superscripts/subscripts, through punctuation, or
       through the middle of text.
3. Place vertical separators in the whitespace gap between
      adjacent text columns. Never place them through letters,
       digits, symbols, units, formulas, superscripts, or
      subscripts.
4. Geometry has priority over uniform spacing. Do not evenly
      distribute borders unless the text bands, alignment, and
       visible rules support that placement.
5. If a separator cannot be placed without touching text,
      choose the nearest existing printed rule or the widest
      available whitespace gap.
6. Do not duplicate nearby horizontal or vertical rules. If a
      visible printed rule already represents the logical
      boundary, use that boundary once.

Critical row-topology rules:
1. A horizontal separator is valid only where adjacent cells
      actually split at that y-position.
2. A full-width horizontal separator is valid only when every
      column participates in the same row split.
3. If a cell visually spans multiple subordinate rows, the
      separator between those subordinate rows must not pass
      through the spanning cell. Start or stop the separator
      at the column boundary of the cells that actually split.
4. Multiple printed lines inside one cell are not multiple
      rows. Do not draw a horizontal separator between wrapped
       lines, continued labels, units, formulas, coordinates,
      parenthetical text, or multi-line descriptions unless
      the surrounding columns also show a consistent row split.

5. Section-like, bold, italic, or standalone text does not
      automatically create a full-width merged row. Use a
      merged/full-width row only when the visual layout shows
      that the row owns the full table width.
6. Prefer partial horizontal segments over full-width
      horizontal segments when only part of the table splits
```

```
      at that y-position.

Annotation policy:
1. Preserve all table text. Borders may overlap existing
      printed rules, but must not cross, touch, or obscure
      text glyphs.
2. Use the existing top rule, header rule, bottom rule,
      alignment, spacing, and content ownership to infer the
      table bounding box and borders.
3. Do not default to a full rectangular internal grid. Use a
      complete grid only when the text alignment and visible
      structure show that all rows and columns share the same
      boundaries.
4. Do not create structure from typography alone. Bold, italic,
       superscripts, subscripts, significance marks, chemical
      notation, scientific symbols, and font weight are cell
      content only.
5. Wrapped or multi-line text inside one visual cell remains
      one cell. Do not add a horizontal border merely because
      a label, phrase, coordinate, unit, taxonomic name,
      parenthetical description, or long heading wraps onto
      multiple printed lines.
6. A new row exists only when there is a separate aligned
      record, a repeated row pattern, or a clear cell
      ownership change across columns.
7. Repeated prefixes, repeated units, repeated child labels,
      repeated statistic labels, or semantic similarity
      between columns do not create grouped headers unless the
       visual header layout shows a parent-child structure.
8. Isolated punctuation, dashes, or stray marks between header
       groups do not create columns or merged cells unless
      they align with a real body column or header hierarchy.
9. Shared unit, scale, normalization, or measurement
      annotations may appear as extra header rows beneath
      otherwise separate column headers. If such an annotation
       applies across multiple measurement columns, treat it
      as a merged header cell spanning only the applicable
      columns.
10. For grouped headers, preserve each visible parent-header
      underline as its own partial horizontal segment. Do not
      merge adjacent header underlines into one longer segment
       if there is a visual gap, a parent-group boundary, or a
       row-spanning header cell between them.
11. For hierarchical headers, parent headers span exactly
      their visible child columns. Child separators must stop
      at the parent-header boundary and restart only below the
       parent header.
12. Descriptor columns may rowspan across multiple header
      levels when they do not participate in grouped
      subheaders.
13. Repeated child labels under different parent headers are
      distinct columns.
14. Do not invent missing child columns under shortened header
       groups.
15. Unlabeled structural columns are valid. If a body column
      contains labels, terms, formulas, coefficient names, row
       descriptors, explanatory text, or statistic labels but
      its header cell is blank, preserve it as an explicit
      column.
16. If an unlabeled body column contains consistently aligned
      row descriptors, statistic labels, coefficients,
      category labels, units, or repeated label patterns,
      treat it as an explicit structural column even when its
      header cell is blank.
17. A header cell may span multiple underlying body columns
      even when no child header labels are printed.
18. Adjacent aligned components such as estimate, uncertainty,
       error, confidence interval, standard deviation,
      standard error, or significance-letter columns should be
       preserved as separate columns when they form consistent
       vertical alignments.
19. Section or explanatory rows should span the full table
      width only when they visually act as full-width section
      dividers, panel titles, block labels, or standalone
      descriptive rows.
20. Internal vertical borders must stop through full-width
      merged section rows and restart after them.
21. Do not merge a section-like label across the full table
      width merely because it is bold, italic, standalone, or
      followed by related rows.
22. Italic, bold, or standalone section-like labels do not
      automatically span the full table width. If the numeric
      columns remain part of the same row structure, keep the
      vertical column separators and treat the numeric-side
      cells as blank cells.
23. Panel header rows may combine a section or panel label in
      the first descriptor column with repeated column headers
       in the remaining columns.
24. Section-like body rows may be local to a column group or
      panel rather than full-width.
25. A full-width section row may reset the local column
      structure below it.
26. Body rowspans are allowed when a label visually owns
      multiple subordinate rows, especially when repeated
      parent labels are omitted.
27. When descriptor cells such as region names, category names,
       sample counts, orchard counts, treatment labels, site
      labels, or group names visually own multiple subordinate
       rows, treat them as row-spanning cells.
28. Horizontal separators between subordinate rows must not
      pass through row-spanning descriptor cells. They should
      begin at the first column that actually splits into
      subordinate rows.
29. Body rowspans are not limited to descriptor columns.
30. Body records may contain nested subrows. Only the columns
      whose values change should split into subordinate rows.
31. Nested subrows may split independently by column.
32. Explicitly repeated values or row labels in separate
      aligned rows do not imply merging.
33. Empty cells, blank regions, placeholders, missing-value
      symbols, and missing-value text remain valid cells.
34. Sparse tables, triangular matrices, and partially
      populated regions must preserve the full latent grid.
35. Output only border segments that define logical cell
      topology. Keep outer borders, row separators, column
      separators, merged-cell boundaries, grouped-header
      boundaries, section/panel boundaries, and necessary
      partial segments. Remove decorative rules, duplicate
      nearby rules, stray marks, text underlines, and line
      fragments that do not separate cells or define ownership.

36. Preserve partial-line semantics. Output partial borders as
       separate segments.
37. Prefer the simplest valid topology that preserves visible
      grouping, row ownership, column ownership, merged cells,
       rowspans, panel-local grids, shared headers, and empty
      cells.

Body-row placement rules:
1. For body rows, place each horizontal separator between the
      lower edge of the text in the row above and the upper
      edge of the text in the row below.
2. Do not place a separator through the center, baseline,
      descenders, superscripts, subscripts, or punctuation of
      either adjacent row.
3. For body rows with long descriptor text, parenthetical
      units, coordinates, soil names, taxonomic names,
      chemical notation, or wrapped descriptions, treat all
      wrapped lines as one cell unless the adjacent columns
      also show a consistent separate row split.
4. Do not add extra horizontal borders inside a single wrapped
       cell.
5. If a row contains blank cells in some columns, preserve
      those blank cells rather than merging the row unless a
      visual full-width section or rowspan is clearly
      indicated.

Segment format:
Horizontal segments must be [x_start,x_end,y].
Vertical segments must be [x,y_start,y_end].

Sorting:
Sort horizontal segments from top to bottom, then left to
      right.
Sort vertical segments from left to right, then top to bottom.

Final coordinate check before answering:
All numbers in table_bbox_norm, horizontal_segments_norm, and
      vertical_segments_norm must be between 0.000000 and
      1.000000.
Do not output any number greater than 1.000000.
Do not output pixel coordinates.
Return only the JSON object.
```

S5 has no additional VLM border-generation prompt; its refinement protocol is described in Section 4.3 and Appendix E.

B.3 Parsing and Validation

The evaluation expects the top-level keys `table_bbox_norm` and `completed_borders`; the latter must contain `horizontal_segments_norm` and `vertical_segments_norm`. The generated text is trimmed to the first complete JSON object before parsing. Parse success and schema validity are reported separately; both are required by the LBRS validity gate defined in Section 5.4. Table 6 reports the shared frozen decoding settings.

Table 6: Frozen vision–language model (VLM) decoding settings shared by the generation-producing systems S1–S4. Rows report the maximum output length in tokens, sampling status, beam count, and repetition penalty; S5 performs no additional VLM generation.

Setting	Value
Maximum new tokens	5,120
Sampling	Disabled
Number of beams	1
Repetition penalty	1.02

C TeybolStruct Annotation Protocol and Ambiguity Policy

C.1 Editor Representation

TeybolStruct represents the main-body target using a pixel-coordinate table box, complete horizontal and vertical boundary axes, bounded partial segments, and erased portions. The editor retains these objects and assisted evidence for reversible addition, removal, trimming, and erasure.

The target contains only `table_bbox_norm` and the `horizontal_segments_norm` and `vertical_segments_norm` arrays under `completed_borders`; OCR, cell text, dimensions, and debug fields are excluded. For image width W, height H, and pixel box $[l, t, r, b]$,

$$B_{\text{norm}} = \left[\frac{l}{W}, \frac{t}{H}, \frac{r}{W}, \frac{b}{H} \right], \tag{4}$$

while a horizontal pixel segment $[x_0, x_1, y]$ is converted to

$$\left[\frac{x_0 - l}{r - l}, \frac{x_1 - l}{r - l}, \frac{y - t}{b - t} \right], \tag{5}$$

while a vertical segment $[x, y_0, y_1]$ becomes

$$\left[\frac{x - l}{r - l}, \frac{y_0 - t}{b - t}, \frac{y_1 - t}{b - t} \right]. \tag{6}$$

Values are clamped to $[0, 1]$ and serialised to six decimal places.

C.2 Assisted Initialisation

Visible-line candidates are extracted by grayscale conversion, adaptive mean thresholding, horizontal/vertical morphological opening, and the fixed length filters in Table 7. Margin-adjacent candidates may be supplied to the optional `Qwen/Qwen3-VL-8B-Instruct` assistant, which proposes only editable top, bottom, left, and right coordinates; it neither generates internal annotations nor participates in the five evaluated systems.

At export, detected rules exceeding the overlap threshold may enter the target. An erased portion marks its axis as non-complete, preventing automatic reintroduction as a full line; manual correction therefore takes precedence over assisted evidence.

C.3 Human Editing Procedure

The annotation procedure was:

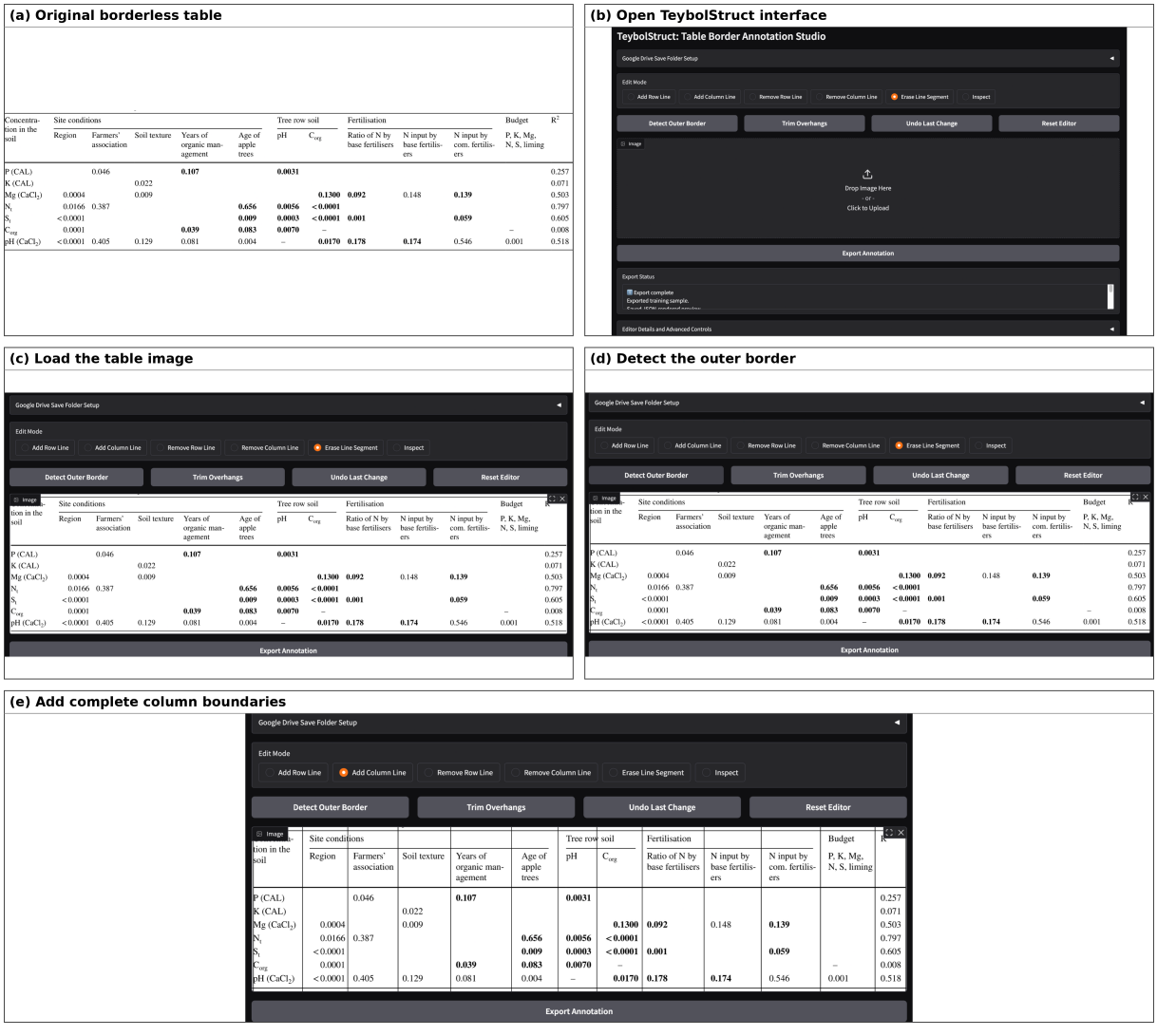
1. load the extracted table image and establish or correct the outer table region;
2. inspect detected rule candidates and, when used, the editable four-boundary VLM proposal;
3. add, remove, or reposition complete horizontal and vertical boundaries;
4. use bounded partial segments, including erasures from otherwise complete axes, to represent local separators without dividing grouped, merged, or spanning regions;
5. trim overhangs and resolve near-duplicate coordinates;
6. export the normalised target and inspect the rendering generated from the exact validated JSON.

Figures 2 and 3 show this procedure sequentially on one representative example.

C.4 Export Validation and Quality Assurance

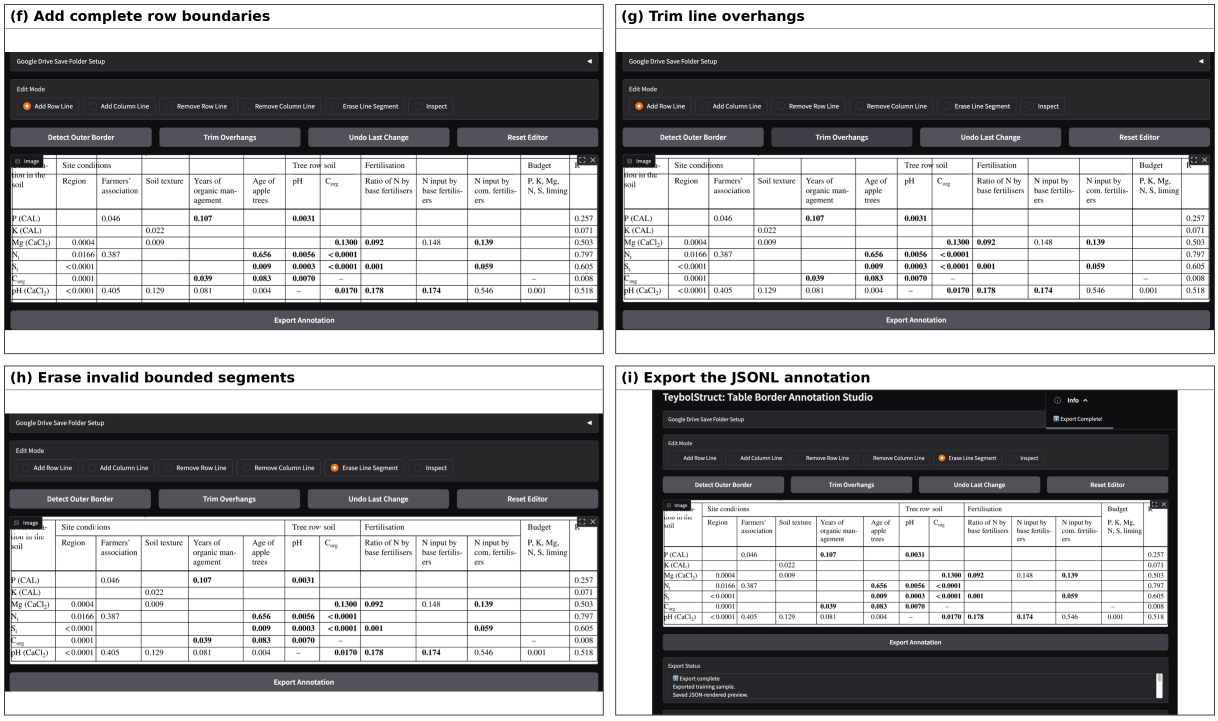
The exporter normalises the final pixel geometry, denormalises it, and blocks export if any reconstructed box or segment coordinate differs by more than one pixel. Validation also requires finite values in $[0, 1]$, increasing endpoints, exactly the expected keys, and full-width/full-height outer borders.

Before JSONL writing, the exact validated JSON is rendered for review. Export tests cover horizontal and vertical segmented-axis suppression and preservation of the outer rectangle.



Concentration in the soil	Site conditions					Tree row soil		Fertilisation			Budget	R^2
	Region	Farmers' association	Soil texture	Years of organic management	Age of apple trees	pH	C_{org}	Ratio of N by base fertilisers	N input by base fertilisers	N input by com. fertilisers	P, K, Mg, N, S, liming	
P (CAL)		0.046		**0.107**		**0.0031**						0.257
K (CAL)			0.022									0.071
Mg ($CaCl_2$)	0.0004		0.009				**0.1300**	**0.092**	0.148	**0.139**		0.503
N_t	0.0166	0.387			**0.656**	**0.0056**	**<0.0001**					0.797
S_t	<0.0001				**0.009**	**0.0003**	**<0.0001**	**0.001**		**0.059**		0.605
C_{org}	0.0001			**0.039**	**0.083**	**0.0070**	–				–	0.008
pH ($CaCl_2$)	<0.0001	0.405	0.129	0.081	0.004	–	**0.0170**	**0.178**	**0.174**	0.546	0.001	0.518

Figure 2: Early stages of the TeybolStruct annotation workflow on a representative borderless table: (a) the original table image; (b) the annotation interface; (c) the loaded image; (d) the detected and, if needed, corrected outer table border; and (e) complete column boundaries added only where they form valid full-table splits.



(j) Inspect the JSON-rendered bordered target

Concentration in the soil	Site conditions					Tree row soil		Fertilisation			Budget	R^2
	Region	Farmers' association	Soil texture	Years of organic management	Age of apple trees	pH	C_{org}	Ratio of N by base fertilisers	N input by base fertilisers	N input by com. fertilisers	P, K, Mg, N, S, liming	
P (CAL)		0.046		**0.107**		**0.0031**						0.257
K (CAL)			0.022									0.071
Mg ($CaCl_2$)	0.0004		0.009				**0.1300**	**0.092**	0.148	**0.139**		0.503
N_t	0.0166	0.387			**0.656**	**0.0056**	**<0.0001**					0.797
S_t	<0.0001				**0.009**	**0.0003**	**<0.0001**	**0.001**		**0.059**		0.605
C_{org}	0.0001			**0.039**	**0.083**	**0.0070**	–				–	0.008
pH ($CaCl_2$)	<0.0001	0.405	0.129	0.081	0.004	–	**0.0170**	**0.178**	**0.174**	0.546	0.001	0.518

Figure 3: Later stages of the TeybolStruct annotation workflow on a representative borderless table: (f) add complete row boundaries; (g) trim line overhangs; (h) erase invalid bounded segments that would split grouped or spanning regions; (i) export the annotation as JSON Lines (JSONL); and (j) inspect the bordered-table rendering generated from the exported JSON. The panels proceed from geometry editing to export and final verification.

Table 7: Fixed TeybolStruct settings and values for the annotation workflow, grouped by line extraction, candidate filtering and merging, and export and geometry validation. Pixel-based values are reported in px; ratios are dimensionless.

Setting	Value
Line extraction	
Adaptive threshold block size	25
Adaptive threshold constant	15
Horizontal / vertical kernel divisor	30 / 30
Minimum detected length	15 px and 0.03 of image axis
Candidate filtering & merging	
Outer-candidate margin ratio	0.18
Outer-candidate minimum span ratio	0.25
Line-merge / duplicate tolerance	3 px
Partial-segment anchor snap tolerance	12 px
Erase-selection tolerance	8 px
Export & geometry validation	
Export detected-rule overlap ratio	0.20 horizontal / 0.20 vertical
Coordinate precision	6 decimal places
Maximum geometry round-trip error	1 px

One primary annotator prepared the labels, and each serialised-target rendering was reviewed before dataset admission. Ambiguity review and acceptance are specified below. Primary annotation averaged approximately five minutes per image (roughly 70 hours for 841 tables); correction and review time were not recorded separately.

C.5 Ambiguity and Adjudication

When multiple geometries are plausible, annotators select the simplest topology that preserves observable row and column ownership, grouped headers, spanning cells, panel-local structure, and empty structural fields. The decision rules are:

1. Use a complete separator only when every region along its extent participates in the same split; otherwise use the minimum bounded extent that expresses the split.
2. Treat aligned, repeated fields as structural even when their header or body entry is blank. Do not create a boundary from typography, whitespace, or a decorative rule alone.
3. Keep wrapped or multiline text in one cell unless consistent neighbouring evidence establishes another row or column. Where text obscures or interrupts an otherwise aligned boundary, join the fragments only if the shared plane is supported on both sides and no content crosses the inferred corridor.
4. Preserve merged cells, rowspans, colspans, grouped headers, and local panels by omitting or bounding separators that would split them. A panel-local separator does not extend through neighbouring panels without evidence of the same split.
5. Use the minimal outer rectangle enclosing the logical table content and rules; exclude captions, footnotes, and decorative material that do not participate in the table grid.

These rules apply equally to fully borderless, partially ruled, and broken-rule regions: visible ink is evidence, whereas the target records the supported logical boundary. If the rules do not yield one defensible geometry, the case is escalated rather than resolved by visual regularity alone.

The primary annotator inspected the source and serialised-target rendering; unresolved cases were reviewed weekly and, if necessary, escalated to the biweekly full-team meeting. The agreed geometry was revised before admission, yielding one reference per table. This was collaborative review rather than independent duplicate annotation, so formal inter-annotator agreement is unavailable. External-test references were reviewed and frozen before S1–S5 evaluation without access to predictions, LBRS scores, or model-specific errors. Table 7 was fixed across all splits, but assisted settings never overrode manual correction and export validation.

D Training and Augmentation Configuration

This appendix records the matched fine-tuning configurations for S3 (non-augmented) and S4 (augmented). Shared settings are reported once; condition-specific data and run records are identified explicitly below.

D.1 Frozen Augmentation Procedure

For each selected source record, the released augmentation module executes the following ordered procedure:

1. Validate the source image path and completed-border JSON, then convert the image-relative table box and table-relative border segments to pixels.

2. Merge nearby axes, construct completed-border edge arrays, optionally repair outer edges, and recover cells by joining adjacent elementary rectangles whose shared border is absent.
3. Remove known rule pixels, skip near-empty crops, and prepare usable crops by border wiping, background normalisation, contrast-limited adaptive histogram equalisation (CLAHE), light denoising, and unsharp masking.
4. Transcribe cells in size-aware batches, recursively splitting CUDA out-of-memory batches and retrying irreducible crops at smaller scales. Normalise scripts without dataset-specific corrections and use geometric detection for missed dash-only placeholders.
5. Assign detected visual words to recovered cells, estimate base font size and cell-level boldness, and deterministically replace letters and digits with case-, width-, and shape-compatible alternatives while preserving punctuation, spacing, subscripts, and superscripts.
6. Mask safely assigned non-dash words while protecting rules, inpaint their pixels, and fit replacement strings to the original word boxes using the configured font-size and horizontal-scale ranges.
7. Accept a variant only if dimensions are unchanged, at least one safe replacement exists, the saved image and per-sample JSONL pass integrity checks, and the raw and parsed assistant targets exactly equal the source targets.

As described in Section 4.3, reference geometry is confined to this offline procedure and is not an inference input; exact target equality is enforced by the final acceptance check. Table 8 records the fixed pipeline parameters.

Here, *graph-guided* means that the recovered cell-adjacency graph—formed by joining adjacent elementary rectangles when their shared border is absent—guides cell assignment and text replacement; it is not a separate learned model.

Cell transcription in this offline augmentation procedure uses the same GLM-OCR model identified in the main text. This training-data preparation role is separate from its use for content-geometry validation in the S5 inference-time refinement stage.

Variant acceptance and rejection. A variant is rejected if validation, image loading, cell recovery, OCR/rendering, or a final integrity check fails. The run accepted 754 of 757 requested variants; Table 9 places these records in the matched S3–S4 data and model configuration, and the released rejection log records the three case-specific causes.

Table 8: Fixed configuration of the offline typography-matched text-augmentation pipeline used to create S4 training variants. Rows report parameter–value pairs grouped by pipeline stage; OCR denotes optical character recognition, and px denotes pixels.

Parameter	Value
Structural preparation	
Axis merge tolerance	5 px
Edge coverage tolerance	8 px
Outer-edge repair	Enabled
Header rows	2
Row-header columns	1
OCR preparation	
OCR model	zai-org/GLM-OCR
OCR scale	2×
OCR padding	12 px
Minimum prepared crop	224 × 80 px
Maximum crop side	768 px
OCR token limit	64
OCR batch size	8
Padded-pixel limit	2,400,000
Preparation buffer	32
Minimum ink ratio	0.0012
Crop enhancement	Enabled
Dash fallback	Enabled
Text and rule masking	
Dark threshold	250
Erase threshold	253
Visual-word linking kernel	7 × 2 px
Minimum word ink	6 px
Normal-word height	6 px
Rule-protection thickness	4 px
Text-mask dilation	1 px
Inpainting radius	2 px
Typography synthesis	
Base font-size search range	14–52 px
Horizontal rendering scale range	0.70–1.35
Reproducibility	
Global seed	42
Variants per image	1

Table 10 reports the shared optimisation settings and the condition-specific run, termination, and compute records.

Token audit. Inputs reach at most 13,027 tokens and assistant targets at most 3,783 tokens; no record exceeds the respective 15,360-token sequence or 5,120-token generation limit.

Run identification. The retained manifests identify S3 with `augmentation_enabled=false`

Table 9: Matched data, model, and integrity configuration for S3 (unaugmented) and S4 (augmented) fine-tuning. S4 adds 754 accepted graph-guided, typography-matched variants to the 757 original training records; validation and test data remain unaugmented. Sequence and generation lengths are in tokens. LoRA, MLP, and rsLoRA denote Low-Rank Adaptation, multilayer perceptron, and rank-stabilized LoRA, respectively.

Configuration	Value
S3–S4 data comparison	
Training / validation split	757 / 84
S3 training records	757 total
S4 training records	757 originals + 754 accepted variants
S4 rejected variants	3
S3 augmentation	Disabled
S4 augmentation	Graph-guided, typography-matched text randomisation
Shared model configuration	
Base model	`unsloth/Qwen3.5-9B`
Model revision	`005429cee5cb648998c f2b70eebdd83175989c 9a`
Adaptation	4-bit LoRA; full fine-tuning disabled
Gradient checkpointing	Unsloth
LoRA rank r	64
LoRA α	64
LoRA dropout	0.02
LoRA bias	none
rsLoRA	False
Trainable parameters	203,913,216 (3.43%)
Targeted modules	Vision, language, attention, and MLP
Sequence length	15,360
Generation allowance	5,120
Integrity controls	
Validation / test augmentation	Disabled
Ancestry and target checks	Passed

and S4 with `augmentation_enabled=true`; original manifests and hashes provide traceability.

Table 10: Shared optimisation settings, observed run and termination records, compute usage, and software versions for S3 (fine-tuned without augmentation) and S4 (fine-tuned with structure-preserving augmentation). Rows list configuration items and their values; shared settings are reported once, and slash-separated run-specific values are ordered S3/S4. GPU time and memory are reported in hours (h) and gigabytes (GB), respectively. Token loss is diagnostic only; final checkpoints are selected by validation Latent-Border Reconstruction Score (LBRS; higher is better).

Configuration	Value
Shared optimisation settings	
Optimizer	8-bit AdamW
Learning rate	1×10^{-4}
Weight decay	0.01
Learning-rate scheduler	Cosine
Maximum gradient norm	1.0
Per-device batch size	1
Gradient accumulation	8
Effective batch size	8
GPUs	1
Evaluation / saving interval	Every 25 optimizer steps
Logging interval	Every 5 optimizer steps
Early-stopping patience	40 evaluations
Early-stopping threshold	1×10^{-4}
Warm-up ratio	0.05
Observed run records (S3 / S4)	
Steps per epoch	95 / 189
Resolved warm-up steps	475 / 945
Run endpoint	1,475 / 1,700 optimizer steps
Endpoint epoch	15.528 / 8.995
Termination	Early stopping / training endpoint
Approx. training-record presentations	11,755 / 13,592
Compute (S3 / S4)	
GPU time	6.18 h / 7.15 h
Peak allocated GPU memory	36.02 GB / 36.03 GB
Peak reserved GPU memory	67.06 GB / 64.13 GB
Software environment	
Python	3.12.13
PyTorch / CUDA	2.8.0 / 12.8
Transformers / Transformers Reinforcement Learning (TRL) / Parameter-Efficient Fine-Tuning (PEFT) packages	5.3.0 / 0.22.2 / 0.19.1

E S5 GLM-OCR-Assisted Geometry-Refinement Algorithm

Inputs and integrity constraints. The frozen implementation consumes the source image and

parsed JSON from validation-selected S4 checkpoint 1,425; it reads no reference borders, metrics, or external-test annotations. Within the predicted table box, candidate text-like components are validated by `zai-org/GLM-OCR` with `do_sample=False`. Border transformations are deterministic conditional on the accepted content geometry. The raw S4 JSONL remains unchanged, and the validated refinement is written separately for S5 evaluation.

Ordered refinement procedure. The executable notebook applies the following stages in exactly this order:

1. **Content geometry extraction.** Detect candidate text-like boxes within the predicted table bbox, validate them with the OCR model, refit accepted boxes to tight ink, and optionally retain sufficiently ink-dense unrecognised candidates as safety boxes.
2. **First-internal-column gap completion.** Complete missing pieces only on the first internal vertical plane and only between existing fragments; a proposed piece is blocked when validated content or tight ink crosses its corridor.
3. **First-internal-row dangling-column completion.** Repair the first internal horizontal plane within independent header blocks, extending to qualifying dangling verticals without crossing top-connected hard barriers.
4. **First-row vertical colspan trimming.** Use first-row content anchors to shorten or remove intervening verticals that would incorrectly split a spanning header cell.
5. **Horizontal alignment.** Snap planes to safe whitespace corridors between content bands without changing segment x-extents.
6. **Whole-segment horizontal pruning.** Remove unsupported, duplicate, or multiline-gap horizontal segments under conservative visible-rule and topology guards.
7. **Vertical alignment.** Move planes to text-safe content-gap midpoints without changing segment y-ranges.
8. **Horizontal-to-vertical junction repair.** Relock raw-topology horizontal connections to the aligned vertical plane.
9. **Reciprocal vertical endpoint row lock.** Relock raw-topology vertical connections to the retained final horizontal row.
10. **First-column rowspan trimming.** Remove only the first-column portion of selected horizontal separators when content occupancy indicates a descriptor cell spanning subordinate rows.
11. **First-column content-envelope refinement.** Use OCR-confirmed width plus actual ink to refine first-column separator extents and protect true content envelopes.
12. **First-column content-top completion.** Add or complete a first-column top separator when content and crossing guards support the missing boundary.
13. **Post-first-column auto-connect.** Canonicalise the first internal column, close compatible vertical gaps, and resynchronise nearby horizontal endpoints without crossing content.
14. **Second-column local trim/top completion.** Apply a conservative coverage-aware correction to the second column, including an optional missing top separator and immediate-separator trimming under long-empty-region guards.
15. **Outer-boundary enforcement.** Replace near-bbox fragments with the exact predicted-bbox edges and restore the normalised schema.

Pipeline pseudocode.

```
input: source image I, raw S4 JSON P=(B,H,V)
C, masks <- detect_text_like_candidates(I, B)
C <- OCR_validate_and_refit(C, I)
V <- complete_first_internal_column_gaps(H,V,B,C,masks)
H,V <- complete_first_internal_row_dangling(H,V,B)
V <- trim_first_row_verticals_by_content(H,V,B,C)
H <- snap_horizontal_to_content_gaps(H,B,C,masks)
H <- prune_unsupported_horizontal_segments(H,V,B,C,masks)
V <- align_vertical_to_content_gap_midpoints(V,B,C,masks,H)
H <- repair_horizontal_endpoints_to_aligned_verticals(H,V)
V <- lock_vertical_endpoints_to_final_horizontal_rows(V,H)
H <- trim_first_column_rowspan_separators(H,V,B,C)
H <- refine_first_column_by_content_envelopes(H,C,masks)
H <- complete_first_column_content_top_separators(H,C,masks)
H,V <- post_first_column_autoconnect(H,V,B,C,masks)
H <- second_column_local_trim_and_top_completion(H,V,B,C)
H,V <- enforce_bbox_as_outer_boundary(H,V,B)
output: separate refined JSONL; raw S4 JSONL remains unchanged
```

Frozen implementation parameters. All stages are enabled. The `zai-org/GLM-OCR` model uses 64 maximum new tokens, batch size 8, a maximum padded-batch budget of 2.4M pixels, minimum crop size 192×64, and maximum crop side 640 px. Candidate boxes use 3 px recognition padding and 1 px tight geometry padding; candidates require at

Table 11: Fixed line-metric settings applied during validation-LBRS checkpoint selection and external-test evaluation. Rows report boundary-plane clustering and matching tolerances, snap-distance thresholds, and the metric implementation version; distances are in pixels (px).

Setting	Value
Boundary-plane clustering tolerance	3 px
Boundary-plane matching tolerance	30 px
Snap thresholds	5 / 10 / 20 / 30 px
Line-metric version	v1

least 4 ink pixels and 8 px^2 area, while unrecognised boxes are retained when ink ratio is at least 0.010. Horizontal and vertical plane grouping tolerances are 3 px. Outer-boundary matching/enforcement uses 6 px. Horizontal safe-corridor width is 3 px with 2 px content clearance; horizontal midpoint search uses 0.5 px steps. Horizontal pruning preserves visible rules at an overlap ratio of at least 0.50, uses a 2 px rule-search radius, a 0.15 minimum supported-x ratio, and 2–3 px duplicate plane/band tolerances. Vertical alignment uses 2 px content clearance, a 3 px minimum empty corridor, 0.5 px midpoint search, and a 3 px target-agreement tolerance. Junction and endpoint association tolerances are 8 px. First-column and second-column local rules use the exact constants in the frozen S5 parameter configuration distributed with the authoritative implementation.

Failure modes. A false OCR box may block a valid completion, whereas an OCR miss may expose an unsafe corridor. Pruning can remove a legitimate short separator, and local first/second-column rules can over-regularise unusual layouts.

F Complete LBRS Evaluation Procedure

Table 11 records the fixed line-metric settings used for both validation checkpoint selection and external-test evaluation.

This appendix specifies metric version `lbrs_interval_dice_no_bbox_v1`, which is the implementation used for checkpoint selection and the five-system evaluation. It defines the component formulae, algorithms, coordinate conventions, and edge cases underlying the compact score in Equation 2.

F.1 Validity Gate and Component Clipping

Let

$$\begin{aligned} P_i &= \mathbb{1}[\text{prediction } i \text{ is parsed as JSON}], \\ S_i &= \mathbb{1}[\text{prediction } i \text{ satisfies the schema}], \end{aligned} \tag{7}$$

so that $G_i = P_i S_i$. The schema requires exactly the top-level keys `table_bbox_norm` and `completed_borders`, whose border object contains the horizontal and vertical segment lists. Bbox and segment coordinates must be finite and in $[0, 1]$; bbox and segment endpoints must increase. Missing, non-numeric, or non-finite values are invalid, and any parse or schema failure sets $G_i = 0$.

The implementation applies

$$\operatorname{clip}^*_{[0,1]}(x) = \begin{cases} 0, & x \text{ invalid}, \\ \min(1, \max(0, x)), & \text{otherwise}, \end{cases} \tag{8}$$

to every weighted component.

F.2 Coordinate Conversion

The table box is normalised relative to the complete image, whereas border-segment coordinates are normalised relative to that table box. Predicted and reference segments are each converted using their corresponding box. For an image of width W_i and height H_i, and a table box (l, t, r, b), a table-relative coordinate (u, v) is mapped to pixels as

$$\begin{aligned} \pi_x(u) &= \operatorname{round}[(l + u(r - l))(W_i - 1)], \\ \pi_y(v) &= \operatorname{round}[(t + v(b - t))(H_i - 1)]. \end{aligned} \tag{9}$$

Thus, a horizontal segment $[u_1, u_2, v]$ becomes the pixel interval $[\pi_x(u_1), \pi_x(u_2)]$ on plane $\pi_y(v)$, while a vertical segment $[u, v_1, v_2]$ becomes $[\pi_y(v_1), \pi_y(v_2)]$ on plane $\pi_x(u)$. The four bbox edges are generated from (l, t, r, b) using the same $W_i - 1$ and $H_i - 1$ convention.

For predicted and reference image-normalised boxes $\widehat{B}_i$ and $B_i^\star$, table localisation is continuous-coordinate intersection over union,

$$B_i = \frac{|\widehat{B}_i \cap B_i^\star|}{|\widehat{B}_i \cup B_i^\star|}, \tag{10}$$

where non-overlapping boxes have zero intersection. Schema-valid boxes have positive area, so the union denominator is non-zero; invalid boxes are handled by the validity gate.

F.3 Boundary-Plane Extraction and Clustering

Broken, duplicated, or fragmented pieces on approximately the same axis are collapsed into one structural plane. For horizontal segments, the plane coordinate is y; for vertical segments, it is x. Let $z_1 \le \cdots \le z_m$ be the sorted finite plane coordinates. Sequential clustering assigns z_j to the current cluster $\mathcal{C}$ when

$$|z_j - \mathrm{median}(\mathcal{C})| \le \delta_{\mathrm{cluster}}, \quad \delta_{\mathrm{cluster}} = 3 \text{ px}; \tag{11}$$

otherwise, it begins a new cluster; each cluster is represented by its median coordinate.

For count and snap-position scoring, the top and bottom bbox edges are added to the horizontal set and the left and right bbox edges to the vertical set before clustering. For interval Dice, only explicitly predicted and annotated segments are clustered; bbox-derived edges are excluded.

F.4 Boundary-Derived Counts

Let $P_{H,i}$ and $E_{H,i}$ be the predicted and expected numbers of bbox-inclusive horizontal planes, and let $P_{V,i}$ and $E_{V,i}$ be the corresponding vertical-plane counts. The boundary-derived row and column counts are

$$\begin{aligned} \widehat{r}_i &= \max(0, P_{H,i} - 1), \\ r_i &= \max(0, E_{H,i} - 1), \\ \widehat{c}_i &= \max(0, P_{V,i} - 1), \\ c_i &= \max(0, E_{V,i} - 1). \end{aligned} \tag{12}$$

For predicted and reference counts $\widehat{n}$ and n, respectively, the count component is

$$S_{\mathrm{count}}(\widehat{n}, n) = \mathrm{clip}_{[0,1]}\left(1 - \frac{|\widehat{n} - n|}{\max(1, n)}\right). \tag{13}$$

The score is one when both counts are zero. These are boundary-derived rather than semantic row and column counts.

F.5 Greedy One-to-One Plane Matching

For predicted plane positions $\widehat{\mathcal{Z}} = \{\widehat{z}_1, \ldots, \widehat{z}_P\}$ and expected positions $\mathcal{Z} = \{z_1, \ldots, z_E\}$, candidate pairs are

$$\mathcal{A} = \{(j,k) : |\widehat{z}_j - z_k| \le \delta_{\mathrm{match}}\}, \quad \delta_{\mathrm{match}} = 30 \text{ px}. \tag{14}$$

Candidates are sorted by ascending absolute distance; equal-distance candidates retain their deterministic lexicographic (j,k) enumeration order. A pair is accepted only if both planes remain unused. This greedy, non-Hungarian procedure is applied independently by axis; unmatched planes remain unpaired.

F.6 Snap-Position Score and MAE

For a matched plane error d, the snap-readiness quality function is

$$q(d) = \begin{cases} 1, & 0 \le d \le 5, \\ 1 - 0.03(d-5), & 5 < d \le 10, \\ 0.85 - 0.025(d-10), & 10 < d \le 20, \\ 0.60 - 0.03(d-20), & 20 < d \le 30, \\ 0, & d > 30. \end{cases} \tag{15}$$

For axis $A \in \{H, V\}$, let $\mathcal{M}_{A,i}$ be the bbox-inclusive match set, $M_{A,i} = |\mathcal{M}_{A,i}|$, and $d^A_{m,i}$ the absolute pixel error of match m. When $P_{A,i} = E_{A,i} = 0$, the snap-position score is defined as one. Otherwise,

$$S^{\mathrm{snap}}_{A,i} = \frac{\sum_{m=1}^{M_{A,i}} q(d^A_{m,i})}{\max(P_{A,i}, E_{A,i})}. \tag{16}$$

This is $R_{s,i}$ for $A = H$ and $C_{s,i}$ for $A = V$. Extra or missing planes contribute zero through the denominator. Diagnostic MAE is the arithmetic mean of matched-plane errors and is unavailable when $M_{A,i} = 0$; unmatched planes do not enter it.

F.7 Interval Unions and Interval Dice

Explicit horizontal and vertical pieces become x- and y-intervals, respectively. Invalid or zero-length pieces are removed; sorted overlapping or endpoint-touching intervals are merged, so duplicate coverage counts once. For union U, $\mu(U)$ is its summed non-overlapping length, and $\mu(\widehat{U} \cap U)$ is computed by a two-pointer traversal of the merged lists.

Explicit predicted and expected planes are greedily matched by the same 30-pixel positional rule. For a matched predicted/reference pair with interval unions $\widehat{U}$ and U, the per-plane similarity is

$$D(\widehat{U}, U) = \frac{2\mu(\widehat{U} \cap U)}{\mu(\widehat{U}) + \mu(U)}. \tag{17}$$

If both interval-union lengths are zero, the implementation defensively returns one, although zero-length pieces are normally removed beforehand.

Let $P^D_{A,i}$ and $E^D_{A,i}$ be the numbers of predicted and expected explicit planes, and $\mathcal{M}^D_{A,i}$ their match

set. When $P^D_{A,i} = E^D_{A,i} = 0$, the axis interval-Dice score is defined as one. Otherwise,

$$S^D_{A,i} = \frac{\sum_{(j,k)\in\mathcal{M}^D_{A,i}} D(\widehat{U}^A_{j,i}, U^A_{k,i})}{\max(P^D_{A,i}, E^D_{A,i})}. \quad (18)$$

This gives $R_{d,i} = S^D_{H,i}$ and $C_{d,i} = S^D_{V,i}$; one-sided or otherwise unmatched explicit planes contribute zero.

F.8 Dataset Aggregation

For N tables, the official score is Equation 3, the unweighted mean of the per-table LBRS values. Each table therefore has equal influence regardless of image dimensions or number of predicted segments. Parse success, schema validity, bbox IoU, count, snap, and interval-Dice metrics are also reported as macro means over per-table values. Horizontal and vertical MAE are averaged only over tables for which the corresponding MAE is available.

The implementation also reports a non-official compatibility diagnostic obtained by inserting macro-averaged components into Equation 2; it may differ because the official gate is applied per table before averaging. The official weights and interpretation are given in Section 5.4; Table 4 reports the Results-section robustness audit.

F.9 Paired Statistical Reporting

For each of the four ordered system contrasts, the implementation computes the later-minus-earlier LBRS difference for every external-test table. It reports the mean, median, and standard deviation of those differences, along with the numbers that are positive, zero, or negative. Paired bootstrap 95% confidence intervals for the mean difference use 20,000 resamples of the 100 aligned table pairs and random seed 42. These intervals quantify variation across the evaluated tables; they do not estimate variability across independently trained models.

G Validation Checkpoint Selection

All checkpoints saved every 25 optimiser steps are evaluated with the same inference configuration on the fixed 32-table validation subset: 59 checkpoints at steps 25–1,475 for S3 and 68 at steps 25–1,700 for S4. Figures 5 and 6 show the two validation trajectories. The unique maximum unrounded macro LBRS is selected independently for each condition; no external-test prediction or score enters either ranking. Table 12 reports the selected and runner-up checkpoints: checkpoint 525 for S3 (96.737, 0.153 points above checkpoint 1,050) and checkpoint 1,425 for S4 (97.107, 0.167 points above checkpoint 1,575).

Table 13 gives the five highest-scoring checkpoints for each condition.

The minimum validation losses are 0.066548 for S3 and 0.066539 for S4. Their checkpoints differ from the task-metric selections: step 475 for S3 obtains 95.022 validation LBRS (1.715 below checkpoint 525), while step 750 for S4 obtains 95.641 (rank 21; 1.466 below checkpoint 1,425). Thus, token loss is diagnostic rather than the structural ranking criterion. The supplementary package includes the complete 59-row and 68-row validation leaderboards and top-10 extracts as machine-readable CSV files, together with the checkpoint-ranking provenance.

H Extended Results, Per-Table Reporting, and Qualitative Audit

H.1 Complete Quantitative Records

The supplementary package provides full component and per-table external-test records for S1–S5, together with paired summaries and per-table differences for each adjacent contrast. The machine-readable external-test records include sample and system provenance; validity and bbox IoU; horizontal and vertical count, snap, interval-Dice, and MAE metrics; LBRS; the selected checkpoint; and geometry-refinement status. Figure 11 provides a compact per-table view. Appendix G documents the separate validation leaderboards.

H.2 Extended Aggregate and Paired Diagnostics

The following records preserve the complete aggregate, conditional-validity, distributional, and component-level evidence underlying the concise RQ summaries in Section 5.

Prompting diagnostics. Under S1, 45 outputs are unparsable and a further 21 fail the schema; the validity gate therefore assigns zero to 66 tables and yields a median LBRS of zero. Among the 34 valid predictions, mean bbox IoU is 97.544% and mean LBRS is 45.997, but horizontal-count recovery (20.853%) remains far below vertical-count recovery (74.428%). Thus, occasional strong geometry—including a best-table LBRS of 88.408—does not offset unreliable compliance and row-structure recovery.

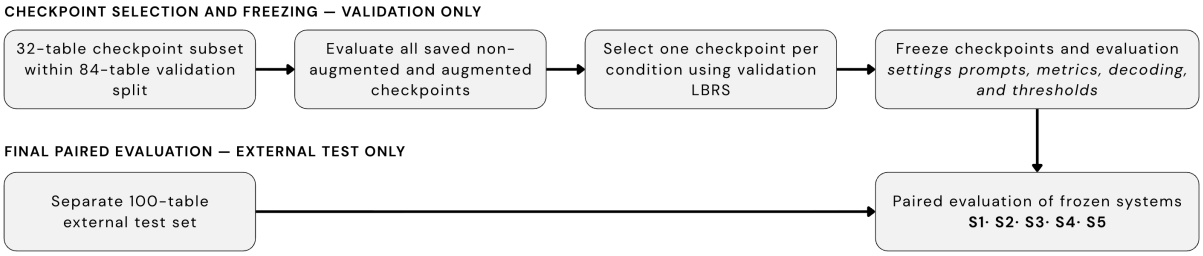


Figure 4: Validation-only checkpoint selection and external-test evaluation protocol. In the upper flow, all saved checkpoints from the non-augmented S3 and augmented S4 fine-tuning conditions are evaluated on a fixed 32-table subset of the 84-table validation split; one checkpoint per condition is selected using the Latent Border Reconstruction Score (LBRS), after which checkpoints, prompts, metrics, decoding settings, and thresholds are frozen. In the lower flow, the separate 100-table external test set is used only for paired evaluation of the frozen systems—prompting baselines S1–S2, fine-tuned systems S3–S4, and S5 refinement of S4. Arrows indicate the sequence and show that the frozen configurations and held-out test data meet only at final evaluation.

Table 12: Top-ranked and runner-up checkpoints for the S3 no-augmentation and S4 augmentation conditions, selected independently on the fixed 32-table validation subset using macro Latent Border Reconstruction Score (LBRS; 0–100, ↑ higher is better). Columns report the optimiser steps and LBRS values of the selected and runner-up checkpoints, the selected-minus-runner-up gap in LBRS points, and tie-break status. **Boldface** marks each selected step and score; both maxima are unique, so no tie-break was required.

Condition	Selected step	Val. LBRS ↑	Runner-up step	Runner-up LBRS	LBRS gap	Tie-break
S3: no augmentation	**525**	**96.737**	1,050	96.584	+0.153	Not required
S4: augmentation	**1,425**	**97.107**	1,575	96.940	+0.167	Not required

S2 reduces the gated-zero count to 26 (25 unparsable outputs and one additional schema failure) and raises median LBRS to 41.021. Among its 74 valid predictions, mean bbox IoU is 97.977% and mean LBRS is 52.098. The paired median gain over S1 is 22.832 points: 66 tables improve, 19 are unchanged, and 15 degrade. The validity transitions are 47 invalid-to-valid, 27 valid-to-valid, seven valid-to-invalid, and 19 invalid-to-invalid. Vertical interval Dice improves by only 2.467 points, so prompting recovers divisions more readily than their bounded extents. Conditional MAE is not directly comparable across systems with markedly different validity coverage. These diagnostics support the RQ1 prompting result in Section 5.

Fine-tuning diagnostics. Validation-selected checkpoint 525 (S3) has a median LBRS of 91.987, an interquartile range of 87.587–94.239, and a range of 55.242–99.356; 69 tables score at least 90, 20 at least 95, and nine below 80. Relative to S2, the median paired gain is 51.414 points: 98 tables improve and two degrade, and all 26 previously invalid outputs become valid while the other 74 remain valid. The complete component values are in Table 3 and the supplementary records; the distribution and validity transitions substantiate the dominant fine-tuning effect reported for RQ1 in Section 5.

Augmentation diagnostics. The independently selected checkpoints are 525 for S3 and 1,425 for S4, using the same fixed validation protocol and matched documented configuration. Their training sets contain 757 original records and, for S4, 754 additional accepted text-randomised variants. Because the runs present approximately 11,755 and 13,592 records, respectively, this is an augmentation-with-additional-exposure comparison rather than an isolation of text diversity.

S4 has a median LBRS of 94.265, an interquartile range of 91.908–97.167, and a range of 63.486–99.787; 89 tables score at least 90, 43 at least 95, and four below 80. Relative to S3, the median gain is 1.793 points, with 69 improvements and 31 regressions. Vertical interval Dice supplies the largest component gain (+9.420 points). Horizontal MAE falls by 1.054 pixels, whereas vertical MAE rises by 2.371 pixels despite improved vertical snap and interval overlap; these metrics therefore capture complementary properties of matched boundaries. The aggregate RQ2 result and exposure qualification are given in Section 5.

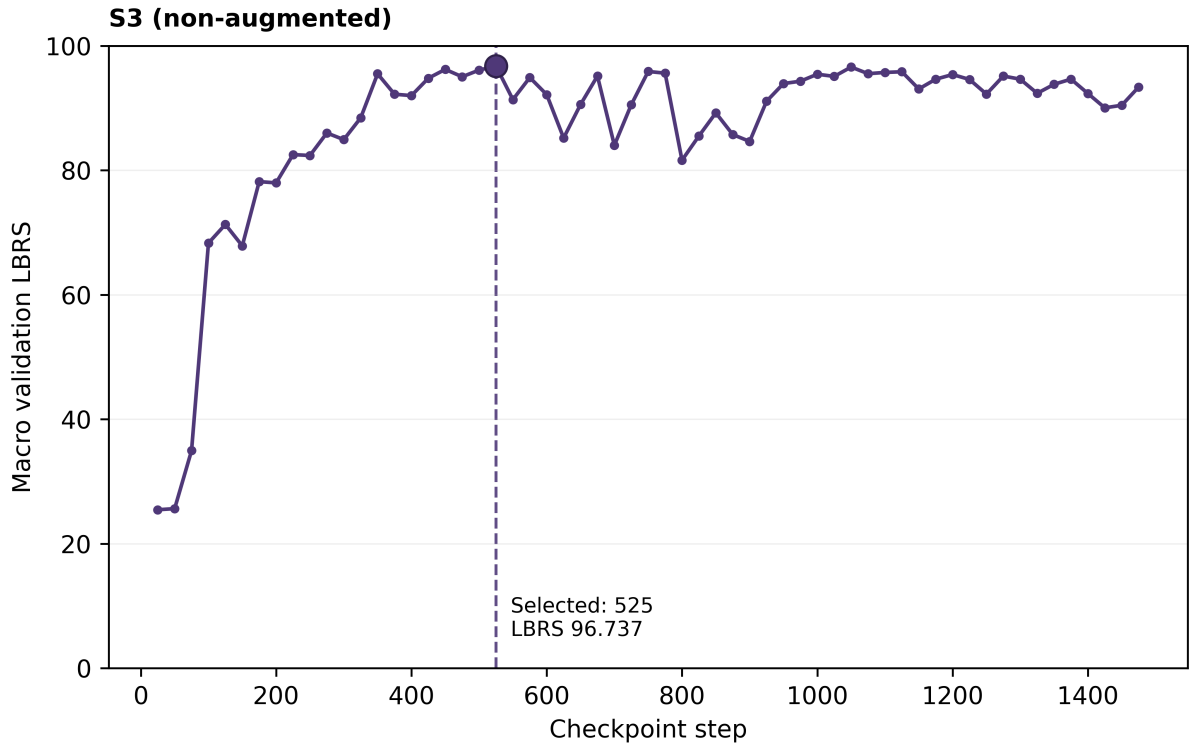


Figure 5: Validation checkpoint selection for S3, the non-augmented fine-tuned system, on the fixed 32-table validation subset. Points show macro Latent Border Reconstruction Score (LBRS; 0–100, higher is better) for 59 saved checkpoints at optimiser steps 25–1,475. The enlarged point and vertical dashed line mark checkpoint 525, selected as the unique maximum (LBRS 96.737); solid connecting lines guide visual tracing between discrete evaluations only.

Geometry-refinement diagnostics. The S5–S4 comparison is strictly paired because refinement operates on the exact raw S4 outputs. Although 90 tables improve and ten degrade, the paired median gain of 3.191 points contrasts with the mean gain of 0.467 reported in Section 5.

Refinement raises vertical snap-position from 49.374% to 82.512% and vertical interval Dice from 64.941% to 90.531%, while reducing vertical MAE from 13.886 to 7.770 pixels. The trade-off is a fall in vertical-count accuracy from 99.923% to 92.340%. Eight regressions exceed 30 LBRS points, each corresponding to a collapse in otherwise near-perfect S4 vertical-count recovery; Figure 10 shows the largest case. These diagnostics support the Section 5 interpretation of S4 as the primary predictor and S5 as a refinement analysis.

H.3 Qualitative Selection

Table 14 fixes one typical case, the two paired refinement extremes, and one complex-layout case, avoiding post-hoc visual selection. Figures 7, 8, 9, and 10 show these four cases. Each figure uses the same eight panels: input, reference, S1–S5, and an S4→S5 operation view in which retained, moved or trimmed, added, and removed borders are distinguished.

H.4 Unified Per-Table S1–S5 Heatmap

Figure 11 shows the paired progression for all 100 external-test tables. Rows are ordered by descending S4 LBRS, concentrating the hardest raw-model cases at the bottom; the principal pattern is the broad fine-tuning gain followed by smaller, non-uniform refinement changes, including conspicuous regressions.

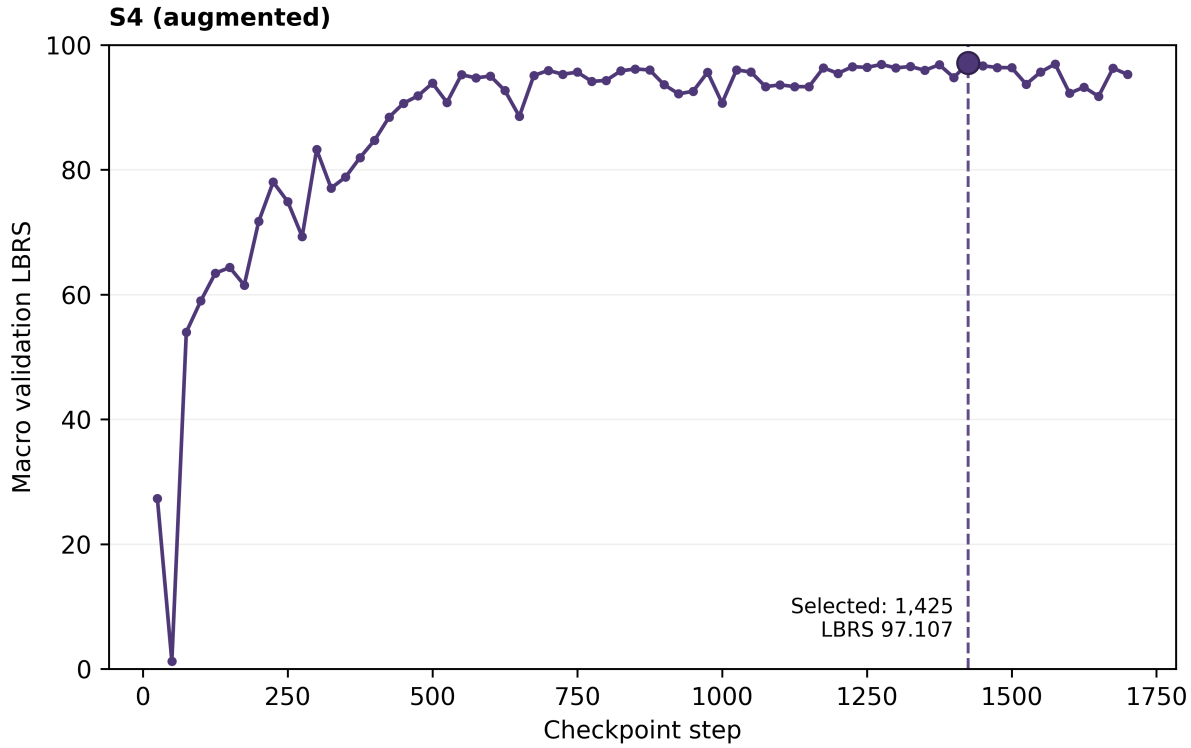


Figure 6: S4 checkpoint selection on the fixed 32-table validation subset. S4 is fine-tuned with structure-preserving augmentation; each point shows macro Latent Border Reconstruction Score (LBRS; 0–100, higher is better) for one of 68 checkpoints at steps 25–1,700. The enlarged point and vertical dashed line mark the unique maximum at step 1,425 (97.107); solid connecting lines are visual guides only.

Table 13: Five checkpoints with the highest macro Latent Border Reconstruction Score (LBRS; 0–100) for each fine-tuning condition—S3 (no augmentation) and S4 (augmentation)—on the fixed 32-table validation subset. Rows rank checkpoints independently within each condition; columns report the optimiser step, validation LBRS used for ranking, and macro diagnostic scores (%) for bounding-box intersection over union (BBox IoU) and boundary-derived row and column counts. ↑ indicates higher is better; **boldface** marks the selected checkpoint's step and LBRS.

Rank	Step	Val. LBRS ↑	BBox IoU (%) ↑	Row count (%) ↑	Col. count (%) ↑
S3: no augmentation					
1	**525**	**96.737**	99.220	98.744	100.000
2	1,050	96.584	99.116	98.906	100.000
3	450	96.210	98.969	98.902	99.609
4	500	96.094	98.954	99.349	100.000
5	750	95.873	99.155	97.725	99.688
S4: augmentation					
1	**1,425**	**97.107**	98.855	99.349	100.000
2	1,575	96.940	99.001	99.141	100.000
3	1,275	96.873	98.890	99.010	100.000
4	1,375	96.828	98.859	98.718	100.000
5	1,450	96.658	98.910	98.828	99.760

Table 14: Selection criteria and analysis purposes for four qualitative audit cases from the fixed 100-table external test set. Rows give the case label, sample ID, selection criterion, and intended qualitative purpose, covering typical S5 performance, the largest positive and negative refinement changes, and a preselected complex layout. S4 is the unrefined augmented fine-tuned system, S5 applies GLM-OCR-assisted geometry refinement to S4, and LBRS denotes Latent-Border Reconstruction Score (0–100; higher is better).

Case	Sample	Selection criterion	Primary purpose
Q1	65	Closest to median S5 LBRS	Typical refined-system behaviour
Q2	89	Largest S5–S4 LBRS increase	Largest refinement improvement
Q3	63	Largest S5–S4 LBRS decrease	Largest refinement regression
Q4	6	Preselected complex-layout case	Grouped headers, spans, and local-panel structure

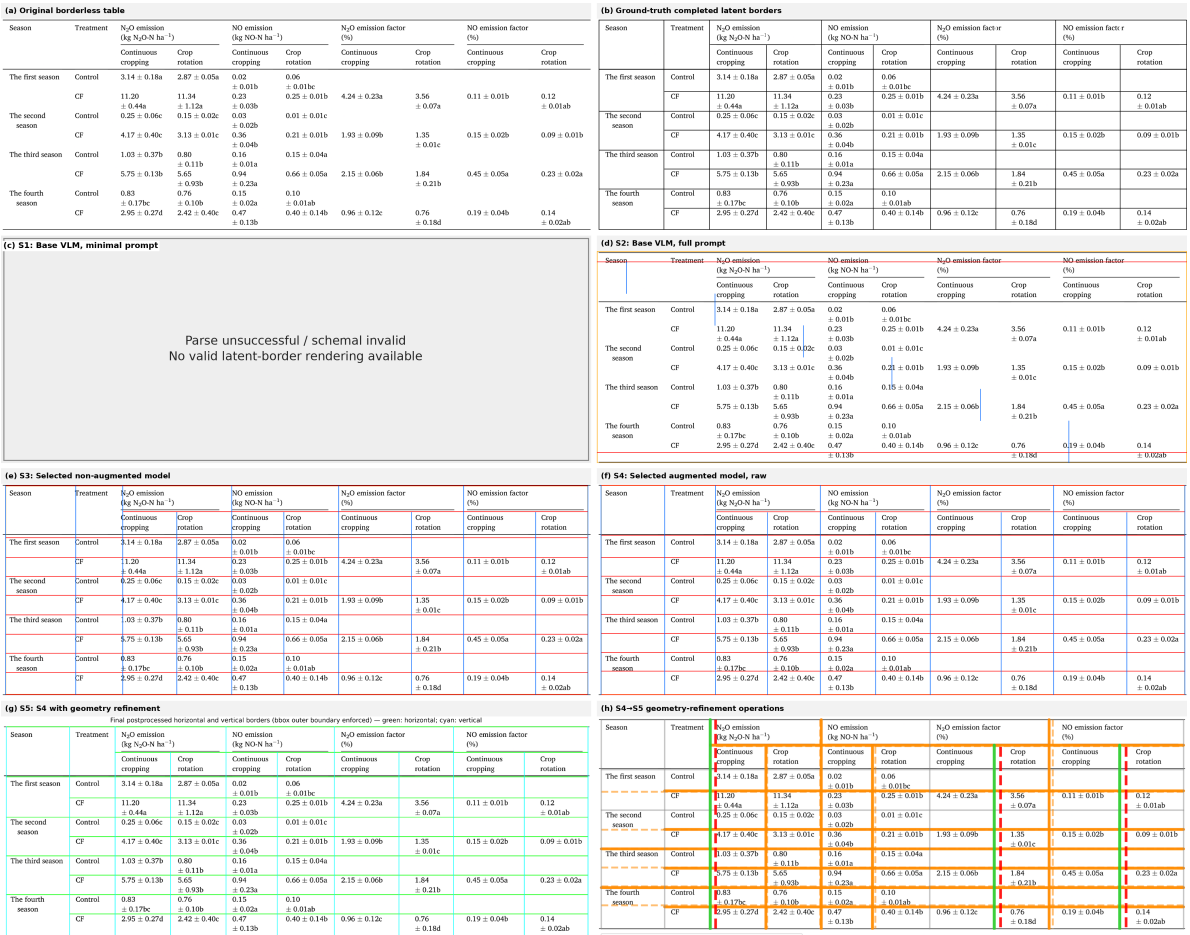


Figure 7: Qualitative latent-border reconstruction for Q4 (sample 6), the preselected complex-layout external-test case with grouped headers, row spans, and local partial boundaries. (a) Original borderless table; (b) ground-truth completed borders; and (c–g) outputs from S1–S5, respectively: base vision–language model (VLM) with minimal and full prompts, non-augmented and augmented fine-tuned models, and geometry-refined S4. Panel (c) has no rendering because the S1 output is unparsable or schema-invalid. In (d–f), red and blue denote horizontal and vertical predicted borders; in (g), green and cyan denote the refined horizontal and vertical borders. (h) S4-to-S5 operations: gray retained, orange moved or trimmed, green added, and red dashed removed. S4 recovers the principal topology, while S5 primarily adjusts geometry and preserves the spanning structure.

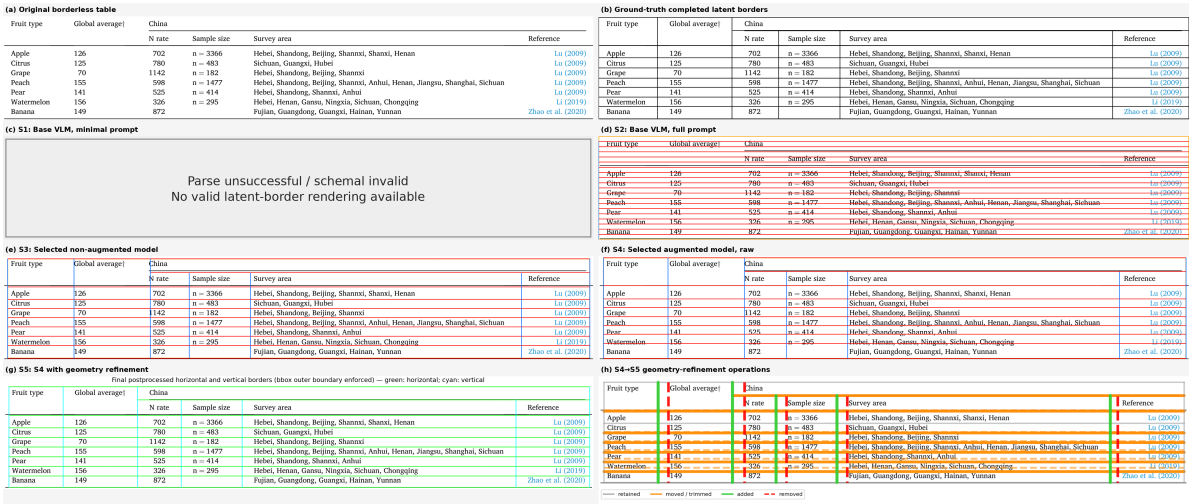
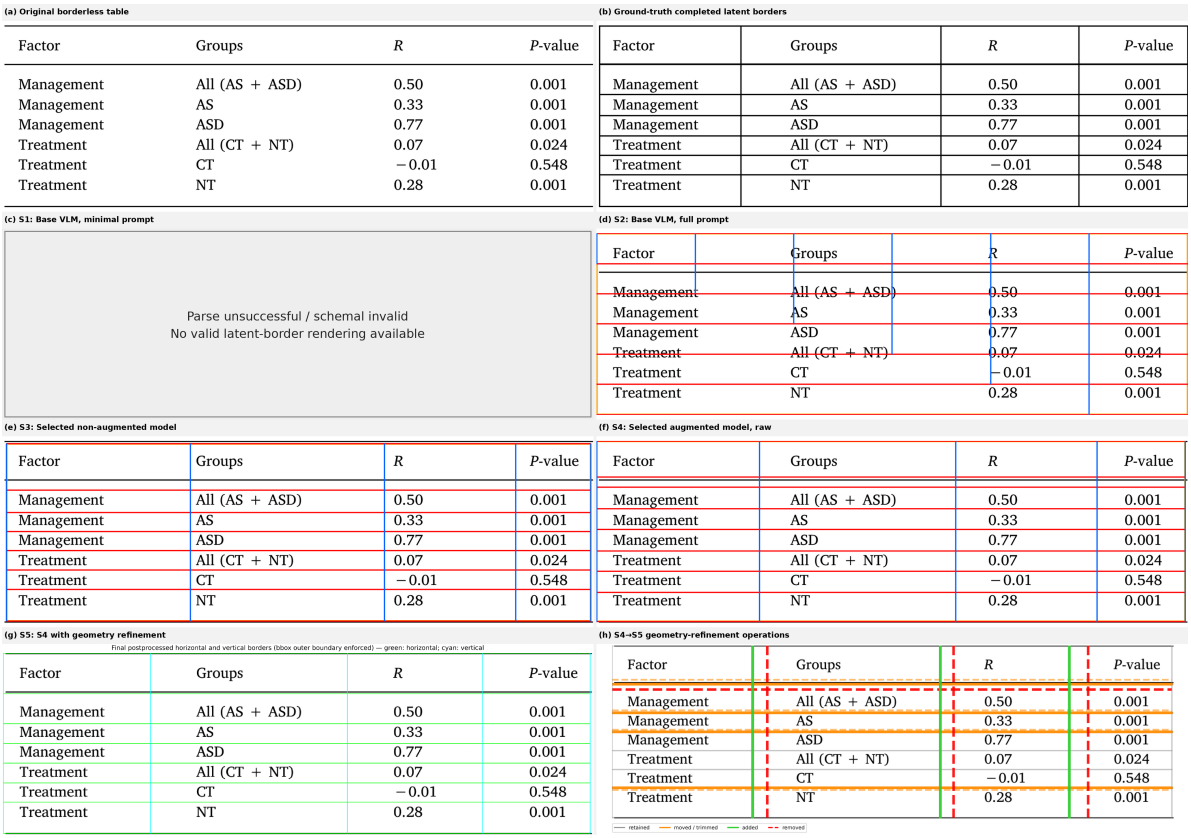


Figure 8: Qualitative latent-border reconstruction for Q1 (external-test sample 65), selected as the case closest to the median S5 Latent Border Reconstruction Score (LBRS). (a) Original borderless table; (b) ground-truth completed borders; and (c–g) outputs from S1–S5, respectively: the base vision–language model (VLM) with minimal and full prompts, the non-augmented and augmented fine-tuned models, and geometry-refined S4. Panel (c) has no rendering because the S1 output is parse- or schema-invalid. In (d–f), red and blue denote predicted horizontal and vertical borders; in (g), green and cyan denote the refined horizontal and vertical borders. (h) S4-to-S5 refinement operations: gray retained, orange moved or trimmed, green added, and red dashed removed.



Factor	Groups	*R*	*P*-value
Management	All (AS + ASD)	0.50	0.001
Management	AS	0.33	0.001
Management	ASD	0.77	0.001
Treatment	All (CT + NT)	0.07	0.024
Treatment	CT	−0.01	0.548
Treatment	NT	0.28	0.001

Figure 9: Qualitative latent-border reconstruction for case Q2 (external-test sample 89), selected for the largest S5–S4 improvement in Latent Border Reconstruction Score (LBRS; +12.406 points, higher is better). (a) Original borderless table; (b) ground-truth completed latent borders; and (c–g) outputs from S1–S5, respectively: the base vision–language model (VLM) with minimal and full prompts, the non-augmented and augmented fine-tuned models, and the geometry-refined S4 output. Panel (c) has no rendering because the S1 output is parse- or schema-invalid. In (d–f), red and blue denote predicted horizontal and vertical borders; in (g), green and cyan denote their refined counterparts. (h) S4-to-S5 operations: gray retained, orange moved or trimmed, green added, and red dashed removed.

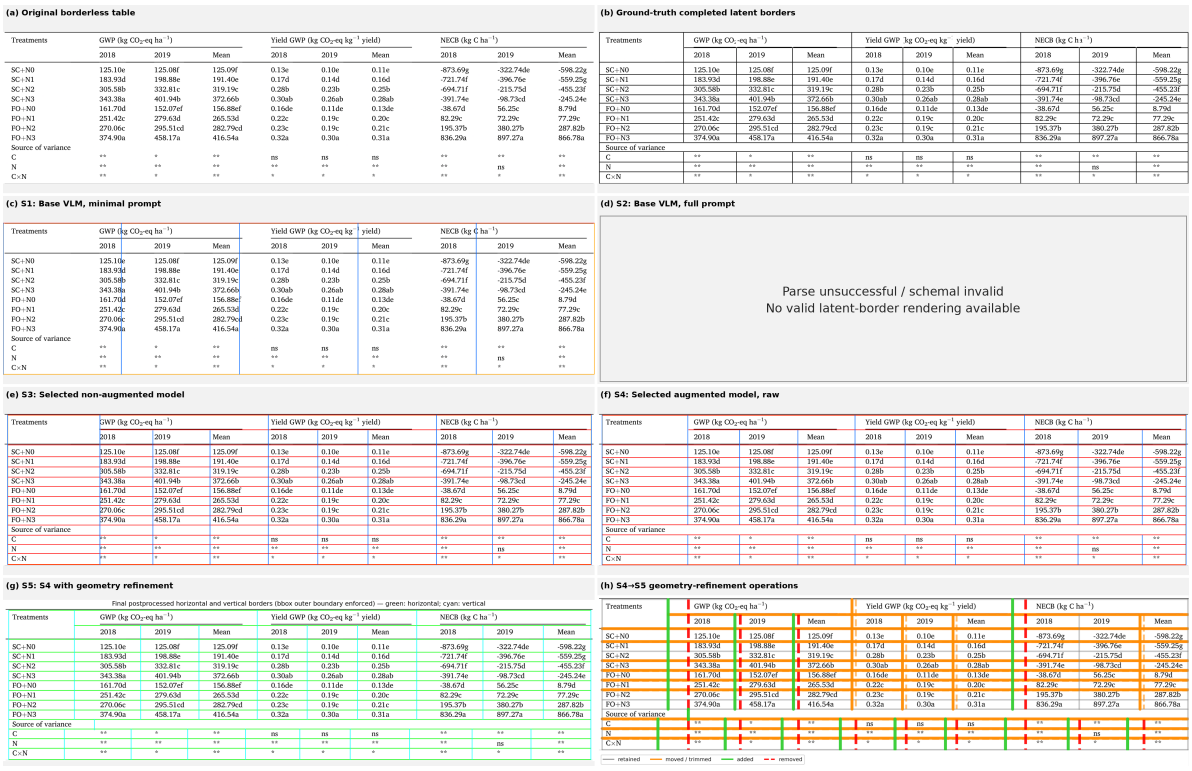


Figure 10: Qualitative latent-border reconstruction for Q3 (external-test sample 63), selected for the largest S5–S4 decrease in Latent Border Reconstruction Score (LBRS; -41.802 points on a 0–100 scale, higher is better). (a) Original borderless table; (b) ground-truth completed borders; and (c–g) outputs from S1–S5, respectively: the base vision–language model (VLM) with minimal and full prompts, the non-augmented and augmented fine-tuned models, and the geometry-refined S4 output. Panel (d) has no rendering because the S2 output is unparsable or schema-invalid. In (c), (e), and (f), red and blue denote predicted horizontal and vertical borders; in (g), green and cyan denote their refined counterparts. (h) S4-to-S5 refinement operations: gray retained, orange moved or trimmed, green added, and red dashed removed.

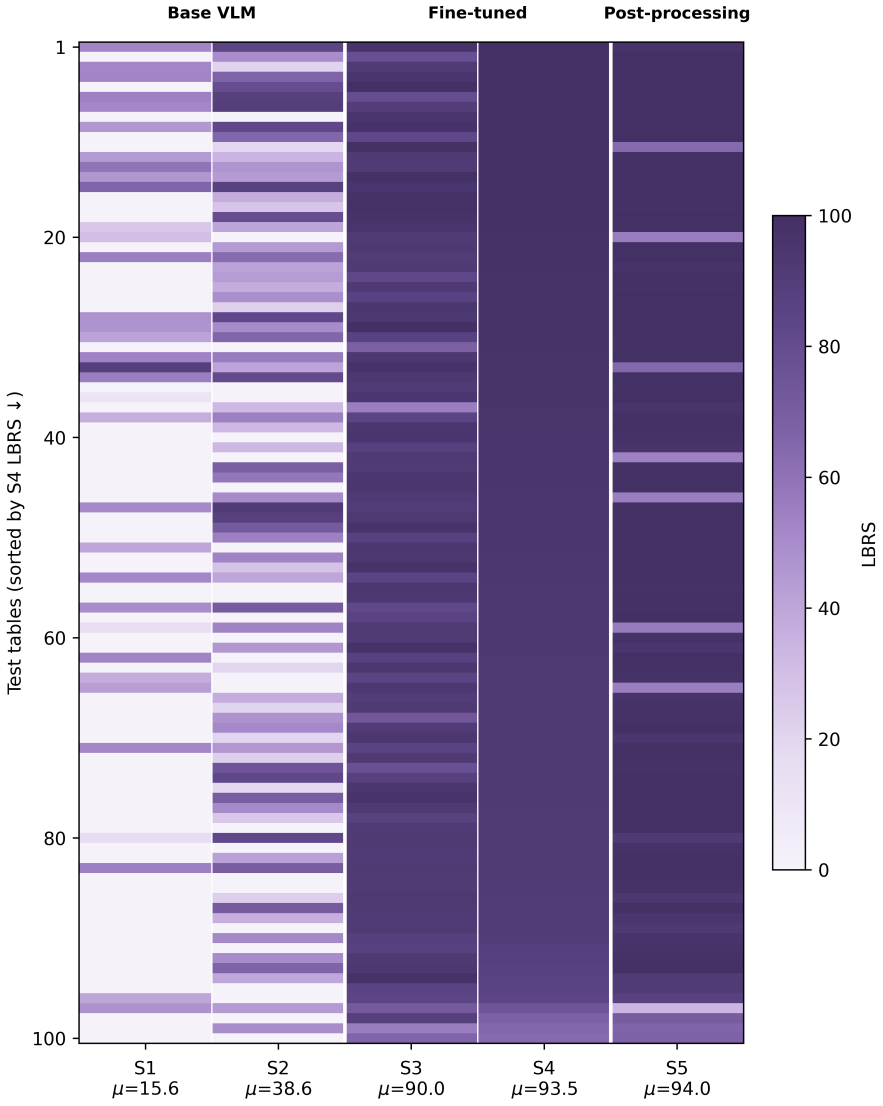


Figure 11: Per-table Latent Border Reconstruction Score (LBRS; 0–100, higher is better) for five systems on the 100-table external test set. Columns show prompting-only base vision–language model (VLM) baselines S1–S2, fine-tuned systems S3–S4, and S5 post-processing of S4; each cell is one table's score, with darker purple indicating higher LBRS. Rows align the same table across systems and are sorted from top to bottom by decreasing S4 score. μ denotes each system's mean over all 100 tables, and paired S4–S5 cells show the effect of refinement on each S4 output.

Silent Structured-Content Corruption in Document Parsing: A Span-Typed Diagnostic and a Released Cross-Parser Resource

Ruiyang Zhang
FLock.io & Ryonix Labs Inc.
zhangruiyang36@gmail.com

Abstract

Document-AI pipelines increasingly begin by converting a PDF to text with a parser—classical extractors, OCR, layout models, or vision-language models—before any retrieval or reasoning. Overall-quality metrics (edit distance, TEDS) score these parsers as if all characters mattered equally. They do not: a corrupted number, date, identifier, or table cell changes meaning while barely moving a similarity score. We isolate this failure with a span-typed diagnostic over six parsers, ten structured span types, and two conditions (digital and scanned), on a controlled corpus with exact ground truth plus a real arXiv subset. Three findings. (i) Preservation is highly parser- and type-dependent: the best parser preserves 99.4% of spans while others fail specific types catastrophically (Docling drops MARKUP to 6.0%; MarkItDown returns nothing on scanned pages), so "which parser" is a per-span-type risk decision, not a single score. (ii) Document-level text similarity does *not* predict span loss (Spearman $\rho \leq 0.46$ across all parsers and conditions; ≈ 0 for VLMs), so standard metrics are blind to it. (iii) Downstream, corrupted spans are dangerous precisely when the parse *looks* successful: among questions whose evidence span was corrupted but whose document still returned text, 96.5% are answered with a confident wrong value rather than an abstention—an undetectable error—whereas total parse failure is comparatively safe because it forces an abstention. We release the resource behind these numbers: 566,904 per-span verdicts over 660 documents and 7,560 parser outputs, under CC-BY, with an independent verifier (49,720 checks) that re-derives every reported number.

1 Introduction

Retrieval-augmented and document-QA systems rarely see a PDF directly. A *parser* first turns it into text or Markdown—PyMuPDF, Tesseract, layout models such as Docling, or a vision-language model prompted to transcribe the page—and everything downstream consumes that text. The parser is therefore the point at which document evidence either survives or is silently corrupted before any model reasons over it.

The field evaluates parsers with document-level fidelity metrics: normalized edit distance for text, TEDS (Zhong et al., 2020) for tables. These treat every character as equally important. But in the documents that matter—financial, legal, scientific, clinical—meaning is carried by *structured spans*: numbers, dates, currency amounts, identifiers, URLs, code, markup, quotations, and table cells. Corrupting one of these (a transposed digit, a dropped currency symbol, a mangled ID) changes what the document says while barely denting an edit-distance score, and no downstream component flags it. Existing parser benchmarks (Ouyang et al., 2025; Xu et al., 2026; Li et al., 2025) rank overall quality; a growing line shows OCR quality does not predict downstream RAG performance (Sun et al., 2026). We complement these with a targeted, *span-typed* audit of the specific failure mode—silent, meaning-changing corruption—and release the per-span verdicts so parsers can be audited on this axis.

Contributions. (i) A controlled, span-typed measurement of structured-content preservation across six parsers, ten span types, and digital/scanned conditions, with exact ground truth (§3). (ii) Evidence that document similarity does not predict span loss—metric blindness (§4). (iii) A downstream QA probe showing that the dangerous regime is *partial* corruption behind a successful-looking parse (96.5% confident-wrong), not total failure (§5); plus a corruption taxonomy, an in-table penalty, and honest OCR-resolution and real-PDF controls (§6). (iv) A released CC-BY resource of 566,904 verdicts and a 49,720-check verifier.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 273–281
October 24, 2026 ©2026 Association for Computational Linguistics

type	inj. recall	prose prec.	prose rec.
DATE	1.00 (5408/5408)	1.00	0.96
ID	1.00 (3473/3473)	0.91	1.00
CURRENCY	1.00 (6912/6912)	1.00	1.00

Table 1: Detector validation for the added span types (hand-audited). Recall on injected spans is exact; prose precision/recall are on held-out human-labelled samples.

parser	cond.	docs	USD
GPT-4.1	digital	600	4.75
GPT-4.1	scanned	600	4.78
GPT-4.1-mini	digital	600	1.58
GPT-4.1-mini	scanned	600	1.59
VLM real-PDF subset (60 each)		120	1.42

Table 2: API cost. Local parsers (PyMuPDF4LLM, MarkItDown, Docling, Tesseract) are free; total VLM spend ≈$14.

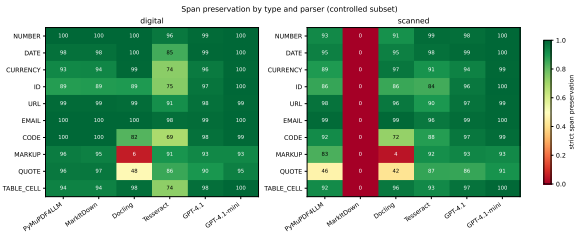


Figure 1: Span preservation by parser × type × condition (all ten types). The best parser preserves 99.4%; the damage concentrates in specific parser/type cells, not uniformly.

2 Setup

Corpus and ground truth. We render a controlled corpus of 600 documents from text carrying *known* structured spans, to digital PDF and to a rasterized "scanned" image PDF (so the OCR path is exercised). Ground truth is the exact injected span strings and their loci; frozen detectors (validated below) recover them from parser output. A real subset of 60 public arXiv PDFs provides ecological validity, with spans taken from each paper's own text layer.

Parsers. Four local, free, CPU-only tools—PyMuPDF4LLM (Artifex Software, 2024), MarkItDown (Microsoft, 2024), Docling (Auer et al., 2024), Tesseract (Smith, 2007)—and two vision-language parsers, GPT-4.1 and GPT-4.1-mini prompted to transcribe each page. Both VLMs get one frozen prompt at temperature 0, and both conditions are downscaled to a common pixel budget so the conditions differ in image *quality* and not in how many pixels the model is handed; the prompt is reproduced verbatim in Appendix C. No GPU was used; total API cost was ≈$14.

Span types and matcher. Ten types: NUMBER, DATE, CURRENCY, ID, URL, EMAIL, CODE, MARKUP, QUOTE, TABLE_CELL. A span is *preserved* iff it appears intact in the output under a pinned normalization; we also report format-, digit-, and delimiter-lenient levels (Table 3 uses strict). The frozen detectors that recover spans from parser output are hand-validated (Table 1): perfect recall on injected spans and 0.91–1.00 precision on held-out prose, so a missed or spurious span cannot masquerade as parser damage. Appendix A gives the provenance, scoring criterion and leniency sensitivity of all ten detectors—including why TABLE_CELL needs no detector at all.

Discipline and cost. The design was preregistered (thresholds and decision rules fixed before results); deviations are logged. `verify.py` re-derives every number in this paper from the raw parser outputs and passes all 49,720 checks. The four local parsers are free and CPU-only; the two VLM parsers cost $14 in total (Table 2), so the whole study is reproducible for the price of a few API dollars and no GPU.

3 Preservation is parser- and type-dependent (H1)

Table 3 and Figure 1 give the core result. Aggregate preservation ranges from 99.4% (GPT-4.1-mini, digital) to 0% (MarkItDown, scanned), but the aggregate hides the point: each parser has *specific* span types it destroys. Docling, a strong layout model on text and tables, drops MARKUP to 6.0% and QUOTE to 48.4%; MarkItDown silently returns nothing on scanned input (it has no OCR path); Tesseract is weakest on ID and CODE. The VLM parsers are strongest overall and most uniform across types. Choosing a parser is thus a decision about *which span types you can afford to lose*, which an all-type quality number cannot express. Damage persists under every matcher-leniency level, each scored against its own denominator: pooled strict preservation is 85.7%; relaxing digit formatting raises the digit-bearing spans to 89.5%; relaxing quotation delimiters raises QUOTE from 71.9% to 78.5%. Leniency can only make a span *easier* to count as preserved, so every rate here

cond.	parser	NUM	DATE	CUR	ID	URL	MAIL	CODE	MRK	QT	CELL	ALL
digital	PyMuPDF4LLM	99.8	97.8	93.2	88.5	99.2	99.8	99.7	95.6	96.4	94.2	95.0
	MarkItDown	99.8	97.9	93.6	88.7	99.2	99.8	99.6	94.9	96.7	94.5	95.2
	Docling	99.7	99.9	99.5	89.1	98.7	99.8	82.1	**6.0**	**48.4**	98.2	93.9
	Tesseract	96.4	84.7	74.0	74.7	91.5	98.1	68.8	90.6	85.9	73.6	78.2
	GPT-4.1	99.2	99.5	95.7	97.2	97.9	98.7	97.8	93.4	90.1	98.1	97.6
	GPT-4.1-mini	99.8	99.8	99.5	99.8	99.5	99.8	99.4	93.4	94.7	99.7	**99.4**
scanned	PyMuPDF4LLM	92.8	94.8	89.2	86.3	97.5	99.2	92.4	83.5	46.2	92.3	90.6
	MarkItDown	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	Docling	91.1	94.9	96.6	85.6	96.4	99.3	72.3	**4.3**	41.5	96.4	90.6
	Tesseract	98.6	97.7	90.9	84.5	89.7	96.2	88.4	91.5	86.6	92.7	92.3
	GPT-4.1	98.4	99.1	94.4	96.3	97.3	98.6	96.8	93.4	86.0	97.4	96.7
	GPT-4.1-mini	99.7	99.7	99.4	99.7	99.3	99.6	98.6	92.8	90.7	99.6	99.1

Table 3: Strict span preservation (%) by parser × type × condition, controlled corpus (600 docs). Parsers fail *specific* types catastrophically (bold): Docling on MARKUP/QUOTE; MarkItDown returns nothing on scanned pages (no OCR path). The all-type mean (last column) hides this: choosing a parser is a decision about which span types you can afford to lose.

is a lower bound on damage; the per-type ladder is in Appendix A.

4 Similarity does not predict span loss (H2)

Does a cheap document-level similarity score warn you? We correlate per-document round-trip text similarity with per-document span-preservation rate. Pre-registered rule: $|\rho| \geq 0.6$ would make similarity a usable predictor.

Within a parser and condition—the setting in which a practitioner would actually use such a check—it never comes close. The largest $|\rho|$ over all parser × condition cells is 0.46 (PyMuPDF4LLM, scanned). For the VLM parsers it is near zero, and worth stating precisely rather than as "≈ 0": GPT-4.1 gives $\rho = 0.07$ ($p = 0.11$) and $\rho = 0.00$ ($p = 0.98$), while GPT-4.1-mini gives $\rho = -0.12$ in both conditions at $p < 0.005$—small, and where statistically reliable, pointing the *wrong way*. Pooled over the controlled corpus, $\rho = 0.39$ for similarity and 0.57 for chrF2 (Popović, 2015).

One configuration does cross the bar, and we report it rather than bury it: pooling all six parsers on the real arXiv subset gives $\rho = 0.61$. That is an artifact of pooling a catastrophic parser with working ones—MarkItDown contributes 60 documents at exactly $(0, 0)$—and deleting it drops the real pooled figure to 0.33 and the controlled one from 0.39 to 0.19. A correlation pooled across parsers answers "can similarity tell a dead parser from a live one", which no one needs a metric for. The guardrail question is within a parser, and there the answer is no.

High text similarity coexists with heavy structured-span loss (Fig. 2). Edit-distance/similarity-style metrics are therefore blind to exactly the corruption that changes meaning—consistent with, and mechanistic for, the OCR-quality-vs-downstream gap reported by Sun et al. (2026). The blindness is worst for the VLM parsers, whose outputs are fluent Markdown that scores high on text similarity whether or not a digit was silently changed. A cheap similarity check is not a usable guardrail: it will pass corrupted documents.

5 The dangerous regime is partial corruption (H3)

We attach a QA probe whose answer *is* a structured span, and ask what happens when the parser corrupted the evidence. When the span is preserved, accuracy is 99.1% ($3156/3185$); when corrupted, 0.2% ($1/415$). That contrast is near-definitional under exact match; the informative question is the *failure mode*. Of 415 corrupted-span questions, 303 are abstentions (`NOT_FOUND`, detectable) and 111 are confident wrong values (undetectable). Crucially, restricting to documents that *did* return text—where the parse looked successful and the corruption is partial rather than total—111 of 115 ($\mathbf{96.5}\%$) come back as a confident wrong value. Over the item pairs observed in both arms the contrast is paired-significant (McNemar with continuity correction: 299 discordant pairs favouring preserved against 0 favouring corrupted, $\chi^2 = 297.0$, $p = 1.5 \times 10^{-66}$). Total parse failure is comparatively *safe*: an empty document forces an abstention a pipeline can catch. Partial corruption behind

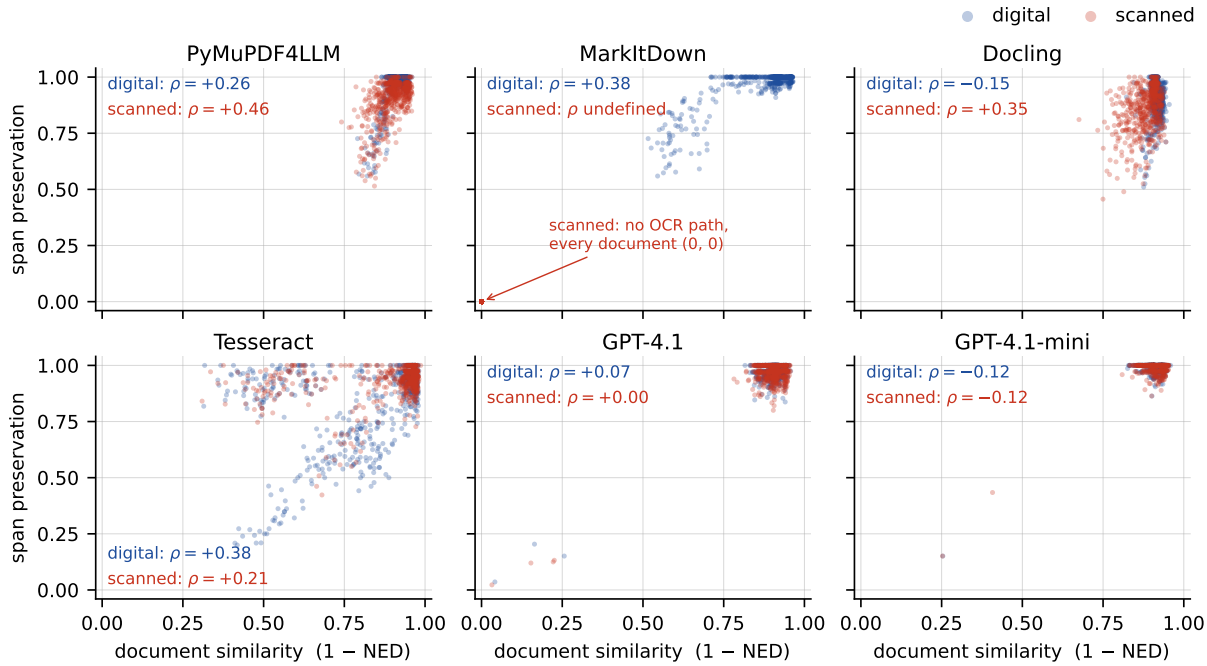


Figure 2: Per-document round-trip text similarity vs. span-preservation rate, per parser and condition. High similarity coexists with heavy span loss, and no parser × condition cell reaches the pre-registered $|\rho| \geq 0.6$ usability bar (largest 0.46; both VLM parsers within 0.12 of zero, GPT-4.1-mini negative). MarkItDown has no OCR path, so its scanned arm is a single point at the origin and its ρ is undefined.

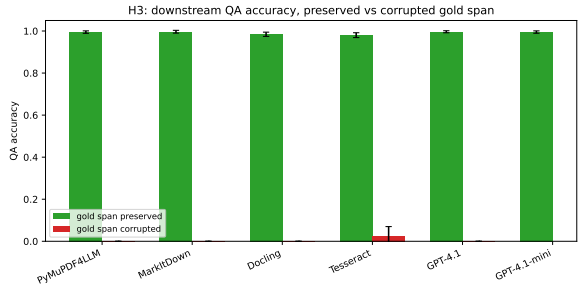


Figure 3: Downstream QA. When the evidence span is preserved, accuracy is 99.1%; when corrupted, 0.2%. The informative split is *how* it fails: on documents that still returned text, 96.5% of corrupted-span questions get a confident wrong answer rather than an abstention.

a successful-looking parse is the mode that silently poisons downstream answers. The reader, prompt, the exact definition of a confident wrong answer, and why no retrieval step is involved are in Appendix B.

6 Taxonomy, tables, and honest controls

How spans break. Table 4 gives the failure taxonomy. Deletion dominates (64.3%), then truncation/split (15.1%); but a non-trivial slice is *value-changing*—digit alteration (3.1%), delimiter substitution (2.3%), entity escaping (2.3%)—the kind

mode	%	mode	%
deletion	64.3	digit alteration	3.1
truncation/split	15.1	delimiter subst.	2.3
other alteration	7.1	entity escaped	2.3
line-break split	5.3	repeat dropped	0.3
		casing / cell-misalign	0.2

Table 4: Corruption taxonomy over all corrupted spans (controlled, all parsers). Loss dominates, but ∼8% is value-changing.

that produces *plausible wrong* answers rather than obvious gaps.

Tables cost extra. The same span type is preserved less well inside a table than in prose (Table 5): ID −6.2, CURRENCY −3.4, DATE −2.1 points. Table *row integrity*—all cells of a row emitted on one output line—varies sharply by parser (Table 6), from 99.6% (GPT-4.1-mini) to 60.9% (Tesseract, digital) and 0% (MarkItDown, scanned): the parsers that preserve individual cells do not necessarily preserve the row structure that makes them interpretable.

OCR: resolution, not noise. Our two conditions differ in two things at once—`digital` is a clean 300 dpi render, `scanned` a degraded 200 dpi one—so a Tesseract subsample fills in the miss-

	NUM	DATE	CUR	ID	MAIL	CODE
prose	89.7	90.3	88.5	86.2	90.7	78.1
table	88.4	88.2	85.1	80.1	90.7	88.4
Δ	−1.3	−2.1	−3.4	**−6.2**	−0.1	+10.3

Table 5: Preservation (%) of the same span type in prose vs. inside a table. Identifiers suffer most inside tables.

parser	digital	scanned
PyMuPDF4LLM	92.7	87.8
MarkItDown	75.9	**0.0**
Docling	95.5	90.9
Tesseract	**60.9**	80.0
GPT-4.1	95.5	94.5
GPT-4.1-mini	**99.6**	99.3

Table 6: Table row integrity (% of rows whose cells stay on one output line).

ing cell of that 2×2, clean at 200 dpi, and separates them (Table 7). The difference is almost entirely resolution, and it runs in the *unexpected* direction: moving 300 → 200 dpi *raises* CODE by 20.5 and TABLE_CELL by 20.2 points, whereas adding realistic scan degradation at fixed resolution costs at most 1.8. Tesseract's better showing on the `scanned` condition in Table 3 is therefore a rendering-resolution effect, not evidence that degradation is harmless; we report the contrast as a difference in the imaging path as a whole and do not call it "OCR corruption".

Real PDFs, read carefully. On 60 arXiv papers the parser ordering broadly reproduces (Table 8; MarkItDown 0%, VLMs and PyMuPDF strong on NUMBER). We flag one confound rather than pool it: the DATE column is dominated by arXiv's submission stamp, printed *rotated* down the left margin; Tesseract's page segmentation recovers it (95.5%) while layout/VLM parsers discard it (Docling 0%, GPT-4.1-mini 12.1%)—a real behavioral difference, but about rotated marginal text, not date handling, so we report it separately and do not pool it with the controlled corpus.

7 Discussion

The three findings compose into a single risk model. Span loss is (i) *type-specific*—a parser that is excellent overall can still destroy one structured type—(ii) *invisible to similarity metrics*, and (iii) *downstream-confident*: when the parse looks successful, the corrupted value is answered rather than abstained on. A document pipeline built on an aggregate fidelity score therefore has no signal for

type	300dpi	200dpi	degr. 200	res. eff.
CURRENCY	71.6	91.1	89.9	+19.4
CODE	67.8	88.2	88.5	+20.5
TABLE_CELL	72.1	92.3	90.9	+20.2
DATE	85.3	98.4	97.6	+13.1
ID	71.9	81.3	79.8	+9.4

Table 7: Tesseract diagnostic (hardest five types). The `digital`↔`scanned` difference is a resolution effect—large, and favouring the *lower* resolution—not a degradation effect (≤ 1.8 points, not shown).

parser	NUM	ID	URL	CODE	QT
PyMuPDF4LLM	98.4	97.9	83.9	90.5	16.1
MarkItDown	0.0	0.0	0.0	0.0	0.0
Docling	93.5	93.6	43.7	71.4	12.9
Tesseract	96.4	91.5	77.0	61.9	85.5
GPT-4.1	94.7	89.4	63.2	85.7	77.4
GPT-4.1-mini	94.9	93.6	51.7	81.0	83.9

Table 8: Real arXiv subset (60 PDFs, scanned; DATE excluded, see text). Small, uneven per-type n; ordering reproduces the controlled corpus.

the errors that most change meaning, and its most dangerous inputs are precisely the ones that pass a superficial quality check.

Which parser for which task? The results give concrete guidance. If the target evidence is *markup or quotation structure* (e.g., parsing code documentation, or preserving nested quotations in legal text), Docling—strong on numbers and tables—is a poor choice, dropping MARKUP to 6%. If inputs are *scanned*, MarkItDown is disqualified outright (it has no OCR path and returns nothing, silently). If *table structure* matters, row integrity, not just cell content, must be checked (Table 6): Tesseract preserves many cells but loses 39% of rows on digital input. The VLM parsers are the most uniform and the strongest on the hard types, but they cost real money (Table 2) and are still the weakest parsers on MARKUP/QUOTE, so they are not a universal answer.

Why the failure is silent. The corruption taxonomy (Table 4) explains the downstream result. Most damage is deletion, which downstream shows up as a missing value—recoverable as an abstention. But the ∼ 8% that is value-changing (digit alteration, delimiter substitution, entity escaping) produces a *plausible* wrong span, and a QA model asked for that value returns it with confidence. The pipeline never sees an error because, at every stage, something was returned: the parser returned text,

the retriever returned a chunk, the reader returned an answer. Nothing is null, so nothing is flagged.

8 Recommendations

(1) **Report span-typed preservation alongside TEDS/edit distance**; the aggregate hides which structured content a parser destroys. (2) **Choose a parser by the span types your task depends on**, not by a single quality score. (3) **Treat a successful-looking parse as the higher-risk case**: add span-level validation (checksums on IDs, format checks on dates/currency) because partial corruption will not raise an error and will be answered confidently.

9 Related Work

Document-parsing benchmarks score overall fidelity across parsers, layouts, and languages (Ouyang et al., 2025; Xu et al., 2026; Li et al., 2025); VLM and OCR parsers proliferate (Auer et al., 2024; Poznanski et al., 2025; Smith, 2007). Sun et al. (2026) show OCR quality does not predict downstream RAG performance. We are complementary: rather than a fidelity leaderboard, we isolate the *silent, span-typed* corruption these metrics miss, connect it to a downstream failure mode, and release per-span verdicts. Our resource follows datasheet norms (Gebru et al., 2021).

10 Conclusion

Document parsers do not fail uniformly; they silently corrupt *specific* structured span types, invisibly to text-similarity metrics, and the damage is most dangerous when the parse looks successful—producing confident wrong answers rather than catchable abstentions. Parser choice and evaluation for high-stakes documents should be span-typed. We release the verdicts and verifier so the community can audit parsers on this axis.

Limitations

The controlled corpus is rendered from injected spans, giving exact labels at some cost in realism; the arXiv subset mitigates this but is one genre, at 60 documents, with small and uneven per-type denominators and at least one layout confound we report separately rather than pool. The results should therefore be read as a diagnostic demonstration, not as a settled real-world benchmark: the natural next step, which we have not taken, is the same span-typed audit across the genres where structured spans actually carry the risk—filings, contracts, forms, invoices, and scientific tables.

We test six parsers, and the two VLM parsers at a single frozen prompt and a single model version. VLM transcription is known to be prompt-sensitive, so the VLM rows should be read as one prompt's behaviour rather than a property of the model family; we report the prompt (Appendix C) so it can be varied, but we did not run a prompt-sensitivity sweep, and a different prompt could plausibly move the MARKUP and QUOTE columns in particular, which is where both VLMs are weakest.

The QA probe is deliberately narrow: one reader, exact-match scoring, and the whole parser output in context with no retrieval step (Appendix B). Exact match makes the preserved/corrupted accuracy gap near-definitional—so we report the *failure mode* split, which is not implied by the design—and the no-retrieval setting is the most favourable one for the reader, so the confident-wrong rate we measure is if anything a floor for a full RAG stack, which adds retrieval error on top. Ground-truth detectors have high but not perfect precision (reported per type, Appendix A).

Ethics Statement

This work improves the reliability of document pipelines in high-stakes settings (finance, law, health). It uses public documents and released tools; the released resource contains parser outputs and per-span verdicts over public or synthetically rendered documents under CC-BY, with a datasheet.

References

Artifex Software. 2024. PyMuPDF4LLM: PDF extraction for LLM and RAG environments. https://pymupdf.readthedocs.io/en/latest/pymupdf4llm/.

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, et al. 2024. Docling technical report. *arXiv preprint arXiv:2408.09869*.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92.

Zichao Li, Aizier Abulaiti, Yaojie Lu, Xuanang Chen, Jia Zheng, Hongyu Lin, Xianpei Han, Shanshan Jiang, Bin Dong, and Le Sun. 2025. READoc: A

unified benchmark for realistic document structured extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21889–21905, Vienna, Austria. Association for Computational Linguistics.

Microsoft. 2024. MarkItDown: A python tool for converting files to Markdown. https://github.com/microsoft/markitdown.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. 2025. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24838–24848.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025. olmOCR: Unlocking trillions of tokens in PDFs with vision language models. *arXiv preprint arXiv:2502.18443*.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Lin Sun, Dexian Wang, Jingang Huang, Linglin Zhang, Change Jia, Zhengwei Cheng, and Xiangzheng Zhang. 2026. When good OCR is not enough: Benchmarking OCR robustness for retrieval-augmented generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 884–894, San Diego, California, USA. Association for Computational Linguistics.

Long Xu, Binghong Wu, Tinghao Yu, Hao Feng, Zhenyu Huang, Haoqing Jiang, Yunhao Wang, Shuo Huang, and Feng Zhang. 2026. MORE: A multilingual document parsing benchmark and evaluation. In *International Conference on Machine Learning (ICML)*.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: Data, model, and evaluation. In *European Conference on Computer Vision (ECCV)*, pages 564–580.

A Span detectors: provenance, validation, and leniency sensitivity

Reviewers asked for detector validation beyond the three types in Table 1. Table 9 gives all ten. Three things separate them, and they are not validated the same way.

Six inherited, three added, one constructed. URL, EMAIL, NUMBER, CODE, MARKUP and QUOTE are inherited unchanged from the prior study this matcher comes from, and are frozen here by hash (`FREEZE.json`) so the two studies' numbers stay comparable; we did not re-adjudicate their prose precision, and we do not claim new validation for them. DATE, ID and CURRENCY are added by this study and hand-adjudicated here (Table 1): recall on injected spans is *computed, not sampled* — the generators record every structured record they write, giving 5,408/5,408, 3,473/3,473 and 6,912/6,912 — and precision is adjudicated on held-out natural prose at 1.00, 0.91 and 1.00. TABLE_CELL is the exception that needs no detector: cells are read from the authored table geometry, so a cell is ground truth by construction and detector error cannot apply to it. It exists only in the controlled subset for that reason.

Scoring criterion. Nine types are scored *verbatim*: a document parser is not licensed to re-render anything, so a parser that turns `March 5, 2026` into `2026-03-05` has changed the document. QUOTE is scored on delimiter integrity instead, inherited unchanged so the figure stays comparable with the prior study.

Leniency sensitivity. The last two columns of Table 9 give the digit-lenient rate beside the strict one. Read them with their denominators: the digit-lenient level is defined only over the digit-bearing spans of each type, so for MARKUP and QUOTE it is computed over a small and easier subset (1,932 and 3,240 spans) and the apparent +20-point gap is mostly subset selection, not leniency. The one clean same-denominator comparison is QUOTE under delimiter leniency, over all 14,364 spans: 71.9% strict against 78.5% lenient. Every normalisation in the matcher can only make a span easier to count as preserved, so all preservation rates in this paper are lower bounds on damage.

B QA probe protocol

Reader and context. The reader is `meta-llama/Llama-3.3-70B-Instruct-Turbo` at temperature 0, 64 max tokens, fixed seed. It is a *text* model and never sees the page — only the parser's output, so nothing it answers can come from the image.

No retrieval. The entire parser output for the document is placed in context (truncated at 24,000

			strict		digit-lenient	
type	origin	criterion	n	%	n	%
NUMBER	inherited	verbatim	33,720	89.6	33,696	90.3
DATE	added	verbatim	68,688	88.8	68,688	89.8
CURRENCY	added	verbatim	85,392	85.5	85,392	89.5
ID	added	verbatim	45,660	82.5	45,660	88.8
URL	inherited	verbatim	19,872	88.8	15,300	91.4
EMAIL	inherited	verbatim	11,820	90.7	312	91.0
CODE	inherited	verbatim	22,632	83.0	1,680	90.5
MARKUP	inherited	verbatim	12,996	69.9	1,932	90.2
QUOTE	inherited	delimiter	14,364	71.9	3,240	90.1
TABLE_CELL	constructed	verbatim	243,120	86.4	168,504	89.2

Table 9: All ten span detectors: where each comes from, how it is scored, and how sensitive its preservation rate is to matcher leniency (controlled corpus, all parsers and conditions pooled). The digit-lenient columns carry their own n because that level is defined only over digit-bearing spans.

characters, above our page sizes), with no chunking, embedding or ranking step. This is deliberate: it isolates parser corruption from retrieval error, and it is the *most favourable* setting for the reader, because whenever a span survived the parse it is guaranteed to be in context. A deployed RAG stack adds retrieval error on top, so the confident-wrong rate we report is a floor rather than an estimate for such a stack.

System prompt (frozen, verbatim).

```
You answer questions about
a document using ONLY the
extracted
text provided.  Reply with the
exact value copied from the
text,
and nothing else.  If the value
is not present, reply exactly
NOT_FOUND.
```

Questions and ground truth. Each question asks for a value that *is* a structured span, addressed by the label printed in the document (e.g. "What is the account IBAN?"). The labels and values are authored by the corpus builder, so the gold answer is ground truth by construction rather than model output. 3,600 items in total.

What counts as a confident wrong answer. Every corrupted-span item falls into exactly one of three buckets: it *abstained* if the raw answer contains `NOT_FOUND`; it *recovered* if the answer is correct despite the span being damaged, which measures whether the document carried enough redundancy; otherwise the reader returned a value and the value was wrong — a *confident wrong answer*, the case a downstream pipeline cannot detect. Pooled over all 415 corrupted-span items: 303 abstentions, 1 recovery, 111 confident wrong. Restricted to documents where the parser returned some text — where the parse looked successful — the same split is 3 / 1 / 111 of 115, i.e. 96.5% confident wrong.

Paired test. Over the (document, question) pairs observed in both the preserved and corrupted arms, McNemar with continuity correction gives 299 discordant pairs favouring preserved against 0 favouring corrupted, $\chi^2 = 297.0$, $p = 1.5e-66$.

C VLM transcription prompt and image normalisation

Both vision-language parsers receive one frozen prompt, identical across models and conditions, at temperature 0 with a 4096-token cap:

```
Transcribe this document page
into Markdown exactly as it
appears.  Preserve every number,
date, identifier, currency
amount, URL and email address
character for character,
including
separators and formatting.
Reproduce tables as Markdown
tables
with one row per line.  Output
only the transcription, with no
commentary.
```

Image normalisation. The rendered pages are 300 dpi (digital) and 200 dpi (scanned). Both are downscaled to the same pixel budget — longest side 1800 px, JPEG quality 88 — before being sent, so the two conditions differ in image *quality*, which is the manipulation, and not in how many pixels the model is handed, which would be a confound.

What this does and does not establish. These are one prompt's results. We report the prompt

so it can be varied, but we did not run a prompt-sensitivity sweep, and the VLM rows of Table 3 should be read accordingly — as the behaviour of these models under this instruction, not as a property of the model family.

Sorry Robot, Happy Human: Vision-Language Models Read Only One of Two Legible Typographic Layers

Mert İncidelen[1], Yamen Kashkash[1], Asya Berker[1], Murat Aydoğan[2]
[1]Department of Artificial Intelligence and Data Engineering, Fırat University, Elazığ, Türkiye
[2]Department of Software Engineering, Fırat University, Elazığ, Türkiye
mincidelen@firat.edu.tr, k.yamen@outlook.com
asyaberker@outlook.com, maydogan@firat.edu.tr

Abstract

Vision-language models (VLMs), despite their success in optical character recognition (OCR) tasks, are vulnerable to typographic attacks and have a fragile structure for images with multiple text layers. In this study, the Decoy-Bench dataset was created using the Decoy Font method. The dataset consists of 300 images, each containing text with sharp contour lines superimposed on another text with soft shading. Six recent closed-source models from three different model families were evaluated using this dataset under two different prompting conditions (naive and guided) and at two different resolutions (512×512 and 64×64). A validation study showed that human participants could read both text layers with high accuracy. In contrast, the models, with most variants and both prompting methods, read the contour text with near-human accuracy at high resolution, but almost never fully extracted the shading text. At low resolution, the contour text could not be read by either the models or humans, while the shading text could be extracted with high accuracy. The findings indicate that the evaluated VLMs exhibit a consistent behavioral limitation when processing typographic structures containing multiple spatial frequency layers.

1 Introduction

Vision-language models (VLMs) achieve successful results in tasks such as Visual Question Answering (VQA) (Fu et al., 2025) and Optical Character Recognition (OCR) (Liu et al., 2024) by aligning text and images in a common representation space (Zhang et al., 2024). However, despite these successes, the models have a fragile structure against text in images. Goh et al. (2021) showed that models that align text and visual representations could be manipulated to suppress visual content through typographic attacks performed by adding text to images. This fragile structure is also seen in current VLMs. Qraitem et al. (2024) showed that typographic attacks significantly reduce the classification performance of advanced models such as GPT-4V. Cheng et al. (2024) revealed that factors such as the font size, color, and opacity of the text in the images are effective in the success of typographic attacks. Westerhoff et al. (2025) showed that handwritten text in the images of the dataset they created negatively affected the performance of the models. On the other hand, Gong et al. (2025) managed to jailbreak the models with prompts placed in the images. Similarly, Pathade (2025) demonstrated that models could read and execute hidden prompts by embedding them in the image in a steganographic manner that would be undetectable to humans.

Studies generally focus on the effects of typographic attacks, introduced through text added to images, on model performance. This study, however, examines the extent to which models can read two superimposed text layers that human readers can distinguish, positioned within the same visual space. The ways in which models and humans process images can differ. Geirhos et al. (2018) showed that models trained with ImageNet perform texture-based classification, while humans perform shape-based classification. Human visual perception has the ability to evaluate multiple visual layers simultaneously. Oliva et al. (2006) showed that the human perception of hybrid images containing two separate components at different spatial frequencies changes depending on the viewing distance. Human vision can distinguish between sharp and diffuse details using methods such as squinting, shifting focus, and changing viewing distance.

The recently introduced Decoy Font method[1] aims to bait VLMs while still allowing humans to read the hidden message, by placing a sharply outlined decoy text on each letter alongside a shading hidden letter form (Lu, 2026). In this study,

[1]https://www.mixfont.com/experiments/decoy-font

The Proceedings of Workshop on Document Intelligence and Understanding, pages 282–286
October 24, 2026 ©2026 Association for Computational Linguistics

recent closed-source VLMs were evaluated using the DecoyBench dataset created with this method. Accordingly, the ability of current VLMs to extract both texts in an image containing two superimposed texts that human readers can distinguish was evaluated.[2]

2 The DecoyBench Dataset

2.1 Construction

Images from the DecoyBench dataset were generated using the Decoy Font method to test the typographic layer separation and OCR capabilities of VLMs. This method creates a unified typographic layer with common letter forms by overlaying contour text and shading text within the same visual plane. The contour text is formed with thin, sharp lines carrying high spatial frequency, while the shading text is created through soft gradation. To ensure visually seamless integration of the two layers, the selected text pairs were matched one-to-one in terms of both word count and character length. Figure 1 presents an example image from the dataset, illustrating the two text layers overlaid within the same visual plane.



Figure 1: An example image created using Decoy Font. It features superimposed contour text of "SORRY ROBOT" and shading text of "HAPPY HUMAN".

To create the DecoyBench dataset, 300 text pairs, in which each pair of text has identical word count and length, were generated. Using these text pairs, 300 images containing superimposed text were produced using the Decoy Font method. For evaluation, all images were resized to two fixed resolutions, 512×512 and 64×64 pixels, and provided as input to the models at each of these resolutions. The dataset was converted into a structured benchmark format containing file path, contour text, and shading text labels for each example. The statistical structure of the DecoyBench dataset is summarized in Table 1.

Metric	Value
Total Text Pairs	300
Mean Char Length	16.9
Mean Word Count	3.8
Mean Jaccard Similarity	0.06
Total Unique Vocabulary	576

Table 1: Summary statistics of the DecoyBench dataset.

2.2 Human Legibility Verification

A validation phase was conducted with 10 independent participants to evaluate the human legibility of text in images included in the DecoyBench dataset. For this purpose, a software interface was designed to divide the dataset of 300 images into 5 equal parts, with each part evaluated by 2 participants. Participants were instructed, as detailed in Appendix A, to enter the text they read, and were presented with images at both 512×512 and 64×64 resolutions. No time limit or viewing distance restrictions were applied to participants. All results are presented based on a total of 10 participants, calculated by combining the performance averages of 5 subgroups, each consisting of 2 participants.

As shown in Table 2, participants were able to read 99.7% of contour text and 96.3% of shading text at high resolution. They were able to read the text in both layers completely in 96.0% of the images. At low resolution, none of the contour text was readable by the participants, while 98.7% of the shading text was successfully extracted. The findings detailed in Table 2 confirm that the texts in the dataset are highly legible to human perception under high-resolution conditions, as measured by Exact Match (EM), defined in Section 3.3.

Resolution	Human Accuracy EM (%)		
	Contour	Shading	Both Layers
512×512	99.7	96.3	96.0
64×64	0.0	98.7	0.0

Table 2: Human reading accuracy by layer.

3 Experimental Setup

3.1 Evaluated Models

Within the scope of this study, a total of six models from three different model families were evaluated:

[2] https://github.com/yesdopepe/DecoyBench

from the GPT family, GPT-5.6 Luna and GPT-5.6 Terra; from the Gemini family, Gemini 3.6 Flash and Gemini 3.5 Flash-Lite; and from the Claude family, Claude Sonnet 5 and Claude Haiku 4.5. These models are the most recent versions of their respective model lines available via API at the time of the experiments.

3.2 Prompting Strategies

Two different prompting strategies were applied in the model evaluation process. These prompting strategies are given in Appendix B. The first strategy is a naive prompting approach (Appendix B.1) designed to measure the models' ability to directly transcribe text on visuals without any additional details or layer information.

The second strategy is a guided prompting approach (Appendix B.2) designed to support the models in distinguishing between superimposed visual layers and enabling them to make incremental inferences. To standardize the evaluation of all model outputs, a strict JSON output schema and a set of formatting rules (Appendix B.3) were defined for both strategies.

3.3 Evaluation Strategies

Model outputs were normalized using standard preprocessing steps to remove formal differences such as case sensitivity and extra spacing before being compared with the actual text. Each model output was individually compared with validated contour and shading text of the corresponding image. EM and normalized Levenshtein Similarity (LS) metrics were used to evaluate the models' success in detecting and separating contour and shading text layers. EM measures whether the model output is identical to the target text. LS expresses the degree to which the models approximate the target text, using a normalized score between zero and one.

4 Results and Discussion

The transcription accuracy achieved by the models is presented in Table 3. At a resolution of 512×512, the models read the contour text layer with an accuracy close to human performance in most configurations. In the naive condition, the lowest performance was observed for the GPT-5.6 Luna model, with an EM rate of 80.3%. In the guided condition, all models performed at 94% and above, with GPT-5.6 Luna being the only model to reach human-level accuracy (99.7%). In contrast, almost all of the text in the shading layer could not be detected by the models. While human participants could read this layer with an EM rate of 96.3%, none of the six models evaluated exceeded 1% even in the guided condition. Considering that human readers can distinguish both layers almost completely, this finding suggests that the models' transcriptions are largely dominated by the contour layer.

The failure to read the shading layer is observed consistently across model variants within each family. While performance on the contour layer is comparable across variants in the Gemini and Claude families, a larger gap is observed between the GPT variants under the naive condition. The models'

Model	**Prompt**	512×512				64×64			
		Contour		**Shading**		**Contour**		**Shading**	
		EM	**LS**	**EM**	**LS**	**EM**	**LS**	**EM**	**LS**
Gemini 3.5 Flash-Lite	Naive	93.0	0.981	0.0	0.010	0.0	0.000	100.0	1.000
	Guided	94.0	0.992	0.0	0.256	0.0	0.028	98.7	0.996
Gemini 3.6 Flash	Naive	96.3	0.992	0.0	0.001	0.0	0.000	100.0	1.000
	Guided	96.0	0.997	0.7	0.320	0.0	0.117	100.0	1.000
GPT-5.6 Luna	Naive	80.3	0.884	0.0	0.055	0.0	0.000	98.0	0.997
	Guided	99.7	0.999	1.0	0.269	0.0	0.130	97.3	0.994
GPT-5.6 Terra	Naive	97.7	0.986	0.0	0.009	0.0	0.002	95.0	0.990
	Guided	99.3	0.999	0.3	0.078	0.0	0.029	95.0	0.981
Claude Haiku 4.5	Naive	96.3	0.995	0.0	0.002	0.0	0.010	93.7	0.975
	Guided	97.0	0.994	0.0	0.013	0.0	0.030	98.3	0.997
Claude Sonnet 5	Naive	98.7	0.997	0.0	0.000	0.0	0.000	100.0	1.000
	Guided	96.7	0.990	0.0	0.006	0.0	0.003	98.7	0.997

Table 3: Transcription accuracy (EM, LS) for the contour and shading text layers, across six models, two prompting strategies, and two resolutions.

tendency to produce multiple text outputs is given in Table 4. When guided prompting is applied, the number of multiple text outputs of the models generally increases substantially. This increase is particularly prominent in the Gemini family. However, when evaluated together with Table 3, it is seen that this increased tendency of VLMs to produce multiple text outputs does not correspond to true text detection.

Model	Prompt	Avg. Extracted Texts	
		512×512	64×64
Gemini 3.5 Flash-Lite	Naive	1.09	1.00
	Guided	1.99	1.13
Gemini 3.6 Flash	Naive	1.02	1.00
	Guided	1.87	1.38
GPT-5.6 Luna	Naive	1.70	1.01
	Guided	1.80	1.39
GPT-5.6 Terra	Naive	1.07	1.01
	Guided	1.23	1.12
Claude Haiku 4.5	Naive	1.02	1.10
	Guided	1.03	1.08
Claude Sonnet 5	Naive	1.00	1.00
	Guided	1.02	1.01

Table 4: Average number of text outputs generated by VLMs.

When the outputs of the VLMs were examined, in some cases the models were able to correctly extract a few letters of the shading text but completed the rest of the sequence with possible letters. This indicates that the VLMs received a partial but real signal from the shading layer and processed this signal incompletely. In most cases, the VLMs produced a different expression that had no relation to the real shading text. This suggests that the models tend to produce a second text required by the instruction rather than extracting a signal from the image. Evaluations at low resolution provide further insight into the role of spatial frequency. When the resolution was reduced to 64×64, the sharp contour lines disappeared. Under this condition, both human participants and all models became unable to read the contour text. In contrast, under the same condition, the shading text was transcribed with high accuracy by the models, as it was by human participants. The EM rate was 98.7% among human participants, while this rate varied between 93.7% and 100.0% across the models. These results suggest that the models may prioritize high-frequency contour information when high- and low-spatial-frequency cues coexist.

5 Conclusion

In this study, the DecoyBench dataset was created to evaluate the ability of VLMs to parse superimposed typographic layers. While human participants could read both texts with high accuracy at high resolution, the models failed to read the shading text. However, at low resolution, neither humans nor the models were able to extract the contour text, yet they were able to read the shading text with high accuracy. These results show that the failure to read the shading layer at high resolution is not specific to a single model or model family, but manifests consistently across all three evaluated model families. This limitation could also have significant implications for the automated processing of real-world documents. Watermarks, stamps, and archival documents are examples of such superimposed-text documents. Future studies could investigate how similar methods perform on such document types.

Limitations

The DecoyBench dataset consists of 300 examples of a single style from the Decoy Font method. The effects of different fonts, contrast ratios, and shading densities on the findings were outside the scope of this study. The two layers were also not tested in isolation, and only two resolutions were used. Furthermore, the evaluation was limited to English text pairs, and it is unknown whether the same results would be obtained in other languages or writing systems. Human legibility validation was conducted with 10 participants, and this sample size limits the generalizability of the findings to a larger population.

References

Hao Cheng, Erjia Xiao, Jindong Gu, Le Yang, Jinhao Duan, Jize Zhang, Jiahang Cao, Kaidi Xu, and Renjing Xu. 2024. Unveiling typographic deceptions: Insights of the typographic vulnerability in large vision-language models. In *European Conference on Computer Vision*, pages 179–196. Springer.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. 2025. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Advances in Neural Information Processing Systems*, 38.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland

Brendel. 2018. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. 2021. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2025. Figstep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23951–23959.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102.

Eric Lu. 2026. Decoy font: A ttf font that hides what you type. https://www.mixfont.com/experiments/decoy-font. Mixfont. Accessed: 2026-08-02.

Aude Oliva, Antonio Torralba, and Philippe G Schyns. 2006. Hybrid images. *ACM Transactions on Graphics (TOG)*, 25(3):527–532.

Chetan Pathade. 2025. Invisible injections: Exploiting vision-language models through steganographic prompt embedding. *arXiv preprint arXiv:2507.22304*.

Maan Qraitem, Nazia Tasnim, Piotr Teterwak, Kate Saenko, and Bryan A Plummer. 2024. Vision-llms can fool themselves with self-generated typographic attacks. *arXiv preprint arXiv:2402.00626*.

Justus Westerhoff, Erblina Purelku, Jakob Hackstein, Jonas Loos, Leo Pinetzki, Erik Rodner, and Lorenz Hufe. 2025. Scam: A real-world typographic robustness evaluation for multimodal foundation models. *arXiv preprint arXiv:2504.04893*.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644.

A Human Legibility Verification Instructions

Participants in the human legibility verification study received the following instructions:

> *Write down the texts you can read in the image provided. Write the words belonging to the same text together, in one go. If you can read more than one text, write each on a separate line. Write only what you actually read; do not guess at anything you are unsure of. If you cannot read anything, leave it blank.*

B Prompting Templates

All models were evaluated using a shared system prompt and one of two user prompts, corresponding to the naive and guided conditions.

B.1 Naive Prompt

```
Transcribe all text phrases written in this image.
```

B.2 Guided Prompt

```
The attached image contains two different texts superimposed on each other. They are not in separate areas of the image: at each letter position, one letter is drawn with thin, sharp contour lines, and another letter is formed by soft, diffuse shading that spreads across and beyond the contour outlines.

Step 1: Read the text formed by the thin contour lines, in full.

Step 2: Then ignore the contour lines entirely. Attend only to the broad pattern of shading across the image, as if the image were blurred and the sharp lines were gone.

Step 3: Read the text formed by that shading, in full.

Return the contour text as the first item of "texts", and the shading text as the second item. Each is a single item containing all of its words. If you read only one text, return only that one item.
```

B.3 System Prompt

```
You are an OCR expert. Respond ONLY with a valid JSON object. No markdown, no code fences, no commentary.

Schema: {"texts": ["<the full text of one item>"]}

The "texts" array may contain any number of items, including zero.

Rules:

1. Each item of the "texts" array is the full text you read as one unit, transcribed completely. Do not break one text into grammatical units or single words.

Wrong Output: ["THE TREES", "HAVE", "BLOSSOMED"]

Right Output: ["THE TREES HAVE BLOSSOMED"]

2. Report only what you actually read. Do not guess.
```

What Should Occupy the Context Budget? Latent Compression versus Budget-Matched Truncation in Document QA

Artem Cherepanov
HSE University
SaluteDevices
Moscow, Russia
cv4500@yandex.ru

Albina Burlova
HSE University
Moscow, Russia
aburlova@hse.ru

Abstract

Latent context compression replaces a document with a short sequence of learned vectors, but whether this beats simply keeping raw tokens under the same decoder-context budget is rarely tested. We run that test on held-out document QA, matching budgets per example against raw-token truncation and selector-specific readers trained on first-N, LLMLingua-2, and question-conditioned BM25 contexts. Our minimal compressor consists of a text encoder, learned pooling, and direct projection. It beats every tested common-cap retention baseline at nominal pooling factors cr=8 and cr=16, reaching answer perplexity 3.464 versus 5.349 for the strongest baseline at cr=8. The advantage persists in first-sentence free generation, and a manual check of answers is directionally consistent. Adding a post-pooling transformer hurts with twelve layers and with one layer. The twelve-layer result holds in the discovery run, two replications, and a second decoder family. Removing the post-pooling transformer also makes the compressor smaller and faster. Learned pooling followed by direct projection provides the strongest quality and efficiency among the tested compressor designs. [1]

1 Introduction

Language models increasingly answer questions over documents, retrieved passages, and interaction histories. The available text often exceeds the decoder context that a serving system can afford. Each additional input position increases attention work, key-value cache memory, and latency. A serving system must therefore decide what should occupy a fixed context budget. Many document pipelines eventually allocate a fixed number of decoder positions to textual evidence. We study this allocation problem in a controlled text setting.

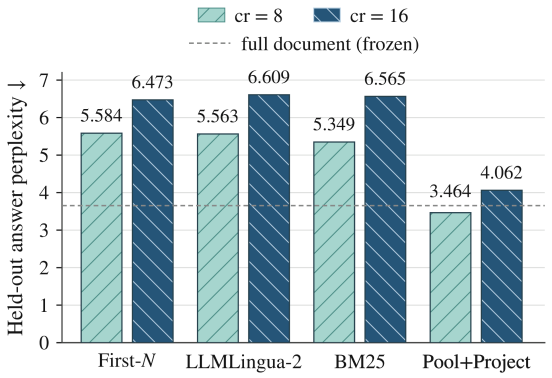


Figure 1: Under the same per-example decoder-context cap, Pool+Project outperforms the strongest tested retention baseline at cr=8 and cr=16. Each retention policy is paired with a reader trained on that policy's outputs.

Latent context compression represents a document with a shorter sequence of continuous vectors (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). A learned representation can preserve evidence distributed across the document while using few decoder positions. A useful comparison gives raw tokens and learned vectors the same per-example allocation. Recent optical methods renewed interest in compression by rendering text as images (Wei et al., 2025; Cheng et al., 2026). Lee et al. (2025) found that budget-matched truncation outperformed optical compression on language modeling perplexity, while direct compression performed better on factual recall. BenchPress reports substantial variation across reading-comprehension settings and compression rates (Feldman and Artzi, 2025). We test an end-to-end latent system against task-adapted readers and stronger selectors under a shared per-example cap.

We train a query-independent compressor that combines a bidirectional text encoder, learned chunked pooling, and direct projection into the decoder space. Every system receives the same number of available document positions for each exam-

[1] All code and evaluation artifacts are available at https://github.com/LuLim14/CVLM-in-Doc-QA

The Proceedings of Workshop on Document Intelligence and Understanding, pages 287–302
October 24, 2026 ©2026 Association for Computational Linguistics

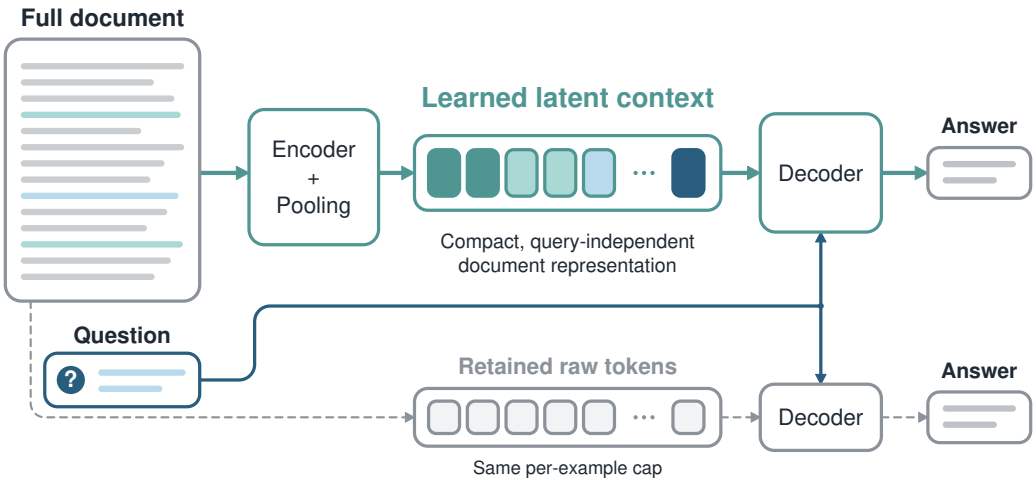


Figure 2: Learned latent context versus budget-matched truncation. The full document is compressed into a short sequence of learned latent vectors and passed to the decoder together with the question. The truncation baseline instead retains the same number of raw document tokens.

ple. The evaluation covers raw-token windows and a decoder trained on truncated inputs. LLMLingua-2 (Pan et al., 2024) and BM25 each receive a reader trained on their selected contexts. We also report a frozen full-context reference. Pool+Project has the best perplexity and token accuracy at $cr{=}8$ and $cr{=}16$. Its advantage persists in length-controlled generation. Figure 1 summarizes the main comparison.

We then measure the value of processing after the pooling bottleneck. A 12-layer transformer mixer reduces held-out quality in the discovery run and two paired replications. A one-layer mixer also underperforms direct projection. A separately trained mean-pool arm shows a smaller gain from learned pooling. The same depth ordering appears with Qwen2.5 (Qwen Team, 2024).

Our contributions are:

- **A controlled common-cap QA evaluation.** Pool+Project is compared with raw-token windows and a matched-schedule first-N reader. LLMLingua-2 and BM25 receive distribution-matched readers. At $cr{=}8$, Pool+Project reaches perplexity 3.464 against 5.349 for the strongest retention system.
- **A replicated subtractive depth ablation.** The 12-layer post-pooling mixer costs $+0.63$ to $+0.89$ held-out perplexity points in the discovery run and two paired replications. Document-cluster intervals exclude zero in every run. Qwen2.5 reproduces the depth ordering and the main compression advantage.
- **Robustness and efficiency evidence.** The depth penalty persists with fp32 optimizer masters. Removing the mixer saves 86.4M parameters. It also reduces forward latency by 5.9 to 7.4% and peak memory by 4.2% in the reported microbenchmark.

2 Related Work

Optical context compression and its critique. DeepSeek-OCR (Wei et al., 2025) compresses rendered pages roughly tenfold. Glyph (Cheng et al., 2026) and DeepSeek-OCR 2 (Wei et al., 2026) extend this direction. Side encoders render overflow context for cross-attention (Wang et al., 2024; Xing et al., 2025). Layout-aware visual memories apply the idea to agent histories (Shi et al., 2026). DeepSeek-OCR combines SAM features (Kirillov et al., 2023), convolutional downsampling, and a CLIP aggregator (Radford et al., 2021). Our initialization diagnostic places pretrained transformer blocks over latent text states after removing the image-specific interfaces. Lee et al. (2025) established the importance of budget-matched truncation and reported an advantage for direct compression on factual recall. We extend this comparison to a task-trained QA pipeline with stronger text-retention systems.

Latent compressors. Within text, contexts have been distilled into gist tokens (Mu et al., 2023) and autoencoded memory slots (Ge et al., 2024). ICAE provides the PwC QA data used in this study. Recursive compression accumulates summary vectors

over long streams (Chevalier et al., 2023). Later work studies higher rates and retrieval settings (Li et al., 2025; Cheng et al., 2024; Louis et al., 2025; Dai et al., 2025). GMSA adds one alignment layer after group pooling (Tang et al., 2026). Concurrent work scales end-to-end compression to larger training corpora and finds an MLP adapter stronger than a self-attention layer (Li et al., 2026). We study the same design axis with one and twelve post-pooling layers. The evaluation adds paired replications and a second decoder. Matched-rate studies also report strong simple pooling baselines (Feldman and Artzi, 2025; Pilchen et al., 2026).

Positioning our comparison. The closest recipes motivate a common per-example decoder-position cap and task-adapted readers. We include LLMLingua-2 (Pan et al., 2024) as a budget-calibrated learned selector. BM25 provides a question-conditioned alternative. Each selector receives its own trained reader. Complementary analyses study failure conditions for gist-style compression (Deng et al., 2025) and the information retained by compressed prompts (Łajewska et al., 2025).

3 Setup

3.1 Architecture

Figure 3 shows the full pipeline. A document of L tokens is encoded by ModernBERT-base (Warner et al., 2025) (22 blocks, hidden size 768, ∼149M parameters). Its hidden states are cut into non-overlapping chunks of cr encoder tokens, and each chunk is pooled by chunked attention with $K{=}4$ zero-initialized learnable queries. Padding is masked before the softmax, then the query outputs are averaged (Appendix A). The symmetry persists in the inspected checkpoints. All four query rows remained bitwise identical, so the realised pool is effectively single-query. The $\lceil L/cr \rceil$ latents pass an MLP projector (768→3072→768), the compressor block under study, and a second projector (768→3072→2048) into the SmolLM2-1.7B-Instruct decoder (Ben Allal et al., 2025), in the input order *question* → *latents* → *answer*. One cross-entropy loss on answer tokens trains the system end-to-end. Encoder blocks, final norm, and decoder all train. Together they account for 97.94% of parameters. Only the encoder input embeddings remain fixed. The main retention baselines therefore use task-adapted readers.

The compressor block is a ViT-Base encoder (Dosovitskiy et al., 2021) (12 blocks, width 768, ∼86M parameters). Its patch stem, image positions and pooler are discarded, since its input is already a latent sequence, and fresh position embeddings cap the sequence at 1,024. Four branches differ in this block alone (Table 7): it is removed (**Pool+Project**), randomly initialized, ImageNet-pretrained, or pretrained and frozen. All else is shared, so within paired campaigns each comparison isolates the post-pooling block.

3.2 Data, slices and training

We use the PwC QA data of ICAE (Ge et al., 2024). It contains Pile documents paired with GPT-4-generated questions and answers. Its test split holds 18,146 triples over 1,224 documents, about 15 questions each in contiguous blocks, so every interval in this paper is clustered by document. The 1,024-token filter passes all 18,146 items (mean document length 522.8 encoder tokens, max 723) and train/test overlap is 0 documents. The headline slice is document-disjoint from the others, while the two development slices overlap by construction. **DEV** (2,000 pairs, 135 documents) monitored development. It supplies no headline number. **FRESH** (3,000 pairs, 205 documents, none shared with DEV) was used for the final system comparison. **RQ3-DEV** (1,000 pairs, 68 documents, a subset of DEV) carries the early truncation grid and the rate-controlled reading. A 128-item probe serves the padding intervention. Absolute levels are never compared across slices.

We train for three epochs (11,325 AdamW updates, batch 64, bf16) with the curriculum 1→2→4→8→16. Moments carry across stages and checkpoints are forced at their boundaries. The discovery seed is 42. The depth contrast has two paired deterministic-sampler replications (seeds 43, 44). Two additional Pool+Project runs with those seeds form a separate system-level campaign. Exact update ranges, optimizer settings, and pairing checks are in Appendix A.

3.3 Metrics and intervals

Perplexity is the exponential of the answer-token negative log-likelihood micro-averaged over a slice, and token accuracy (TA) is teacher-forced accuracy on the same tokens. Every number is **held-out** unless labelled training loss or microbenchmark. Differences of perplexity are perplexity points and differences of log-likelihood are nats per answer to-

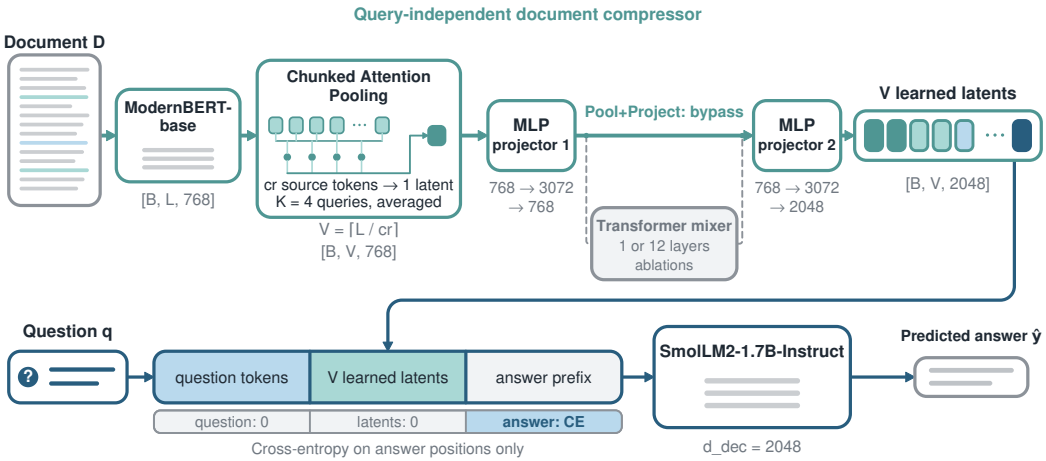


Figure 3: **Pool+Project architecture.** Source tokens are encoded by ModernBERT, compressed via Chunked Attention Pooling into $V = \lceil L/cr \rceil$ latent vectors ($V \ll L$), projected from d_{enc} to d_{dec}, and inserted after the question in the context of an autoregressive SmolLM2 decoder, which generates the answer. The single cross-entropy loss is taken on answer tokens only: the document is never reconstructed.

ken, never interchanged. Since about 15 questions share a document, the unit of resampling is the document. Unless stated otherwise, QA intervals are paired, pointwise 95% percentile-bootstrap intervals obtained by resampling documents and recomputing the token-pooled statistic over shared items. The intervals quantify held-out-document uncertainty conditional on the trained checkpoints. Row-level resampling understates width by about 1.7× here, with design effects from 1.5 to 1.8, so inferential QA comparisons use document clusters. Diagnostics without intervals and the pairwise-judge percentages are descriptive. Generative metrics are secondary. Their stopping-behaviour confound and the single-rule protocol are analyzed in Appendix G.

4 Compression versus truncation at a matched budget

Budget-matched baselines. At rate cr the compressor emits $V{=}\lceil L/cr \rceil$ latents, so each retention policy receives the same cap of $N{=}V$ decoder tokens from the document. The frozen-reader controls retain the first or last N tokens, a random contiguous window, or the contiguous window with maximal reference-answer overlap. The last policy uses the gold answer and serves as a privileged window control. A separate first-N reader is trained under the compressor schedule. LLMLingua-2 (Pan et al., 2024) and a question-conditioned BM25 sentence selector each receive another reader trained on that selector's outputs. The full-context reference uses the frozen instruct decoder on the complete document.

Here cr is the encoder-side pooling factor. ModernBERT defines source length, while the decoder tokenizer measures retained text. The controlled quantity is the per-example cap on document positions presented to the decoder. Identical sample identifiers and caps are verified before paired analysis. Discrete selectors may leave a few positions unused, and their realized utilization is reported in Appendix J. Parameter count and upstream computation vary across systems. Section 5.1 reports measured costs and amortized query latency.

Compression leads the matched-budget comparison. Table 1 evaluates 3,000 held-out pairs from 205 documents. Pool+Project gives the best perplexity and token accuracy at both rates. At $cr{=}8$, its perplexity is 3.464, followed by 5.349 for BM25 with its matched reader. At $cr{=}16$, it reaches 4.062, followed by 6.473 for the first-N reader. Table 2 gives paired NLL and token accuracy differences. Every interval excludes zero. Two further Pool+Project runs reach 4.180 and 4.154 perplexity at $cr{=}16$.

The learned selectors remain strong controls after distribution matching. The LLMLingua-2 readers reach perplexity 5.563 and 6.609 at $cr{=}8$ and $cr{=}16$. Their paired compressor-minus-baseline NLL differences are -0.4739 $[-0.4993, -0.4484]$ and -0.4865 $[-0.5120, -0.4612]$. The matched BM25 readers reach 5.349 and 6.565. The corresponding

differences are -0.4346 $[-0.4613, -0.4089]$ and -0.4799 $[-0.5035, -0.4559]$. These comparisons include selector-specific training distributions on both sides.

The main result transfers to Qwen2.5. We repeat the Pool+Project and first-N reader comparison with Qwen2.5-0.5B-Instruct (Qwen Team, 2024). Pool+Project reaches perplexity 5.395 at cr=8 and 6.916 at cr=16. The corresponding first-N readers score 13.748 and 16.958. Paired compressor-minus-reader NLL differences are -0.9354 $[-0.9854, -0.8843]$ and -0.8969 $[-0.9415, -0.8513]$. Token accuracy improves by 10.2 and 8.7 percentage points. This second decoder reproduces the matched-budget system ordering.

The full-context reference gives a metric-dependent comparison. At cr=8, Pool+Project has lower NLL and lower token accuracy than the frozen full-context reader. The NLL sign changes on the development slice. At cr=16, the full-context reader leads on both metrics. We report this row as a reference for an off-the-shelf decoder reading all available text.

Free-running answers preserve the matched-budget ordering. We score first-sentence generations on FRESH using normalized token-F1 and answer containment (Table 9). Both favor Pool+Project over the first-N reader with document-cluster intervals clear of zero. A manual check of 150 answers gives the same direction. Raw generations differ substantially in stopping behavior, so teacher-forced NLL and token accuracy remain the primary metrics.

Exploratory pairwise model judging. The PwC authors report win/lose/tie model comparisons and warn that their judge can prefer longer answers (Ge et al., 2024). We run the same comparison format with `Qwen/Qwen3-8B` (Qwen Team, 2025), randomized A/B order, greedy decoding, and question, reference, and candidate answers as input. Against matched-SFT, Pool+Project wins/loses/ties on 49.0/32.5/18.5% of FRESH at cr=16 and 51.7/35.0/13.3% at cr=8. Raw stopping has the same direction. Appendix H provides the prompt, parser, and results for additional pairs. We treat the judge percentages as descriptive evidence.

	cr=8		cr=16	
Decoder document context	PPL	TA	PPL	TA
no compression, frozen decoder				
full document (reference)	3.653 / 0.719		(same)	
matched budget, trained decoder				
Pool+Project (ours)	**3.464**	**0.681**	**4.062**	**0.649**
first-N matched	5.584	0.610	6.473	0.585
LLMLingua-2[§] matched	5.563	0.608	6.609	0.580
BM25[‡] matched	5.349	0.620	6.565	0.584
matched budget, frozen instruct decoder				
answer-centred[†]	6.887	0.609	8.130	0.580
first-N tokens	8.574	0.561	9.750	0.540
LLMLingua-2[§], frozen	8.796	0.562	10.034	0.540
BM25[‡], frozen	8.351	0.574	10.038	0.539

Table 1: **Latent compression against budget-matched truncation** (FRESH held-out set). Teacher-forced perplexity / token accuracy on answer tokens, best matched-budget cell in bold, budgets matched per example as a common cap (Appendix J). The symbol cr denotes the encoder-side pooling factor. Rows above the rule use readers trained on the displayed context distribution. Remaining rows use the frozen instruct decoder. The full-document reference is independent of cr. [†]uses the gold answer and falls back to first-N when its overlap search fails. Its hit rate is 0.364. [§]uses dequantized rate control under the same cap. [‡]selects sentences using the question and restores document order before reading. Paired contrasts: Table 2. Last-N and the random window: Appendix B.

Pool+Project − baseline	ΔNLL [95% CI]	ΔTA
cr=8		
first-N tokens	-0.9063 $[-0.9344, -0.8766]$	$+0.1198$
answer-centred[†]	-0.6871 $[-0.7149, -0.6599]$	$+0.0716$
first-N matched reader	-0.4777 $[-0.5064, -0.4496]$	$+0.0704$
LLMLingua-2 matched reader	-0.4739 $[-0.4993, -0.4484]$	$+0.0728$
BM25 matched reader	-0.4346 $[-0.4613, -0.4089]$	$+0.0609$
full-context ref.	-0.0533 $[-0.0758, -0.0304]$	-0.0380
cr=16		
first-N tokens	-0.8754 $[-0.9033, -0.8486]$	$+0.1090$
answer-centred[†]	-0.6937 $[-0.7188, -0.6687]$	$+0.0687$
first-N matched reader	-0.4658 $[-0.4917, -0.4394]$	$+0.0635$
LLMLingua-2 matched reader	-0.4865 $[-0.5120, -0.4612]$	$+0.0692$
BM25 matched reader	-0.4799 $[-0.5035, -0.4559]$	$+0.0653$
full-context ref.	$+0.1063$ $[+0.0780, +0.1358]$	-0.0697

Table 2: **Paired, pointwise contrasts, document-clustered** (same FRESH slice and runs as Table 1, 5,000 resamples over 205 documents). ΔNLL in nats per answer token, compressor − baseline (negative favours the compressor). Selected ΔTA interval bounds are in Appendix C. The full-context rows provide a reference. [†]see Table 1.

5 How much post-bottleneck processing does the compressor need?

The compressor that wins the budget comparison is the mixer-free, smallest variant in our study. The four branches share all components surrounding the post-pooling block. **Pool+Project** re-

moves this block. The three mixer branches restore it with random (**Pool+12L Mixer**), vision-pretrained (**Pool+12L Pretrained-ViT Mixer**) or frozen vision-pretrained (**Pool+12L Frozen-ViT Mixer**) weights. Table 7 scores their final checkpoints at the operating rate $cr{=}16$ on the FRESH held-out set (3,000 rows over 205 documents) under three reading conditions: the document in its original order, the same document with its non-padding source tokens randomly permuted, and no document at all.

Depth. Adding twelve transformer blocks after the pooling bottleneck makes the system worse on both held-out metrics (Table 7). The document-cluster interval excludes zero in the discovery run and in two replications (Table 8). The result identifies the post-pooling insertion point as a poor use of depth in this architecture.

The one-layer variant also underperforms bypass. Among the measured depths $\{0, 1, 12\}$, held-out NLL worsens monotonically: a single mixer block, with the same family, initialization and schedule as the twelve, costs $+0.027$ nats over no block at $cr{=}16$, and twelve is worse still (intervals in Appendix F). Direct projection after pooling is the best choice among these depths. ImageNet initialization increases the gap, and freezing those weights increases it again. The initialization analysis appears in Appendix E. A representational analysis of the trained checkpoints reports 3.9% weight drift for pretrained weights and 29.5% for random initialization. The effective ranks after the block are 85 and 162. The bypass has effective rank 170. Linear CKA with the bypass output is 0.63 for pretrained weights and 0.78 for random initialization (Appendix N).

Source-order sensitivity diagnostic. Two diagnostic columns of Table 7 separate a weak compressor from a broken run. Every branch is better with a document than without one. Their sensitivity to token order differs sharply. Permuting the real source tokens raises Pool+Project perplexity by 4.41 points. The increase is only 0.13 for the pretrained-ViT branch. The lower-quality branches are also less sensitive to source order. This intervention preserves the document's token multiset and serves as a behavioral diagnostic.

The depth result replicates. The discovery comparison and two paired deterministic-sampler replications all favour Pool+Project at $cr{=}16$. Point-wise document-cluster intervals exclude zero for NLL and token accuracy in every run (Table 8). The discovery run used the original sampler. The two replications use the deterministic sampler described in Appendix A.

The depth result transfers to a second decoder. Both branches were retrained end-to-end with Qwen2.5-0.5B-Instruct in place of SmolLM2-1.7B-Instruct under the same training and evaluation protocol, with decoder-specific tokenization and interfaces, and scored on the same FRESH slice at $cr{=}16$. The ordering is unchanged: held-out perplexity is 6.917 for Pool+Project against 16.820 for the 12-layer mixer, a paired document-clustered penalty of $+0.8887$ nats $[+0.8462, +0.9308]$. The token-permutation diagnostic also repeats: Pool+Project loses 14.3 perplexity points when source order is destroyed, while the mixer loses 4.9. The lower-quality branch is again less sensitive to token order. Together with the matched-budget comparison in Section 4, this experiment reproduces both main orderings on a second decoder family.

Learned pooling earns its keep. Zeroing the *trained* pool queries, which reduces the pool to uniform averaging, costs held-out quality at both rates. This intervention measures the contribution of the learned weights at inference. A separately trained mean-pool arm is worse by $+0.0193$ nats at $cr{=}16$ in a paired comparison. Its $cr{=}8$ contrast is cross-seed and therefore descriptive (Appendix F). Together, the two interventions support a small local benefit from learned pooling in this setup.

5.1 Robustness checks

An inference-time padding intervention. Padding exposure provides an alternative explanation for the depth gap because the mixer attends over padded latent slots. On fixed checkpoints, toggling the key-padding mask changes PPL by at most 0.53% on a 128-item probe. For the random mixer at $cr{=}16$ the change is 0.05%, compared with a 15.4% Pool+12L-versus-bypass PPL gap on FRESH. A synthetic-padding probe gives similarly small point shifts (Appendices L, M). The scale of these shifts is far below the observed depth gap. At $cr{=}1$ the latent budget saturates, so the synthetic probe there measures lost content.

A six-rate, fixed-checkpoint diagnostic. The curriculum confounds rate with elapsed training,

so Table 5 (appendix) scores one checkpoint per branch, taken at the end of the $cr{=}8$ stage, at every $cr \in \{1, 2, 4, 8, 16, 32\}$, varying only the evaluation rate within each branch. Pool+Project leads in point estimates at every rate through $cr{=}16$, and the depth penalty is *non-monotone*, peaking at $cr{=}8$. Elapsed training therefore contributes to the monotone trend observed across curriculum stages. The sweep is qualitative. At $cr{=}32$, beyond the rates seen by this checkpoint, perplexity and token accuracy disagree. We treat this point as unstable extrapolation.

Precision control: the depth penalty persists with fp32 optimizer masters. We batch-pair a full replication in which the mixer's master weights, gradients, and optimizer moments are fp32 while forward parameters remain bf16. The fp32 mixer is worse than its bf16 counterpart by $+0.3379$ nats, and the mixer-versus-bypass depth penalty remains positive in both regimes. The precision control increases the observed depth gap. Full intervals and implementation details are in Appendix K.

Systems: parameters, latency, memory. The ablation also moves three system costs in one direction (Table 6, appendix). Dropping the twelve blocks removes parameters, shortens the forward pass at both rates, and lowers peak memory. The same branch also wins held-out quality. Document latents are query-independent and can be cached across questions. At the 15 questions per document in this corpus, measured amortised latency stays within 2.3% of truncation (Appendix I).

6 Discussion

The matched-reader experiments sharpen the system comparison. First-N, LLMLingua-2, and BM25 each have a reader trained on their own context distribution. Pool+Project leads all three at both compression rates on SmolLM2. The same comparison with Qwen2.5 gives a larger NLL advantage. The gain therefore persists across two decoder families and across retention policies that range from prefix selection to question-conditioned retrieval.

The architecture study points to a simple allocation rule inside the compressor. Learned pooling provides a small benefit, while transformer depth after pooling reduces quality. The 12-layer result repeats across three SmolLM2 runs and transfers to Qwen2.5. The one-layer arm follows the same ordering. Order permutation further shows that Pool+Project responds more strongly to source order than the lower-quality mixer branches. These results favor preserving information at the bottleneck and projecting it directly into the decoder space.

The practical comparison joins quality with measured cost. Removing the mixer saves 86.4M parameters and lowers latency and peak memory. Document latents can be cached once and reused for multiple questions. With the query density of PwC QA, amortized latency remains within 2.3% of raw-token retention. Latent compression is therefore most attractive when a document supports repeated queries under a fixed context allocation.

7 Conclusion

We asked whether a trained latent compressor is worth its context slots, giving every matched system the same per-example cap. Pool+Project achieves the strongest quality among the tested common-cap systems on SmolLM2. Its advantage holds against first-N, LLMLingua-2, and BM25 readers trained on their respective inputs. Qwen2.5 reproduces the main comparison. Among post-pooling depths $\{0, 1, 12\}$, direct projection gives the best held-out quality. The 12-layer ordering repeats in three SmolLM2 runs and with Qwen2.5. The same removal saves 86.4M parameters and reduces measured latency and memory. Effective latent compression in this setting comes from learning what to preserve at the bottleneck and presenting it directly to the decoder.

Limitations

The evaluation uses one English document-QA dataset whose source documents fit within $1{,}024$ encoder tokens. Its questions and answers were generated by GPT-4, and the answers are usually short. The reported evidence therefore characterizes short-document QA under this data construction.

The experiments use SmolLM2-1.7B-Instruct and Qwen2.5-0.5B-Instruct. The common-cap study controls decoder positions for every example. Training costs and upstream modules remain system-specific. Teacher-forced NLL and token accuracy are primary because the systems have different generation length distributions.

Ethics Statement

We train and evaluate on the PwC QA data of ICAE (Ge et al., 2024), which pairs Pile-derived news articles with questions and answers generated by GPT-4. The corpus contains third-party copyrighted material. We cite the upstream source and distribute no source documents. The supervision is model-generated, so the metrics measure agreement with a model-written reference. The source articles can concern identifiable people, and generated answers can contain factual errors. Released artifacts therefore contain code, evaluation scripts, and aggregate metrics.

Acknowledgments

This work is an output of a research project (HSE-BR-2025-025) implemented as part of the Basic Research Program at HSE University.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Clémentine Fourrier, Hugo Larcher, Ben Burtenshaw, Haojun Zhao, Caleb Fahlgren, Mathieu Morlon, Agustín Piqueres Lajarín, Xuan-Son Nguyen, Cyril Zakka, and 3 others. 2025. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, Yushi Bai, Jie Tang, Hongning Wang, and Minlie Huang. 2026. Glyph: Scaling context windows via visual-text compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 37145–37158, San Diego, California, United States. Association for Computational Linguistics.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *Advances in Neural Information Processing Systems*, volume 37, pages 109487–109516. Curran Associates, Inc.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732, Vienna, Austria. Association for Computational Linguistics.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Xinting Huang, Dong Yu, and Zhicheng Dou. 2025. A silver bullet or a compromise for full attention? a comprehensive study of gist token-based context compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4861–4879, Vienna, Austria. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR 2021)*.

Yair Feldman and Yoav Artzi. 2025. No mean feat: Simple, strong baselines for context compression. *Preprint*, arXiv:2510.20797.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Weronika Łajewska, Momchil Hardalov, Laura Aina, Neha Anna John, Hang Su, and Lluis Marquez. 2025. Understanding and improving information preservation in prompt compression for LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17520–17541, Suzhou, China. Association for Computational Linguistics.

Ivan Yee Lee, Cheng Yang, and Taylor Berg-Kirkpatrick. 2025. Optical context compression is just (bad) autoencoding. *Preprint*, arXiv:2512.03643.

Ang Li, Sean McLeish, Haozhe Chen, Nimit Kalra, Zaiqian Chen, Artem Gazizov, Venkata Anoop Suhas Kumar Morisetty, Bhavya Kailkhura,

Harshitha Menon, Zhuang Liu, Brian R. Bartoldson, Tom Goldstein, Sanae Lotfi, Micah Goldblum, and Pavel Izmailov. 2026. End-to-end context compression at scale. *Preprint*, arXiv:2606.09659.

Zongqian Li, Yixuan Su, and Nigel Collier. 2025. 500xCompressor: Generalized prompt compression for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25081–25091, Vienna, Austria. Association for Computational Linguistics.

Maxime Louis, Hervé Déjean, and Stéphane Clinchant. 2025. PISCO: Pretty simple compression for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15506–15521, Vienna, Austria. Association for Computational Linguistics.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, pages 19327–19352. Curran Associates, Inc.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.

Hippolyte Pilchen, Edouard Grave, and Patrick Pérez. 2026. ARC-Encoder: Learning compressed text representations for large language models. *Transactions on Machine Learning Research*.

Qwen Team. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Olivier Roy and Martin Vetterli. 2007. Effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference (EUSIPCO)*, pages 606–610.

Yaorui Shi, Shugui Liu, Yu Yang, Wenyu Mao, Yuxin Chen, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, and An Zhang. 2026. MemOCR: Layout-aware visual memory for efficient long-horizon reasoning. *Preprint*, arXiv:2601.21468.

Jiwei Tang, Zhicheng Zhang, Shunlong Wu, Jingheng Ye, Lichen Bai, Zitai Wang, Tingwei Lu, Lin Hai, Yiming Zhao, Hai-Tao Zheng, and Hong-Gee Kim. 2026. GMSA: Enhancing context compression via group merging and layer semantic alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28690–28704, San Diego, California, United States. Association for Computational Linguistics.

Alex Jinpeng Wang, Linjie Li, Yiqi Lin, Min Li, Lijuan Wang, and Mike Zheng Shou. 2024. Leveraging visual tokens for extended text contexts in multi-modal learning. *Preprint*, arXiv:2406.02547.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. DeepSeek-OCR: Contexts optical compression. *Preprint*, arXiv:2510.18234.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2026. DeepSeek-OCR 2: Visual causal flow. *Preprint*, arXiv:2601.20552.

Ling Xing, Alex Jinpeng Wang, Rui Yan, Xiangbo Shu, and Jinhui Tang. 2025. Vision-centric token compression in large language model. In *Advances in Neural Information Processing Systems*, volume 38, pages 33080–33110. Curran Associates, Inc.

A Architecture and training details

Chunked attention pool. The encoder states of a document are cut into non-overlapping chunks of cr tokens. Writing $C_j \in \mathbb{R}^{cr\times d}$ for chunk j and $q_1, \ldots, q_K \in \mathbb{R}^d$ for the learnable queries, the latent for that chunk is

$$o_j \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathrm{softmax}\!\left(\frac{C_j q_k^{\top}}{\sqrt{d}}\right)^{\top} C_j, \qquad (1)$$

with padded positions masked before the softmax. We instantiate $K{=}4$ queries at exactly zero. Because the initialization, attention computation, and final average are permutation-symmetric, all four rows remained bitwise identical in every inspected checkpoint. Equation 1 gives the implemented form. The realised model has one distinct query vector because all four rows remain identical.

Optimization. Training uses one accelerator, bf16, gradient checkpointing, and a fused linear cross-entropy kernel for the decoder. AdamW has $(\beta_1, \beta_2)=(0.9, 0.999)$, $\epsilon=10^{-8}$, zero weight decay, and global gradient-norm clipping at 1.0. Batch size is 64 with no gradient accumulation. The learning rate warms linearly from zero to 10^{-4} for 100 updates and is then constant. There are 11,325 optimizer updates over three epochs. The internal counter begins at one and increments after each update, so the final checkpoint is indexed 11,326. The curriculum is $cr=1$ for updates 1–1,499, $cr=2$ for 1,500–2,999, $cr=4$ for 3,000–4,499, $cr=8$ for 4,500–5,999, and $cr=16$ thereafter. Optimizer moments carry across stage boundaries and a checkpoint is forced at each boundary, so branches are compared at equal update counts within a stage. Source documents are filtered at 1,024 encoder tokens. This fixed filter passes all 18,146 test items. We cap prompts at 512 and answers at 1,024 decoder tokens, and the latent sequence is capped at 1,024 positions by the position-embedding table. The truncated-SFT reference decoder is retrained under the compressor's own schedule (three epochs, effective batch 64, constant learning rate 10^{-4} after warmup). An earlier, lighter schedule (two epochs, rate 10^{-5}) scored 5.687/6.544, so the margin over truncated-SFT is stable under the stronger schedule. The first-N, LLMLingua-2, and BM25 readers are trained separately on the contexts produced by their respective selection policies. Each uses the same three-epoch schedule.

For paired replications, the epoch permutation is a deterministic function of seed and epoch, with a separate DataLoader generator. Branch-specific modules are constructed under one seed and shared projectors and positional embeddings are then reseeded identically. Initialization hashes and the SHA-256 hash of the first batch verify pairing. The discovery seed-42 run used the previous loader implementation and is therefore always reported separately from deterministic-sampler replications 43 and 44.

The separate system-level Pool+Project runs at seeds 43 and 44 score 4.180 and 4.154 PPL at $cr=16$. Their paired contrasts use the earlier lighter-schedule SFT reader at 6.544 PPL. The differences are -0.4483 $[-0.4736, -0.4230]$ and -0.4541 $[-0.4804, -0.4281]$ nats. These checkpoints belong to a separate campaign from the depth replications in Table 8.

Evaluation. Batch 8. Evaluation covers trained compressor checkpoints, frozen and task-adapted readers on selected raw text, and a frozen instruct reader on the full document. The slices are those of §3.2. DEV has 135 documents, FRESH has 205, and RQ3-DEV has 68. The padding probe contains 128 rows from 9 documents. Beyond the slice statistics quoted in §3.2, near-duplicate document pairs (Jaccard > 0.5) number 2 of 1,224 (0.16%) and the answer string appears in the question in 0 of 18,146 items.

Logged rate quantities. Bits per source token is a constant rescaling of mean loss in this implementation. Their ratio is 0.1409 at every compression rate. The mean training ratio combines encoder and decoder tokenizers. We therefore report the measured decoder-slot cap for each item (Appendix J).

B Budget-matched truncation grid

Context given to the decoder ($cr=4$)	PPL	TA
Pool+Project latents (ours)	3.358	0.6884
answer-centred window†	5.816	0.6368
query-aware BM25	6.807	0.6116
first-N tokens	7.205	0.5904

Table 3: **An additional post-hoc operating rate.** The final seed-42 Pool+Project checkpoint is evaluated on FRESH at $cr=4$. teacher-forced perplexity and token accuracy are computed on answer tokens. The retention policies are read by the frozen instruct decoder at the compressor's own per-example budget. Paired, pointwise document-clustered contrasts in nats per answer token, compressor − baseline: -0.7635 $[-0.7923, -0.7356]$ against first-N, -0.7065 $[-0.7325, -0.6798]$ against question-conditioned retrieval and -0.5493 $[-0.5773, -0.5210]$ against the privileged window, all excluding zero. †consults the gold answer and serves as a privileged control.

At rate cr the compressor emits $V=\lceil L/cr \rceil$ latents, and each truncation policy receives a cap of $N=V$ raw document tokens, so the configuration receives the same per-example document-context cap (§4, realized lengths in Appendix J). Table 4 adds the two policies the body omits, last-N and a random window. It is scored on the **RQ3-DEV** slice (1,000 items over 68 documents) with the compressor taken at the end of the $cr=8$ curriculum stage. Table 1 uses FRESH, so absolute levels are compared only within each table.

The **answer-centred window** is a privileged baseline. It takes the contiguous N-token span with

maximal overlap with the reference answer. Such a span is found for 36.4% of held-out items (hit rate 0.364) and the policy falls back to first-N on the rest. It provides one privileged contiguous-window control. The **full document** row is a frozen-reader reference that is independent of cr.

Context given to the decoder	PPL	TA
cr=8 *(matched budget)*		
Pool+Project latents (ours)	**3.610**	**0.673**
first-N tokens	8.094	0.572
last-N tokens	9.346	0.554
random N-token window	9.068	0.552
answer-centred window†	6.437	0.624
truncated-SFT decoder‡	5.562	0.608
cr=16 *(matched budget)*		
Pool+Project latents (ours)	**5.099**	**0.610**
first-N tokens	9.153	0.550
last-N tokens	10.370	0.537
random N-token window	10.059	0.538
answer-centred window†	7.615	0.593
truncated-SFT decoder‡	6.376	0.583
full document (reference)	3.356	0.737

Table 4: Budget-matched truncation grid, held-out, 1,000 samples over 68 documents on RQ3-DEV. Table 1 uses FRESH, so absolute levels are compared only within each table. The table gives teacher-forced PPL and token accuracy (TA) on answer tokens, best matched-budget cell per rate in bold. Free generation is reported separately and is *secondary*. ROUGE-L is 0.342 at cr=8 and 0.312 at cr=16 for the truncated-SFT row, and falls in the 0.010–0.030 band for every row scored with the frozen instruct decoder, which emits no end-of-sequence token and generates to the length cap. This band mainly reflects output length. †privileged answer-centred baseline (hit rate 0.364, falls back to first-N otherwise). ‡decoder fine-tuned on documents truncated to the same per-item budget. All other rows use the frozen instruct decoder. Table 10 reports length-controlled generation.

C Paired differences for the RQ3 contrasts

The four pre-specified token-accuracy intervals promised in the caption of Table 2 (FRESH, pointwise document-clustered, 5,000 resamples) all exclude zero: at cr=8, $[+0.1145, +0.1249]$ against first-N, $[+0.0666, +0.0767]$ against the privileged window, $[+0.0656, +0.0752]$ against truncated-SFT, and $[-0.0429, -0.0330]$ against the full-context reference. At cr=16 they are $[+0.1046, +0.1137]$, $[+0.0641, +0.0733]$, $[+0.0592, +0.0679]$, and $[-0.0758, -0.0638]$ respectively.

D The rate-controlled grid

Table 5 is the rate-controlled grid read qualitatively in the robustness checks.

cr	Pool+ Project	Pool+12L Mixer	+Pretr. ViT	Δ depth (PPL pts)
1	**3.833**	4.216	7.586	+0.383
2	**3.138**	3.571	7.513	+0.434
4	**3.029**	3.619	7.458	+0.590
8	**3.610**	4.308	7.453	+0.698
16	**5.099**	5.594	7.577	+0.496
32	7.182	**7.119**	7.719	−0.063

Table 5: **Rate-controlled reading: one checkpoint, six rates.** Held-out, teacher-forced perplexity on answer tokens. One checkpoint per branch from the end of the cr=8 curriculum stage is scored at every $cr \in \{1, 2, 4, 8, 16, 32\}$ on a single 1,000-row held-out set of 68 documents. Best per row in bold. **This is the paper's only six-rate controlled sweep**. Within each branch, weights, data, and elapsed training are fixed while evaluation rate varies. It separates rate from the curriculum-stage confound. Δ depth = Pool+12L Mixer − Pool+Project in perplexity *points*, positive meaning the added block is worse. It is non-monotone and reaches its largest value at cr=8. It changes sign only in the cr=32 row, where all three branches have degraded to perplexity 7.1–7.7 and token accuracy near 0.56 and where token accuracy still marginally favours Pool+Project (0.568 against 0.565). Because this checkpoint had trained only to cr=8, its cr=16 and cr=32 rows are extrapolation beyond the range it saw. Different evaluation set from Table 7. Absolute levels are compared only within each table. One run per branch. The grid is read qualitatively and no intervals are attached.

	Params	Forward (ms)		Peak mem.
	(non-dec.)	cr=8	cr=16	cr=8
Pool+Project	**163.2M**	**68.5**	**60.2**	**3.92 GB**
Pool+12L Mixer	249.6M	72.8	65.0	4.09 GB
difference	−86.4M	−5.9%	−7.4%	−4.2%

Table 6: **Cost of the post-pooling block (microbenchmark).** Parameter counts cover every non-decoder module of the compressor (encoder, pooling stage, both projectors, and the post-pooling block where present). The decoder is identical in the two branches, so it is excluded from the parameter column. Timing and memory include it. *Batched teacher-forcing forward latency, batch=8, source=1024, prompt=16, answer=32, A100, bf16, excluding generation.* These are forward-pass measurements. The difference row is Pool+Project minus Pool+12L Mixer. A negative entry means Pool+Project is smaller, faster or leaner. Every entry here is a microbenchmark measurement. Held-out quality for the same two branches is in §5.

Document given	Pool+ Project	Pool+12L Mixer	+Pretr. ViT	+Frozen ViT
held-out PPL (lower is better)				
original order	**4.062**	4.688	7.504	7.782
tokens permuted	8.475	8.165	7.636	7.825
none	9.924	9.674	9.975	10.096
held-out token accuracy (higher is better)				
original order	**0.649**	0.625	0.560	0.556
tokens permuted	0.546	0.551	0.558	0.555
none	0.535	0.537	0.539	0.537

Table 7: **Depth and prior ablation with two diagnostic reading conditions.** Held-out, teacher-forced perplexity and token accuracy on answer tokens at cr=16, final checkpoints, on the FRESH evaluation set (3,000 rows over 205 documents). Columns are the four branches (abbreviated), rows are the reading condition fed to the compressor. "Tokens permuted" independently applies a random permutation to each item's non-padding source tokens while keeping padding at the tail. It preserves the token multiset. This and the no-document row are diagnostics. Depth intervals: Table 8. Remaining contrasts (nats/answer token): pretrained−random $+0.4705$ $[+0.4513, +0.4900]$, frozen−pretrained $+0.0364$ $[+0.0322, +0.0407]$. Shuffle penalties (token-permuted minus original-order NLL) are $+0.735$ / $+0.555$ / $+0.018$ / $+0.005$ nats for Pool+Project / 12L / Pretrained / Frozen, a $\sim$40$\times$ span, every interval excluding zero.

E The ImageNet-initialization diagnostic

Optical stacks place a pretrained aggregator immediately after their compressor, and our mixer occupies the position-matched slot. Alongside the random initialization of §5 we therefore also ran the same twelve blocks warm-started from supervised ImageNet classification (`google/vit-base-patch16-224`), trainable and frozen. These weights were trained on images and transferred to pooled text latents. We use the transfer as an initialization diagnostic. It underperforms here: 7.504/0.560 trainable and 7.782/0.556 frozen, against 4.688/0.625 for the identical architecture initialized randomly (FRESH, cr=16, pretrained−random $+0.4705$ nats $[+0.4513, +0.4900]$, freezing a further $+0.0364$ $[+0.0322, +0.0407]$). In this setup, an ImageNet transformer-block initialization is a weaker starting point for the text-latent mixer. The observed gap is an optimization penalty for this transferred block.

Run	Pool+Proj.	+12L Mixer	ΔPPL [95% CI]
discovery (42)	4.0618	4.6876	$+0.6258$ $[+0.5702, +0.6795]$
replication (43)	4.0605	4.7145	$+0.6540$ $[+0.6025, +0.7046]$
replication (44)	4.0574	4.9512	$+0.8938$ $[+0.8239, +0.9639]$

Table 8: **The depth contrast replicates.** Held-out perplexity at cr=16 on the FRESH set, and the paired difference $\Delta\text{PPL} = \text{PPL}(\text{Pool+12L Mixer}) - \text{PPL}(\text{Pool+Project})$ in perplexity points, so a positive value means the added block is worse, with pointwise document-cluster intervals over the 205 documents. The token-accuracy differences also exclude zero and favour Pool+Project: -0.0243 $[-0.0268, -0.0217]$, -0.0258 $[-0.0281, -0.0235]$ and -0.0342 $[-0.0368, -0.0316]$, reported as mixer − Pool+Project. On the additive scale the same three differences are $+0.143$, $+0.149$ and $+0.199$ nats per answer token. The first row is labelled the *discovery* run because it was trained under the previous sampler implementation.

F Interval bounds for the one-layer and pooling contrasts

All four branches emit the same number of latents per item at cr=16 and are therefore scored at an identical decoder budget. Table 5 is read qualitatively and carries no intervals by design.

The one-layer and pooling contrasts quoted qualitatively in §5 carry the following FRESH-slice document-cluster intervals, in nats per answer token. One mixer block over no block at cr=16 (paired): $+0.0272$ $[+0.0231, +0.0313]$. One block against twelve: -0.122 $[-0.131, -0.113]$. Zeroing the trained pool queries: $+0.0936$ $[+0.0846, +0.1028]$ at cr=8 and $+0.1855$ $[+0.1732, +0.1982]$ at cr=16. The trained-from-scratch mean-pool variant against learned pooling: $+0.0193$ $[+0.0149, +0.0236]$ at cr=16 (paired, replication seed) and $+0.030$ $[+0.025, +0.035]$ at cr=8 (cross-seed).

G Free generation and length control

The manual row below is a post-hoc 150-item check. Its annotation artifact records one binary acceptability judgment for each system answer against the reference (`cvlm_ok` and `sftm_ok`). One annotator produced the labels. Its document-bootstrap interval reflects item sampling conditional on those labels.

Free-generation scores are *secondary* throughout this paper. The truncated-SFT training data appends an end-of-sequence token to answers. The compressor training data omits it, so 75–82% of its generations reach the 256-token cap (about 1,172 characters against 239 for the reference) while 2–

FRESH, cr=16	ours	matched-SFT	Δ [95% CI]
token-F1	0.3686	0.3383	+0.0303 [+0.0230, +0.0381]
containment			+0.0642 [+0.0566, +0.0723]
manual (%)	46.7	36.0	+10.7 [−0.7, +21.4]

Table 9: **Free-running and manual evaluation** (FRESH, one stopping rule: first sentence). Paired differences with document-clustered intervals, positive favours the compressor. The manual row scores 150 items and is suggestive at this sample size. Given twice the budget, matched-SFT at cr=8 reaches ΔF1 −0.000 [−0.008, +0.007] against the cr=16 compressor while containment still favours the compressor by +0.037. Under raw stopping the F1 sign flips (−0.197). Generation protocols: Appendix G.

3% of the SFT decoder's do (about 246 characters). Any precision-weighted overlap score therefore measures stopping behaviour. Table 10 scores the same two systems under four protocols on the RQ3-DEV slice (68 documents). We make **first_sentence** primary because it operates independently of reference length. The two truncation protocols use oracle length. The reversal appears for Pool+Project. Against a matched-SFT decoder given twice the budget, the cr=16 compressor has equal F1 and higher containment.

Protocol	P+P	SFT	Δ [95% CI]
length-free			
first-sent. R-L F	**0.3302**	0.3032	+0.0269 [+0.0125, +0.0411]
first-sent. BLEU-4	**12.04**	9.97	+2.07 [+1.23, +2.85]
raw (confounded by length)			
raw R-L F	0.1351	**0.3122**	−0.1770 [−0.1928, −0.1614]
raw R-L recall	**0.5115**	0.3256	+0.1860 [+0.1727, +0.1993]
raw R-L precision	0.0852	**0.3400**	−0.2548
oracle-length			
trunc-word R-L F	**0.3710**	0.3259	+0.0451 [+0.0324, +0.0573]

Table 10: Free generation, held-out, on the RQ3-DEV slice (1,000 items over 68 documents). P+P = Pool+Project at cr=16 and SFT = the truncated-SFT decoder at cr=16. R-L = ROUGE-L. Δ is P+P − SFT, with document-clustered 95% intervals. The *first-sent.* block is primary: it is the only protocol here that operates independently of reference length. The *raw* block is reported to make the artifact visible: F-score favours the SFT decoder and recall favours the compressor by a comparable margin, because the compressor generates to the cap, and the *trunc-word* row truncates the prediction to the reference's word count and is therefore *oracle-length*, which is why no claim rests on it. Exact match is 0.0000 for every configuration scored with the frozen instruct decoder, which also emits nothing on 70.8–90.6% of items, so frozen-decoder rows are excluded from generation tables and exact match is dropped entirely (§3.3). The branch ordering is identical under all four protocols and matches the perplexity ordering.

H Pairwise-judge protocol

The exploratory judge in §4 uses the Hugging Face repository identifier `Qwen/Qwen3-8B` through its tokenizer chat template, with `add_generation_prompt=True` and thinking disabled. Inputs contain the PwC question, the reference answer stored in the Pool+Project dump, and the two generated answers. The source document is excluded. For first-sentence evaluation, each answer is cut at the first occurrence of ".", "!", or "?" followed by whitespace. A Python RNG with seed 42 independently assigns Pool+Project to side A or B for each item. The exact user message is:

```
You are grading two answers to a question
about a document.
Question: {q}
Reference answer: {ref}
Answer A: {a}
Answer B: {b}
Which answer is more consistent with the
reference answer? Judge only factual
agreement with the reference, not length
or style. Reply with exactly one token:
A, B, or TIE.
```

Generation uses batches of 16, greedy decoding (`do_sample=False`), and at most four new tokens. After uppercasing and stripping the decoded continuation, outputs beginning with A or B are mapped back through the randomized order. Every other output, including an explicit `TIE`, is counted as a tie. There are no retries. The tie category therefore includes nonconforming output. Per-item verdicts were stored. The reported summaries have no document-clustered uncertainty and are descriptive.

Every completed pair contains all 3,000 FRESH items. Under first-sentence stopping, Pool+Project has descriptive win/lose/tie rates of 44.8/33.9/21.3% against the SmolLM2 12-layer mixer at cr=16. The within-Qwen depth pair is 51.0/20.3/28.7%. For Pool+Project–matched-SFT at cr=16, raw stopping gives 53.0/35.0/12.0%, the same observed direction as first-sentence stopping.

I Amortised serving cost

The compressor adds a document encoder to the serving path. Its latents are query-independent, so encoding is paid once per document and reused across questions. At cr=16, the measured document-side compression path for a 512-token document takes 28.5 ms. A query that reuses the

cached latents then takes 83.9 ms, the same measured latency as a truncation query that re-reads raw tokens. A first, cold query therefore costs 112.4 ms, while the cost amortised over the 15 questions this corpus asks per document is 85.8 ms, 2.3% above truncation. The independently measured amortised values at $cr{=}8$ are 92.5 against 90.8 ms, a 1.9% overhead. The cached document representation is 0.125 MB at $cr{=}16$ and 0.25 MB at $cr{=}8$. These are microbenchmark timings on one card with synthetic inputs of fixed length, measuring prefill to the first token. They characterize the reported setup.

J Evaluation alignment

Four things a reader is entitled to check about Table 1 are separated below.

Identical items. Paired FRESH analyses are configured to require the same 3,000 sample identifiers and 205 document clusters in both per-sample dumps. Incomplete dumps cause the analysis to stop. The earlier RQ3-DEV grid contains 1,000 items over 68 documents and is reported separately. No comparison in §4 crosses evaluation sets.

Identical decoder-slot cap. For item i, all matched-budget policies receive the same document cap $V_i = \min(\lceil L_i/cr \rceil, V_{\max})$. Text selectors are retokenized before evaluation and may use fewer than V_i positions because their selection units are discrete. Every output stays within the cap. The comparison uses a matched per-example cap, while realized counts can differ.

LLMLingua-2 calibration. The budget-calibrated selector of §4 uses `llmlingua==0.2.2` with `microsoft/llmlingua-2-xlm-roberta-large-meetingbank`. Only the document is compressed. The question is excluded from selector input. We score each unique document once and bisect the continuous word-probability percentile for 32 iterations, retaining the largest attainable output within the SmolLM2-token cap. Forced tokens and consecutive-token dropping are disabled, and the renderer is required to match the package API byte-for-byte on 25 documents. No selection exceeds the cap or truncates a prefix. Mean utilization is 0.9874 at $cr{=}8$ and 0.9746 at $cr{=}16$. Exact-budget fractions are 0.6820 and 0.6211. The median shortfall is zero tokens and the ninetieth percentile is two tokens. The reader is trained on these selected contexts under the shared reader schedule.

BM25 granularity. BM25 selects sentence units and is evaluated under an implementation rule that admits a sentence only when it fits the same cap. Selected sentences are restored to document order. The reader is trained on BM25 contexts under the shared reader schedule. Sentence-level selection can leave a small portion of the cap unused.

For the four ablation branches the budget is identical by construction: they share the pooling stage, so at $cr{=}16$ each emits $\lceil L/16 \rceil$ latents for the same document.

K Precision-control details

The control gives fp32 master weights, gradients, and moments to exactly 86,389,248 mixer parameters while retaining bf16 forward parameters and the legacy path elsewhere. It runs the full curriculum on replication seed 43 and is batch-paired with its bf16 counterpart. On FRESH at $cr{=}16$, fp32-master minus bf16 for the mixer is $+0.3379$ $[+0.3219, +0.3535]$ nats. The depth penalty is $+0.1491$ $[+0.1387, +0.1596]$ under bf16 and $+0.4870$ $[+0.4633, +0.5106]$ under fp32 masters. These pointwise intervals resample documents. The intervention jointly changes update accumulation and optimizer-state precision. The control covers random initialization under both precision regimes. Held-out contrasts provide the reported evidence.

L Padding intervention: full numbers

The body reports that masking padded latent keys moves held-out perplexity by at most 0.53%. Table 11 gives the four signed differences behind that statement. The convention is (mask on) − (mask off) evaluated on *the same* weights, so a positive entry means the mask made perplexity slightly worse. The comparison uses the magnitude $|\Delta|$. Pool+Project is absent by construction: it has no post-pooling self-attention for a key mask to act on.

M Synthetic-latent-padding probe

The second check appends explicitly masked latent slots amounting to 25/50/75% of the pooled sequence and re-scores the same items. At $cr{=}16$ the padded sequence stays far below the latent cap. The observed PPL shifts range from -0.06 to -0.01 across branches and mask settings. These are small point differences without intervals.

At $cr{=}1$ the same intervention measures truncation at the latent cap. The number of latents is

Branch	cr	Δ PPL	Δ rel.
12L Mixer	1	-0.0087	-0.17%
12L Mixer	16	$+0.0025$	$+0.05\%$
12L Pretrained-ViT Mixer	1	$+0.0120$	$+0.14\%$
12L Pretrained-ViT Mixer	16	$+0.0445$	$+0.53\%$

Table 11: Inference-time padding-mask intervention on a single final checkpoint per branch (128 held-out items over 9 documents, seed 42). All four magnitudes are at most 0.53%. The random-mixer change at cr=16 is 0.05%, versus a 15.4% held-out PPL gap between the 12-layer mixer and bypass on FRESH. The intervention provides no evidence that inference-time padding explains the magnitude of the depth gap on this probe. These point estimates characterize the inference-time intervention.

$\lceil L/cr \rceil$ with L capped at 1,024 source tokens and the latent budget capped at 1,024 slots, so at cr=1 the budget is exactly saturated and any lengthening of the source pushes the tail of the document past the cap, where it is silently dropped. The resulting degradation (diagnostic scale: $4.72 \rightarrow 5.76$ PPL, a difference of $+1.04$, at 75% appended slots) measures lost content. It is largest for Pool+Project, which has no post-pooling attention at all. The cr=1 row of the probe is consequently excluded from the padding argument, which rests on cr=16 and on the mask on/off intervention of Table 11. The saturation itself is a brittleness of the architecture at the latent-budget boundary and is listed among the limitations.

N Representational analysis of the post-pooling block

The initialization diagnostic of Appendix E shows a quality gap for ImageNet-pretrained weights. This section probes the trained checkpoints at cr=16 on 256 held-out documents with representational metrics summarized in Figure 4.

Weight drift. The pretrained ViT moves only 3.9% from its ImageNet initialization, versus 29.5% for the random weights, a $7.5\times$ ratio (Figure 4a). Because the initializations and parameter norms differ, this ratio is descriptive.

Effective rank and alignment. Effective rank (Roy and Vetterli, 2007) quantifies the dimensionality actually used by the representation. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_{\min(n,d)}$, define $p_i = \sigma_i / \sum_j \sigma_j$. The effective rank is

$$\mathrm{erank}(\mathbf{X}) = \exp\Big(-\sum_i p_i \log p_i\Big), \qquad (2)$$

the exponential of the Shannon entropy of the normalized singular-value distribution. A matrix using all dimensions equally has $\mathrm{erank} = \min(n, d)$. A matrix concentrated on a single direction has $\mathrm{erank} = 1$.

The projector outputs feeding the bypass, pretrained, and random branches have effective ranks 170, 69, and 54, respectively (Figure 4b). After the block, the latter two ranks are 85 and 162. These are branch-level descriptive measurements after separate joint end-to-end training. The pretrained branch's rank increases across its block and remains below the random and bypass branches.

Cosine alignment between input and output of the block (Figure 4c) measures how much the post-pooling transformation rotates the representation vectors. For each latent vector $\mathbf{h}$, we compute $\cos(\mathbf{h}_{\mathrm{in}}, \mathbf{h}_{\mathrm{out}})$ where $\mathbf{h}_{\mathrm{in}}$ and $\mathbf{h}_{\mathrm{out}}$ are the input and output of the post-pooling block, and report the mean over all vectors and documents. A value of 1.0 means the block preserves direction. Lower values indicate larger rotation. The pretrained initialization rotates representations the most (0.30), the random initialization less (0.44), and bypass preserves direction (1.00).

Cross-branch similarity. We measure representational similarity with linear Centred Kernel Alignment (CKA) (Kornblith et al., 2019). Given two representation matrices $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d_2}$ (rows are examples, columns are features), linear CKA is

$$\mathrm{CKA}(\mathbf{X}, \mathbf{Y}) = \frac{\|\mathbf{Y}^\top \mathbf{X}\|_F^2}{\|\mathbf{X}^\top \mathbf{X}\|_F \, \|\mathbf{Y}^\top \mathbf{Y}\|_F}, \qquad (3)$$

where both matrices are column-centred beforehand. CKA is invariant to orthogonal transformations and isotropic scaling, making it suitable for comparing representations of different dimensionality or learned under different initializations.

At the ViT output stage (Figure 4d) the random initialization is more similar to bypass (0.78) than the pretrained initialization is (0.63). This is a descriptive cross-branch comparison.

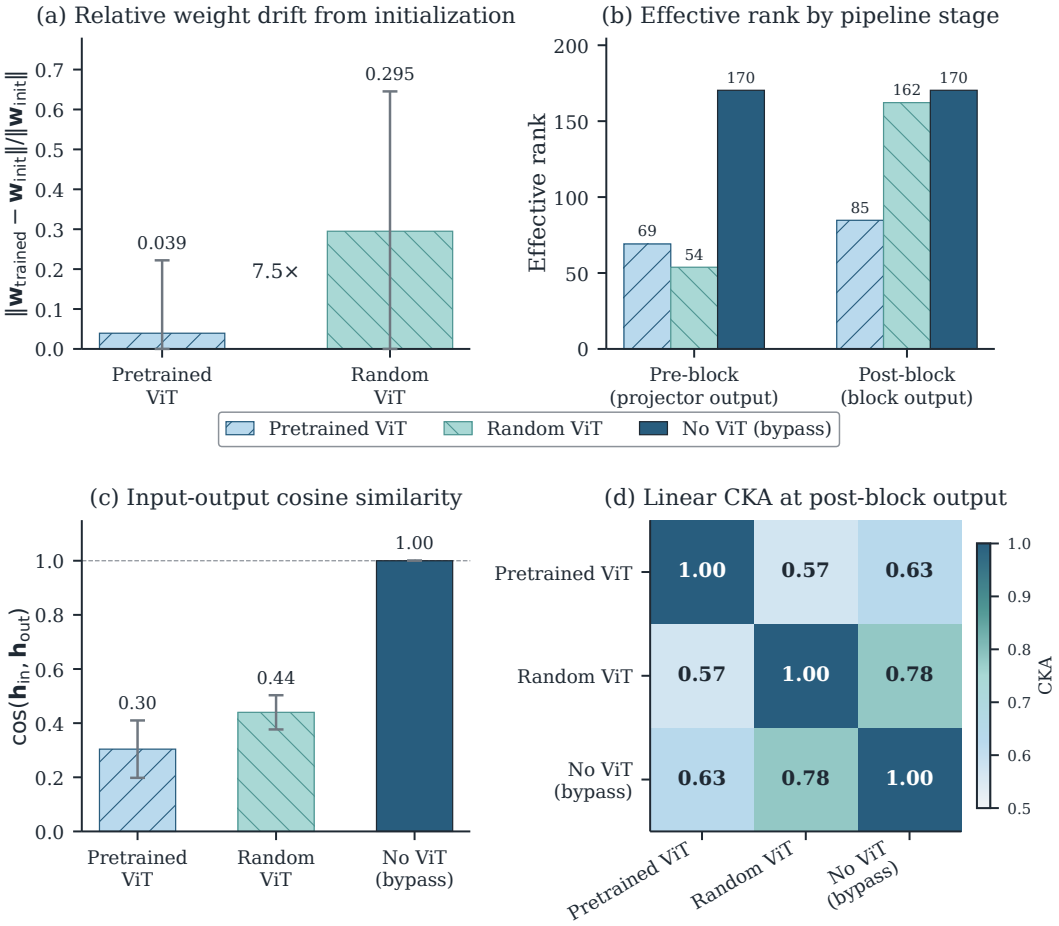


Figure 4: **Representational diagnostics for the three post-pooling branches** at cr=16, 256 held-out documents. (a) Mean relative weight drift from initialization, with min/max error bars for pretrained and random initializations. Bypass has no ViT weights. (b) Effective rank (Roy and Vetterli, 2007) at two pipeline stages: before the ViT block (projector output) and after. (c) Mean cosine similarity between the block's input and output vectors, with standard-deviation error bars. Bypass is 1.0 by construction. (d) Linear CKA (Kornblith et al., 2019) similarity matrix at the ViT output stage. All diagnostics are descriptive. The error bars summarize documents within each trained checkpoint.

RAG Components Are Not Independently Rankable

Vishnu Prasad Vijaya Kumar[1,2] **Iaroslav Neverov**[1] **Pavel Chizhov**[1,2]
Anton Changalidis[1] **Vaiva Kazanaviciute**[1] **Pierre-Carl Langlais**[1]
Anastasia Stasenko[1] **Ivan P. Yamshchikov**[1,2]

[1]PleIAs
[2]CAIRO, Technical University of Applied Sciences Würzburg-Schweinfurt
Correspondence: vishnu@pleias.fr

Abstract

RAG systems are usually compared as complete pipelines, making it difficult to attribute improvements to individual components. We evaluate chunking, retrieval, fusion, and reranking through a full factorial design with paired component substitutions and equal context budgets. Across five benchmarks, the effect of a chunker depends on the retriever that consumes its output, BM25 fusion changes direction by domain, and reranking is the only intervention we test that remains beneficial across upstream configurations. Claim-level transitions further show that similar net recall can conceal very different changes in visible evidence. These findings show that, for evidence delivery, RAG components cannot be ranked independently, and motivate interaction-aware evaluation instead of winner-take-all pipeline comparisons.

1 Introduction

Retrieval-augmented generation is built from interchangeable parts. A collection is chunked, chunks are retrieved, candidates are optionally reordered by a reranker, and some prefix of the result reaches the generator's context window. Practitioners choose among these parts using end-to-end benchmark scores.

Such comparisons cannot identify what a component does. A leaderboard reports which complete configuration won, not why—the failure to identify the sources of empirical gains that Lipton and Steinhardt (2019) describe. When semantic chunking beats fixed-size chunking, the chunker may be better, or may merely suit the retriever it was paired with, or the dataset may favour the resulting evidence geometry: different claims with different consequences that no full-pipeline comparison can separate.

The comparison is also unfair. Passages are not a fixed unit of context, so five 512-token chunks expose twice the text of five 256-token chunks, and a system can score higher simply by supplying more material.

We stop before generation. Answer accuracy is a noisy proxy for evidence delivery: a generator can succeed from parametric knowledge despite poor retrieval, or fail despite sufficient evidence. Adding generators would also add another factor whose own behaviour—such as sensitivity to where evidence appears in the context—would mix with the component effects we want to isolate. Evidence delivery is the stage that chunking, retrieval, and reranking directly control, and a claim that never reaches the context cannot be grounded in it.

Instead we run the complete $4\times13\times4$ factorial grid at five equal token capacities. Because every combination exists, a component effect is a paired difference: hold the question, dataset, capacity, and every other component fixed, change one component, and average.

Three findings follow, and they do not agree.

Chunking and retrieval are complementary rather than separately rankable. On FinanceBench, semantic chunking relative to Fixed-512 costs 2.6 points under dense retrieval and gains 2.7 under late interaction—an interaction of +5.24 points [2.37, 8.23]. A global chunker ranking averages this away.

Lexical fusion is not a safe default. Adding BM25 to a matched neural parent helps on MultiHop-RAG, QMSum and TAT-QA and harms on FinanceBench and LegalBench-RAG. The pooled effect is +0.91 points with an interval spanning zero, describing none of the five domains.

Reranking is the exception. It stays positive after upstream choices vary: in all 240 dataset $\times$ chunker $\times$ retrieval-family $\times$ reranker cells, and in 776 of 780 cells at the level of individual retrievers.

Our contributions are:

- **A paired-substitution framework.** Token-normalized comparison, claim-level recall,

The Proceedings of Workshop on Document Intelligence and Understanding, pages 303–323
October 24, 2026 ©2026 Association for Computational Linguistics

and paired analysis each exist in prior work; we combine them in one complete $4 \times 13 \times 4$ grid, which yields exact counterfactual pairs for component and interaction effects.

- **Component effects are conditional upstream but stable downstream**. The better chunker depends on the retriever, and the effect of BM25 fusion depends on the dataset; both reverse sign. Reranking is positive in all 240 dataset $\times$ chunker $\times$ retrieval-family $\times$ reranker cells.

- **Claim transitions distinguish interventions with equal net effects**. Separating claims gained from claims lost shows whether a substitution exchanges visible evidence or adds it.

2 Related work

Claim-level RAG evaluation. Automated RAG evaluation frameworks assess dimensions such as context relevance, answer faithfulness, and answer relevance, including RAGAs and ARES (Es et al., 2024; Saad-Falcon et al., 2024). Fine-grained factual evaluation has also decomposed generated text into independently verifiable units: FActScore introduced atomic-fact evaluation, while RefChecker represented claims as claim triplets (Min et al., 2023; Hu et al., 2024). Most directly, RAGChecker defines retriever claim recall as the proportion of reference-answer claims entailed by retrieved chunks, alongside several generator diagnostics (Ru et al., 2024). We adapt this claim-level construct specifically to upstream evidence delivery: reference answers are decomposed into claims and each visible context is checked for claim support, but no answer is generated and no generator-quality metrics are computed.

Context normalization and chunking. Comparisons between retrieval and long context-processing have emphasized the trade-off between answer quality, context length, and computation (Li et al., 2024). Chunk granularity itself has been studied through passages, sentences, and propositions: Dense X evaluates four dense retrievers and limits retrieved context to fixed token budgets, demonstrating that granularity affects both retrieval and downstream QA (Chen et al., 2024). Thus, neither token normalization nor multi-retriever chunking evaluation is new individually. Our distinction is to estimate paired chunker effects across sparse, dense, late-interaction, fused, and reranked pipelines, revealing when the effect of a chunker changes with retrieval architecture or dataset.

Interaction-aware evaluation. IR evaluation has long recognized that system effects can vary across topics, motivating resampling and interaction-aware statistical analysis (Voorhees et al., 2017). Concurrent preprints apply factorial designs to compound-AI answer quality and to complementarity among document-, query-, and answer-side RAG interventions (Lukassen et al., 2026; Chong, 2026). Our scope is different. To our knowledge, this is the first large-scale factorial audit of upstream RAG evidence delivery that combines equal-token exposure with claim-level semantic recall and paired interventions across chunking, retrieval architecture, lexical fusion, and reranking. This design identifies not only average component effects, but also chunker–retriever compatibility, domain-conditioned sign reversals, and whether an intervention makes new reference claims visible or removes previously visible ones.

3 Controlled Factorial Evaluation of RAG Components

3.1 Benchmarks

We evaluate on five benchmarks spanning financial, legal, multi-document, meeting, and numerical reasoning tasks: FinanceBench (Islam et al., 2023), LegalBench-RAG (Pipitone and Alami, 2024), MultiHop-RAG (Tang and Yang, 2024), QMSum (Zhong et al., 2021), and TAT-QA (Zhu et al., 2021). Table 1 summarizes the task, evaluation subset, document count, and question count for each benchmark.

3.2 Factorial Component Grid

A configuration is $x = (c, r, z)$, comprising a chunker c, first-stage retrieval condition r, and reranking condition z. As summarized in Table 2, we evaluate the complete $4 \times 13 \times 4$ factorial grid rather than selecting a small number of end-to-end pipelines. Every configuration is evaluated on every dataset at five equal-token context capacities.

All retrievers return at most 20 candidates. Rerankers reorder this fixed set and cannot introduce new fragments, whereas fusion may change candidate membership. The cutoff bounds reranking by construction: evidence below rank 20 is invisible to every downstream condition, so our reranker effects are lower bounds on what an unre-

Table 1: Evaluation datasets.

Benchmarks	Task and evaluation subset	Documents	Questions
FinanceBench (Islam et al., 2023)	Financial-report QA; public 150-question set	11,948	150
LegalBench-RAG (Pipitone and Alami, 2024)	Legal retrieval; complete frozen archive	714	6,889
MultiHop-RAG (Tang and Yang, 2024)	Multi-document news QA; non-null queries	609	2,255
QMSum (Zhong et al., 2021)	Query-focused meeting summarization	35	244
TAT-QA (Zhu et al., 2021)	Financial table-and-text QA; gold test set	277	1,660
Total		**13,583**	**11,198**

Table 2: RAG components evaluated in the factorial experiment. The pipeline grid contains $4 \times 13 \times 4 = 208$ configurations, each evaluated at five context capacities.

Component	Group	Variants	Levels
Chunking		Fixed-256, Fixed-512, Sentence-512, Semantic-512	4
Retrieval	Sparse	BM25	13
	Dense	EmbeddingGemma-300M, Qwen3-Embedding-0.6B, GTE-ModernBERT, LFM2.5-Embedding-350M	
	Late interaction	GTE-ModernColBERT, LFM2.5-ColBERT-350M	
	Fusion	BM25–RRF with each of the six neural retrievers	
Reranking		None, GTE-Reranker-ModernBERT, Qwen3-Reranker-0.6B, Llama-Nemotron-Rerank-1B-v2	4
Context capacity		256, 512, 1,024, 2,048, 4,096 visible content tokens	5

stricted pool would allow. Context is normalized by tokens rather than by k. Ranked chunks are added until the capacity is reached, and the remaining budget is filled with an exact prefix of the boundary chunk. This prevents differences in chunk length from changing how much text the evaluator sees.

We group first-stage retrieval into three architecture families—sparse, dense, and late interaction—and treat BM25–RRF fusion separately, since fusion is an intervention applied to a neural retriever rather than a distinct architecture. Section 4.3 analyses it as a paired contrast against each matched neural parent; averaging over six parents would conflate architecture with fusion, so it is excluded from the chunker $\times$ family analysis.

3.3 Claim-level outcomes and paired substitutions

We adapt RAGChecker's atomic-claim evaluation to retrieval only. In **Stage 1**, reference answers and official supports are used once to construct 38,685 question-level reference claims. In **Stage 2**, a pointwise judge receives only the question, reference claims, and one retrieved fragment, returning one Boolean support decision per claim. It does not observe the dataset, pipeline identity, rank, score, capacity, reference answer, or official evidence. We generate no answers and compute no generator-quality metrics. Across the grid, Stage 2 comprises 5,441,601 fragment-evaluation requests and 18,679,678 claim-level support decisions.

The judge is `google/gemma-4-31B-it`, with reasoning disabled, temperature 0, and a fixed decoding seed. Prompts and schemas are given in Appendix A.

For dataset d and question q, let $G_{d,q}$ be the set of reference claims. A pipeline configuration $x = (c, r, z)$ consists of a chunker c, a first-stage retriever r, and a reranker z. Let $V_{d,q}(x, b)$ be the set of fragments visible under x at a context capacity of b tokens: the complete chunks that fit, plus the prefix of the boundary chunk. Writing $f \models g$ when fragment f supports claim g, our primary outcome is question-level claim recall:

$$Y_{d,q}(x,b) = \frac{1}{|G_{d,q}|} \sum_{g \in G_{d,q}} \mathbf{1}[\exists f \in V_{d,q}(x,b) : f \models g], \tag{1}$$

where $\mathbf{1}[\cdot]$ is 1 if the condition holds and 0 otherwise. A claim is therefore covered when at least one visible fragment supports it.

We do not report fragment precision: the number of visible fragments varies with chunk size at fixed capacity, so precision is not comparable across the chunker substitutions we estimate.

We estimate component effects using paired substitutions. For example, the effect of replacing chunker c with c' is

$$\Delta^{c' \leftarrow c}_{d,q,r,z,b} = Y_{d,q}(c', r, z, b) - Y_{d,q}(c, r, z, b). \quad (2)$$

The question, dataset, retriever, reranker, and capacity are identical on both sides; only the chunker changes. Retrieval, fusion, and reranking effects are defined analogously, and interactions are differences between paired effects: whether the same substitution behaves differently under sparse, dense, or late-interaction retrieval. Every effect is reported against a fixed baseline—Fixed-512, BM25, the matched neural parent, and no reranker—so a positive value always means the named condition delivers more claim coverage.

Questions are weighted equally within each dataset, the five datasets equally in pooled estimates, and capacities and unchanged component levels evenly across the grid. Confidence intervals use 2,000 deterministic paired bootstrap replicates with seed 42, resampling questions within datasets and hierarchically for pooled estimates. Our causal interpretation is local to substitutions within this frozen grid; it implies neither universal component effects nor answer-level correctness.

A blinded human audit of 100 judge-positive and 100 judge-negative decisions confirmed all negatives but only 68 of the positives: the judge rarely misses support but grants it too readily (Appendix A.4).

Complete model and retrieval configurations, including representations, indexing settings, and inference parameters, are provided in Appendix B.

4 Results

The factorial analysis reveals a clear asymmetry between upstream and downstream interventions. Chunking and fusion effects vary across datasets and retrieval regimes, whereas learned reranking remains beneficial after nearly every upstream configuration.

4.1 Chunk Geometry Does Not Determine the Best Chunker

Dataset manifests reveal substantially different evidence geometry. In QMSum, only 4% of official evidence spans are fully contained within a Fixed-256 chunk, compared with 17% for Fixed-512 and 19% for Sentence-512. A reference span consequently touches 5.57 Fixed-256 chunks on average, compared with 3.27 Fixed-512 chunks. Evidence in MultiHop-RAG and TAT-QA, by contrast, is usually contained within a single chunk.

These intrinsic differences do not produce a consistent chunker ranking. Figure 1 reports paired changes in claim recall for each chunker substitution relative to Fixed-512. The direction and magnitude of these effects vary across datasets, and no alternative dominates universally. QMSum makes the disconnect concrete: it has the worst Fixed-256 containment of any dataset, yet Fixed-256 improves claim recall there under dense retrieval (Section 4.2). Geometry describes the representation given to retrieval; whether it helps depends on the retriever that consumes it.

4.2 Chunking and Retrieval Are Complementary

The effect of a chunking choice depends on the retriever that consumes its output. Figure 2 compares each chunker with Fixed-512 under sparse, dense, and late-interaction retrieval, with reranking disabled. Effects are averaged across the five capacities and equally across models within each family.

On FinanceBench, Semantic-512 changes claim recall by $+0.27$ percentage points under sparse retrieval, -2.59 under dense, and $+2.65$ under late interaction, a late-interaction-minus-dense interaction of $+5.24$ points $[2.37, 8.23]$.

The pattern also holds within matched backbones, so it does not come from averaging four dense models against two late-interaction ones: the effect rises from -3.1 to $+1.4$ for GTE and from $+1.6$ to $+3.9$ for LFM2.5 when moving from dense to late interaction (Figure 7).

Reversals occur elsewhere too. On LegalBench-RAG, Fixed-256 changes recall by -2.17 under sparse but $+0.75$ under dense, an interaction of $+2.92$ $[2.50, 3.30]$; on QMSum the same substitution gives -1.63 and $+2.01$, an interaction of $+3.64$ $[1.47, 5.86]$.

MultiHop-RAG bounds the claim: Fixed-256 helps under sparse ($+5.34$), dense ($+4.34$), and late interaction ($+4.64$). Its evidence is short and almost always sits inside one chunk, so smaller chunks concentrate it rather than splitting it, and every family benefits. Component effects are conditional, not arbitrary: where evidence geometry is uniform across a corpus, a chunker choice can transport.

Overall, chunker effects change direction across

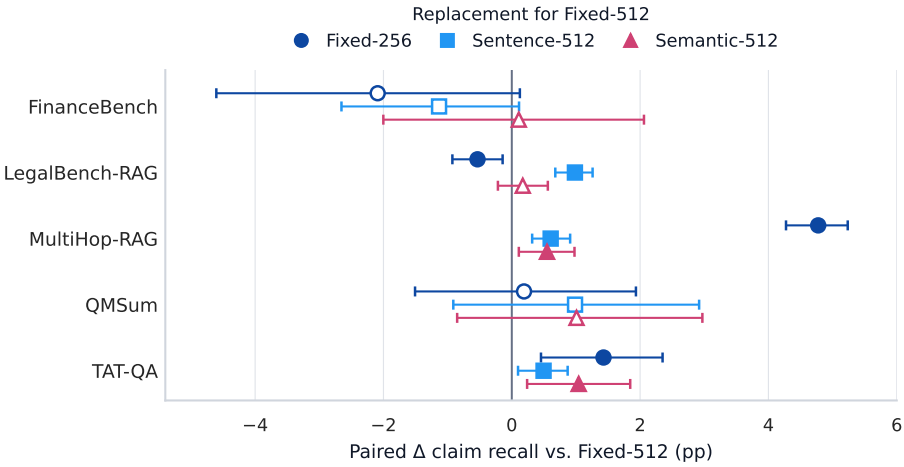


Figure 1: **The effect of replacing Fixed-512 varies across datasets.** Points show paired changes in claim recall, averaged equally across the sparse, dense, and late-interaction retrieval families and across context capacities, with reranking disabled. BM25–RRF fusion conditions are excluded (Section 3.2). Error bars denote 95% paired-bootstrap confidence intervals; filled markers indicate intervals that exclude zero.

retriever families in four of the five datasets. This means that a chunker should not be judged on its own: whether it helps depends on the retriever used with it.

4.3 BM25 Fusion Is Conditional on the Upstream Pipeline

Figure 3 compares each BM25–RRF condition with its exact neural parent while holding the dataset, chunker, context capacity, and no-reranker condition fixed. Averaged over the six matched neural parents and four chunkers, fusion exhibits a dataset-level sign reversal: claim recall decreases by 5.92 percentage points on FinanceBench and 2.85 points on LegalBench-RAG, but increases by 4.53 points on MultiHop-RAG, 2.46 points on QMSum, and 6.33 points on TAT-QA.

The reversal is not driven by one parent model. On FinanceBench and LegalBench-RAG, none of the six neural retrievers improves after fusion under any chunker; on MultiHop-RAG, all six improve under every chunker, with five or six on QMSum and four or five on TAT-QA. The same matched intervention reliably harms two datasets and helps three. Magnitude is less even: on TAT-QA, fusion with LFM Dense gains 18–21 points, several times any other parent, so that dataset's +6.33 mean rests largely on one model pair (Appendix C).

Averaging these opposing effects produces a pooled gain of only 0.91 points, with a 95% confidence interval spanning zero $[-3.09, 4.82]$. This pooled estimate conceals a qualitative interaction: BM25 fusion changes not only in magnitude but also in direction across datasets. Fusion should therefore be treated as a domain-conditioned intervention rather than a generally beneficial retrieval enhancement.

4.4 Net Effects Conceal Claim-Level Transitions

4.5 Reranking Is Robust to Upstream Pipeline Choices

Unlike chunking and fusion, reranking produces positive effects in nearly every tested configuration. Figure 4 compares each learned reranker with no reranker, holding the dataset, question, chunker, first-stage retriever, and capacity fixed. Averaged equally across datasets and retrieval families, GTE-Reranker-ModernBERT improves claim recall by +5.74 percentage points $[4.10, 7.87]$, Qwen3-Reranker-0.6B by +6.50 $[4.73, 7.68]$, and Llama-Nemotron-Rerank-1B-v2 by +8.76 $[6.83, 10.51]$.

The consistency holds below the pooled average. All 240 dataset × chunker × retrieval-family × reranker comparisons are positive (here fusion counts as a fourth upstream group, since reranking is applied after it), as are 776 of 780 cells at the level of individual retrievers; the four exceptions are small and localized. The Nemotron reranker is positive on every dataset, from +5.28 on QMSum to +11.05 on FinanceBench, and varies with capacity +7.46 at 256 tokens, +11.30

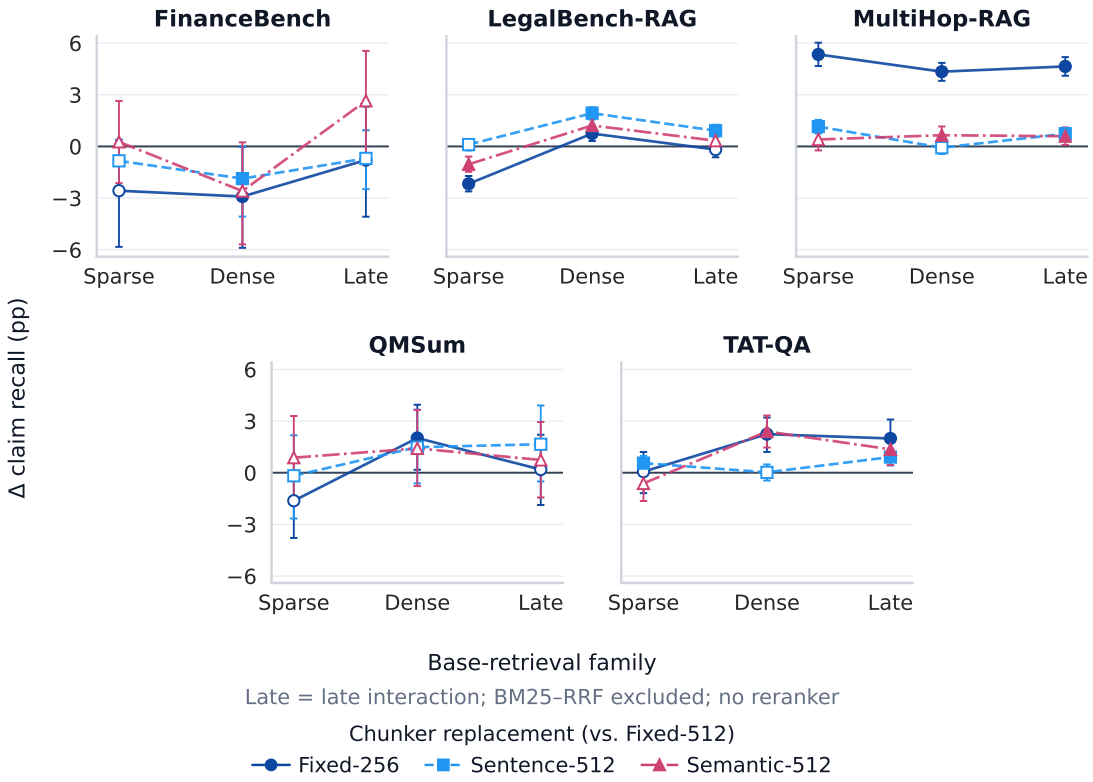


Figure 2: **Chunker effects depend on retrieval family and dataset.** Each panel shows one dataset. Lines represent chunker replacements relative to Fixed-512, while the horizontal axis varies the retrieval family. Points show paired changes in claim recall, averaged across context capacities and equally across models within each family, with reranking disabled. Error bars denote 95% paired-bootstrap confidence intervals; filled markers indicate intervals excluding zero.

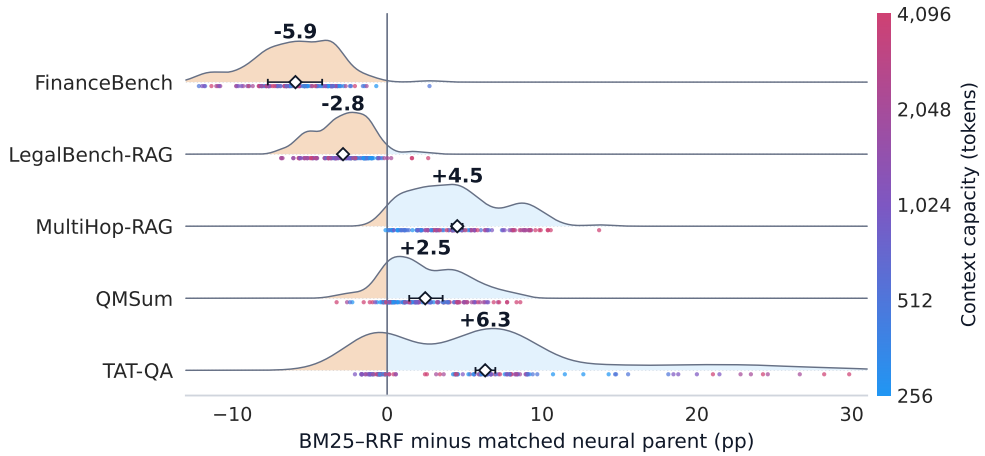


Figure 3: **BM25 fusion has dataset- and pipeline-dependent effects.** Each rug mark represents one fixed chunker, neural parent, and context capacity. Ridges summarize paired changes in claim recall across the tested conditions. Diamonds and lines show dataset-level mean effects and paired-bootstrap 95% confidence intervals.

at 1,024, +5.15 at 4,096—without reversing. The fixed pool explains the shape: at 256 tokens few chunks fit whatever their order, at 4,096 much of the top 20 is visible whatever the ranking, so reordering matters most in between.

Reranking differs structurally from the preceding interventions: it reorders a fixed pool rather than changing the chunk representation or adding a retrieval source. Under a constrained budget it can lift evidence above the visibility threshold without

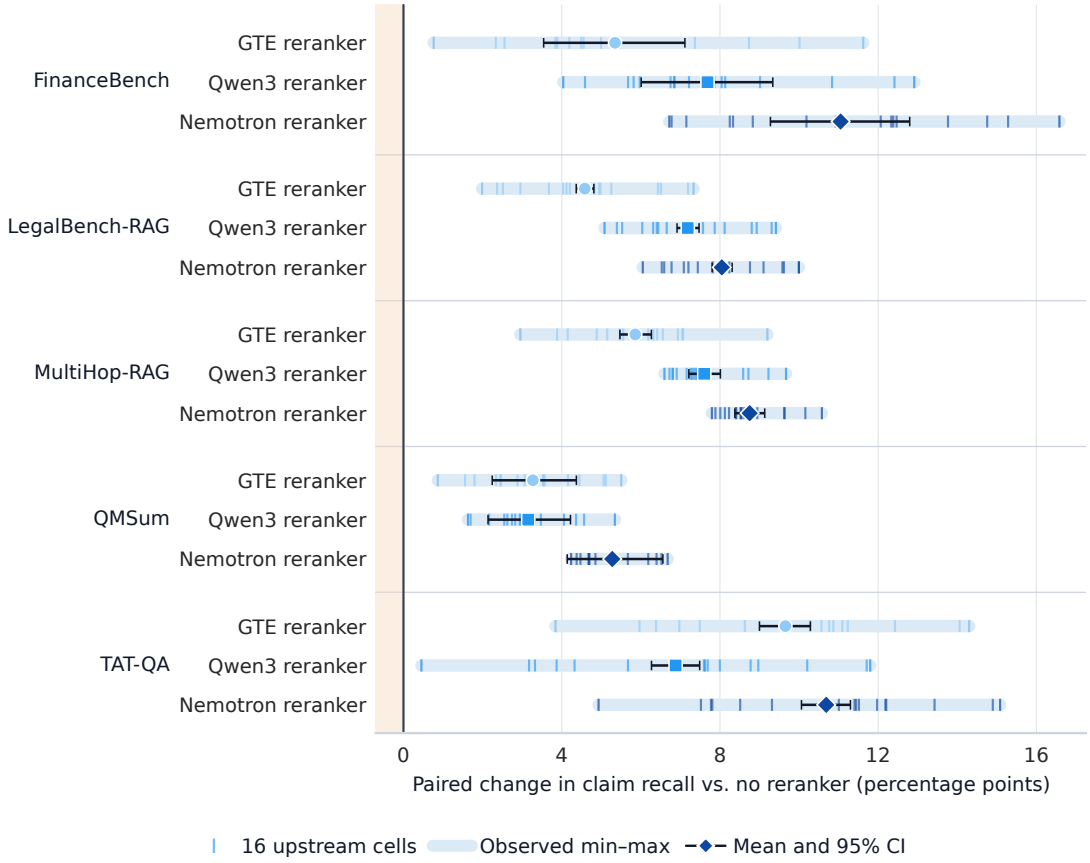


Figure 4: **Learned reranking improves claim recall across upstream configurations and datasets.** Each row compares one reranker with no reranker. Pale segments show the observed minimum–maximum effect across 16 upstream configurations (four chunkers $\times$ four upstream groups: sparse, dense, late interaction, and fusion), and small ticks mark the individual configurations. Colored markers show dataset-level paired means; black whiskers denote 95% paired-bootstrap confidence intervals.

depending on a particular chunker–retriever pairing. Within our frozen grid it is therefore more stable than chunking or fusion—a conclusion local to the evaluated pools, rerankers, datasets, and capacities, not a claim that every reranker helps. Claim recall also ignores order within the visible context: a claim counts as covered wherever it appears. Reranking changes where evidence sits, and generators are sensitive to position (Liu et al., 2024), so a gain in delivered evidence need not become an equal gain in answer quality.

Aggregate recall changes do not reveal which claims change coverage state. We therefore decompose each paired intervention into two mutually exclusive transitions: claims that become covered under the intervention and claims that cease to be covered. Their difference recovers the net recall effect:

$$\Delta\text{Recall} = P(0\to1) - P(1\to0). \qquad (3)$$

Figure 5a reports chunker transitions under the complete-grid estimand, averaging across rerankers as well as retrievers and capacities; the chunker effects in Section 4.2 instead condition on the no-reranker setting, so the two are not numerically identical. Figure 5b conditions reranker transitions on the upstream retrieval family, since a reranker is defined only relative to a first-stage ranking. All estimates weight capacities and datasets equally.

Replacing Fixed-512 with Fixed-256 illustrates why a near-zero average effect need not imply that the intervention is inconsequential. Fixed-256 newly covers 7.95% of claims but no longer covers 8.12%, a net change of only -0.16 percentage points (95% CI $[-3.10, 2.64]$; computed from unrounded shares). Thus, approximately 16.1% of claims change coverage state even though aggregate recall changes little. The substitution changes which evidence is visible rather than uniformly improving or degrading coverage.

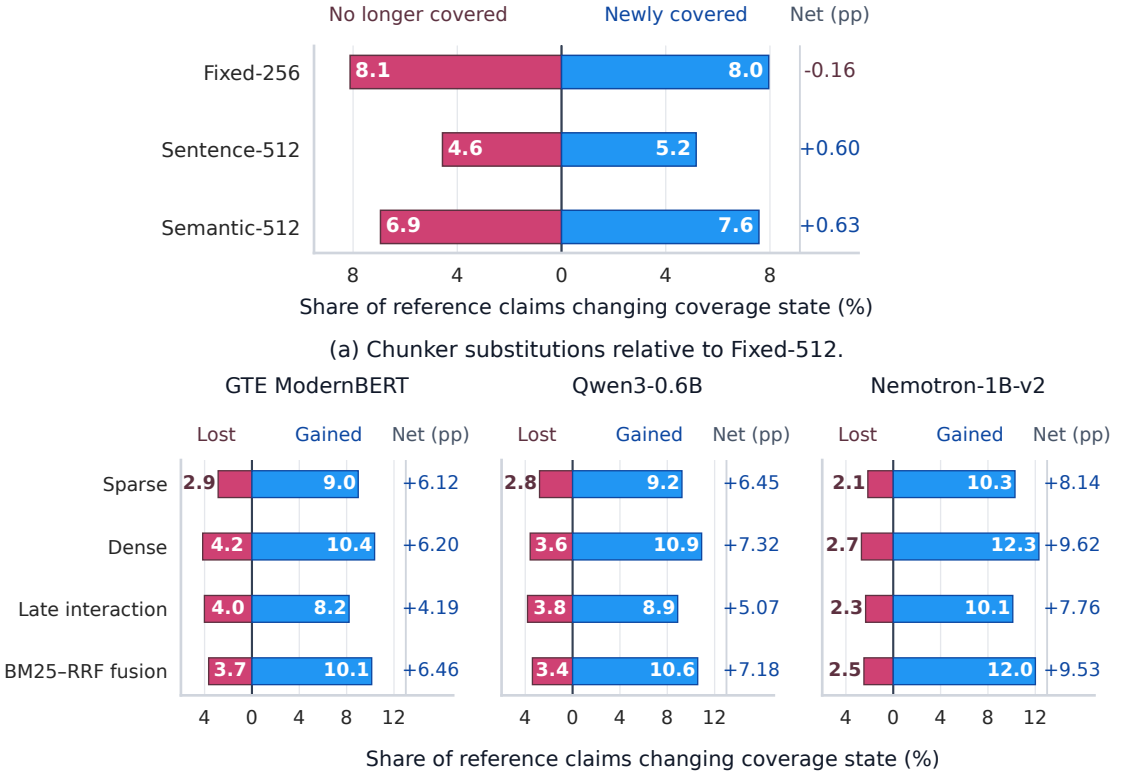


Figure 5: **Component substitutions change which reference claims are covered.** Blue bars show newly covered claims and pink bars show claims no longer covered. Net changes are reported in percentage points. Chunker effects are averaged across datasets, capacities, retrievers, and rerankers. Reranker effects are averaged across datasets, capacities, chunkers, and the models within each retrieval family.

Reranking produces a qualitatively different pattern. Across every reranker and retrieval family, newly covered claims outnumber claims no longer covered. The Nemotron reranker yields the largest net improvement in all four families: $+8.14$ points with sparse retrieval, $+9.62$ with dense retrieval, $+7.76$ with late interaction, and $+9.53$ with BM25–RRF fusion. Its newly covered share ranges from 10.10% to 12.30%, whereas only 2.15%–2.69% of claims cease to be covered. The GTE and Qwen3 rerankers are also repair-dominant, although their gains are consistently smaller. For all three rerankers, the smallest net improvement occurs after late-interaction retrieval, suggesting that stronger first-stage retrieval leaves less uncovered evidence for reranking to recover.

These results show that net recall can conceal materially different effect compositions. Chunking may exchange one set of visible claims for another, whereas reranking predominantly adds coverage in this grid. Transition analysis describes how coverage changes under a controlled substitution; it does not by itself identify the mechanism producing that change.

5 Discussion

5.1 Component Quality Is Relational

The pattern in our results is not uniform variance. Chunking and fusion change which candidates exist; reranking only reorders a fixed pool. The interventions that change the representation are the ones whose sign depends on what consumes it, while the one that leaves the candidate set alone transports across upstream regimes. A component's transportability tracks how much of the pipeline it changes. This qualifies our third finding: reranking may be more stable because it is a smaller intervention, not because reordering is inherently more reliable than representation.

5.2 Implications for Evaluation

Three evaluation practices follow for studies of evidence delivery. Report chunker results together with the retriever they were measured against, since a chunker effect without its retrieval family cannot be interpreted. Compare fusion against its matched neural parent on each target domain; our pooled estimate of $+0.91$ points describes none of the five datasets. And report sign-consistency counts alongside pooled means, which otherwise cannot distinguish a small effect from opposing large ones.

These practices concern how evidence delivery is measured. Whether the same component choices improve generated answers is a separate question our design does not test. What our results do show is that a leaderboard cannot say which component to change next: the answer depends on the dataset and on what sits either side of the component.

6 Conclusion

We presented a controlled factorial evaluation of RAG evidence delivery across chunking, retrieval, fusion, reranking, and context capacity. Rather than ranking complete pipelines, we estimated paired component substitutions while holding the remaining pipeline fixed and controlling visible context by token capacity.

The results show that component effects are conditional. The effect of a chunker depends on the retriever that consumes its output, and BM25–RRF fusion helps on some datasets while harming others. Reranking improves claim recall in nearly every configuration we test, with smaller gains after late-interaction retrieval. Claim-transition analysis further shows that similar net effects can hide very different changes: chunking often exchanges which claims are covered, whereas reranking mostly adds coverage.

For evidence delivery, these findings argue against treating RAG components as independently rankable modules. Evaluation should combine equal-context comparison, paired substitutions, interaction analysis, and claim-level diagnostics, so that it shows not only which pipeline performs best but which components drive the difference and under what conditions.

Limitations

Our conclusions are local to the five datasets, component implementations, candidate pools, context capacities, and retrieval evaluator used in this study. The factorial design identifies controlled effects within this frozen grid, but it does not establish that the same effects will hold for other domains, models, indexing strategies, or context windows. Candidate pools are capped at the first-stage top 20 (Section 3.2), so we cannot measure the ceiling an unrestricted pool would permit.

We evaluate evidence delivery rather than generated answers. Claim recall indicates whether retrieved fragments contain reference claims, but it does not measure answer correctness, faithfulness, fluency, or downstream reasoning. Delivering a claim is necessary for grounding an answer in it, not sufficient: generators differ in how well they use the evidence they receive. Testing whether the component effects reported here carry over to answer quality, using several generators on a subset of the grid, is the most direct next step.

The same frozen model constructs the reference claims and judges support, so the judge may more readily confirm claims phrased the way it would phrase them. A blinded audit confirmed 68 of 100 judge-positive and all 100 judge-negative decisions (Appendix A.4), so the judge is permissive and absolute claim recall is overstated. Paired effects are affected only if the false-positive rate differs between compared conditions—for example, if reranked contexts expose the judge to more fragments that are on-topic but do not support the claim. A larger audit stratified by chunker, reranker, and claim type would bound this more tightly.

Support is judged one fragment at a time, which interacts with chunk size in two opposite ways. Evidence split across fragments cannot be combined, so a claim whose support crosses a chunk boundary is undercounted, as can occur in MultiHop-RAG and TAT-QA; this penalizes finer chunking. At the same time, smaller chunks yield more fragments within a fixed capacity, each another chance for a false positive; this favors finer chunking. Because the number of fragments depends only on the chunker and capacity, this effect is the same on both sides of every reranking and fusion contrast and cancels in the chunker $\times$ retrieval-family interactions. It can bias chunker main effects, such as Fixed-256 against Fixed-512, in a direction the audit cannot determine.

Retrieval implementations are not symmetric. Late-interaction candidates come from approximate FastPLAID search with 4-bit residual compression, whereas dense retrieval is exact and exhaustive, so part of any dense versus late-

interaction difference may reflect implementation rather than architecture. The families are also unbalanced: four dense models against two late-interaction models. Matched GTE and LFM2.5 backbones show the same direction (Section 4.2), which addresses model composition but not search approximation; repeating the comparison with exact MaxSim retrieval would separate the two.

Finally, equal-token capacity controls context exposure but does not capture all deployment costs. Latency, index size, retrieval throughput, reranking cost, and generator sensitivity to evidence ordering remain outside the present analysis.

References

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics.

Ng ST Chong. 2026. Three sides of retrieval: Factorial evidence for document-side, query-side, and answer-side complementarity in RAG. *arXiv preprint arXiv:2607.24781*.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Knowledge-centric hallucination detection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6953–6975, Miami, Florida, USA. Association for Computational Linguistics.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. FinanceBench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 881–893, Miami, Florida, US. Association for Computational Linguistics.

Zachary C Lipton and Jacob Steinhardt. 2019. Research for practice: troubling trends in machine-learning scholarship. *Communications of the ACM*, 62(6):45–53.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Fabian Lukassen, Christoph Weisser, Thomas Kneib, and Alexander Silbersdorff. 2026. CAFE: A Compound-AI factorial evaluation framework. *arXiv preprint arXiv:2607.10380*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Nicholas Pipitone and Ghita Houir Alami. 2024. LegalBench-RAG: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Jiayang Cheng, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, volume 37, pages 21999–22027.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Ellen M. Voorhees, Daniel V. Samarov, and Ian Soboroff. 2017. Using replicates in information retrieval evaluation. *ACM Transactions on Information Systems*, 36(2):12:1–12:21.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages

5905–5921, Online. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Evaluation Prompts, Schemas, and Validation

This appendix reports the complete instructions used for reference-claim construction and pointwise claim-support evaluation. Gemma 4 31B-IT was used as reported in Section 3.3. Thinking was disabled, temperature was 0, top-p was 1, and the random seed was 42. Structured outputs were generated with strict JSON schemas using the `guidance` backend. The static prompt text below is reproduced verbatim; expressions enclosed in angle brackets denote values inserted separately for each question or retrieved fragment.

A.1 Reference-Claim Construction

The first evaluation stage converted each benchmark reference answer into a set of standalone reference claims. Official benchmark evidence was supplied to ground the claims and retain provenance.

Stage 1: Claim-construction system message

```
You create reference claims for evaluating retrieved
evidence. The user provides a question, reference
answer, claim-coverage requirement, optional answer
annotations, and labelled official evidence. Create a
complete, non-redundant set of standalone atomic
claims that follows the claim-coverage requirement.

Coverage comes before minimization. Include every
material assertion, explanation, qualifier,
comparison, and calculation required by the specified
coverage. Remove a claim only when it repeats the same
proposition and removing it does not reduce the
required coverage.

Each claim must state one independently verifiable
proposition and be understandable without the
question, answer, other claims, or evidence. Resolve
pronouns and preserve every necessary entity,
relationship, date, unit, scope, qualifier,
definition, and comparison direction. Keep an
exception or limitation in the same claim as the
proposition it qualifies; never separate their
meanings.

For a prose answer, decompose every material assertion
and explanation. For a short entity or scalar answer,
combine the question and answer into standalone
propositions; never write "the answer is X." For a
calculated answer, create claims for the required
operand facts and one final calculation claim
containing the operation, operands, result, and unit
or scale. If that result directly establishes a yes/no
or comparison conclusion, include it in that
calculation claim instead of repeating the result.

Use official evidence as grounding and provenance for
claims needed to establish the reference answer. Do
not create a claim merely to cite every supplied item:
evidence may contain headings, bridge text, redundant
context, adjacent rows, or distractors. Leave an item
uncited when it adds no material proposition required
by the answer. Ignore unrelated details, rebuttals, or
counterarguments in the same item. In answer-first
mode, evidence does not introduce extra claims.

Every claim must cite all and only the official
evidence items that support it. evidence_ids may
contain only supplied labels such as E001. Evidence
IDs represent provenance, not dependencies between
claims. Never put claim IDs in evidence_ids, evidence
labels in claim text, or use outside knowledge. Before
returning, verify completeness, atomicity, grounding,
arithmetic, units, and non-redundancy. Remove any
claim whose deletion would not reduce required answer
or evidence coverage. Return only the required
structure.
```

The corresponding user message was rendered from the following template.

Stage 1: User-message template

```
QUESTION:
<question>

REFERENCE ANSWER:
<reference answer>

<optional reference-answer annotations>

CLAIM COVERAGE REQUIREMENT:
<coverage requirement>

ALLOWED OFFICIAL EVIDENCE IDS:
<E001, E002, ...>

OFFICIAL EVIDENCE:

[E001]
<official evidence text>

[E002]
<official evidence text>

...
```

When benchmark annotations were available, they were inserted using the following fields. Only available fields were included.

Optional reference-answer annotations

```
REFERENCE ANSWER ANNOTATIONS:
Answer type: <answer type>
Answer unit or scale: <scale>
Official answer rationale:
<rationale>
Official derivation: <derivation>
Official facts:
- <fact>
...
```

Coverage policies. FinanceBench, LegalBench-RAG, and QMSum used the following answer-first coverage requirement.

Answer-first coverage requirement

```
Represent every material assertion and explanation in
the reference answer. Use official evidence and answer
annotations to ground and validate those claims. Do
not create claims from redundant, structural,
incidental, or unrelated text.
```

MultiHop-RAG and TAT-QA used the following evidence-fallback requirement. This policy was used when short reference answers did not themselves express all supporting operands or evidence requirements.

Evidence-fallback coverage requirement

```
The reference answer alone does not contain the
complete evidence requirements. Use the question,
answer annotations, and official evidence to construct
them. Represent every nonredundant evidence fact
required to establish the answer.
```

A.1.1 Claim Output Schema

Claim responses were generated under a strict JSON schema. At request time, the `enum` under `evidence_ids` was replaced by the exact set of official evidence labels supplied for the question.

Stage 1: Claim-construction schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["claims"],
  "properties": {
    "claims": {
      "type": "array",
      "minItems": 1,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": [
          "claim_id",
          "text",
          "evidence_ids"
        ],
        "properties": {
          "claim_id": {
            "type": "string"
          },
          "text": {
            "type": "string"
          },
          "evidence_ids": {
            "type": "array",
            "minItems": 1,
            "items": {
              "type": "string",
              "enum": [
                "E001",
                "E002",
                "..."
              ]
            }
          }
        }
      }
    }
  }
}
```

A.2 Pointwise Claim-Support Evaluation

The second stage evaluated whether each retrieved fragment supported each reference claim. Every request contained one retrieved fragment and all reference claims for the corresponding question. Decisions were therefore pointwise with respect to a single fragment: evidence from different fragments could not be combined.

Stage 2: Pointwise-support system message

```
Decide whether the single retrieved chunk supports
each reference claim. A claim is supported only when
this chunk by itself entails the complete claim. Do
not use outside knowledge, the question alone, or
information from another chunk. A partial match is not
enough.
For a calculated quantitative claim, mark it supported
when the chunk contains every required input with the
labels and units needed to interpret it, and the
stated calculation follows correctly from those
inputs; the derived result does not need to appear
verbatim in the chunk. When the claim explicitly
states a formula or operational definition, evaluate
that stated formula or definition; do not replace it
with a different conventional definition. Return
exactly one decision for every reference claim in the
required structure.
```

The corresponding user message was rendered as follows.

Stage 2: User-message template

```
QUESTION:
<question>

REFERENCE CLAIMS:
[CL001] <claim text>
[CL002] <claim text>
...

RETRIEVED CHUNK:
[C001] <retrieved fragment>
```

A `supported=true` decision therefore means that the single displayed fragment entails the complete target claim by itself. A fragment that supports another claim, or only part of the target claim, receives `supported=false` for that target claim.

A.2.1 Support Output Schema

For a request containing N reference claims, both `minItems` and `maxItems` were set to N. The `claim_id` enumeration was also replaced by the exact claim identifiers supplied in that request.

Stage 2: Pointwise-support schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["claim_results"],
  "properties": {
    "claim_results": {
      "type": "array",
      "minItems": <N>,
      "maxItems": <N>,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": [
          "claim_id",
          "supported"
        ],
        "properties": {
          "claim_id": {
            "type": "string",
            "enum": [
              "CL001",
              "CL002",
              "..."
            ]
          },
          "supported": {
            "type": "boolean"
          }
        }
      }
    }
  }
}
```

A.3 Validation and Retry Policy

A.3.1 Claim Validation

Claim responses were accepted only when they completed before the 8,192-token output ceiling, decoded as a JSON object containing at least one claim, and passed the following protocol checks:

- every claim was represented by a JSON object;
- claim text was non-empty and not duplicated within the response;
- evidence labels such as `[E001]` did not appear inside claim text;
- every claim cited at least one official evidence ID;
- evidence IDs were strings and were unique within each claim; and
- every cited evidence ID occurred in the allowed set supplied with the question.

After validation, claim identifiers were assigned canonically as `CL001`, `CL002`, ... in output order. The original model-generated identifier did not determine the final claim identifier.

The validator did not require every supplied official evidence item to be cited. Official evidence could contain headings, bridge text, adjacent rows, redundant context, or distractors. Uncited supplied evidence IDs were retained in an audit field but did not invalidate an otherwise valid claim set.

These checks validate response structure and evidence provenance; they do not independently establish semantic entailment between a claim and its cited official evidence. We deliberately used no lexical-overlap, number-matching, OCR-normalization, or arithmetic heuristics to accept or reject claims. Semantic grounding, arithmetic, units, completeness, and atomicity were governed by the frozen prompt and inspected through manual auditing.

A.3.2 Claim Retries

A failed initial claim response received at most three retries, producing a maximum of four attempts. Every retry reused the complete original request and appended the following instruction.

Stage 1: Retry instruction

```
Recreate the complete claim set from the original
input and apply every system rule.
Ensure the claims jointly establish the reference
answer, every claim is necessary and grounded,
calculations and units are exact, and evidence_ids
contain only supplied official evidence labels—not
claim IDs. Return the complete corrected structure.
```

For every attempt, we retained the exact request body, raw response, proposed claim objects, failure code, evidence IDs, token usage, finish reason, and latency. Failed-attempt claims were never passed to the support-evaluation stage.

Following audited retry exhaustion, two responses remained invalid solely because they repeated byte-identical claim text. These duplicates were losslessly consolidated: one copy of each distinct text was retained and its evidence-ID list was replaced by the union of the duplicated copies. No claim text was rewritten and no evidence ID was invented. This removed three duplicate claim objects across two questions.

A.3.3 Support Validation

Support responses were accepted only when they:

- completed before the 2,048-token output ceiling;
- decoded as a valid JSON object;

- returned a Boolean decision for every supplied claim ID;
- returned each supplied claim ID exactly once; and
- contained no unknown or duplicated claim IDs.

A failed support response received one retry, for a maximum of two attempts. The retry reused the original request and appended the following instruction.

Stage 2: Retry instruction

```
Check the result again. Return exactly one decision
for every supplied claim. Mark supported as true only
when the single retrieved chunk entails the complete
claim by itself.
```

No support request required a retry in the final production run.

A.4 Human Audit of Support Decisions

The Stage 2 judge marked 539,201 of 18,679,678 claim–fragment decisions as supported (2.9%). We audited a blinded sample of 100 judge-positive and 100 judge-negative decisions. On this balanced sample, overall agreement was 84.0% ($\kappa = 0.68$): the annotator confirmed 68 of 100 positive decisions (95% Wilson interval 58.3–76.3%) and all 100 negative decisions (96.3–100.0%). Reweighting the same cases to the production frequencies of dataset, chunker, and judge label gives 68.6% agreement on positives, 100.0% on negatives, and 99.1% overall ($\kappa = 0.81$); the intervals above describe the sampled cases, not the reweighted rates. The judge therefore rarely misses support but grants it too readily, so absolute claim recall is overstated. Because this bias is present on both sides of every paired substitution, it affects effect estimates only where the false-positive rate differs between the compared conditions. Table 3 reports the breakdown by dataset.

A.5 Inference Parameters

Claim construction used a 22,000-token model context, an 8,192-token output ceiling, tensor parallelism of two, and at most three retries after the initial attempt. Pointwise support evaluation used a 16,384-token model context, a 2,048-token output ceiling, and at most one retry. Both stages used BF16 inference, FlashAttention 4, eager execution, strict guided JSON decoding, disabled thinking, temperature 0, top-$p = 1$, and seed 42.

Scope	Judge+	A_+	A_-	Overall	κ
All	2.9	69	100	99.1	0.81
FinanceBench	2.0	63	100	99.2	0.77
LegalBench-RAG	3.0	66	100	99.0	0.79
MultiHop-RAG	3.2	71	100	99.1	0.82
QMSum	1.5	55	100	99.3	0.70
TAT-QA	2.1	85	100	99.7	0.91

Table 3: Human audit of Stage 2 support decisions ($n = 200$; 40 per dataset). Judge+ is the production share of positive decisions (%). A_+ and A_- are agreement on judge-positive and judge-negative decisions, reweighted to production frequencies; Overall and κ are prevalence-weighted. Per-dataset rows rest on about 20 audited decisions per label and are indicative only.

B Model and Retrieval Configuration

Neural retrieval. Table 4 summarizes the six neural retrievers. Prefixes are concatenated directly with the input. Dense embeddings are L2-normalized and searched using exhaustive float32 cosine similarity with TF32 disabled. Late-interaction models use L2-normalized token embeddings and MaxSim.

Semantic chunking. Semantic-512 uses `jina-embeddings-v5-text-nano-text-matching`. Three-sentence windows are encoded with the `Document:` prefix, an 8,192-token model limit, 768-dimensional normalized embeddings, BF16, and FlashAttention 4. Gradient-based boundaries use a sentence buffer of one. The resulting source-exact chunks have zero overlap and are hard-capped at 512 `o200k_base` tokens.

BM25 and fusion. BM25 uses BM25S with Lucene term weighting and IDF, $k_1 = 1.2$, and $b = 0.75$. Text is normalized with NFKC, Unicode case-folded, and split into Unicode alphanumeric runs. We apply neither stemming nor stopword removal. Each hybrid condition combines the saved BM25 and neural top-20 rankings using equal-weight reciprocal-rank fusion:

$$s_{\mathrm{RRF}}(u) = \sum_{m \in \{\mathrm{BM25},\, \mathrm{neural}\}} \frac{1}{60 + \mathrm{rank}_m(u)}.$$

Late-interaction retrieval. Both ColBERT models use FastPLAID with 4-bit residuals, four k-means iterations, `seed=42`, `n_ivf_probe=8`, `n_full_scores=8192`, and no token pooling. Dense retrieval performs an exhaustive corpus scan, whereas FastPLAID candidate generation is approximate; we discuss this implementation asymmetry as a limitation.

Table 4: Neural retriever models and representations.

Model	Representation
EmbeddingGemma-300M	Native pooling; 768-dimensional L2-normalized embedding
Qwen3-Embedding-0.6B	Last-token pooling; 1,024-dimensional L2-normalized embedding
GTE-ModernBERT	CLS pooling; 768-dimensional L2-normalized embedding
LFM2.5-Embedding-350M	CLS pooling; 1,024-dimensional L2-normalized embedding
GTE-ModernColBERT	128-dimensional L2-normalized token embeddings with MaxSim
LFM2.5-ColBERT-350M	128-dimensional L2-normalized token embeddings with MaxSim

Reranking We use the GTE, Qwen3, and Nemotron rerankers (Table 2). Each reranker scores only the raw question and a retrieved chunk and preserves the first-stage top-20 candidate set. GTE uses the tokenizer's question–document pair format with an 8,192-token limit. Nemotron uses `question:{query} \n \n passage:{document}` with an 8,192-token limit. Qwen uses its retrieval instruction and scores $\mathrm{logit}(\mathrm{yes}) - \mathrm{logit}(\mathrm{no})$ with a 32,768-token limit. Over-length inputs remove document tokens before question tokens.

C Complete Per-Configuration and Per-Model Results

This appendix reports the complete model-level results underlying the aggregated analyses in Section 4. All displayed values use the arithmetic mean of question-level claim recall across the five context capacities (256–4,096 tokens). Figure 6 is descriptive; Figures 7–11 report paired component substitutions.

Reranker	Dataset	Chunker	Retriever	Δ (pp)
GTE	FinanceBench	Fixed-256	EmbeddingGemma	-0.40
GTE	FinanceBench	Fixed-256	Qwen3 Emb.	-2.00
GTE	FinanceBench	Sentence-512	Qwen3 Emb.	-1.20
Qwen3	TAT-QA	Fixed-256	GTE ColBERT	-0.02

Table 5: The four negative reranker effects among the 780 cells. Δ is the paired change in claim recall relative to no reranker, averaged across the five capacities.

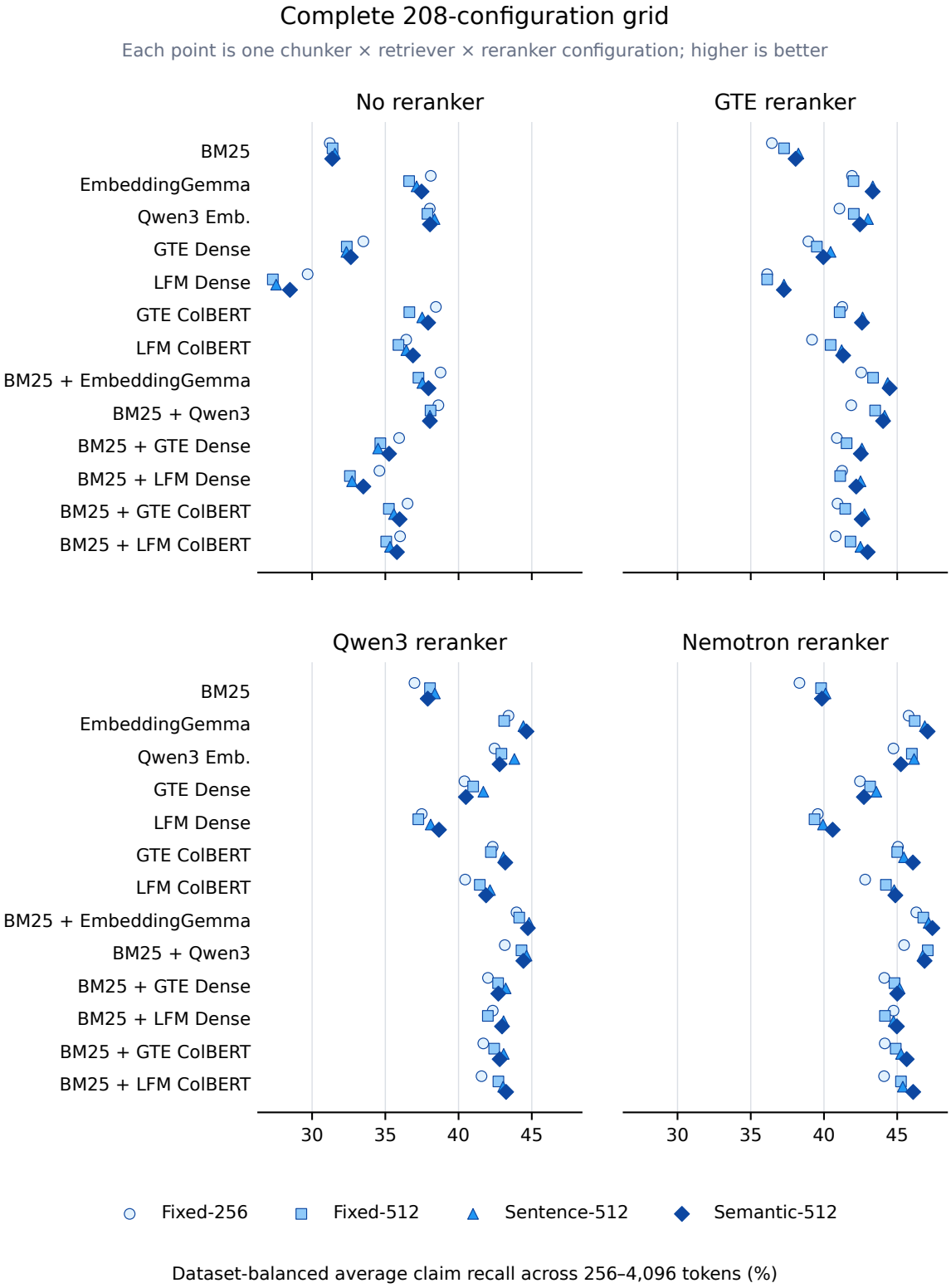


Figure 6: **Descriptive results for all 208 pipeline configurations.** Each point represents one chunker–retriever combination within the indicated reranking condition. Values are dataset-balanced average claim recall across the five context capacities. This figure describes complete-pipeline performance and is not interpreted as a paired component effect.

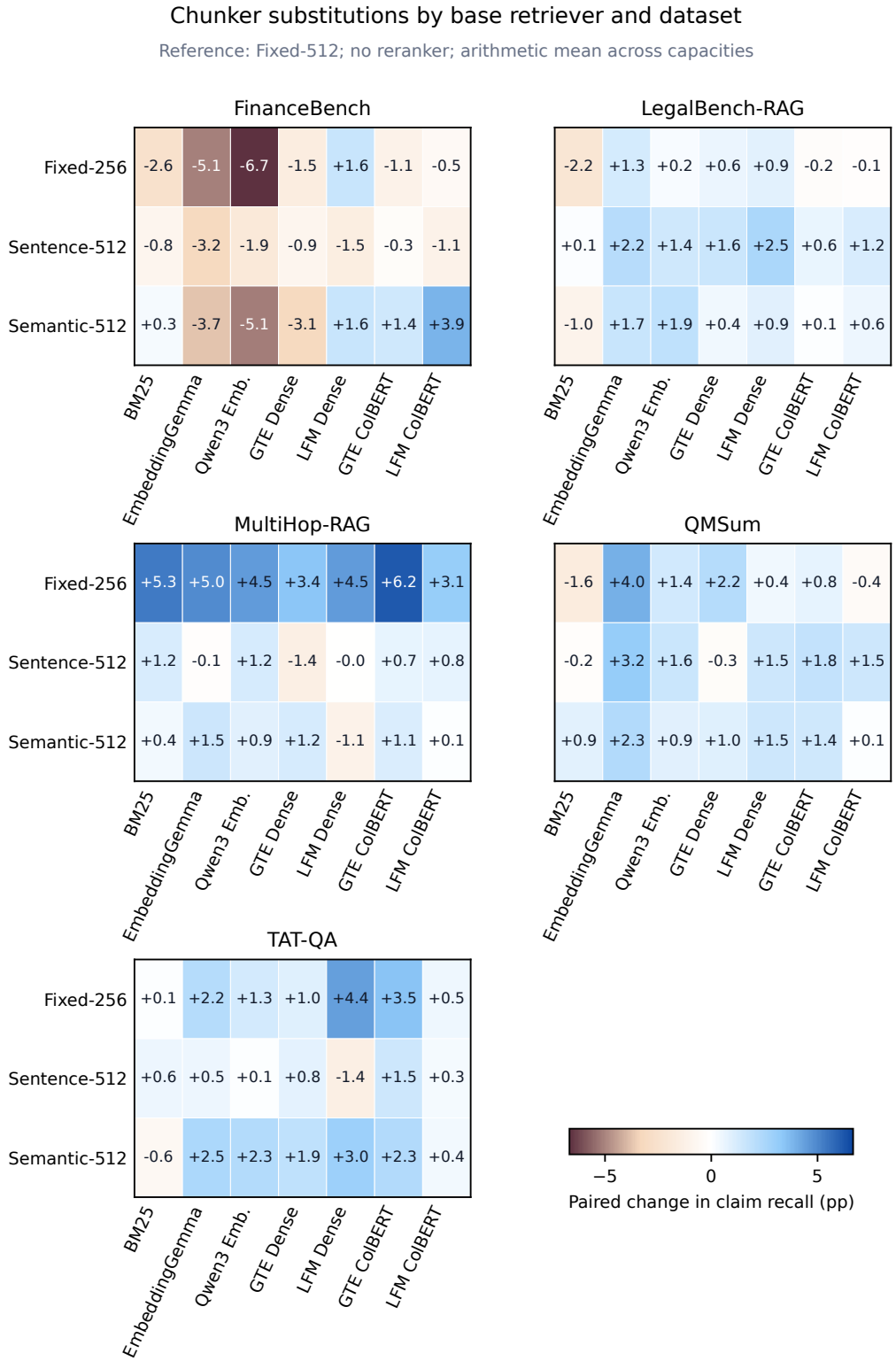


Figure 7: **Chunker effects for every dataset and base retriever.** Each cell reports the paired change in claim recall from replacing Fixed-512 with the indicated chunker, while holding the dataset, retriever, capacity, and no-reranker condition fixed. Values are percentage-point effects averaged equally across capacities.

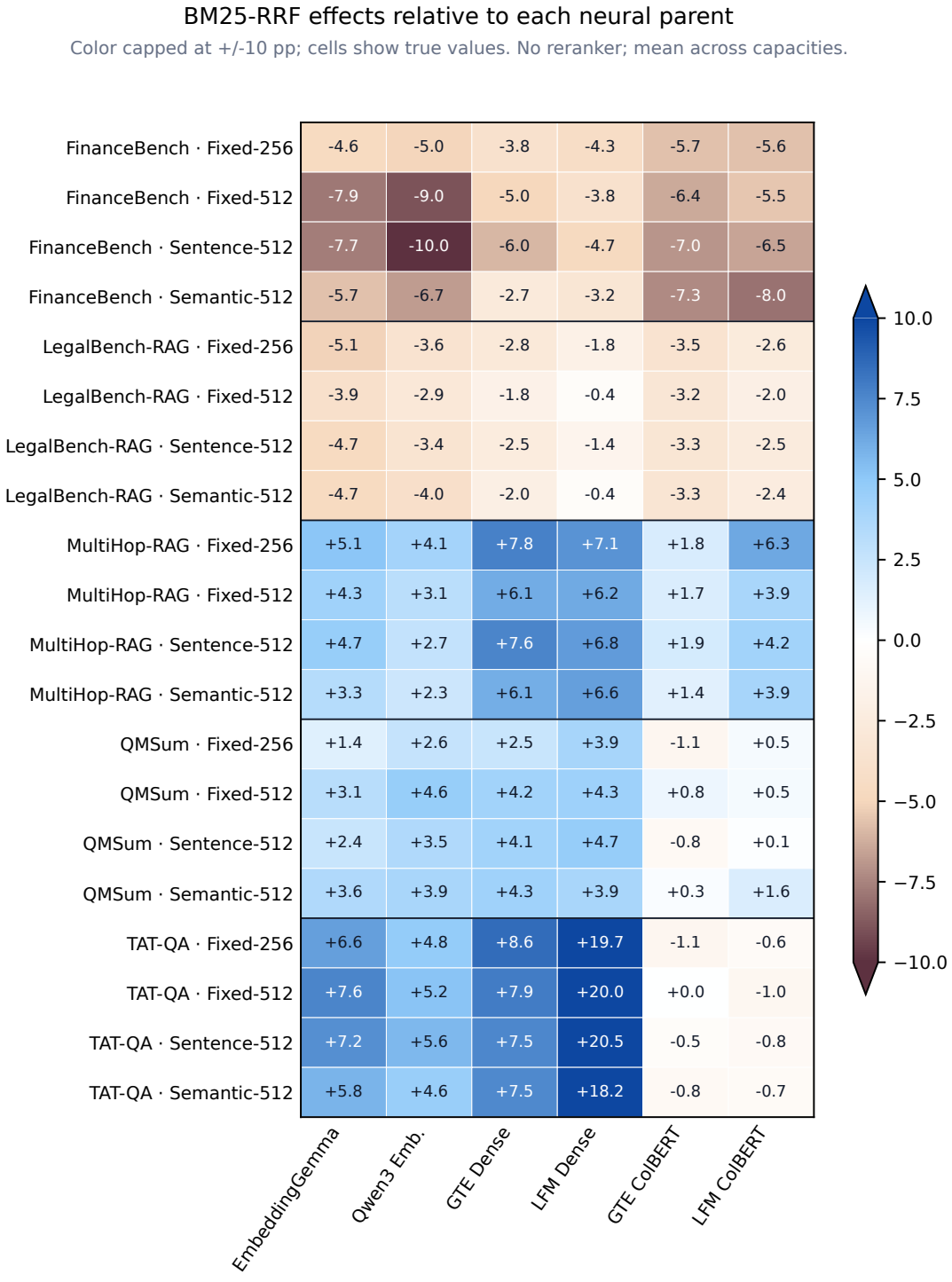


Figure 8: **All 120 matched BM25–RRF fusion effects.** Each cell compares a BM25–RRF retriever with its exact neural parent under the same dataset, chunker, capacity, and no-reranker condition. Values are paired percentage-point changes in claim recall, averaged equally across capacities. Colors are capped at ± 10 points so that smaller effects remain visible; cells show the true values.

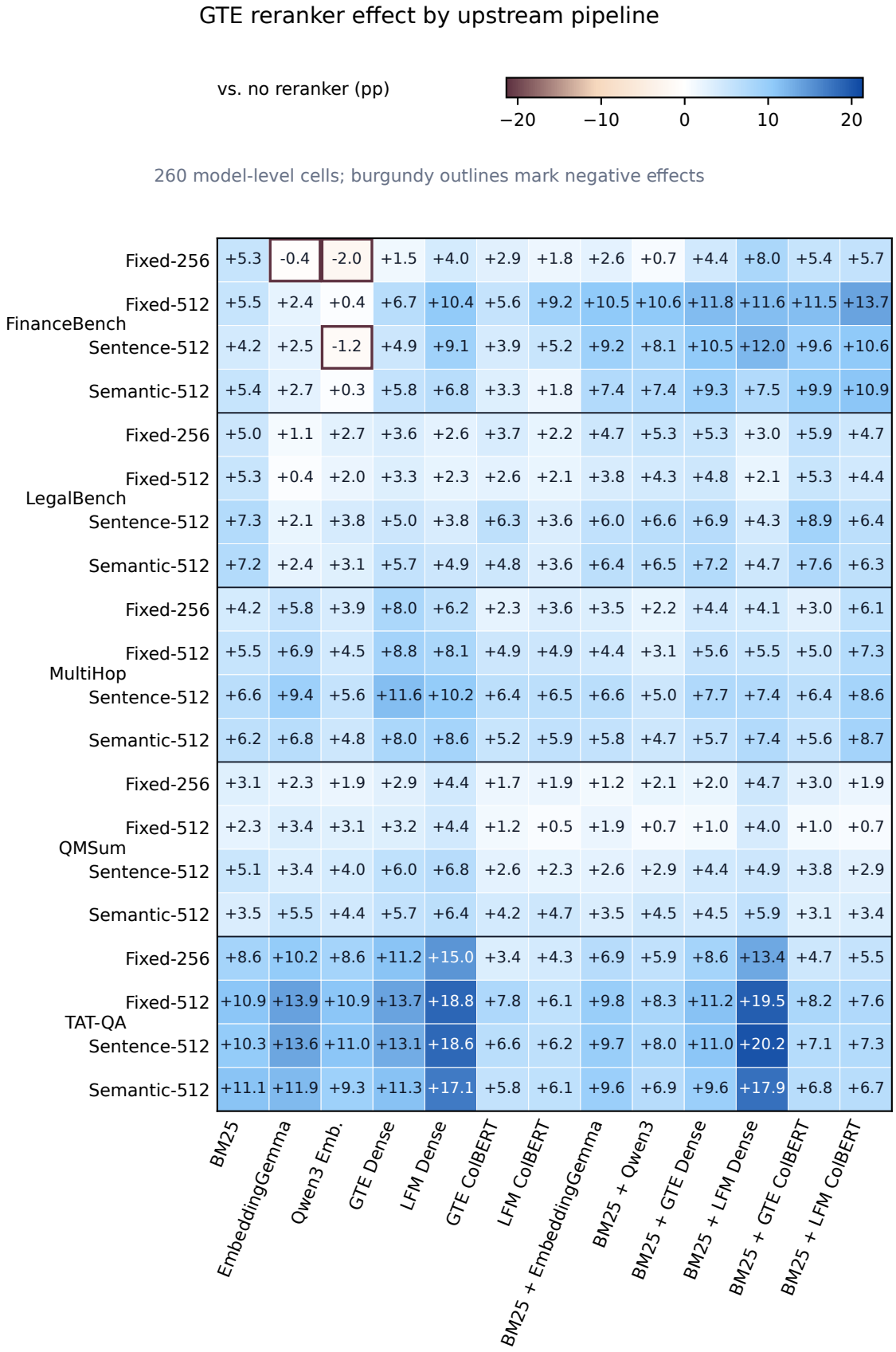


Figure 9: **GTE reranker effects across 260 upstream cells.** Each cell is the paired percentage-point change in claim recall relative to no reranker for the same dataset, chunker, and retriever, averaged equally across the five token capacities. Burgundy outlines mark negative effects.

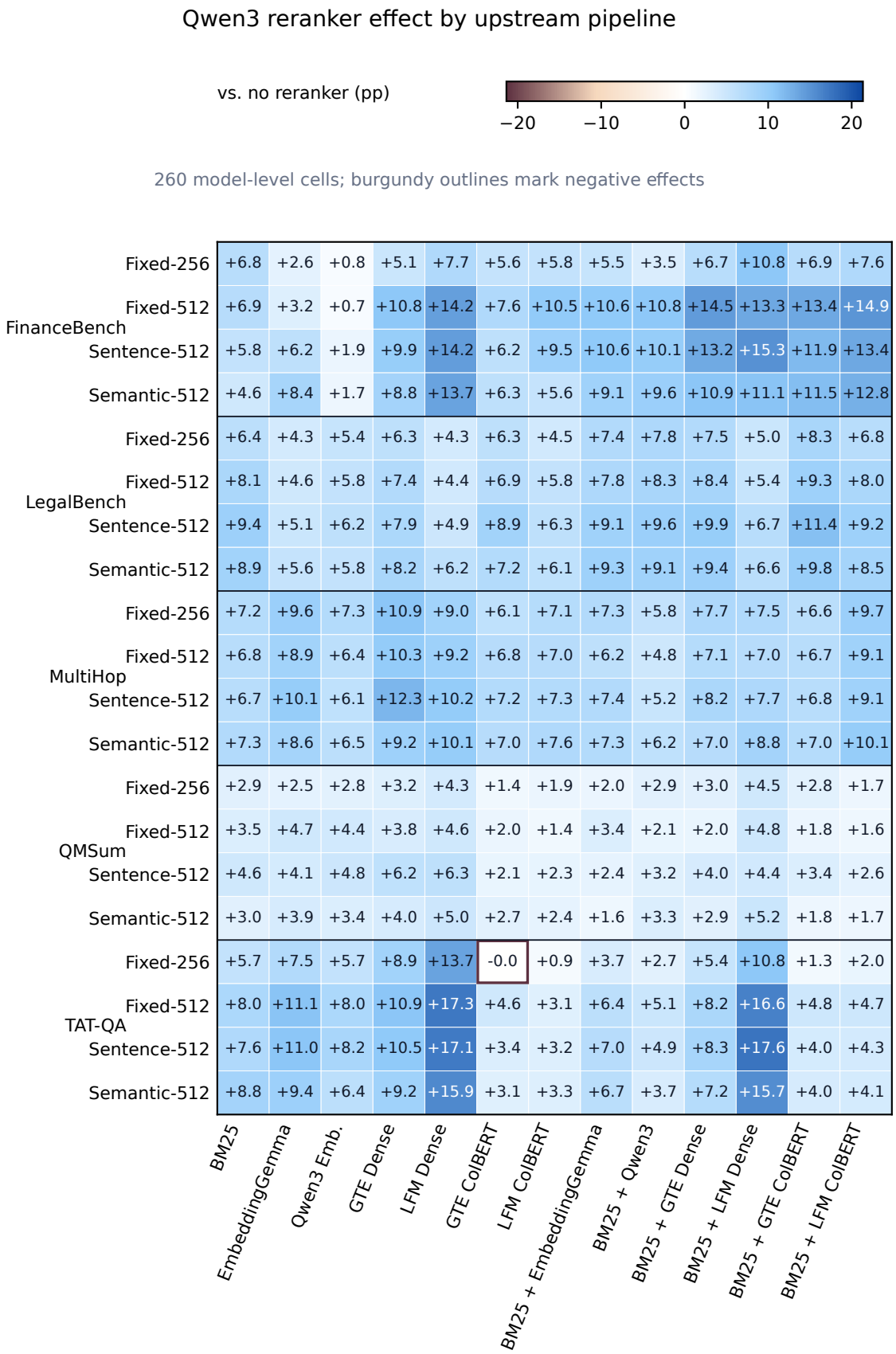


Figure 10: **Qwen3 reranker effects across 260 upstream cells.** Cells use the same paired comparison and capacity averaging as Figure 9. Burgundy outlines mark negative effects.

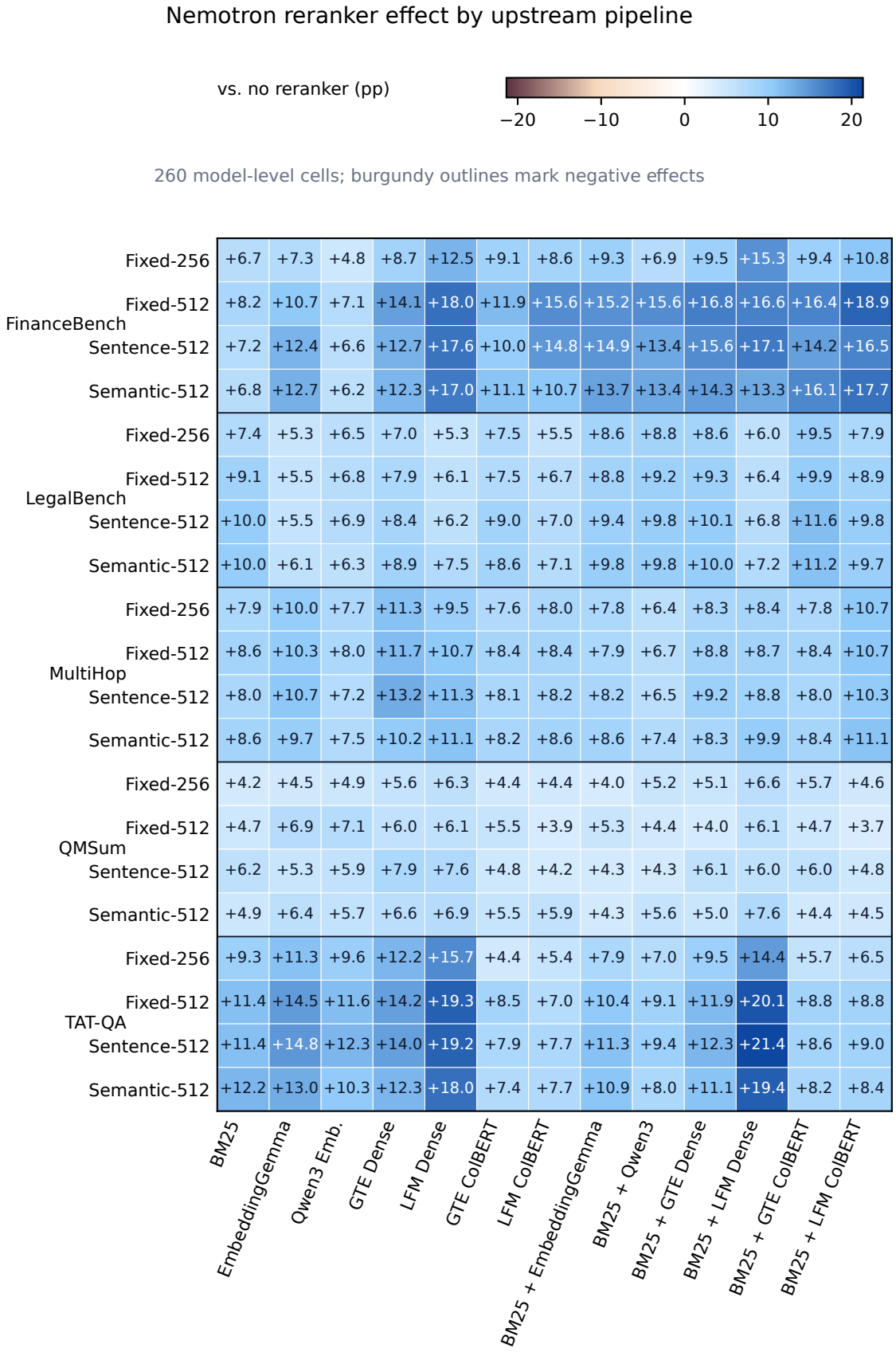


Figure 11: **Nemotron reranker effects across 260 upstream cells.** Cells use the same paired comparison and capacity averaging as Figure 9. Across these three figures, 776 of the 780 effects are positive; burgundy outlines identify the four negative cells.

LaTeXTrans: Structure-Aware Translation of Complex LaTeX Documents

Ziming Zhu[1*], **Chenglong Wang**[1*], **Haosong Xu**[1], **Shunjie Xing**[1], **Yifu Huo**[1], **Fengning Tian**[2], **Quan Du**[2], **Di Yang**[1,2], **Chunliang Zhang**[1,2], **Tong Xiao**[1,2†] **and Jingbo Zhu**[1,2]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]NiuTrans Research, Shenyang, China
{zhuzm0721, clwang1119}@gmail.com, {xiaotong, zhujingbo}@mail.neu.edu.cn

Abstract

Translating LaTeX documents is challenging because natural-language content is tightly interleaved with executable structures such as commands, equations, environments, and cross-references. Direct translation may therefore corrupt document structure and lead to formatting errors or compilation failures. We present LaTeXTrans, a structure-aware framework that separates semantic translation from executable LaTeX structure through four stages: structure-aware parsing, protected translation, structural validation, and document reconstruction. LaTeXTrans isolates non-translatable structures with protected placeholders, validates structural constraints after translation, and reconstructs the translated content into the original multi-file project structure. We evaluate LaTeXTrans on 100 real-world LaTeX projects across multiple domains and translation directions. Experimental results show consistent improvements in translation quality over direct large language model (LLM) translation while maintaining strong structural reliability. On English–Chinese translation, LaTeXTrans achieves a 97% compilation success rate with only 0.43 formatting errors per document on average. Controlled experiments further show that its structural performance remains stable when previously unseen user-defined command and environment identifiers are introduced.

1 Introduction

LaTeX (Urban, 1986) is a widely used macro system built on TeX for typesetting complex and structured documents, and has become the standard infrastructure for scholarly publishing worldwide. Although LaTeX itself is language-agnostic, academic communication remains overwhelmingly dominated by English: nearly 98% of scientific papers are published in English (Kleidermacher and Zou, 2025), while only about 5% of the global population is proficient in the language as of 2025 (Dwyer, 2025). This asymmetry creates an additional burden for non-native English speakers when reading scientific literature.

A straightforward solution is to first compile LaTeX source files into PDFs and then perform machine translation at the PDF level (*a.k.a.*, PDF translation). However, PDF translation often suffers from document parsing errors and, more importantly, cannot preserve the original editable source structure. A more promising solution is to directly translate LaTeX source files and subsequently recompile them into documents in the target language. While prior work has demonstrated that source-level translation can better preserve document formatting than PDF translation (Hoy et al., 2025), translating real-world LaTeX projects remains challenging. Unlike plain text, LaTeX source files tightly interleave natural language with mathematical expressions, citation commands, cross-references, and user-defined macros. These elements jointly constitute an executable document structure whose syntax and dependencies must remain valid throughout translation. As a result, LaTeX translation is not merely a text-to-text machine translation problem, but a *structure-constrained document transformation* task.

This problem involves two main structural challenges. First, structural information such as commands, equations, labels, and references must be preserved, since even small modifications may lead to formatting errors or compilation failures. Second, real-world LaTeX projects are often structurally heterogeneous, containing multiple interdependent files, customized commands, and domain-specific macros that are difficult to cover with fixed rules. Additionally, scientific papers are typically long documents, where technical terms and contextual information may span multiple sections, making document-level context useful in some transla-

[*]Authors contributed equally.
[†]Corresponding author.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 324–343
October 24, 2026 ©2026 Association for Computational Linguistics

tion scenarios. A practical system should therefore prioritize structural preservation of complex LaTeX constructions, while allowing document-level context to be incorporated when needed.

Recent LLM-based agent systems have demonstrated the potential of autonomous workflows that can decompose complex objectives, and iteratively act on environmental feedback, enabling substantial progress on complex software-engineering tasks (Yang et al., 2024; Wang et al., 2024a). This motivates us to formulate LaTeX translation as an agentic document transformation task. To this end, we propose **LaTeXTrans**, a structure-aware framework that directly translates complex LaTeX documents. Instead of requiring a translation model to jointly handle natural language and LaTeX syntax, LaTeXTrans separates semantic translation from executable document structures through four stages: *structure-aware parsing*, *protected translation*, *structural validation*, and *document reconstruction*. Specifically, the parser decomposes the input project into translation units while protecting non-translatable structures. The Translator then focuses on translating natural-language content, while the Validator checks structural consistency and provides feedback for necessary revisions. Finally, LaTeXTrans restores the translated content into the original document structure and reconstructs a compilable LaTeX project. Additionally, LaTeXTrans supports optional document-level context augmentation through a running summary and a terminology dictionary.

To evaluate LaTeXTrans, we construct a multi-domain and multilingual benchmark for LaTeX source-level translation using real-world TeX projects from arXiv. Based on this benchmark, we compare LaTeXTrans with strong formatted-text translation baselines in terms of both translation quality and structural preservation. We further analyze the contribution of different system components, generalization to previously unseen user-defined LaTeX constructs, and the cost–quality trade-off of using LaTeXTrans compared with direct translation. Experimental results show that LaTeXTrans can robustly translate complex LaTeX projects while maintaining high structural fidelity. In the En–Zh setting, it achieves a one-pass compilation success rate of 97%, with an average of only 0.43 formatting errors per document.

2 Related Work

LLM-based Machine Translation. The emergence of LLMs has introduced a new paradigm for machine translation, moving beyond traditional supervised training on parallel corpora toward more general-purpose language understanding (Gain et al., 2025). By leveraging in-context learning, LLM-based translation has demonstrated competitive performance in zero-shot and few-shot settings (Vilar et al., 2023; Luo et al., 2025), particularly for high-resource language pairs. Compared with conventional neural machine translation systems, LLMs can more flexibly incorporate task instructions and contextual information during generation. However, translating long scientific documents independently at the sentence or section level may still lead to insufficient use of document-level context and inconsistent terminology.

Document-Level Translation. Document-level machine translation aims to improve translation quality by incorporating information beyond individual sentences, such as previously translated content, discourse context, and document-level terminology. With the development of LLMs, recent studies have explored maintaining contextual information throughout the translation process. For example, DELTA (Wang et al., 2024b) introduces multi-level memory into online document-level translation to improve contextual coherence and translation consistency. Such approaches demonstrate the importance of document-level context for long-form translation. However, they mainly focus on linguistic consistency and do not explicitly consider the executable structures contained in formatted documents such as LaTeX.

Structured Document Translation. Structured documents such as LaTeX and XML interleave natural language with commands, tags, mathematical expressions, and other elements encoding document structure (Kleidermacher and Zou, 2025). Recent work has explored the translation of scientific and formatted documents, including source-level LaTeX translation (Hoy et al., 2025). Compared with PDF-level translation, directly operating on LaTeX source provides better access to the original document structure and avoids reconstructing source-level information from compiled documents. Practical open-source systems such as

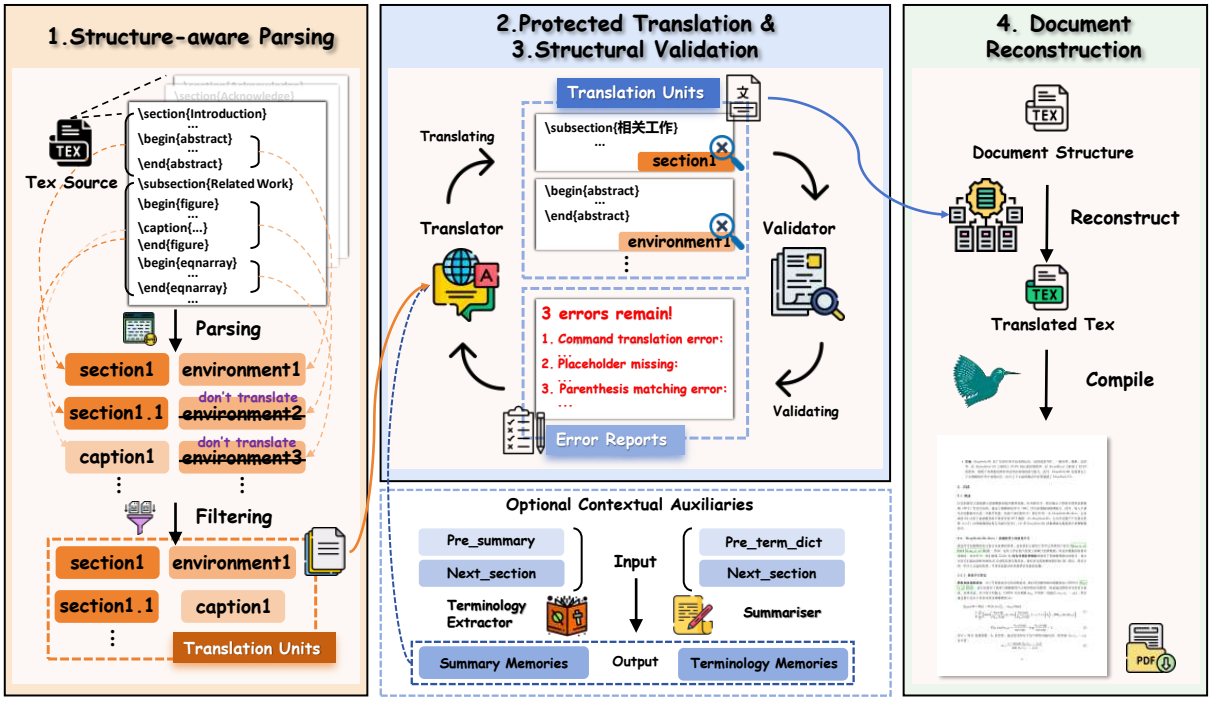


Figure 1: Overview of the structure-aware translation framework in LaTeXTrans. The core pipeline consists of structure-aware parsing, protected translation, structural validation, and document reconstruction, while summary and terminology memories provide auxiliary document-level context.

MathTranslate* and GPT-Academic† also provide LaTeX-aware translation pipelines. However, real-world LaTeX projects may contain multiple source files, customized commands, user-defined environments, and domain-specific macros. Many of these constructs are project-specific and may use previously unseen identifiers, making it undesirable for a translation system to depend on a fixed inventory of known command or environment names. Generalization to such user-defined constructs therefore represents an important aspect of structured document translation. Moreover, conventional MT metrics such as BLEU (Papineni et al., 2002) and COMET (Rei et al., 2020) mainly measure linguistic quality and do not directly assess whether the translated source remains structurally valid and compilable. Therefore, structured document translation requires considering both translation quality and structural reliability.

3 Structure-Aware Translation Framework

LaTeXTrans translates structured LaTeX documents by explicitly separating semantic translation from executable document structure. As illustrated in Figure 1, the core framework follows four stages: *structure-aware parsing*, *protected translation*, *structural validation*, and *document reconstruction*. The input LaTeX project is first resolved and decomposed into translation units, while structural components that should not be modified are isolated from the translation process. The resulting units are then translated under structural constraints and explicitly validated before being restored to the original document structure. Beyond the core pipeline, LaTeXTrans optionally uses summary and terminology memories for document-level context.

3.1 Structure-Aware Parsing

A LaTeX project is not a plain-text sequence. Natural-language content is tightly coupled with mathematical expressions, commands, environments, figures, references, and user-defined structures. Directly translating the complete source therefore exposes a large amount of executable syntax to the translation model, increasing both unnecessary processing and the risk of structural corruption. LaTeXTrans addresses this issue by first resolving the source files, isolating LaTeX structures that should not be translated, and organizing the remaining content into translation units through content-based filtering.

Project Resolution. Real-world LaTeX projects often distribute their content across multiple source

*https://github.com/SUSYUSTC/MathTranslate
†https://github.com/binary-husky/gpt_academic

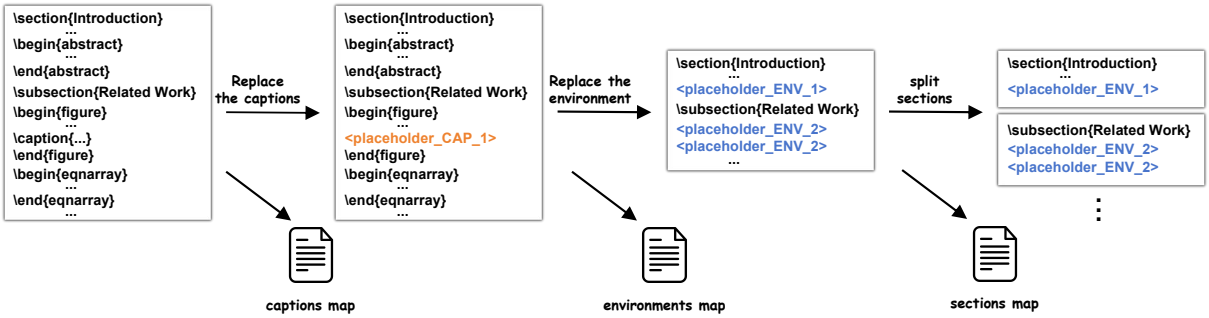


Figure 2: Structure-aware parsing in LaTeXTrans. Structural components are isolated with protected placeholders, while translatable content is organized into translation units at different granularities.

files. LaTeXTrans first resolves the source files required by the main document and merges them into a unified representation for subsequent processing. For merged files, boundary placeholders are inserted to record their original positions so that the corresponding content can be restored during reconstruction. This step allows the following translation stages to operate on a coherent document context while retaining the structural information required to recover the original project organization.

Protected Structure Extraction. The central idea of structure-aware parsing is to reduce the amount of executable LaTeX syntax exposed to the translation model. Rather than requiring the LLM to identify and preserve every structural element during generation, LaTeXTrans temporarily isolates components that should remain unchanged and replaces them with protected placeholders.

The extraction process is illustrated in Figure 2. LaTeXTrans first separates translatable captions from their surrounding figure or table structures and represents them independently. It then extracts LaTeX environments from the remaining source. Common environments whose structural roles are known can be directly protected according to predefined rules, whereas customized or otherwise uncertain environments are subsequently examined by the content-based Filter. Protected environments are replaced with environment-level placeholders. The remaining source is then organized according to the document hierarchy.

This procedure converts the mixed LaTeX source into translation units at different granularities. Section- and environment-level units preserve broader contextual information, while captions are represented as sentence-level units. More importantly, structures that should remain unchanged are removed from the editable translation space. As a result, the translation model operates primarily on semantic content rather than the complete LaTeX source, reducing the opportunity for accidental modifications to executable syntax.

Translation Unit Construction and Filtering. Not all extracted environments can be classified using predefined LaTeX rules. In particular, user-defined environments may use arbitrary names while containing either natural-language content or purely structural material. For environments that are not explicitly protected by predefined rules, translation is treated as the default behavior. LaTeXTrans therefore introduces an LLM-based Filter to further examine their content and determine whether translation is required. Environments classified as translatable are retained as translation units, whereas those classified as non-translatable are protected from subsequent generation. This design avoids relying on an exhaustive enumeration of LaTeX environment names and allows the same filtering procedure to be applied to user-defined environments with previously unseen identifiers.

3.2 Protected Translation

After structure-aware parsing, LaTeXTrans translates only the units identified as translatable, while protected structures remain represented by placeholders. This separation limits the editable content exposed to the translation model and reduces the possibility of modifying executable LaTeX syntax during generation.

For each translation unit, the Translator is instructed to translate only the natural-language content while preserving all protected placeholders and any remaining LaTeX-specific expressions. The latter may include user-defined commands that are not explicitly interpreted by the Parser. LaTeXTrans does not require semantic understanding of such command names during parsing; instead, the Translator is instructed to preserve them unchanged

during generation. Depending on the document context, a running summary and a terminology dictionary can additionally be provided to support contextual understanding and consistent terminology usage. The complete translation prompts are provided in the Appendix F.

Although most complex structures have already been isolated during parsing, some LaTeX commands, including project-specific commands, may remain inside the translation units because they are tightly coupled with natural-language content. Instruction-based preservation alone is insufficient to guarantee their structural correctness. LaTeXTrans therefore further checks these commands during structural validation.

3.3 Structural Validation

Even when the translation model is explicitly instructed to preserve LaTeX syntax, generation errors may still occur. For example, commands such as "`\textbf{}`" may be omitted, while "`\left`" may be incorrectly interpreted as natural language. Such errors can alter document formatting or even result in compilation failures. Instead of relying solely on the instruction-following ability of the Translator, LaTeXTrans explicitly validates the structural integrity of translated units before reconstruction.

As illustrated in Figure 1, the Validator examines translated units from three aspects: *command preservation*, *placeholder preservation*, and *parenthesis matching*. First, it checks whether LaTeX commands appearing in the source unit are incorrectly translated, removed, or otherwise modified. Importantly, this check is applied to the commands observed in each source unit rather than to a fixed vocabulary of predefined LaTeX commands. Therefore, user-defined commands that remain inside translation units are subject to the same preservation check as standard LaTeX commands. Second, the Validator verifies that all protected placeholders are retained after translation, since a missing placeholder prevents the corresponding structure from being correctly restored. Third, it checks the matching of structural parentheses to identify syntax corruption introduced during generation.

If a translated unit violates any of these constraints, the Validator generates an error report describing the detected problem. The original source unit, its current translation, and the corresponding error report are then provided to the Translator for revision. Only the erroneous units are re-translated, avoiding unnecessary regeneration of units that have already passed validation. This translation–validation loop progressively removes structural errors before the document enters the reconstruction stage.

3.4 Contextual Auxiliaries

Beyond the core structure-preserving pipeline, LaTeXTrans optionally supports two contextual auxiliaries for applications that require additional document-level information. Inspired by document-level translation methods (Wang et al., 2024b), these auxiliaries provide a running summary and a terminology dictionary to the Translator.

Summary Memory. A running summary is maintained throughout the translation process to provide compact contextual information from previously processed sections. After a translation unit is completed, the Summarizer combines the previous summary with the source content of the current unit and produces an updated summary. The resulting summary is then provided as additional context when translating subsequent units. In this way, the Translator can access relevant document-level information without repeatedly processing the complete preceding document.

Terminology Memory. LaTeXTrans additionally maintains a bilingual terminology dictionary for technical terms that appear during translation. After each unit is translated, the Terminology Extractor identifies domain-specific terms in the source and their corresponding translations, and updates the dictionary accordingly. For subsequent units, previously recorded term pairs are included in the Translator input to encourage consistent translation of repeated terminology across the document.

Note that these two components are optional extensions rather than part of the core structure-preserving pipeline. They can be enabled when additional document-level context or terminology control is desired.

3.5 Document Reconstruction

After translation and structural validation, LaTeXTrans reconstructs the translated project by restoring the protected structures extracted during parsing. Each placeholder is replaced with its corresponding original content according to the mappings retained throughout the pipeline, while translated captions, environments, and section-level units are inserted back into their original positions.

For projects containing multiple source files, the boundary placeholders introduced during project resolution are further used to restore the original file boundaries and regenerate the corresponding `.tex` files. In this way, LaTeXTrans preserves not only the internal document structure but also the original multi-file project organization.

The reconstructed project is then compiled to verify that the translated source remains executable. LaTeXTrans automatically selects an appropriate compilation engine, such as pdfLaTeX or XeLaTeX, according to the document configuration, and produces the final target-language PDF. This reconstruction stage closes the structure-aware translation pipeline by converting the validated translation units back into a complete LaTeX project.

4 Experiments

4.1 Experimental Setup

Dataset. As no public benchmark is currently available for LaTeX source-level document translation, we construct an evaluation set using the source projects of 100 English papers collected from arXiv. The dataset covers three scientific domains, including 50 papers from computer science, 30 from physics, and 20 from mathematics. We consider three translation directions overall: English–Chinese (En–Zh), English–Japanese (En–Ja), and English–Korean (En–Ko), with individual analyses using subsets of these directions as specified below. For the generalization analysis in Section 4.3, we additionally construct controlled variants of projects containing user-defined commands and environments. We replace their original identifiers with randomly generated names while consistently updating the corresponding definitions and usages, leaving the document content and executable semantics unchanged. This paired construction allows us to isolate whether LaTeXTrans depends on familiar command or environment names.

Baselines. We compare LaTeXTrans with two groups of baselines: *direct source translation* and *LaTeX-aware translation systems*. For direct source translation, we include NiuTrans and Google Translate as conventional MT systems, together with Qwen3-14B (Team, 2025), DeepSeek-V3 (DeepSeek-AI, 2024), and GPT-4o (OpenAI, 2024) as LLM-based baselines. Since directly processing a complete LaTeX project is infeasible for several baseline systems, we segment the source into section-level units and translate them independently without structure-aware parsing or structural validation. We further include GPT-Academic[‡] as a practical LaTeX-aware translation baseline for format-preservation comparison.

Evaluation Metrics. We evaluate the systems from both translation quality and structural reliability. For translation quality, we use COMETKiwi (Rei et al., 2022) as a reference-free MT metric and an LLM-based evaluator that scores translation quality using GPT-4o. For structural reliability, we report the *Compilation Success Rate* (CSR), defined as the percentage of translated projects that can be compiled without manual intervention, and the *Formatting Error Count* (FEC), which measures the average number of formatting-related errors in each translated document. Detailed definitions of formatting errors and the LLM-based evaluation protocol are provided in the Appendix C and Appendix F.

For the controlled generalization experiment, we additionally report the *Construct Preservation Rate* (CPR), which measures the proportion of user-defined commands and environments that are correctly preserved in the translated source. The specific definitions of CSR, FEC and CPR can be found in the Appendix B. For the cost–benefit analysis, we report the average API cost per translated document.

Implementation Details. For the direct-translation baselines, all systems operate on the same section-level units to ensure a consistent evaluation protocol. For LaTeXTrans, the Filter uses a temperature of 0 with a maximum output length of 50 tokens, while the remaining LLM-based components use a maximum output length of 8,192 tokens. The summary and terminology memories described in Section 3.4 are disabled in all experiments, and LaTeXTrans refers to the core structure-preserving pipeline. Additional evaluation details are provided in the Appendix A. LaTeXTrans supports local deployment through both command-line and web-based interfaces and outputs the translated multi-file LaTeX project together with the compiled target-language PDF, the specific usage methods and web operation procedures are as shown in the Appendix D. With locally hosted models, the entire translation process can remain within the local environment.

[‡] https://github.com/binary-husky/gpt_academic

Subject	En-Zh		En-Ja		En-Ko	
	CSR (↑)	FEC (↓)	CSR (↑)	FEC (↓)	CSR (↑)	FEC (↓)
Physics	100%	0.37	93%	0.43	90%	0.47
Mathematics	100%	1.15	95%	1.30	100%	1.35
Computer Science	94%	0.18	76%	0.20	86%	0.25
Total	97%	0.43	85%	0.49	90%	0.54

Table 1: Structural preservation of LaTeXTrans across different scientific domains and translation directions.

4.2 Main Results

Structural Preservation. We first evaluate whether LaTeXTrans can preserve executable document structure during translation. Table 1 reports the CSR and FEC across three scientific domains and three translation directions. LaTeXTrans achieves a CSR of 97% on En–Zh, 85% on En–Ja, and 90% on En–Ko, while maintaining low FEC values of 0.43, 0.49, and 0.54 errors per document, respectively. These results show that the structure-aware pipeline can reliably reconstruct compilable target-language projects across different domains and target languages.

We further observe that structural performance varies across domains. For En–Zh, all physics and mathematics projects compile successfully, while computer science achieves a CSR of 94%. These results establish the structural reliability of LaTeXTrans on naturally occurring scientific projects. However, they do not isolate whether the system depends on familiar LaTeX command or environment names. We therefore conduct a controlled generalization analysis in Section 4.3, where user-defined identifiers are replaced with previously unseen random names while the underlying document structure and content remain unchanged.

Translation Quality. In addition to structural reliability, a structure-aware translation system should preserve the linguistic quality of the translated content. Table 2 compares LaTeXTrans with conventional MT systems and direct LLM-based translation using COMETKiwi and LLM-score. Across all three LLM backbones, LaTeXTrans consistently improves COMETKiwi over direct source translation. For example, on En–Zh, COMETKiwi increases from 67.22 to 73.59 with GPT-4o. Similar improvements are observed on En–Ja. The LLM-score results are comparatively stable, with small increases or decreases depending on the backbone and translation direction. This suggests that the structural processing introduced by LaTeXTrans does not substantially compromise overall linguistic quality.

System	En-Zh		En-Ja	
	COMETKiwi	LLM-score	COMETKiwi	LLM-score
NiuTrans	64.69	7.93	65.49	8.19
Google Translate	46.23	5.93	56.21	7.01
Qwen3-14B	68.18	8.76	72.84	8.66
LaTeXTrans $_{\text{Qwen3-14B}}$	71.37	8.97	74.68	8.51
DeepSeek-V3	67.26	**9.02**	72.17	**9.00**
LaTeXTrans $_{\text{DeepSeek-V3}}$	73.48	9.01	**75.39**	8.89
GPT-4o	67.22	8.58	71.16	8.91
LaTeXTrans $_{\text{GPT-4o}}$	**73.59**	8.92	74.47	8.93

Table 2: Translation quality comparison between direct translation baselines and LaTeXTrans. Bold indicates the best result within each backbone group.

System	Compiled	Phys. (30)			Math. (20)			CS (50)		
		A	B	C	A	B	C	A	B	C
LaTeXTrans	97/100	25	3	2	14	3	3	45	1	1
GPT-Academic	98/100	24	1	4	9	6	5	40	7	2

Table 3: Human evaluation of format preservation on 100 En–Zh projects. "Compiled" reports the number of projects successfully compiled without manual intervention. Phys., Math., and CS denote physics, mathematics, and computer science, respectively.

Human Evaluation. Compilation success alone does not fully reflect the visual quality of a translated document. We therefore conduct a human evaluation on all 100 En–Zh projects to compare the format preservation of LaTeXTrans and GPT-Academic. Successfully compiled documents are further categorized into three levels: (A) the original formatting is fully preserved; (B) minor formatting errors are present but do not affect readability; and (C) severe formatting corruption substantially affects readability.

As shown in Table 3, GPT-Academic successfully compiles 98 out of 100 projects, compared with 97 for LaTeXTrans. However, among the successfully compiled documents, LaTeXTrans achieves better visual format preservation: 84 of 97 documents (86.6%) are rated as level A, compared with 73 of 98 (74.5%) for GPT-Academic. Moreover, only 6.2% of LaTeXTrans outputs are rated as level C, compared with 11.2% for GPT-Academic. These results indicate that LaTeXTrans provides better visual format preservation once a valid target-language document is produced.

4.3 Generalization to Unseen LaTeX Identifiers

To examine whether LaTeXTrans depends on familiar LaTeX identifiers, we conduct controlled experiments on user-defined commands and environments. For each construct type, we select 30 eligible projects and compare each original project with a randomized counterpart, where the corresponding identifier is replaced consistently in its

Setting	CSR (↑)	FEC (↓)	CPR (↑)
Cmd. Original	100%	0.23	98.7%
Cmd. Randomized	100%	0.27	98.2%
Env. Original	100%	0.33	96.3%
Env. Randomized	100%	0.30	96.4%

Table 4: Generalization to previously unseen user-defined LaTeX identifiers on 30 eligible projects for each construct type.

definition and all occurrences while leaving the document semantics and functionality unchanged. Since this experiment focuses on structural generalization rather than language or model variation, we conduct all experiments on the En–Zh task using GPT-4o. Table 4 reports the results.

Randomizing user-defined identifiers results in almost no degradation in structural performance. For user-defined commands, CSR remains at 100%, while FEC changes only slightly from 0.23 to 0.27 and CPR decreases by merely 0.5 percentage points. A similar pattern is observed for user-defined environments: CSR remains unchanged at 100%, while FEC and CPR remain nearly identical before and after randomization. These results indicate that LaTeXTrans does not strongly depend on the specific names of user-defined commands and environments, and remains effective when previously unseen identifiers are introduced.

4.4 Ablation Study

We evaluate the contribution of the two core structure-preserving components of LaTeXTrans on the En–Zh task using GPT-4o. We use *Direct Translation* as the baseline, where the LaTeX source is segmented into section-level units and translated directly without structure-aware parsing or structural validation. As shown in Table 5, structure-aware parsing substantially improves both translation quality and structural reliability. COMETKiwi increases from 67.22 to 74.47 and LLM-score from 8.58 to 8.89, while CSR increases from 77% to 84% and FEC decreases from 3.17 to 1.24. These results suggest that isolating non-translatable LaTeX structures reduces interference from executable syntax and allows the translation model to focus more effectively on semantic content. Structural validation primarily improves structural reliability rather than translation quality. While COMETKiwi decreases slightly from 74.47 to 73.59, LLM-score remains largely stable. Its contribution is instead clearly reflected in structural reliability: CSR further increases from 84% to 97%, while FEC decreases from 1.24 to 0.43. These results show that parsing and validation play complementary roles: parsing improves both translation quality and structural preservation, whereas validation primarily corrects residual structural errors before document reconstruction.

Setting	COMETKiwi	LLM-score	CSR (↑)	FEC (↓)
Direct Translation	67.22	8.58	77%	3.17
+ Parsing	74.47	8.89	84%	1.24
+ Validation	73.59	8.92	97%	0.43

Table 5: Ablation study of the core LaTeXTrans pipeline. Components are added cumulatively from top to bottom. "Parsing" denotes structure-aware parsing, while "Validation" denotes structural validation.

Model	System	En–Zh ($)	En–Ja ($)
DeepSeek-V3	Direct Translation	0.02	0.03
	LaTeXTrans	0.10	0.13
GPT-4o	Direct Translation	0.13	0.11
	LaTeXTrans	0.35	0.45

Table 6: Average API cost per document for direct translation and LaTeXTrans on the En–Zh and En–Ja tasks.

4.5 Cost–Benefit Analysis

Using LaTeXTrans introduces additional LLM calls for structure-aware processing and validation compared with direct translation. We therefore compare the average API cost per document between direct translation and LaTeXTrans using DeepSeek-V3 and GPT-4o on the En–Zh and En–Ja tasks. The direct setting uses the same section-level translation protocol described in Section 4.1, whereas LaTeXTrans uses the full core pipeline.

As shown in Table 6, depending on the model and translation direction, LaTeXTrans increases the average API cost by approximately 2.7–5.0× compared with direct translation. However, this additional cost is accompanied by the translation-quality and structural-reliability improvements reported in Section 4.2. Across both GPT-4o and DeepSeek-V3, LaTeXTrans substantially improves COMETKiwi while providing reliable reconstruction of compilable LaTeX projects. These results indicate that the additional API cost provides a practical trade-off for improved structural reliability in source-level LaTeX document translation.

5 Conclusion

In this paper, we present LaTeXTrans, a structure-aware framework for translating complex LaTeX documents. LaTeXTrans integrates structure-aware parsing, protected translation, structural validation,

and document reconstruction to preserve LaTeX structures during translation. Experimental results demonstrate that LaTeXTrans improves both translation quality and structural reliability, providing a practical solution for structured LaTeX document translation.

Limitations

Our experiments cover a limited set of LLMs, translation directions, and scientific domains, and therefore may not reflect the full diversity of LaTeX translation scenarios. In addition, our generalization experiment evaluates previously unseen identifiers of user-defined commands and environments, rather than arbitrary macro semantics or highly customized TeX behaviors. Extending the evaluation to broader models, languages, and LaTeX constructs remains future work.

References

DeepSeek-AI. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Roland Dwyer. 2025. How many countries speak english in 2026.

Baban Gain, Dibyanayan Bandyopadhyay, and Asif Ekbal. 2025. Bridging the linguistic divide: A survey on leveraging large language models for machine translation.

Calvin Hoy, Samuel Frontull, and Georg Moser. 2025. Latexmt: Machine translation for latex documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 739–748.

Hannah Calzi Kleidermacher and James Zou. 2025. Science across languages: Assessing llm multilingual translation of scientific papers.

Yingfeng Luo, Tong Zheng, Yongyu Mu, Bei Li, Qinghong Zhang, Yongqi Gao, Ziqiang Xu, Peinan Feng, Xiaoqian Liu, Tong Xiao, and Jingbo Zhu. 2025. Beyond decoder-only: Large language models can be good encoders for machine translation.

OpenAI. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Ricardo Rei, Marcos Treviso, Nuno M. Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José G. C. de Souza, Taisiya Glushkova, Duarte Alves, Luisa Coheur, Alon Lavie, and André F. T. Martins. 2022. CometKiwi: IST-unbabel 2022 submission for the quality estimation shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 634–645, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Qwen Team. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Michael Urban. 1986. *An introduction to LATEX*. TEX users group.

David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. 2023. Prompting PaLM for translation: Assessing strategies and performance. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15406–15427, Toronto, Canada. Association for Computational Linguistics.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024a. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

Yutong Wang, Jiali Zeng, Xuebo Liu, Derek F. Wong, Fandong Meng, Jie Zhou, and Min Zhang. 2024b. Delta: An online document-level translation agent based on multi-level memory.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

A Additional Detailed Settings of the Evaluation

When computing COMETKiwi and LLM-scores, we used `pylatexenc`[§] to convert each LaTeX translation unit into plain text. Although LaTeXTrans parses structured LaTeX documents into fine-grained translation units, we followed the baseline's evaluation protocol by using section-level translation units for computing both COMETKiwi and LLM-scores. Furthermore, to assess contextual consistency in the LLM-score evaluation, we concatenated section-level translation units into paired paragraphs and then scored them using GPT-4o.

B Evaluation Metric Definitions

We provide detailed definitions of the structural metrics introduced in our experiments. Let $\mathcal{D} = \{d_i\}_{i=1}^{N}$ denote an evaluation set containing N translated LaTeX projects.

Compilation Success Rate. Compilation Success Rate (CSR) is defined as

$$\mathrm{CSR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\mathrm{Compile}(d_i) = 1] \times 100\% \quad (1)$$

where $\mathbb{I}[\cdot]$ denotes the indicator function, and $\mathrm{Compile}(d_i) = 1$ if the translated project is successfully compiled. In this paper, "one-pass" compilation refers to one complete end-to-end execution of LaTeXTrans without manual intervention, including the internal translation–validation revision process.

Formatting Error Count. Formatting Error Count (FEC) measures the average number of formatting-related errors in each translated document. Let e_i denote the number of formatting errors identified in d_i. FEC is calculated as

$$\mathrm{FEC} = \frac{1}{N}\sum_{i=1}^{N} e_i \quad (2)$$

When multiple downstream anomalies are caused by the same structural corruption, they are counted as a single structure-induced error.

Construct Preservation Rate. Construct Preservation Rate (CPR) measures the proportion of evaluated user-defined LaTeX constructs that are correctly preserved after translation. Let K_i denote the number of evaluated constructs in document d_i, and let C_i denote the number of correctly preserved constructs. We define

$$\mathrm{CPR} = \frac{\sum_{i=1}^{N} C_i}{\sum_{i=1}^{N} K_i} \times 100\% \quad (3)$$

For user-defined commands, a construct is considered preserved when its identifier remains unchanged at the corresponding occurrences. For user-defined environments, preservation requires the identifiers in both `\begin{...}` and `\end{...}` to remain unchanged and correctly matched.

[§] https://github.com/phfaist/pylatexenc

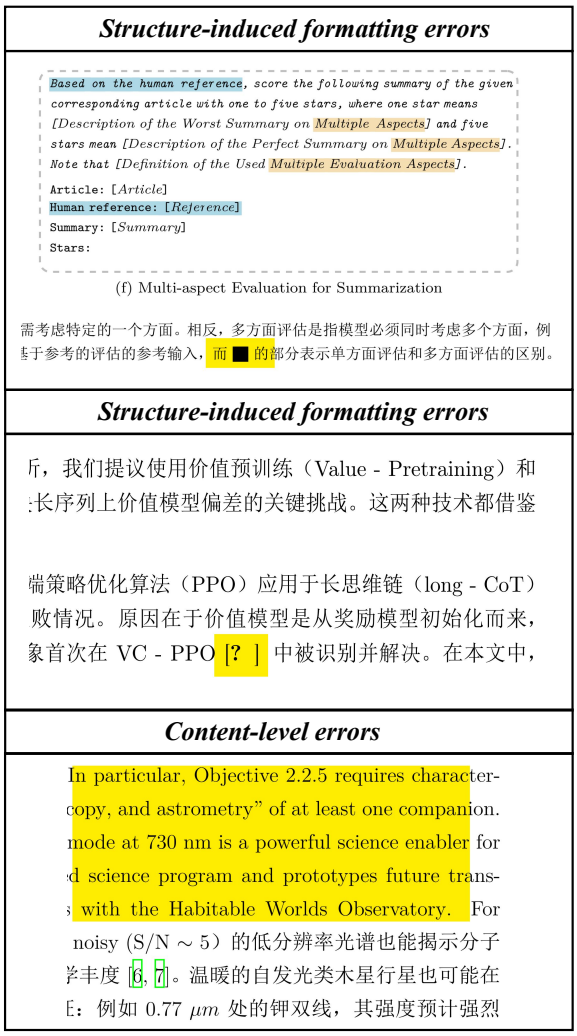


Figure 3: Examples of formatting errors.

C Formatting Error Categories

We categorize formatting-related errors into two major types:

- *Structure-induced formatting* errors, arising from corrupted or improperly restored LaTeX commands. These may result in missing formatting effects (e.g., bold text not rendered),

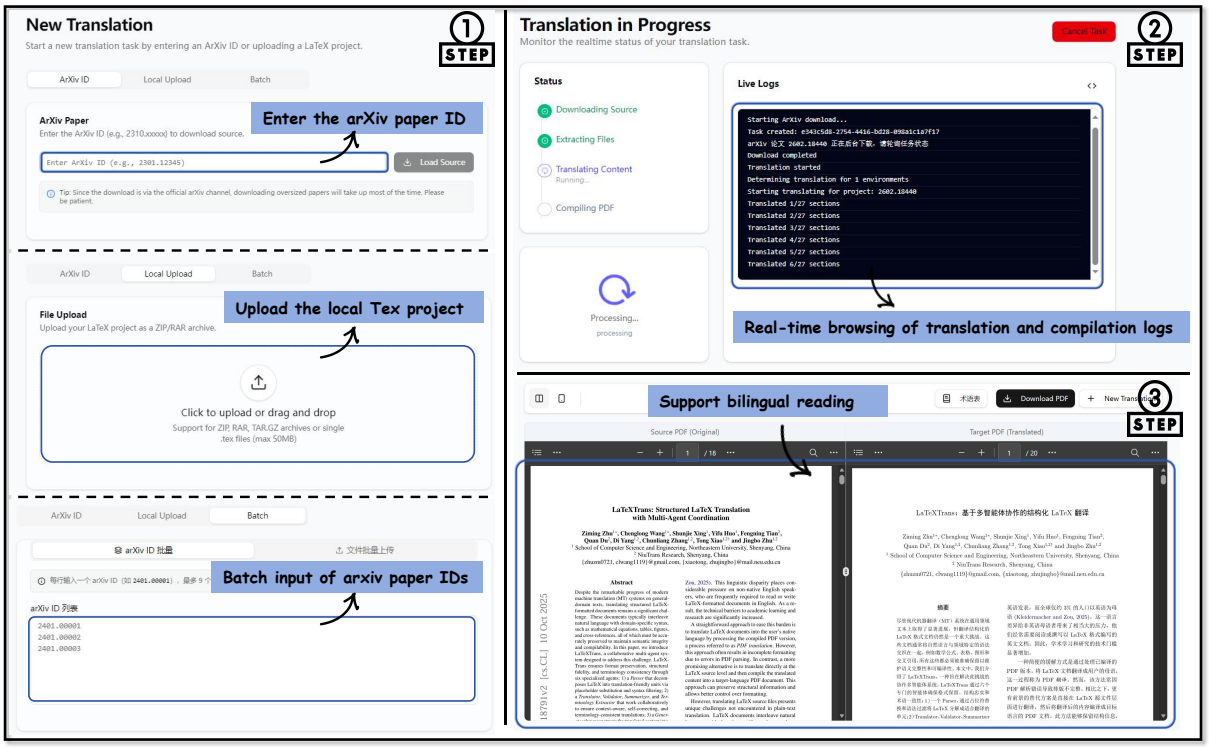


Figure 4: Illustration of the locally deployed LaTeXTrans interface and workflow.

incorrect cross-references, or garbled content in the compiled document.

- *Content-level errors*, involving incorrect or unintended textual outputs, such as untranslated segments or irrelevant words.

Figure 3 illustrates representative examples of formatting errors.

To ensure consistent evaluation, we adopt the following counting rules. For structure-induced errors, if a single corrupted LaTeX command causes multiple downstream anomalies (e.g., numerous missing cross-references triggered by one faulty reference definition), the error is counted only once per document. In contrast, content-level errors are counted independently at the instance level. This design prevents inflated error counts caused by cascading structural failures while preserving fine-grained sensitivity to semantic and textual inaccuracies.

D System Deployment and Usage

Command-Line Tool. LaTeXTrans supports local deployment through a command-line interface. By providing an arXiv paper ID, the system retrieves the associated LaTeX project and produces a translated PDF in a single command:

```
LaTeXTrans -arxiv xxxx.xxxxx
```

Upon completion, the system generates both the translated PDF and the complete translated LaTeX source project. Additionally, detailed translation and compilation logs are provided to facilitate debugging and ensure transparency.

Local Web Interface. In addition to the command-line tool, LaTeXTrans can be deployed locally with a web-based interface for interactive document translation. Users can submit LaTeX projects or arXiv paper IDs, monitor translation and compilation logs during execution, and inspect the generated bilingual documents through the interface. The workflow of the locally deployed interface is illustrated in Figure 4.

E System Performance Display

We select six cases to visually demonstrate the translation performance of LaTeXTrans, focusing on En-Zh and En-Ja translation tasks, as illustrated in Figure 5 to Figure 10. All six cases are translation cases of the LaTeX source code of papers by LaTeXTrans. In each case, we have selected two relatively complex parts to present. Among the six cases, there are three En-Zh translation tasks and three En-Ja translation tasks, respectively.

F Prompt Templates for LLM-Based Components in LaTeXTrans

Figures 12 through 16 show the prompt templates used within the LaTeXTrans.

3. We systematically integrate techniques from prior work, such as Clip-Higher and Token-level Loss from DAPO [29], Value-Pretraining and Decoupled-GAE from VC-PPO [30], self-imitation learning from SIL [14], and Group-Sampling from GRPO [22]. Additionally, we further validate their necessity through ablation studies.

VAPO is an effective reinforcement learning system that brings together these improvements. These enhancements work together smoothly, leading to a combined result that's better than the sum of the individual parts. We conduct experiments using the Qwen2.5-32B pre-trained model, ensuring no SFT data is introduced in any of the experiments, to maintain comparability with related works (DAPO and DeepSeek-R1-Zero-Qwen-32B). The performance of **VAPO** improves from vanilla PPO a score of 5 to 60, surpassing the previous SOTA value-model-free methods DAPO [29] by 10 points. More importantly, **VAPO** is highly stable — we don't observe any crashes during training, and the results across multiple runs are consistently similar.

2 Preliminaries

This section presents the fundamental concepts and notations that serve as the basis for our proposed algorithm. We first explore the basic framework of representing language generation as a reinforcement learning task. Subsequently, we introduce Proximal Policy Optimization and Generalized Advantage Estimation.

2.1 Modeling Language Generation as Token-Level MDP

Reinforcement learning centers around the learning of a policy that maximizes the cumulative reward for an agent as it interacts with an environment. In this study, we cast language generation tasks within the framework of a Markov Decision Process (MDP) [17].

Let the prompt be denoted as x, and the response to this prompt as y. Both x and y can be decomposed into sequences of tokens. For example, the prompt x can be expressed as $x = (x_0, \ldots, x_m)$, where the tokens are drawn from a fixed discrete vocabulary $\mathcal{A}$.

We define the token-level MDP as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, R, d_0, \omega)$. Here is a detailed breakdown of each component:

- **State Space ($\mathcal{S}$)**: This space encompasses all possible states formed by the tokens generated up to a given time step. At time step t, the state s_t is defined as $s_t = (x_0, \ldots, x_m, y_0, \ldots, y_t)$.
- **Action Space ($\mathcal{A}$)**: It corresponds to the fixed discrete vocabulary, from which tokens are selected during the generation process.
- **Dynamics ($\mathbb{P}$)**: These represent a deterministic transition model between tokens. Given a state $s_t = (x_0, \ldots, x_m, y_0, \ldots, y_t)$, an action $a = y_{t+1}$, and the subsequent state $s_{t+1} = (x_0, \ldots, x_m, y_0, \ldots, y_t, y_{t+1})$, the probability $\mathbb{P}(s_{t+1}|s_t, a) = 1$.
- **Termination Condition**: The language generation process concludes when the terminal action ω, typically the end-of-sentence token, is executed.
- **Reward Function ($R(s, a)$)**: This function offers scalar feedback to evaluate the agent's performance after taking action a in state s. In the context of Reinforcement Learning from Human Feedback (RLHF) [18, 23], the reward function can be learned from human preferences or defined by a set of rules specific to the task.
- **Initial State Distribution (d_0)**: It is a probability distribution over prompts x. An initial state s_0 consists of the tokens within the prompt x.

2.2 RLHF Learning Objective

We formulate the optimization problem as a KL-regularized RL task. Our objective is to approximate the optimal KL-regularized policy, which is given by:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi, s_0 \sim d_0} \left[\sum_{t=0}^{H} \left(R(s_t, a_t) - \beta \mathrm{KL}\left(\pi(\cdot|s_t) \| \pi_{\mathrm{ref}}(\cdot|s_t)\right)\right)\right] \tag{1}$$

(a) The first part of the English PDF of case 1.

GRPO [22] 的组采样。此外，我们通过消融研究进一步验证了它们的必要性。

VAPO 是一个有效的强化学习系统，将这些改进结合在一起。这些增强措施协同工作，导致合并结果优于各个部分的简单相加。我们使用Qwen2.5-32B预训练模型进行实验，确保在任何实验中都没有引入SFT数据，以保持与相关工作的可比性（DAPO 和 DeepSeek-R1-Zero-Qwen-32B）。**VAPO** 的性能从原始PPO的得分5提高到60，超过了之前的SOTA无价值模型方法DAPO [29] 10分。更重要的是，**VAPO** 高度稳定——我们在训练期间没有观察到任何崩溃，并且多次运行的结果始终相似。

2 预备知识

本节介绍作为我们所提算法基础的基本概念和符号。我们首先探讨将语言生成表示为强化学习任务的基本框架。随后，我们介绍近端策略优化和广义优势估计。

2.1 将语言生成建模为令牌级MDP

强化学习的核心是学习一种策略，使代理在与环境交互时最大化累积奖励。在本研究中，我们将语言生成任务置于马尔可夫决策过程（MDP）的框架内 [17]。

令提示表示为 x，对该提示的响应表示为 y。x 和 y 都可以分解为令牌序列。例如，提示 x 可以表示为 $x = (x_0, \ldots, x_m)$，其中令牌来自固定的离散词汇 $\mathcal{A}$。

我们将令牌级MDP定义为元组 $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, R, d_0, \omega)$。以下是每个组件的详细分解：

- **状态空间 ($\mathcal{S}$)**：此空间包含了在给定时间步之前生成的所有可能状态。在时间步 t，状态 s_t 定义为 $s_t = (x_0, \ldots, x_m, y_0, \ldots, y_t)$。
- **动作空间 ($\mathcal{A}$)**：它对应于固定的离散词汇表，从中选择生成过程中的标记。
- **动态模型 ($\mathbb{P}$)**：这些表示标记之间的确定性转换模型。给定状态 $s_t = (x_0, \ldots, x_m, y_0, \ldots, y_t)$，动作 $a = y_{t+1}$，以及后续状态 $s_{t+1} = (x_0, \ldots, x_m, y_0, \ldots, y_t, y_{t+1})$，则概率 $\mathbb{P}(s_{t+1}|s_t, a) = 1$。
- **终止条件**：语言生成过程在终止动作 ω 执行时结束，通常是句子结束标记。
- **奖励函数 ($R(s, a)$)**：此函数提供标量反馈，以评估智能体在状态 s 下执行动作 a 后的表现。在从人类反馈中进行强化学习（RLHF）[18, 23] 的背景下，奖励函数可以从人类偏好中学习，或通过特定任务的规则集定义。
- **初始状态分布 (d_0)**：这是一个关于提示 x 的概率分布。初始状态 s_0 包含提示 x 内的标记。

2.2 RLHF 学习目标

我们将优化问题表述为一个 KL 正则化的 RL 任务。我们的目标是逼近最优的 KL 正则化策略，其表示为：

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi, s_0 \sim d_0} \left[\sum_{t=0}^{H} \left(R(s_t, a_t) - \beta \mathrm{KL}\left(\pi(\cdot|s_t) \| \pi_{\mathrm{ref}}(\cdot|s_t)\right)\right)\right] \tag{1}$$

(b) The first part of the Chinese PDF of case 1.

In this equation, H represents the total number of decision steps, s_0 is a prompt sampled from the dataset, $R(s_t, a_t)$ is the token-level reward obtained from the reward function, β is a coefficient that controls the strength of the KL-regularization, and π_{ref} is the initialization policy.

In traditional RLHF and most tasks related to LLMs, the reward is sparse and is only assigned at the terminal action ω, that is, the end-of-sentence token `<eos>`.

2.3 Proximal Policy Optimization

PPO [21] uses a clipped surrogate objective to update the policy. The key idea is to limit the change in the policy during each update step, preventing large policy updates that could lead to instability.

Let $\pi_\theta(a|s)$ be the policy parameterized by θ, and $\pi_{\theta_{\mathrm{old}}}(a|s)$ be the old policy from the previous iteration. The surrogate objective function for PPO is defined as:

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the estimated advantage at time step t, and ϵ is a hyperparameter that controls the clipping range.

Generalized Advantage Estimation [20] is a technique used to estimate the advantage function more accurately in PPO. It combines multiple-step bootstrapping to reduce the variance of the advantage estimates. For a trajectory of length T, the advantage estimate $\hat{A}_t$ at time step t is computed as:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} \tag{3}$$

where γ is the discount factor, $\lambda \in [0, 1]$ is the GAE parameter, and $\delta_t = R(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$ is the temporal-difference (TD) error. Here, $R(s_t, a_t)$ is the reward at time step t, and $V(s)$ is the value function. Since it is a common practice to use discount factor $\gamma = 1.0$ in RLHF, to simplify our notation, we omit γ in later sections of this paper.

3 Challenges in Long-CoT RL for Reasoning Tasks

Long-CoT tasks present unique challenges to RL training, especially for methods that employ a value model to reduce variance. In this section, we systematically analyze the technical issues arising from sequence length dynamics, value function instability, and reward sparsity.

3.1 Value Model Bias over Long Sequences

As identified in VC-PPO [30], initializing the value model with a reward model introduces significant initialization bias. This positive bias arises from an objective mismatch between the two models. The reward model is trained to score on the `<EOS>` token, incentivizing it to assign lower scores to earlier tokens due to their incomplete context. In contrast, the value model estimates the expected cumulative reward for all tokens preceding `<EOS>` under a given policy. During early training phases, given the backward computation of GAE, there will be a positive bias at every timestep t that accumulates along the trajectory.

Another standard practice of using GAE with $\lambda = 0.95$ might exacerbates this issue. The reward signal $R(s_T, \text{<EOS>})$ at the termination token propagates backward as $\lambda^{T-t} R(s_T, \text{<EOS>})$ to the t-th token. For long sequences where $T - t \gg 1$, this discounting reduces the effective reward signal to near zero. Consequently, value updates become almost entirely bootstrapped, relying on highly biased estimates that undermine the value model's role as a reliable variance-reduction baseline.

(c) The second part of the English PDF of case 1.

在此方程中，H 表示决策步骤的总数，s_0 是从数据集中采样的提示，$R(s_t, a_t)$ 是从奖励函数中获得的基于 token 的奖励，β 是控制 KL 正则化强度的系数，而 π_{ref} 是初始化策略。

在传统的 RLHF 和大多数与 LLM 相关的任务中，奖励是稀疏的，仅在终端动作 ω，即句子结束 token `<eos>` 时分配。

2.3 近端策略优化

PPO [21] 使用截断的替代目标来更新策略。其关键思想是在每次更新步骤中限制策略的变化，防止过大的策略更新导致不稳定。

设 $\pi_\theta(a|s)$ 为参数化为 θ 的策略，$\pi_{\theta_{\mathrm{old}}}(a|s)$ 为上一迭代中的旧策略。PPO 的替代目标函数定义为：

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

其中 $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ 是概率比，$\hat{A}_t$ 是时间步 t 的估计优势，ϵ 是控制截断范围的超参数。

广义优势估计 [20] 是一种在 PPO 中用来更准确估计优势函数的技术。它结合了多步引导来减少优势估计的方差。对于长度为 T 的轨迹，时间步 t 的优势估计 $\hat{A}_t$ 计算为：

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} \tag{3}$$

其中 γ 是折扣因子，$\lambda \in [0, 1]$ 是 GAE 参数，$\delta_t = R(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$ 是时序差分（TD）误差。这里，$R(s_t, a_t)$ 是时间步 t 的奖励，$V(s)$ 是价值函数。由于在 RLHF 中常用的做法是使用折扣因子 $\gamma = 1.0$，为简化记号，本文后续部分将省略 γ。

3 长链式思维路径强化学习在推理任务中的挑战

长链式思维路径任务对强化学习训练带来了独特的挑战，特别是对于使用价值模型来减少方差的方法。在本节中，我们系统地分析了由序列长度动态、价值函数不稳定性和奖励稀疏性引发的技术问题。

3.1 长序列上的价值模型偏差

如VC-PPO中所指出的 [30]，用奖励模型初始化价值模型会引入显著的初始化偏差。这种正偏差来源于两个模型之间的目标不匹配。奖励模型被训练在`<EOS>`标记上打分，激励其对较早的标记给予较低分数，因为这些标记的上下文不完整。相比之下，价值模型估计在给定策略下`<EOS>`之前所有标记的预期累计奖励。在训练初期阶段，由于GAE的反向计算，每个时间步t都会存在一个正偏差，并沿着轨迹累积。

使用$\lambda = 0.95$的GAE的另一种常见做法可能会加剧这一问题。在终止标记处的奖励信号$R(s_T, \text{<EOS>})$向后传播为$\lambda^{T-t} R(s_T, \text{<EOS>})$到第$t$个标记。对于长序列而言，当$T - t \gg 1$时，这种折扣会将有效的奖励信号降低到接近于零。因此，价值更新几乎完全依赖于高度有偏差的估计，削弱了价值模型作为可靠的方差降低基线的作用。

(d) The second part of the Chinese PDF of case 1.

Figure 5: Case 1 demonstrates the performance of LaTeXTrans on the En-Zh task

1. Introduction

In recent years, Large Language Models (LLMs) have been undergoing rapid iteration and evolution (Anthropic, 2024; Google, 2024; OpenAI, 2024a), progressively diminishing the gap towards Artificial General Intelligence (AGI).

Recently, post-training has emerged as an important component of the full training pipeline. It has been shown to enhance accuracy on reasoning tasks, align with social values, and adapt to user preferences, all while requiring relatively minimal computational resources against pre-training. In the context of reasoning capabilities, OpenAI's o1 (OpenAI, 2024b) series models were the first to introduce inference-time scaling by increasing the length of the Chain-of-Thought reasoning process. This approach has achieved significant improvements in various reasoning tasks, such as mathematics, coding, and scientific reasoning. However, the challenge of effective test-time scaling remains an open question for the research community. Several prior works have explored various approaches, including process-based reward models (Lightman et al., 2023; Uesato et al., 2022; Wang et al., 2023), reinforcement learning (Kumar et al., 2024), and search algorithms such as Monte Carlo Tree Search and Beam Search (Feng et al., 2024; Trinh et al., 2024; Xin et al., 2024). However, none of these methods has achieved general reasoning performance comparable to OpenAI's o1 series models.

In this paper, we take the first step toward improving language model reasoning capabilities using pure reinforcement learning (RL). Our goal is to explore the potential of LLMs to develop reasoning capabilities without any supervised data, focusing on their self-evolution through a pure RL process. Specifically, we use DeepSeek-V3-Base as the base model and employ GRPO (Shao et al., 2024) as the RL framework to improve model performance in reasoning. During training, DeepSeek-R1-Zero naturally emerged with numerous powerful and interesting reasoning behaviors. After thousands of RL steps, DeepSeek-R1-Zero exhibits super performance on reasoning benchmarks. For instance, the pass@1 score on AIME 2024 increases from 15.6% to 71.0%, and with majority voting, the score further improves to 86.7%, matching the performance of OpenAI-o1-0912.

However, DeepSeek-R1-Zero encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates a small amount of cold-start data and a multi-stage training pipeline. Specifically, we begin by collecting thousands of cold-start data to fine-tune the DeepSeek-V3-Base model. Following this, we perform reasoning-oriented RL like DeepSeek-R1-Zero. Upon nearing convergence in the RL process, we create new SFT data through rejection sampling on the RL checkpoint, combined with supervised data from DeepSeek-V3 in domains such as writing, factual QA, and self-cognition, and then retrain the DeepSeek-V3-Base model. After fine-tuning with the new data, the checkpoint undergoes an additional RL process, taking into account prompts from all scenarios. After these steps, we obtained a checkpoint referred to as DeepSeek-R1, which achieves performance on par with OpenAI-o1-1217.

We further explore distillation from DeepSeek-R1 to smaller dense models. Using Qwen2.5-32B (Qwen, 2024b) as the base model, direct distillation from DeepSeek-R1 outperforms applying RL on it. This demonstrates that the reasoning patterns discovered by larger base models are crucial for improving reasoning capabilities. We open-source the distilled Qwen and Llama (Dubey et al., 2024) series. Notably, our distilled 14B model outperforms state-of-the-art open-source QwQ-32B-Preview (Qwen, 2024a) by a large margin, and the distilled 32B and 70B models set a new record on the reasoning benchmarks among dense models.

(a) The first part of the English PDF of case 2.

1. 介绍

近年来，大型语言模型（LLMs）正在经历快速迭代和演变 (Anthropic, 2024; Google, 2024; OpenAI, 2024a)，逐步缩小与人工通用智能（AGI）的差距。

最近，后训练已成为完整训练流程中的重要组成部分。研究表明，它能够增强推理任务的准确性，符合社会价值观，并适应用户偏好，同时与预训练相比需要相对较少的计算资源。在推理能力的背景下，OpenAI 的 o1 系列模型首次通过增加链式思维推理过程的长度，引入了推理时间缩放。这种方法在各种推理任务中取得了显著改进，如数学、编码和科学推理。然而，有效的测试时间缩放仍然是研究界的一个开放问题。一些先前的工作探索了各种方法，包括基于过程的奖励模型 (Lightman et al., 2023; Uesato et al., 2022; Wang et al., 2023)、强化学习 (Kumar et al., 2024) 和搜索算法，如蒙特卡洛树搜索和波束搜索 (Feng et al., 2024; Trinh et al., 2024; Xin et al., 2024)。然而，这些方法都未能实现与 OpenAI 的 o1 系列模型相媲美的通用推理性能。

在本文中，我们迈出了使用纯强化学习（RL）提高语言模型推理能力的第一步。我们的目标是探索 LLMs 在不使用任何监督数据的情况下发展推理能力的潜力，重点在于通过纯 RL 过程进行自我进化。具体来说，我们使用 DeepSeek-V3-Base 作为基础模型，并采用 GRPO (Shao et al., 2024) 作为 RL 框架来提高模型在推理中的性能。在训练过程中，DeepSeek-R1-Zero 自然涌现出许多强大且有趣的推理行为。经过数千次 RL 步骤后，DeepSeek-R1-Zero 在推理基准测试中表现出超强性能。例如，AIME 2024 的 pass@1 得分从 15.6% 提高到 71.0%，并且通过多数投票，得分进一步提高到 86.7%，与 OpenAI-o1-0912 的性能相当。

然而，DeepSeek-R1-Zero 遇到了一些挑战，如可读性差和语言混合。为了解决这些问题并进一步提高推理性能，我们引入了 DeepSeek-R1，其中包含少量冷启动数据和多阶段训练流程。具体来说，我们首先收集数千条冷启动数据来微调 DeepSeek-V3-Base 模型。随后，我们进行类似于 DeepSeek-R1-Zero 的推理导向 RL。当 RL 过程接近收敛时，我们通过 RL 检查点上的拒绝抽样创建新的 SFT 数据，并结合 DeepSeek-V3 在写作、事实问答和自我认知等领域的监督数据，然后重新训练 DeepSeek-V3-Base 模型。在用新数据微调后，检查点经过额外的 RL 过程，考虑所有场景的提示。经过这些步骤，我们获得了一个被称为 DeepSeek-R1 的检查点，其性能与 OpenAI-o1-1217 相当。

我们进一步探索了从 DeepSeek-R1 到更小的密集模型的蒸馏。使用 Qwen2.5-32B (Qwen, 2024b) 作为基础模型，直接从 DeepSeek-R1 蒸馏优于在其上应用 RL。这表明较大基础模型发现的推理模式对于提高推理能力至关重要。我们开源了蒸馏的 Qwen 和 Llama (Dubey et al., 2024) 系列。值得注意的是，我们蒸馏的 14B 模型在推理基准测试中远超最先进的开源 QwQ-32B-Preview (Qwen, 2024a)，蒸馏的 32B 和 70B 模型在密集模型中创下了推理基准测试的新纪录。

(b) The first part of the Chinese PDF of case 2.

- **Others**: DeepSeek-R1 also excels in a wide range of tasks, including creative writing, general question answering, editing, summarization, and more. It achieves an impressive length-controlled win-rate of 87.6% on AlpacaEval 2.0 and a win-rate of 92.3% on ArenaHard, showcasing its strong ability to intelligently handle non-exam-oriented queries. Additionally, DeepSeek-R1 demonstrates outstanding performance on tasks requiring long-context understanding, substantially outperforming DeepSeek-V3 on long-context benchmarks.

2. Approach

2.1. Overview

Previous work has heavily relied on large amounts of supervised data to enhance model performance. In this study, we demonstrate that reasoning capabilities can be significantly improved through large-scale reinforcement learning (RL), even without using supervised fine-tuning (SFT) as a cold start. Furthermore, performance can be further enhanced with the inclusion of a small amount of cold-start data. In the following sections, we present: (1) DeepSeek-R1-Zero, which applies RL directly to the base model without any SFT data, and (2) DeepSeek-R1, which applies RL starting from a checkpoint fine-tuned with thousands of long Chain-of-Thought (CoT) examples. 3) Distill the reasoning capability from DeepSeek-R1 to small dense models.

2.2. DeepSeek-R1-Zero: Reinforcement Learning on the Base Model

Reinforcement learning has demonstrated significant effectiveness in reasoning tasks, as evidenced by our previous works (Shao et al., 2024; Wang et al., 2023). However, these works heavily depended on supervised data, which are time-intensive to gather. In this section, we explore the potential of LLMs to develop reasoning capabilities **without any supervised data**, focusing on their self-evolution through a pure reinforcement learning process. We start with a brief overview of our RL algorithm, followed by the presentation of some exciting results, and hope this provides the community with valuable insights.

2.2.1. *Reinforcement Learning Algorithm*

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \quad (1)$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (2)$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \quad (3)$$

(c) The second part of the English PDF of case 2.

- **其他**: DeepSeek-R1 在广泛的任务中也表现出色，包括创意写作、一般问答、编辑、总结等。在 AlpacaEval 2.0 上取得了 87.6% 的长度控制胜率，在 ArenaHard 上取得了 92.3% 的胜率，展现了其智能处理非考试导向查询的强大能力。此外，DeepSeek-R1 在需要长上下文理解的任务中表现出色，在长上下文基准测试中显著超越了 DeepSeek-V3。

2. 方法

2.1. 概述

以往的研究大量依赖于监督数据来提升模型性能。在本研究中，我们展示了即使不使用监督微调（SFT）作为冷启动，通过大规模强化学习（RL）可以显著提高推理能力。此外，加入少量冷启动数据可以进一步提升性能。在接下来的部分中，我们介绍：(1) DeepSeek-R1-Zero，它直接将 RL 应用于基础模型而不使用任何 SFT 数据，(2) DeepSeek-R1，它从经过数千个长链式思维（CoT）示例微调的检查点开始应用 RL，(3) 将 DeepSeek-R1 的推理能力提炼到小型稠密模型中。

2.2. DeepSeek-R1-Zero：基础模型上的强化学习

强化学习在推理任务中显示出显著的效果，这在我们之前的工作中已经得到了证实 (Shao et al., 2024; Wang et al., 2023)。然而，这些工作在很大程度上依赖于监督数据，而这些数据的收集非常耗时。在本节中，我们探索 LLMs 在 **没有任何监督数据**的情况下发展推理能力的潜力，重点关注它们通过纯粹的强化学习过程实现自我进化。我们首先简要概述我们的 RL 算法，然后介绍一些令人兴奋的结果，并希望这能为社区提供有价值的见解。

2.2.1. 强化学习算法

群组相对策略优化 为了节省强化学习的训练成本，我们采用群组相对策略优化（GRPO）(Shao et al., 2024)，该方法放弃了通常与策略模型大小相同的评论模型，而是通过群组评分来估计基线。具体来说，对于每个问题 q，GRPO 从旧策略 $\pi_{\theta_{old}}$ 中采样一组输出 $\{o_1, o_2, \cdots, o_G\}$，然后通过最大化以下目标来优化策略模型 π_θ：

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \quad (1)$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (2)$$

其中 ε 和 β 是超参数，A_i 是优势，通过使用对应于每个群组内输出的一组奖励 $\{r_1, r_2, \ldots, r_G\}$ 来计算：

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \quad (3)$$

(d) The second part of the Chinese PDF of case 2.

Figure 6: Case 2 demonstrates the performance of LaTeXTrans on the En-Zh task

In the multi-snapshot scenario, the forward T-measurement process is described as

$$\boldsymbol{Y} = \boldsymbol{A} \cdot \boldsymbol{S} + \boldsymbol{N}, \tag{9}$$

where $\boldsymbol{Y} = [\boldsymbol{y}(1), \cdots, \boldsymbol{y}(T)] \in \mathbb{C}^{M\times T}$ is the matrix of received signals across T time snapshots, $\boldsymbol{S} = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_T] \in \mathbb{C}^{L\times T}$ denotes the source signal matrix, and $\boldsymbol{N} \in \mathbb{C}^{M\times T}$ represents the noise.

B. Classical 2D MUSIC Algorithm

The MUSIC algorithm is widely used for AoA estimation through eigenvalue decomposition. Based on the model in (9), the covariance matrix of the received signals is given by

$$\begin{aligned}\boldsymbol{R} &= E[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{H}}] \\ &= \boldsymbol{A}\boldsymbol{R}_{ss}\boldsymbol{A}^{\mathrm{H}} + \sigma^2\boldsymbol{I},\end{aligned} \tag{10}$$

where $\boldsymbol{R}_{ss} = E[\boldsymbol{S}\boldsymbol{S}^{\mathrm{H}}]$ is the correlation matrix of the source signals. The eigenvectors of $\boldsymbol{R}$ associated with the largest D eigenvalues span the signal subspace $\boldsymbol{E}_S$, while the remaining eigenvectors span the noise subspace $\boldsymbol{E}_N$. The 2D MUSIC AoA pseudo-spectrum is defined as

$$\boldsymbol{P_M}(\theta,\phi) = \frac{\boldsymbol{a}^H(\theta,\phi)\boldsymbol{a}(\theta,\phi)}{\boldsymbol{a}^H(\theta,\phi)\boldsymbol{E}_N\boldsymbol{E}_N^H\boldsymbol{a}(\theta,\phi)}. \tag{11}$$

The angles corresponding to the peaks in this pseudo-spectrum provide estimates of the directions of the incident signals.

C. 1-SSMUSIC for 3D AoA

In contrast to 2D AoA estimation, 3D AoA estimation demands significantly higher computational complexity. Moreover, 3D localization tasks are further challenged by the increased severity of multipath propagation. A well-known limitation of subspace-based methods is their degraded performance in the presence of correlated sources, primarily due to rank deficiency in the covariance matrix. A notable solution to mitigate this issue is the spatial smoothing technique.

We now present an improved MUSIC algorithm with 2D spatial smoothing, referred to as 1-SSMUSIC, designed for URAs. Based on (9), the (m_1, m_2)-th smoothed subarrays of size $M_1 \times M_2$ is formally expressed as

$$\boldsymbol{Y}_{m_1m_2} = \boldsymbol{A}_1\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\cdot\boldsymbol{S} + \boldsymbol{N}_{m_1m_2}, \tag{12}$$

where

$$\begin{aligned}\boldsymbol{D}_x &= \mathrm{diag}[u(\theta_1,\phi_1),\cdots,u(\theta_L,\phi_L)], \\ \boldsymbol{D}_y &= \mathrm{diag}[v(\theta_1,\phi_1),\cdots,v(\theta_L,\phi_L)].\end{aligned} \tag{13}$$

Here $\boldsymbol{N}_{m_1m_2}$ is the noise matrix at the (m_1, m_2)-th subarray and $\boldsymbol{A}_1 = [\boldsymbol{a}_1(\theta_1,\phi_1)\ \boldsymbol{a}_1(\theta_2,\phi_2)\ \cdots\ \boldsymbol{a}_1(\theta_L,\phi_L)]$ is the steering matrix, where each $\boldsymbol{a}_1(\theta_l,\phi_l)$ is given by

$$\begin{aligned}\boldsymbol{a}_1(\theta_l,\phi_l) &= \boldsymbol{a}_{y,M_1}(\theta_l,\phi_l)\otimes\boldsymbol{a}_{x,M_1}(\theta_l,\phi_l), \\ \boldsymbol{a}_{x,M_1}(\theta,\phi) &= \left[1\quad u\quad\cdots\quad u^{M_1-1}\right]^\top, \\ \boldsymbol{a}_{y,M_2}(\theta,\phi) &= \left[1\quad v\quad\cdots\quad v^{M_2-1}\right]^\top.\end{aligned} \tag{14}$$

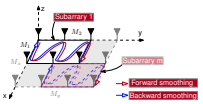


Fig. 6. 1-SSMUSIC of URA with forward-backward spatial smoothing applied to each subarray.

Using (12), we can reformulate the expression in (10). The covariance matrix of the (m_1, m_2)-th subarray is therefore given by

$$\begin{aligned}\boldsymbol{R}_{m_1m_2}^f = \boldsymbol{A}_1\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\boldsymbol{R}_{ss}\left(\boldsymbol{D}_y^{m_2-1}\right)^{\mathrm{H}} \\ \times\left(\boldsymbol{D}_x^{m_1-1}\right)^{\mathrm{H}}\boldsymbol{A}_1^{\mathrm{H}} + \sigma^2\boldsymbol{I}.\end{aligned} \tag{15}$$

In the spatial smoothing scheme, the forward smoothed covariance matrix $\boldsymbol{R}^f$ is obtained by averaging the covariance matrices of all forward subarrays, yielding

$$\boldsymbol{R}^f = \frac{1}{H_xH_y}\sum_{m_1=1}^{H_x}\sum_{m_2=1}^{H_y}\boldsymbol{R}_{m_1m_2}^f = \boldsymbol{A}_1\boldsymbol{R}_{ss}^f\boldsymbol{A}_1^{\mathrm{H}} + \sigma^2\boldsymbol{I}, \tag{16}$$

where $H_x = M_x - M_1 + 1$ and $H_y = M_y - M_2 + 1$. Similarly, we denote the forward-smoothed source covariance matrix by $\boldsymbol{R}_{ss}^f$, which is defined by

$$\begin{aligned}\boldsymbol{R}_{ss}^f = \frac{1}{H_xH_y}\sum_{m_1=1}^{H_x}\sum_{m_2=1}^{H_y}\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\boldsymbol{R}_{ss} \\ \times\left(\boldsymbol{D}_y^{m_2-1}\right)^{\mathrm{H}}\left(\boldsymbol{D}_x^{m_1-1}\right)^{\mathrm{H}}.\end{aligned} \tag{17}$$

The spatially smoothed covariance matrix enables the application of eigenstructure-based methods for AoA estimation, even in the presence of coherent signals.

One limitation of the spatial smoothing algorithm is its tendency to reduce the effective array aperture, which may degrade sensing performance [17]. To mitigate this issue, we introduce a forward-backward spatial smoothing scheme for URAs, as illustrated in Fig. 6. This bidirectional smoothing approach preserves the aperture size by exploiting the conjugate symmetry property of the covariance matrix.

Mathematically, the forward-backward spatially smoothed covariance matrix is expressed as

$$\boldsymbol{R_X} = \frac{1}{2}\left(\boldsymbol{R}^f + \boldsymbol{I_v}\left(\boldsymbol{R}^f\right)^*\boldsymbol{I_v}\right), \tag{18}$$

where $\left(\boldsymbol{R}^f\right)^*$ is the conjugate for matrix $\boldsymbol{R}^f$, and

$$\boldsymbol{I_v} = \begin{bmatrix}0 & \cdots & 0 & 1\\ 0 & \cdots & 1 & 0\\ \vdots & \iddots & \vdots & \vdots\\ 1 & \cdots & 0 & 0\end{bmatrix}_{M\times M}. \tag{19}$$

By computing the pseudo-spectrum in (11) using this smoothed covariance matrix, we enable accurate estimation of the form

$$\begin{cases}\boldsymbol{r}_1 = \boldsymbol{c}_1 + t_1\boldsymbol{d}_1, \\ \quad\vdots \\ \boldsymbol{r}_i = \boldsymbol{c}_i + t_i\boldsymbol{d}_i, \\ \quad\vdots \\ \boldsymbol{r}_n = \boldsymbol{c}_n + t_n\boldsymbol{d}_n,\end{cases} \tag{20}$$

where $\boldsymbol{c}_i \in \mathbb{R}^3$ denotes the center of the i-th URA, and $\boldsymbol{d}_i \in \mathbb{R}^3$ is the direction vector of the arrival ray l_i. To identify the vector $\mathbf{t}_{h,i} = [t_h\ t_i]^\top$ that best approximates the intersection of the h-th and i-th AoA rays, we solve the following equation

$$\begin{bmatrix}-\mathbf{d}_h^\top\mathbf{d}_h & \mathbf{d}_h^\top\mathbf{d}_h\\ -\mathbf{d}_h^\top\mathbf{d}_i & \mathbf{d}_i^\top\mathbf{d}_i\end{bmatrix}\begin{bmatrix}t_h\\ t_i\end{bmatrix} = \begin{bmatrix}(\mathbf{c}_h-\mathbf{c}_i)\cdot\mathbf{d}_i\\ (\mathbf{c}_h-\mathbf{c}_i)\cdot\mathbf{d}_h\end{bmatrix}. \tag{21}$$

If there is no exact intersection, the least-squares solution $\mathbf{t}^* = [t_{h,i}^*\ t_{i,h}^*]^\top$ determines the pair of closest points on the two rays. The coordinates of these points are given by

$$\mathbf{c}_{h,i}^* = \mathbf{c}_h + t_{h,i}^*\mathbf{d}_h, \mathbf{c}_{i,h}^* = \mathbf{c}_i + t_{i,h}^*\mathbf{d}_i. \tag{22}$$

Finally, the estimated position of the source based on all u URAs is computed as

$$\mathbf{c}^* = \frac{1}{u(u-1)}\sum_{h=1}^{u-1}\sum_{i=h+1}^{u}\left(\mathbf{c}_{h,i}^* + \mathbf{c}_{i,h}^*\right) = [x^*, y^*, z^*]. \tag{23}$$

IV. Collaborative 3D Direct Position Determination

For the previously described closest geometric point approach, collaboration is performed at the level of estimated AoAs, as the involved URAs are not synchronized with each other. Given that signal synchronization among elements within each array has now been implemented, a natural question arises: can this synchronization mechanism be further extended to the inter-array level to enable greater cooperative gains? In this section, we develop an inter-array synchronization framework designed to facilitate direct position determination (DPD) [38], [39]. Unlike the preceding closest point estimation method, DPD bypasses intermediate parameter estimation, such as AoA, and instead computes the source position directly in a single step.

To reduce the spatial sampling overhead of the proposed DPD algorithm, we first employ the 1-SSMUSIC and closest point estimation approaches to define a compact localized space of interest (LSoI). By discretizing the LSoI, we derive a measurement model that characterizes the observation process across multiple synchronized URAs. Synchronization among these arrays is achieved by measuring phase differences relative to a common reference signal. Once synchronization is established, the distributed URAs effectively form a virtual large-scale array, enabling the computation of the MUSIC pseudo-spectrum at the spatial sampling points within the LSoI. To further expand the LSoI and enhance estimation fidelity, we introduce a progressive local traversal strategy. The overall process is illustrated in Fig. 8.

For simplicity, the LSoI is configured as a sphere of radius R, centered at the closest geometric point $\boldsymbol{c}^*$ estimated via the 1-SSMUSIC algorithm. The sphere is discretized with a voxel

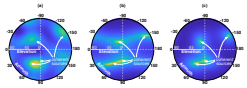


Fig. 7. Spatial spectrum of four correlated sources generated using a 3 × 4 antenna array. (a), (b) and (c) show the 2D spatial spectrums computed by MUSIC, SS-MUSIC and 1-SSMUSIC, respectively.

of correlated signals while mitigating the effects of rank deficiency.

By examining (16) and (18), we observe that the number of forward-only smoothed subarrays, denoted by H, determines the maximum number of resolvable correlated sources, whereas forward-backward smoothing effectively doubles this limit to $2H$. In typical indoor environments, where the number of multipath components is usually fewer than five [13], [17], a single forward-backward smoothing operation ($H = 2$) can decorrelate signals from up to four distinct angles.

We now present a comparative evaluation of conventional MUSIC, MUSIC with forward-only spatial smoothing (SS-MUSIC), and the proposed 1-SSMUSIC for estimating the angles of four correlated signals under identical conditions. The URA consists of 3×4 antennas. Four correlated signal sources emit continuous signals with an SNR of 15 dB, arriving from the following angles: $(21.8^\circ, 90^\circ)$, $(32^\circ, 56^\circ)$, $(15^\circ, -60^\circ)$ and $(60^\circ, -150^\circ)$, respectively. The spatial spectra are illustrated in Fig. 7, from which it is evident that the proposed 1-SSMUSIC outperforms the other methods. The estimated AoAs using 1-SSMUSIC are $(21.8^\circ, 90.8^\circ)$, $(32.4^\circ, 57.2^\circ)$, $(16.4^\circ, -59.6^\circ)$ and $(60.2^\circ, -150.6^\circ)$, respectively. In comparison, while SS-MUSIC is capable of estimating correlated signals, it exhibits notably lower resolution. Its estimated AoAs are $(22.8^\circ, 82.2^\circ)$, $(37.2^\circ, 50.8^\circ)$, $(15.2^\circ, -62^\circ)$ and $(58.8^\circ, -149.6^\circ)$. The standard MUSIC algorithm, by contrast, fails to resolve the correlated sources, resulting in an ambiguous and inaccurate AoA spectrum.

D. Closest Geometric Point Estimation

With AoA estimations obtained from multiple URAs distributed across space, the specific location of the signal source can be determined. Ideally, the estimated AoA vectors intersect at the true position of the source. However, due to measurement errors, a robust closest-point estimation algorithm is required to approximate the actual point of intersection. The proposed geometric positioning (GP) method first identifies the closest points between each pair of AoAs, as illustrated in Stage 1 of Fig. 8. The final position estimate is then computed as the mean of these closest points.

Let l_i denote the estimated arrival ray associated with the i-th URA. Each ray can be represented by a parametric equation

(a) The first part of the English PDF of case 3.

每列对应一个不同的到达方向。向量$\boldsymbol{s}(t) \in \mathbb{C}^{L\times1}$包含源信号，$\boldsymbol{\epsilon}(t)$是零均值方差为$\sigma^2$的复高斯噪声向量。

在多快照场景中，前向T测量过程描述为

$$\boldsymbol{Y} = \boldsymbol{A} \cdot \boldsymbol{S} + \boldsymbol{N}, \tag{9}$$

其中$\boldsymbol{Y} = [\boldsymbol{y}(1), \cdots, \boldsymbol{y}(T)] \in \mathbb{C}^{M\times T}$是跨$T$个时间快照的接收信号矩阵，$\boldsymbol{S} = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_T] \in \mathbb{C}^{L\times T}$表示源信号矩阵，$\boldsymbol{N} \in \mathbb{C}^{M\times T}$代表噪声。

B. 经典二维MUSIC算法

MUSIC算法广泛用于通过特征值分解进行到达角估计。基于模型 (9)，接收到的信号的协方差矩阵表示为

$$\begin{aligned}\boldsymbol{R} &= E[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{H}}] \\ &= \boldsymbol{A}\boldsymbol{R}_{ss}\boldsymbol{A}^{\mathrm{H}} + \sigma^2\boldsymbol{I},\end{aligned} \tag{10}$$

其中 $\boldsymbol{R}_{ss} = E[\boldsymbol{S}\boldsymbol{S}^{\mathrm{H}}]$ 为源信号的相关矩阵。与最大 D 个特征值相关的 $\boldsymbol{R}$ 的特征向量组成信号子空间 $\boldsymbol{E}_S$，而其余特征向量组成噪声子空间 $\boldsymbol{E}_N$。二维MUSIC到达角伪谱定义为

$$\boldsymbol{P_M}(\theta,\phi) = \frac{\boldsymbol{a}^H(\theta,\phi)\boldsymbol{a}(\theta,\phi)}{\boldsymbol{a}^H(\theta,\phi)\boldsymbol{E}_N\boldsymbol{E}_N^H\boldsymbol{a}(\theta,\phi)}. \tag{11}$$

该伪谱中峰值对应的角度提供了入射信号方向的估计。

C. 用于三维到达角的1-SSMUSIC

与二维到达角估计相比，三维到达角估计需要显著更高的计算复杂度。此外，三维定位任务还面临多径传播严重性的增加。一种众所周知的基于子空间的方法的局限性是在存在相关源的情况下性能下降，主要是由于协方差矩阵的秩不足。一个显著的解决方案是空间平滑技术。

我们现在介绍一种改进的MUSIC算法，采用二维空间平滑，称为1-SSMUSIC，专为URA设计。基于 (9)，大小为$M_1 \times M_2$的第(m_1, m_2)个平滑子阵形式上表示为

$$\boldsymbol{Y}_{m_1m_2} = \boldsymbol{A}_1\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\cdot\boldsymbol{S} + \boldsymbol{N}_{m_1m_2}, \tag{12}$$

其中

$$\begin{aligned}\boldsymbol{D}_x &= \mathrm{diag}[u(\theta_1,\phi_1),\cdots,u(\theta_L,\phi_L)], \\ \boldsymbol{D}_y &= \mathrm{diag}[v(\theta_1,\phi_1),\cdots,v(\theta_L,\phi_L)].\end{aligned} \tag{13}$$

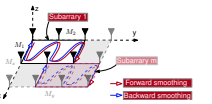


图 6. 对每个子阵列应用前后空间平滑的 URA 的 1-SSMUSIC。

这里$\boldsymbol{N}_{m_1m_2}$是第(m_1, m_2)个子阵的噪声矩阵，$\boldsymbol{A}_1 = [\boldsymbol{a}_1(\theta_1,\phi_1)\ \boldsymbol{a}_1(\theta_2,\phi_2)\ \cdots\ \boldsymbol{a}_1(\theta_L,\phi_L)]$是导向矩阵，其中每个$\boldsymbol{a}_1(\theta_l,\phi_l)$由以下公式给出

$$\begin{aligned}\boldsymbol{a}_1(\theta_l,\phi_l) &= \boldsymbol{a}_{y,M_1}(\theta_l,\phi_l)\otimes\boldsymbol{a}_{x,M_1}(\theta_l,\phi_l), \\ \boldsymbol{a}_{x,M_1}(\theta,\phi) &= \left[1\quad u\quad\cdots\quad u^{M_1-1}\right]^\top, \\ \boldsymbol{a}_{y,M_2}(\theta,\phi) &= \left[1\quad v\quad\cdots\quad v^{M_2-1}\right]^\top.\end{aligned} \tag{14}$$

使用 (12)，我们可以重新表述 (10)中的表达式。因此，第(m_1, m_2)个子阵的协方差矩阵为

$$\begin{aligned}\boldsymbol{R}_{m_1m_2}^f = \boldsymbol{A}_1\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\boldsymbol{R}_{ss}\left(\boldsymbol{D}_y^{m_2-1}\right)^{\mathrm{H}} \\ \times\left(\boldsymbol{D}_x^{m_1-1}\right)^{\mathrm{H}}\boldsymbol{A}_1^{\mathrm{H}} + \sigma^2\boldsymbol{I}.\end{aligned} \tag{15}$$

在空间平滑方案中，前向平滑协方差矩阵$\boldsymbol{R}^f$通过平均所有前向子阵的协方差矩阵获得，得到

$$\boldsymbol{R}^f = \frac{1}{H_xH_y}\sum_{m_1=1}^{H_x}\sum_{m_2=1}^{H_y}\boldsymbol{R}_{m_1m_2}^f = \boldsymbol{A}_1\boldsymbol{R}_{ss}^f\boldsymbol{A}_1^{\mathrm{H}} + \sigma^2\boldsymbol{I}, \tag{16}$$

其中$H_x = M_x - M_1 + 1$和$H_y = M_y - M_2 + 1$。类似地，我们将前向平滑源协方差矩阵记为$\boldsymbol{R}_{ss}^f$，其定义为

$$\begin{aligned}\boldsymbol{R}_{ss}^f = \frac{1}{H_xH_y}\sum_{m_1=1}^{H_x}\sum_{m_2=1}^{H_y}\boldsymbol{D}_x^{m_1-1}\boldsymbol{D}_y^{m_2-1}\boldsymbol{R}_{ss} \\ \times\left(\boldsymbol{D}_y^{m_2-1}\right)^{\mathrm{H}}\left(\boldsymbol{D}_x^{m_1-1}\right)^{\mathrm{H}}.\end{aligned} \tag{17}$$

空间平滑协方差矩阵使得即使在存在相干信号的情况下也可以应用基于特征结构的方法进行到达角估计。

空间平滑算法的一个局限性是其倾向于减小有效阵列孔径，这可能会降低感知性能 [17]。为了解决这一问题，我们为URA引入了一种前向-后向空间平滑方案，如图 6所示。这种双向平滑方法通过利用协方差矩阵的共轭对称性保持孔径大小。

在数学上，前向-后向空间平滑协方差矩阵表示为

$$\boldsymbol{R_X} = \frac{1}{2}\left(\boldsymbol{R}^f + \boldsymbol{I_v}\left(\boldsymbol{R}^f\right)^*\boldsymbol{I_v}\right), \tag{18}$$

(b) The first part of the Chinese PDF of case 3.

图 7. 使用 3 × 4 天线阵列生成的四个相关信号源的空间谱。(a)、(b) 和 (c) 分别显示了由 MUSIC、SS-MUSIC 和 1-SSMUSIC 计算的二维空间谱。

其中$\left(\boldsymbol{R}^f\right)^*$是矩阵$\boldsymbol{R}^f$的共轭矩阵，以及

$$\boldsymbol{I_v} = \begin{bmatrix}0 & \cdots & 0 & 1\\ 0 & \cdots & 1 & 0\\ \vdots & \iddots & \vdots & \vdots\\ 1 & \cdots & 0 & 0\end{bmatrix}_{M\times M}. \tag{19}$$

通过使用这种平滑协方差矩阵计算伪谱在(11)中，我们能够在减轻秩不足影响的同时准确估计相关信号。

通过检查(16)和(18)，我们观察到前向平滑子阵的数量，用H表示，决定了最大可分辨相关源的数量，而前向-后向平滑则有效地将这一限制增加到$2H$。在典型的室内环境中，通常多径分量的数量少于五个 [13], [17]，一次前向-后向平滑操作（$H = 2$）即可使多达四个不同角度的信号去相关。

我们现在对传统MUSIC、仅前向空间平滑的MUSIC（SS-MUSIC）和提出的1-SSMUSIC在相同条件下估计四个相关信号角度进行比较评估。URA由3×4个天线组成。四个相关信号源发射连续信号，信噪比为15 dB，来自以下角度：$(21.8^\circ, 90^\circ)$，$(32^\circ, 56^\circ)$，$(15^\circ, -60^\circ)$和$(60^\circ, -150^\circ)$。空间谱如图7所示，从中可以看出，提出的1-SSMUSIC优于其他方法。使用1-SSMUSIC估计的到达角是$(21.8^\circ, 90.8^\circ)$，$(32.4^\circ, 57.2^\circ)$，$(16.4^\circ, -59.6^\circ)$和$(60.2^\circ, -150.6^\circ)$。相比之下，虽然SS-MUSIC能够估计相关信号，但其分辨率明显较低。其估计的到达角为$(22.8^\circ, 82.2^\circ)$，$(37.2^\circ, 50.8^\circ)$，$(15.2^\circ, -62^\circ)$和$(58.8^\circ, -149.6^\circ)$。而标准MUSIC算法则无法解决相关源，导致不明确和不准确的到达角。

D. 最近几何点估计

通过从多个分布在空间的URA获得的AoA估计，可以确定信号源的具体位置。理想情况下，估计的AoA矢量将在源的真实位置相交。然而，由于测量误差，需要一个强大的最近点估计算法来逼近实际的交点。所提出的几何定位（GP）方法首先识别每对AoA之间的最近点，如图 8的阶段1所示。然后将这些最近点的平均值计算为最终位置估计。

设l_i表示与第i个URA相关联的估计到达射线。每条射线可以由参数方程表示为

$$\begin{cases}\boldsymbol{r}_1 = \boldsymbol{c}_1 + t_1\boldsymbol{d}_1, \\ \quad\vdots \\ \boldsymbol{r}_i = \boldsymbol{c}_i + t_i\boldsymbol{d}_i, \\ \quad\vdots \\ \boldsymbol{r}_n = \boldsymbol{c}_n + t_n\boldsymbol{d}_n,\end{cases} \tag{20}$$

其中$\boldsymbol{c}_i \in \mathbb{R}^3$表示第$i$个URA的中心，$\boldsymbol{d}_i \in \mathbb{R}^3$是到达射线$l_i$的方向向量。为了识别最佳逼近第$h$个和第$i$个AoA射线交点的向量$\mathbf{t}_{h,i} = [t_h\ t_i]^\top$，我们解以下方程

$$\begin{bmatrix}-\mathbf{d}_h^\top\mathbf{d}_h & \mathbf{d}_h^\top\mathbf{d}_h\\ -\mathbf{d}_h^\top\mathbf{d}_i & \mathbf{d}_i^\top\mathbf{d}_i\end{bmatrix}\begin{bmatrix}t_h\\ t_i\end{bmatrix} = \begin{bmatrix}(\mathbf{c}_h-\mathbf{c}_i)\cdot\mathbf{d}_i\\ (\mathbf{c}_h-\mathbf{c}_i)\cdot\mathbf{d}_h\end{bmatrix}. \tag{21}$$

如果没有精确交点，最小二乘解$\mathbf{t}^* = [t_{h,i}^*\ t_{i,h}^*]^\top$确定两条射线上的最近点对。这些点的坐标为

$$\mathbf{c}_{h,i}^* = \mathbf{c}_h + t_{h,i}^*\mathbf{d}_h, \mathbf{c}_{i,h}^* = \mathbf{c}_i + t_{i,h}^*\mathbf{d}_i. \tag{22}$$

最后，基于所有u个URA的源的估计位置计算为

$$\mathbf{c}^* = \frac{1}{u(u-1)}\sum_{h=1}^{u-1}\sum_{i=h+1}^{u}\left(\mathbf{c}_{h,i}^* + \mathbf{c}_{i,h}^*\right) = [x^*, y^*, z^*]. \tag{23}$$

IV. 协作3D直接位置确定

对于前面描述的最近几何点方法，协作是在估计的AoA级别进行的，因为相关的URA彼此之间不同步。鉴于现在已经在每个阵列内的元素之间实现了信号同步，一个自然的问题是：这种同步机制能否进一步扩展到阵列间水平，以实现更大的协作增益？在本节中，我们开发了一种旨在促进直接位置确定（DPD）的阵列间同步框架 [38], [39]。与之前的最近点估计方法不同，DPD绕过了中间参数估计，例如AoA，而是直接在一步中计算源位置。

为了减少所提出的DPD算法的空间采样开销，我们首先采用1-SSMUSIC和最近点估计方法来定义一个紧凑的局部空间感兴趣区（LSoI）。通过对LSoI进行离散化，我们推导出一个测量模型，该模型描述了跨多个同步URA的观测过程。通过相对于公共参考信号测量

(c) The second part of the English PDF of case 3.

(d) The second part of the Chinese PDF of case 3.

Figure 7: Case 3 demonstrates the performance of LaTeXTrans on the En-Zh task

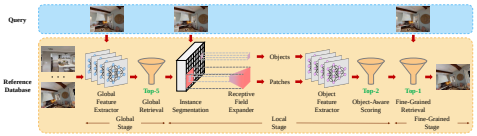


Figure 2. **The AirRoom coarse-to-fine pipeline**. The pipeline begins with the Global Feature Extractor, which captures global context features to retrieve the top-5 reference images. Instance segmentation then generates object masks, followed by the Receptive Field Expander, which extracts object patches. The Object Feature Extractor processes both object and patch features. The Object-Aware Scoring module narrows the selection to the top-2 candidates, and Fine-Grained Retrieval identifies the most suitable reference image.

However, the high performance of most VPR approaches is largely attributed to large-scale training on VPR-specific datasets [16]. Collecting extensive data for outdoor scenes is relatively straightforward due to natural variations in daylight, weather, and seasons. However, such data collection is more challenging in indoor rooms, making large-scale training on indoor datasets difficult and potentially limiting their effectiveness. Our approach effectively tackles this challenge by focusing on object-oriented feature representations, allowing us to leverage mature, pre-trained models for object feature learning. This design enables AirRoom to deliver robust performance without requiring any additional training or fine-tuning on specific datasets.

3. Proposed Approach

We propose a simple yet highly effective pipeline, AirRoom, for room reidentification that leverages multi-level object-oriented information, as shown in Figure 2. We will now systematically introduce each module of the pipeline, following the sequence of stages in which they are executed.

3.1. Global Stage

In this stage, we utilize the Global Feature Extractor to capture global context features, which are derived from the collective presence of objects within the room. These features are then used for Global Retrieval, coarsely selecting semantically similar candidate rooms from the database.

3.1.1. Global Feature Extractor

Indoor rooms exhibit fewer variations compared to outdoor environments. They lack diverse topographies, such as aerial, subterranean, or underwater features, and do not experience temporal changes like day-night or seasonal variations. Consequently, collecting large datasets for each indoor room is challenging, complicating large-scale training as seen in many VPR methods [1, 2, 13].

However, indoor rooms are inherently rich in objects, each contributing to the room's overall semantic context. By leveraging this global context information, we can refine the reference search to specifically focus on rooms with similar semantic features to those in the query image. For this purpose, we prefer backbones pretrained on large image datasets, as they provide strong generalizability and effectively capture informative global context features [17]. Our model selections, therefore, include pretrained CNN-based models such as ResNet [14] and transformer-based self-supervised models like DINOv2 [25].

3.1.2. Global Retrieval

Using the Global Feature Extractor, we extract global context features for M query and N reference images. Let $\mathbf{Q} \in \mathbb{R}^{M \times D_g}$ and $\mathbf{R} \in \mathbb{R}^{N \times D_g}$ denote the query and reference features, respectively, where D_g is the feature dimension. The cosine similarity matrix $\mathbf{S}$ is then computed as:

$$\mathbf{S}_{ij} = \frac{\mathbf{Q}_i \cdot \mathbf{R}_j}{\|\mathbf{Q}_i\| \|\mathbf{R}_j\|}. \tag{1}$$

For each query, we select the top-5 most similar reference candidates using the following formula:

$$\text{Top}_5(\mathbf{S}_{i,:}) = \text{argsort}(-\mathbf{S}_{i,:})[:5], \tag{2}$$

where $\mathbf{S}_{i,:}$ represents the cosine similarity for the i-th query.

3.2. Local Stage

Global context features provide valuable semantic information that helps narrow down the candidate list. However, when faced with many semantically similar rooms, relying solely on global context is insufficient, and local features become increasingly essential. In this stage, we adopt a local perspective by first applying instance segmentation and the Receptive Field Expander to identify objects and patches. We then use the Object Feature Extractor to extract features from both objects and patches, followed by Object-Aware Scoring to further refine the candidate list.

(a) The first part of the English PDF of case 4.

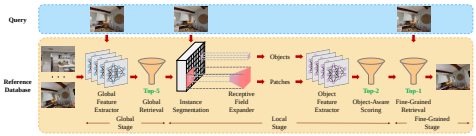


Figure 2. AirRoom **の粗から細へのパイプライン**. このパイプラインは、グローバルな文脈特徴をキャプチャしてトップ 5 の参照画像を検索するグローバル特徴抽出器から始まります。その後、インスタンスセグメンテーションが物体マスクを生成し、受容野拡張器が物体パッチを抽出します。物体特徴抽出器は、物体とパッチの特徴の両方を処理します。物体認識に配慮したスコアリングモジュールが選択肢をトップ 2 の候補に絞り、細かい検索が最も適切な参照画像を識別します。

習ベースのモデルが特徴マップを抽出し、局所特徴を統合して包括的なグローバル記述子を生成するようになりました。

しかし、ほとんどの VPR アプローチの高いパフォーマンスは、VPR 専用のデータセットでの大規模なトレーニングによるものです [16]。屋外のシーンのデータ収集は、昼間の光、天気、季節の変動が自然に存在するため比較的簡単です。しかし、室内の部屋でのデータ収集はより困難であり、屋内のデータセットでの大規模なトレーニングが難しく、その効果を制限する可能性があります。私たちのアプローチは、物体指向の特徴表現に焦点を当てることにより、この課題を効果的に解決します。この設計により、AirRoom は特定のデータセットでの追加のトレーニングや微調整なしで強力なパフォーマンスを提供することができます。

3. 提案手法

我々は、図 Figure 2 に示すように、多段階の物体指向情報を活用した部屋の再識別のためのシンプルでありながら非常に効果的なパイプライン、AirRoom を提案します。次に、パイプラインの各モジュールを、それらが実行される順序に従って体系的に紹介します。

3.1. グローバル段階

この段階では、グローバル特徴抽出器を使用して、部屋内の物体の集合的な存在から得られるグローバルな文脈特徴をキャプチャします。これらの特徴は、グローバル検索に使用され、データベースから意味的に類似した候補部屋を粗く選択するために使用されます。

3.1.1. グローバル特徴抽出器

屋内の部屋は、屋外環境と比較して変化が少ない。航空、地下、水中といった多様な地形的特徴を欠き、昼夜や季節の変化といった時間的変化も存在しない。そのため、各屋内空間ごとに大規模なデータセットを収集することは困難であり、多くの視覚的場所認識（VPR）手法で見られるような大規模な学習を複雑にしている [1, 2, 13]。

しかしながら、屋内の部屋には本質的に多くの物体が存在し、それぞれが部屋全体の意味的文脈に寄与しています。このグローバルな文脈情報を活用することで、参照検索をクエリ画像と意味的に類似した部屋に絞って精緻化することが可能となる。この目的のために、我々は大規模な画像データセットで事前学習されたバックボーンを好んで用いる。これらは高い汎化性能を備え、情報豊富なグローバルな文脈特徴を効果的に捉えることができるためである [17]。そのため、我々のモデル選択には、ResNet [14] のような CNN ベースの事前学習モデルや、DINOv2 [25] のようなトランスフォーマーベースの自己教師ありモデルが含まれる。

3.1.2. グローバル検索

グローバル特徴抽出器を使用して、M 個のクエリ画像と N 個の参照画像に対してグローバルな文脈特徴を抽出します。クエリ特徴を $\mathbf{Q} \in \mathbb{R}^{M \times D_g}$、参照特徴を $\mathbf{R} \in \mathbb{R}^{N \times D_g}$ で表し、ここで D_g は特徴の次元を示します。コサイン類似度行列 $\mathbf{S}$ は次のように計算されます：

$$\mathbf{S}_{ij} = \frac{\mathbf{Q}_i \cdot \mathbf{R}_j}{\|\mathbf{Q}_i\| \|\mathbf{R}_j\|} \tag{1}$$

各クエリについて、次の式を使用して最も類似したトップ 5 の参照候補を選択します：

$$\text{Top}_5(\mathbf{S}_{i,:}) = \text{argsort}(-\mathbf{S}_{i,:})[:5], \tag{2}$$

ここで、$\mathbf{S}_{i,:}$ は i 番目のクエリに対するコサイン類似度を表します。

3.2. ローカル段階

グローバルな文脈特徴は、候補リストを絞り込むために価値のある意味的情報を提供します。しかし、意味的に類似した部屋が多く存在する場合、グローバルな文脈だけに頼ることは不十分であり、局所特徴がますます重要になります。この段階では、最初にインスタンスセグメンテーションと受容野拡張器を適用して物体とパッチを識別し、次に物体特徴抽出器を使用して物体とパッチの

(b) The first part of the Japanese PDF of case 4.

3.2.1. Instance Segmentation

For each query image and its corresponding five candidates, we employ instance segmentation methods, such as Mask R-CNN [15] and Semantic-SAM [20], to identify and delineate individual objects. This process generates each object's mask and bounding box. Next, we calculate the center point c of each object using its bounding box, as shown below:

$$c = (\frac{x+W}{2}, \frac{y+H}{2}). \tag{3}$$

In this equation, x and y represent the pixel coordinates of the top-left corner of the bounding box, while W and H denote the width and height of the bounding box, respectively.

3.2.2. Receptive Field Expander

Single object information alone is not sufficiently discriminative. For example, although different desks may have distinct appearances, they can be found in both dining halls and offices. However, when an object is connected with its neighboring items—such as a desk alongside a computer, keyboard, or notebook—it suggests that the room is more likely to be an office rather than a dining hall. This insight motivates us to expand the receptive field from a single object to a patch containing multiple objects.

Given the center points of all objects in an image, we employ Delaunay triangulation [6] to generate a triangulated graph of object relationships. Specifically, Delaunay triangulation is applied to the set of object centers, ensuring that no object centers are inside the circumcircle of any triangle. This method maximizes the minimum angle of the triangles, preventing narrow, elongated triangles and ensuring more uniform object adjacency. By analyzing the adjacency relationships among the resulting triangles, we can construct the object adjacency matrix, which encodes the spatial and relational proximity of objects within the room.



Figure 3. The Receptive Field Expander broadens the receptive field from individual objects to patches rich in contextual information. Leveraging the object adjacency matrix and each object's bounding box, it expands single objects such as a cupboard, window pane, and chair into object patches like a modular kitchen, multi-pane window, and dining set, respectively.

Given the object adjacency matrix and bounding boxes in an image, for each object, we consider the bounding boxes of its neighboring objects and enlarge the current object's bounding box to encompass all adjacent objects. This expansion increases the receptive field, enabling us to capture richer contextual information, as illustrated in Figure 3. We then apply Non-Maximum Suppression (NMS) to select the highest confidence bounding boxes, removing overlapping ones based on their Intersection over Union (IoU) scores. This results in a set of clean, informative object patches.

3.2.3. Object-Aware Refinement

The Object-Aware Refinement module is composed of three key submodules: Object Feature Extractor, Mutual Nearest Neighbors, and Object-Aware Scoring.

Object Feature Extractor To effectively leverage object patches and object segmentation information, we prioritize global features over local feature aggregation. The latter approach may fail to capture object characteristics effectively and can significantly increase computational complexity and storage demands [49]. As discussed in Section 3.1.1, we continue to rely on models pre-trained on large image datasets. Using the Object Feature Extractor, we obtain features for both query and reference patches and objects. Let $Q_p = \{\mathbf{p}_i^q\}_{i=1}^{n_q}$ and $Q_o = \{\mathbf{o}_i^q\}_{i=1}^{n_q}$ represent the query patch and object feature sets, respectively. For each reference image among the query's five candidates, we define the reference patch and object feature sets as $R_p = \{\mathbf{p}_i^r\}_{i=1}^{n_r}$ and $R_o = \{\mathbf{o}_i^r\}_{i=1}^{n_r}$.

Mutual Nearest Neighbors Given a set of query features $\{\mathbf{f}_i^q\}_{i=1}^{n_q}$ and reference features $\{\mathbf{f}_i^r\}_{i=1}^{n_r}$, we obtain feature pairs by identifying mutual nearest neighbor matches through exhaustive comparison of the two sets. Let P denote the set of cosine similarity scores for these mutual nearest neighbor matches, then we have

$$P = \{\cos(\mathbf{f}_i^q, \mathbf{f}_j^r) \mid i = \text{NN}_r(\mathbf{f}_j^r),\ j = \text{NN}_q(\mathbf{f}_i^q)\} \tag{4}$$

where

$$\text{NN}_q(\mathbf{f}_i^q) = \arg\max_j \left(\frac{\mathbf{f}_i^q \cdot \mathbf{f}_j^r}{\|\mathbf{f}_i^q\| \|\mathbf{f}_j^r\|} \right), \tag{5}$$

$$\text{NN}_r(\mathbf{f}_i^r) = \arg\max_j \left(\frac{\mathbf{f}_i^r \cdot \mathbf{f}_j^q}{\|\mathbf{f}_i^r\| \|\mathbf{f}_j^q\|} \right), \tag{6}$$

$$\cos(\mathbf{f}_i^q, \mathbf{f}_j^r) = \frac{\mathbf{f}_i^q \cdot \mathbf{f}_j^r}{\|\mathbf{f}_i^q\| \|\mathbf{f}_j^r\|}. \tag{7}$$

By utilizing mutual nearest neighbors, we can significantly improve retrieval accuracy, simultaneously narrowing the search space and enhancing overall retrieval efficiency [50].

Object-Aware Scoring The object-aware score s is the sum of the global score s_{global} (calculated in Equation 1), the patch score s_{patch}, and the object score s_{object}:

$$s = s_{\text{global}} + s_{\text{patch}}(Q_p, R_p) + s_{\text{object}}(Q_o, R_o). \tag{8}$$

(c) The second part of the English PDF of case 4.

特徴を抽出します。その後、物体認識に配慮したスコアリングを行い、候補リストをさらに絞り込みます。

3.2.1. インスタンスセグメンテーション

各クエリ画像とその対応する 5 つの候補について、Mask R-CNN [15] や Semantic-SAM [20] などのインスタンスセグメンテーション手法を用いて、個々の物体を識別し、輪郭を描きます。このプロセスでは、各物体のマスクと境界ボックスが生成されます。次に、以下のようにして、各物体の中心点 c をその境界ボックスを用いて計算します：

$$c = (\frac{x+W}{2}, \frac{y+H}{2}). \tag{3}$$

この式では、x および y は境界ボックスの左上隅のピクセル座標を表し、W および H はそれぞれ境界ボックスの幅と高さを示します。

3.2.2. 受容野拡張器

単一の物体情報だけでは十分に識別的ではありません。例えば、異なるデスクは外見が異なるかもしれませんが、食堂とオフィスの両方で見つけることができます。しかし、物体がその隣接物と結びついている場合—例えば、デスクの隣にコンピュータ、キーボード、ノートがある場合—それはその部屋が食堂ではなくオフィスである可能性が高いことを示唆します。この洞察は、受容野を単一の物体から複数の物体を含むパッチに拡張する動機となります。

画像内のすべての物体の中心点が与えられた場合、Delaunay 三角分割 [6] を使用して物体間の関係の三角形グラフを生成します。具体的には、物体の中心点のセットに対して Delaunay 三角分割を適用し、任意の三角形の外接円の中に物体中心が含まれないことを確保します。この方法は三角形の最小角を最大化し、狭い細長い三角形を防ぎ、物体の隣接性をより均等に保ちます。得られた三角形間の隣接関係を分析することにより、部屋内の物体の空間的および関係的近接をエンコードする物体隣接行列を構築できます。

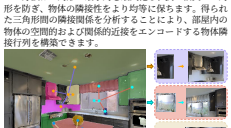


Figure 3. 受容野拡張器は、個々の物体から文脈情報に富んだパッチへの受容野の拡大を行います。物体隣接行列と各物体の境界ボックスを活用して、収納棚、窓ガラス、椅子などの単一の物体を、それぞれモジュラーキッチン、複数の窓ガラス、ダイニングセットのような物体パッチに拡張します。

物体隣接行列と画像内の境界ボックスが与えられた場合、各物体について、その隣接物体の境界ボックスを考慮し、現在の物体の境界ボックスを隣接するすべての物体を含むように拡大します。この拡張により受容野が増加し、より豊かな文脈情報を取得できるようになります（図 Figure 3 に示す通り）。その後、非最大抑制（NMS）を適用して、最も高い信頼度の境界ボックスを選択し、その交差部分に基づいて重複するものを削除します。これにより、クリーンで情報豊富な物体パッチが得られます。

3.2.3. 物体認識に配慮した改良

物体認識に配慮した改良モジュールは、物体特徴抽出器、相互最近傍、物体認識に配慮したスコアリングの 3 つの主要なサブモジュールで構成されています。

物体特徴抽出器 物体パッチと物体セグメンテーション情報を効果的に活用するために、局所特徴の集約よりもグローバル特徴を優先します。後者のアプローチは物体の特徴を効果的に捉えることができず、計算の複雑さやストレージの要求が大幅に増加する可能性があります [49]。セクション 3.1.1 で述べたように、大規模な画像データセットで事前学習されたモデルに引き続き依存します。物体特徴抽出器を使用して、クエリと参照のパッチおよび物体の特徴を取得します。クエリのパッチと物体特徴セットをそれぞれ $Q_p = \{\mathbf{p}_i^q\}_{i=1}^{n_q}$ と $Q_o = \{\mathbf{o}_i^q\}_{i=1}^{n_q}$ とし、各参照画像について、参照のパッチと物体特徴セットを $R_p = \{\mathbf{p}_i^r\}_{i=1}^{n_r}$ と $R_o = \{\mathbf{o}_i^r\}_{i=1}^{n_r}$ と定義します。

相互最近傍 クエリ特徴 $\{\mathbf{f}_i^q\}_{i=1}^{n_q}$ と参照特徴 $\{\mathbf{f}_i^r\}_{i=1}^{n_r}$ のセットが与えられた場合、両セットの徹底的な比較を通じて相互最近傍マッチを識別することにより、特徴ペアを取得します。P はこれらの相互最近傍マッチに対するコサイン類似度スコアのセットを示すとすると、次のように表されます

$$P = \{\cos(\mathbf{f}_i^q, \mathbf{f}_j^r) \mid i = \text{NN}_r(\mathbf{f}_j^r),\ j = \text{NN}_q(\mathbf{f}_i^q)\} \tag{4}$$

ここで

$$\text{NN}_q(\mathbf{f}_i^q) = \arg\max_j \left(\frac{\mathbf{f}_i^q \cdot \mathbf{f}_j^r}{\|\mathbf{f}_i^q\| \|\mathbf{f}_j^r\|} \right), \tag{5}$$

$$\text{NN}_r(\mathbf{f}_i^r) = \arg\max_j \left(\frac{\mathbf{f}_i^r \cdot \mathbf{f}_j^q}{\|\mathbf{f}_i^r\| \|\mathbf{f}_j^q\|} \right), \tag{6}$$

$$\cos(\mathbf{f}_i^q, \mathbf{f}_j^r) = \frac{\mathbf{f}_i^q \cdot \mathbf{f}_j^r}{\|\mathbf{f}_i^q\| \|\mathbf{f}_j^r\|}. \tag{7}$$

相互最近傍を利用することで、検索精度を大幅に向上させ、検索空間を縮小し、全体的な検索効率を高めることができます [50]。

物体認識に配慮したスコアリング 物体認識に配慮したスコア s は、グローバルスコア s_{global}（式 1 で計算）、

(d) The second part of the Japanese PDF of case 4.

Figure 8: Case 4 demonstrates the performance of LaTeXTrans on the En-Ja task

transformer [40] layers, denoted $\{\mathcal{V}_i\}_{i=1}^{L}$. Given an input image $x \in \mathbb{R}^{H\times W\times 3}$, it is divided into M fixed-size patches, each projected into a patch embedding, resulting in $E_0 \in \mathbb{R}^{M\times d_v}$, where M represents the number of patches and d_v the embedding dimension. The initial patch embeddings E_0 are combined with a learnable class token c_0 and positional encodings, forming the input sequence for the transformer layers. Each layer processes this sequence as

$$[c_i, E_i] = \mathcal{V}_i([c_{i-1}, E_{i-1}]) \quad i = 1, 2, \ldots, L$$

After passing through all transformer layers, a patch projection layer, P_v^c, projects the output of the class token, c_L, into a shared V-L latent space,

$$f = P_v^c(c_L)$$

where $f \in \mathbb{R}^d$.

Text Encoding: For an input text, *e.g.*, "A photo of a [CLASS].", it is tokenized and converted into embeddings $T_0 \in \mathbb{R}^{N\times d_t}$, where N is the token length and d_t the embedding dimension. Beginning-of-text (BOT) and end-of-text (EOT) tokens, denoted b_0 and e_0, mark the sequence boundaries. These token embeddings, with positional encodings, are passed through the text encoder's L transformer layers, $\{\mathcal{W}_i\}_{i=1}^{L}$, as follows,

$$[b_i, T_i, e_i] = \mathcal{W}_i([b_{i-1}, T_{i-1}, e_{i-1}]) \quad i = 1, \ldots, L$$

After the final layer, the output of the EOT token, e_L, is projected into the shared V-L space using P_t,

$$w = P_t(e_L)$$

where $w \in \mathbb{R}^d$.

Classification with CLIP: With the image feature f and text features $\{w_c\}_{c=1}^{C}$ for C classes, CLIP calculates the cosine similarity between f and each w_c,

$$\text{sim}(f, w_c) = \frac{f \cdot w_c}{|f||w_c|},$$

where $|\cdot|$ represents the L_2 norm. Class probabilities are then computed using the softmax function,

$$p(y = c \mid f) = \frac{\exp(\text{sim}(f, w_c)/\tau)}{\sum_{i=1}^{C} \exp(\text{sim}(f, w_i)/\tau)}$$

where τ a temperature parameter. The final predicted class is selected as the one with the highest probability score.

3.2. Multi-Modal Representation Learning (MMRL)

Our proposed MMRL aims to address the challenges of adapting pre-trained VLMs using few-shot data while maintaining generalization to new tasks. The training and inference frameworks of MMRL are shown in Fig. 2 and Fig. 3, respectively. In the following, we describe the specifics of the methodology.

3.2.1. Learnable Representation Space

MMRL establishes a shared, learnable representation space $\mathcal{R}$ to facilitate multimodal interactions, initialized through sampling from a Gaussian distribution. Using a learnable mapping function $\mathcal{F}(\cdot)$, implemented as a linear layer, we project the tokens $R \in \mathbb{R}^{K\times d_v}$ in this space—where K is the number of tokens and d_v is the dimension of the representation space—into both visual and textual modalities,

$$R^v = \{R_i^v\}_{i=J-1}^{L-1} \quad R_i^v = \mathcal{F}_i^v(R)$$
$$R^t = \{R_i^t\}_{i=J-1}^{L-1} \quad R_i^t = \mathcal{F}_i^t(R)$$

where $R_i^v \in \mathbb{R}^{K\times d_v}$ and $R_i^t \in \mathbb{R}^{K\times d_t}$ represent the representation tokens for visual and textual modalities, respectively, in the $(i+1)$-th transformer layer. The index J indicates the starting layer from which these representation tokens are integrated into the encoders.

3.2.2. Integration into Higher Encoder Layers

To preserve the generalized knowledge in the lower layers of the pre-trained CLIP model, the representation tokens $\mathcal{R}^v$ and $\mathcal{R}^t$ are integrated into the higher layers of the image encoder $\mathcal{V}$ and the text encoder $\mathcal{W}$, beginning from the J-th layer.

For the image encoder $\mathcal{V}$,

$$\begin{aligned}
[c_i, E_i] &= \mathcal{V}_i([c_{i-1}, E_{i-1}]) \quad i = 1, \ldots, J-1 \\
[c_i, _, E_i] &= \mathcal{V}_i([c_{i-1}, R_{i-1}^v, E_{i-1}]) \quad i = J, \ldots, L-1 \\
[c_i, R_i^v, E_i] &= \mathcal{V}_i([c_{i-1}, R_{i-1}^v, E_{i-1}]) \quad i = L
\end{aligned}$$

For the text encoder $\mathcal{W}$, while previous prompt learning [17] involves replacing parts of T_i to incorporate deep prompts, we retain the entire T_i and insert R_i^t before it, aiming to preserve the original textual information,

$$\begin{aligned}
[b_i, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, T_{i-1}, e_{i-1}]) \quad i = 1, \ldots, J-1 \\
[b_i, _, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, R_{i-1}^t, T_{i-1}, e_{i-1}]) \\
&\qquad i = J, \ldots, L-1 \\
[b_i, R_i^t, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, R_{i-1}^t, T_{i-1}, e_{i-1}]) \quad i = L
\end{aligned}$$

Note that due to the autoregressive nature of the text encoder, we adjust the attention mask matrix to accommodate the increased embedding length.

3.2.3. Representation Learning

Representation learning is designed to leverage representation tokens for dataset-specific adaptation, while the class token preserves the pre-trained knowledge of the original CLIP. Through a set of strategies aimed at retaining generalization during both training and inference, MMRL enables flexible inference for different tasks, as detailed below.

- **Training Phase:** We optimize the features of both the representation tokens and the original class token, with the final MMRL loss function is

$$\mathcal{L}_{MMRL} = \alpha\mathcal{L}_{ce}^{v} + (1-\alpha)\mathcal{L}_{ce}^{t} + \lambda(\mathcal{L}_{cos}^{v} + \mathcal{L}_{cos}^{t})$$

where α controls the balance between the features, and λ is the penalty coefficient.
- **Testing on Base Classes:** For in-distribution classes seen during training, we combine the dataset-specific representation features with the class features that preserve generalizability. The probability of an in-distribution test sample x belonging to the c-th class is

$$p(y = c \mid x) = \alpha \cdot p(y = c \mid f_v) + (1-\alpha) \cdot p(y = c \mid f_r)$$

where f_v and f_r are features extracted from the class token and representation tokens, respectively.
- **Testing on Novel Classes:** For classes unseen during training or for new datasets, we rely solely on the class tokens, which retain generalized knowledge.

$$p(y = c \mid x) = p(y = c \mid f_v)$$

4. Experiments

Details on implementation, datasets, and computational cost are provided in the Supplementary Materials.

4.1. Tasks and Datasets

We conduct four core evaluations to comprehensively assess MMRL's performance: base-to-novel generalization, cross-dataset evaluation, domain generalization, and few-shot learning. Except for few-shot learning, all experiments utilize a 16-shot setting, *i.e.*, only 16 training examples per category.

Base-to-Novel Generalization: In this evaluation, dataset classes are equally divided into base and novel classes. The model is trained exclusively on base classes and tested on both base and novel classes, allowing us to examine its transfer learning effectiveness on base classes as well as its ability to retain the inherent generalization or zero-shot capabilities of pre-trained VLMs for novel classes. We conduct this evaluation across 11 diverse image classification datasets: ImageNet [7], Caltech101 [9], OxfordPets [32], StanfordCars [19], Flowers102 [29], Food101 [3], FGVCAircraft [27], SUN397 [45], UCF101 [30], DTD [6], and EuroSAT [11].

(a) The first part of the English PDF of case 5.

3. 方法

私たちのアプローチは、従来の手法に沿って、事前学習済みの VLM であるCLIP [38]を基盤としています。このセクションでは、MMRL 訓練フレームワークの構築と実装の詳細について説明します。

3.1. 前提

私たちのアプローチで使用する記法を定義することから始めます。CLIP は、2 つのエンコーダから構成されます：画像エンコーダ $\mathcal{V}$ とテキストエンコーダ $\mathcal{W}$ です。
画像エンコーディング: 画像エンコーダ $\mathcal{V}$ は、L 層のトランスフォーマー [40] から構成され、これを $\{\mathcal{V}_i\}_{i=1}^{L}$ で表します。入力画像 $x \in \mathbb{R}^{H\times W\times 3}$ が与えられると、それは M 個の固定サイズのパッチに分割され、それぞれがパッチ埋め込みに投影され、$E_0 \in \mathbb{R}^{M\times d_v}$ が得られます。ここで、M はパッチの数、d_v は埋め込み次元を表します。最初のパッチ埋め込み E_0 は、学習可能なクラス・トークン c_0 および位置エンコーディングと組み合わせられ、トランスフォーマー層の入力シーケンスが形成されます。各層はこのシーケンスを次のように処理します。

$$[c_i, E_i] = \mathcal{V}_i([c_{i-1}, E_{i-1}]) \quad i = 1, 2, \ldots, L$$

すべてのトランスフォーマー層を通過した後、パッチ投影層 P_v^c が、クラス・トークン c_L の出力を共有された V-L 潜在空間に投影します。

$$f = P_v^c(c_L)$$

ここで、$f \in \mathbb{R}^d$ です。
テキストエンコーディング: 入力テキスト、例えば「A photo of a [CLASS].」の場合、それはトークン化され、埋め込み $T_0 \in \mathbb{R}^{N\times d_t}$ に変換されます。ここで、N はトークン長、d_t は埋め込み次元を表します。テキストの開始トークン（BOT）および終了トークン（EOT）は、それぞれ b_0 および e_0 で示され、シーケンスの境界を示します。これらのトークン埋め込みは位置エンコーディングとともにテキストエンコーダの L 層のトランスフォーマー層 $\{\mathcal{W}_i\}_{i=1}^{L}$ を通過します。次のように処理されます。

$$[b_i, T_i, e_i] = \mathcal{W}_i([b_{i-1}, T_{i-1}, e_{i-1}]) \quad i = 1, \ldots, L$$

最終層後、EOT トークン e_L の出力は、P_t を使用して共有 V-L 空間に投影されます。

$$w = P_t(e_L)$$

ここで、$w \in \mathbb{R}^d$ です。
CLIP **による分類**: 画像特徴 f と C クラスのテキスト特徴 $\{w_c\}_{c=1}^{C}$ を用いて、CLIP は f と各 w_c とのコサイン類似度を計算します。

$$\text{sim}(f, w_c) = \frac{f \cdot w_c}{|f||w_c|},$$

ここで、$|\cdot|$ は L_2 ノルムを表します。クラス確率は次のソフトマックス関数を用いて計算されます。

$$p(y = c \mid f) = \frac{\exp(\text{sim}(f, w_c)/\tau)}{\sum_{i=1}^{C} \exp(\text{sim}(f, w_i)/\tau)}$$

ここで、τ は温度パラメータです。最終的に予測されたクラスは、最も高い確率スコアを持つクラスとして選択されます。

3.2. Multi-Modal Representation Learning (MMRL)

我々が提案する MMRL は、少数ショットデータを使用して事前学習済み VLM の適応に関する課題を解決し、同時に新しいタスクへの一般化を維持することを目的としています。MMRL の訓練および推論フレームワークは、それぞれ Fig. 2 および Fig. 3 に示されています。以下に、方法論の詳細について説明します。

3.2.1. 学習可能な表現空間

MMRL は、マルチモーダル相互作用を促進するために共有の学習可能な表現空間 $\mathcal{R}$ を確立します。この空間は、ガウス分布からサンプリングすることで初期化されます。学習可能なマッピング関数 $\mathcal{F}(\cdot)$ を使用し、これは線形層として実装され、トークン $R \in \mathbb{R}^{K\times d_v}$ をこの空間に投影します—ここで K はトークンの数、d_v は表現空間の次元を示します—視覚的およびテキストのモダリティに対して、

$$R^v = \{R_i^v\}_{i=J-1}^{L-1} \quad R_i^v = \mathcal{F}_i^v(R)$$
$$R^t = \{R_i^t\}_{i=J-1}^{L-1} \quad R_i^t = \mathcal{F}_i^t(R)$$

ここで $R_i^v \in \mathbb{R}^{K\times d_v}$ および $R_i^t \in \mathbb{R}^{K\times d_t}$ は、それぞれ視覚的およびテキストのモダリティにおける $(i+1)$ 層目での表現トークンを表します。インデックス J は、これらの表現トークンがエンコーダに統合される開始層を示します。

3.2.2. 高層エンコーダ層への統合

事前学習済み CLIP モデルの下層における一般化された知識を保持するために、表現トークン R^v および R^t は、画像エンコーダ $\mathcal{V}$ およびテキストエンコーダ $\mathcal{W}$ の高層に統合され、J 層目から始まります。
画像エンコーダ $\mathcal{V}$ の場合、

$$\begin{aligned}
[c_i, E_i] &= \mathcal{V}_i([c_{i-1}, E_{i-1}]) \quad i = 1, \ldots, J-1 \\
[c_i, _, E_i] &= \mathcal{V}_i([c_{i-1}, R_{i-1}^v, E_{i-1}]) \quad i = J, \ldots, L-1 \\
[c_i, R_i^v, E_i] &= \mathcal{V}_i([c_{i-1}, R_{i-1}^v, E_{i-1}]) \quad i = L
\end{aligned}$$

テキストエンコーダ $\mathcal{W}$ の場合、従来のプロンプト学習 [17] では T_i の一部を置き換えて深いプロンプトを組み込んでいますが、我々は T_i 全体を保持し、その前に R_i^t を挿入することで、元のテキスト情報を保持するこ

(b) The first part of the Japanese PDF of case 5.



Figure 3. MMRL inference process, where different tasks utilize distinct features.

the primary focus on representation features to preserve pre-trained knowledge. Specifically, the projection layer for the representation tokens is trainable, while that for the class token remains fixed. For the image encoder $\mathcal{V}$, after passing through L transformer layers, we obtain the output $c_L \in \mathbb{R}^{d_v}$ for the class token and $R_L^v \in \mathbb{R}^{K\times d_v}$ for the K representation tokens. The final output of the representation tokens, r_L, is derived by averaging across the K tokens,

$$r_L = \text{Mean}(R_L^v)$$

where $r_L \in \mathbb{R}^{d_v}$. We then apply the patch projection layers to map the outputs of both the class and representation tokens into the common V-L latent space, yielding the class features f_c and representation features f_r.

$$f_c = P_v^c(c_L) \quad f_r = P_v^c(r_L)$$

Here, P_v^c is the original, frozen patch projection layer of CLIP for class features, while P_v^r for representation features is trainable.
For the text encoder $\mathcal{W}$, following the sequential nature of text, we map the EOT token e_L—as in the original CLIP model—after processing through L transformer layers into the common V-L space, yielding the text features.

$$w = P_t(e_L)$$

With the image features f_c, f_r, and the text classifiers $\{w_c\}_{c=1}^{C}$ for C classes, we apply cross-entropy loss to separately optimize the class and representation features,

$$\mathcal{L}_{ce}^{c} = -\sum_{c}^{C} y_c \log p(y = c \mid f_c)$$

$$\mathcal{L}_{ce}^{r} = -\sum_{c}^{C} y_c \log p(y = c \mid f_r)$$

where $y_c = 1$ if the image x belongs to class c, and $y_c = 0$ otherwise. To further preserve the generalization of class features, we maximize the cosine similarity between (f_c, w) and the frozen CLIP features (f_0, w_0), explicitly guiding the training trajectory,

$$\mathcal{L}_{cos}^{v} = 1 - \frac{f_c \cdot f_0}{|f_c||f_0|} \quad \mathcal{L}_{cos}^{t} = 1 - \frac{1}{C}\sum_{c}^{C} \frac{w^c \cdot w_0^c}{|w^c||w_0^c|},$$

(c) The second part of the Chinese PDF of case 5.

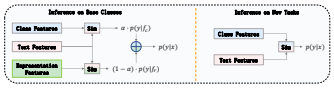


Figure 3. MMRL 推論プロセス、異なるタスクが異なる特徴を利用する。

とを目指します。

$$\begin{aligned}
[b_i, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, T_{i-1}, e_{i-1}]) \quad i = 1, \ldots, J-1 \\
[b_i, _, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, R_{i-1}^t, T_{i-1}, e_{i-1}]) \\
&\qquad i = J, \ldots, L-1 \\
[b_i, R_i^t, T_i, e_i] &= \mathcal{W}_i([b_{i-1}, R_{i-1}^t, T_{i-1}, e_{i-1}]) \quad i = L
\end{aligned}$$

テキストエンコーダの自己回帰的性質により、埋め込み長が増加するため、注意マスク行列を調整していることに注意してください。

3.2.3. 表現学習

表現学習は、表現トークンを活用してデータセットへの適応を行うことを目的としており、一方でクラス・トークンは元の CLIP が持つ事前学習済みの知識を保持します。学習および推論の両方において一般化を維持することを目的とした一連の戦略を通じて、MMRL はさまざまなタスクに柔軟な推論を可能にします。以下にその詳細を示します。

- Training Phase: 表現トークンおよび元のクラス・トークンの特徴を最適化し、事前学習済みの知識を保持するために主に表現特徴に焦点を当てる。具体的には、表現トークン用の投影層は学習可能であるのに対し、クラス・トークン用の投影層は固定されている。画像エンコーダ $\mathcal{V}$ において、L 層のトランスフォーマーを通過した後、クラス・トークンの出力 $c_L \in \mathbb{R}^{d_v}$ と K 個の表現トークンの出力 $R_L^v \in \mathbb{R}^{K\times d_v}$ を得る。表現トークンの最終出力 r_L は、K 個のトークンを平均化することで得られる。

$$r_L = \text{Mean}(R_L^v)$$

ここで、$r_L \in \mathbb{R}^{d_v}$。次に、クラスおよび表現トークンの出力を共通の V-L 潜在空間にマッピングするためにパッチ投影層を適用し、クラス特徴 f_c および表現特徴 f_r を得る。

$$f_c = P_v^c(c_L) \quad f_r = P_v^r(r_L)$$

ここで、P_v^c は CLIP のクラス特徴用の元の凍結されたパッチ投影層であり、P_v^r は表現特徴用の学習可能な層である。
テキストエンコーダ $\mathcal{W}$ においては、テキストの逐次的性質に従い、元の CLIP モデルと同様に、L 層のトランスフォーマー処理後の EOT トークン e_L を共通の V-L 空間にマッピングし、テキスト特徴を得る。

$$w = P_t(e_L)$$

画像特徴 f_c、f_r および C クラスのテキスト分類器 $\{w_c\}_{c=1}^{C}$ により、クラス特徴と表現特徴を別々に最適化するためにクロスエントロピーロスを適用する。

$$\mathcal{L}_{ce}^{c} = -\sum_{c}^{C} y_c \log p(y = c \mid f_c)$$

$$\mathcal{L}_{ce}^{r} = -\sum_{c}^{C} y_c \log p(y = c \mid f_r)$$

ここで、画像 x がクラス c に属する場合は $y_c = 1$、それ以外は $y_c = 0$。さらに、クラス特徴の一般化を保持するために、(f_c, w) と凍結された CLIP 特徴 (f_0, w_0) のコサイン類似度を最大化し、学習の軌道を明示的に誘導する。

$$\mathcal{L}_{cos}^{v} = 1 - \frac{f_c \cdot f_0}{|f_c||f_0|} \quad \mathcal{L}_{cos}^{t} = 1 - \frac{1}{C}\sum_{c}^{C} \frac{w^c \cdot w_0^c}{|w^c||w_0^c|},$$

最終的な MMRL 損失関数は次のようになる。

$$\mathcal{L}_{MMRL} = \alpha\mathcal{L}_{ce}^{v} + (1-\alpha)\mathcal{L}_{ce}^{t} + \lambda(\mathcal{L}_{cos}^{v} + \mathcal{L}_{cos}^{t})$$

ここで、α は特徴間のバランスを制御し、λ はペナルティ係数である。

- Testing on Base Classes: 学習中に観測されたインディストリビューションクラスに対しては、データセット固有の表現特徴と、一般化能力を保持したクラス特徴を組み合わせる。インディストリビューションのテストサンプル x が c 番目のクラスに属する確率は次のように計算される。

$$p(y = c \mid x) = \alpha \cdot p(y = c \mid f_v) + (1-\alpha) \cdot p(y = c \mid f_r)$$

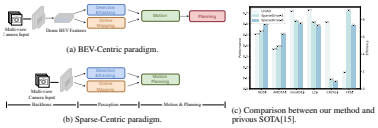
ここで、f_v と f_r はそれぞれクラス・トークンおよび表現トークンから抽出された特徴である。

- Testing on Novel Classes: 学習中に観測されていないクラスや新しいデータセットに対しては、一般化された知識を保持するクラス・トークンのみに依拠する。

$$p(y = c \mid x) = p(y = c \mid f_v)$$

(d) The second part of the Japanese PDF of case 5.

Figure 9: Case 5 demonstrates the performance of LaTeXTrans on the En-Ja task



(a) BEV-Centric paradigm.

(b) Sparse-Centric paradigm.

(c) Comparison between our method and privous SOTA[15].

Figure 1: The comparison of various end-to-end paradigms. (a) The BEV-Centric paradigm. (b) The proposed Sparse-Centric paradigm. (c) Performance and efficiency comparison between (a) and (b).

motion prediction and planning should consider the high-order and bidirectional interactions among road agents. However, previous methods typically adopt a sequential design for motion prediction and planning, ignoring the impact of ego vehicle on surrounding agents. (2) Accurate prediction for future trajectories requires semantic information for scene understanding, and geometric information to predict future movement of agents, which is applicable to both motion prediction and planning. While these information are extracted in upstream perception tasks for surrounding agents, it is overlooked for ego vehicle. (3) Both motion prediction and planning are multi-modal problems with inherent uncertainty, but previous methods only predict deterministic trajectory for planning.

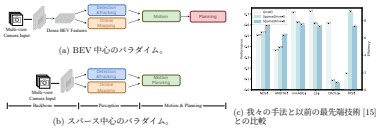
To this end, we propose SparseDrive, a Sparse-Centric paradigm as shown in Fig. 1b. Specifically, SparseDrive is composed of a symmetric sparse perception module and a parallel motion planner. With the decoupled instance feature and geometric anchor as complete representation of one instance (a dynamic road agent or a static map element), **Symmetric Sparse Perception** unifies detection, tracking and online mapping tasks with a symmetric model architecture, learning a fully sparse scene representation. In **Parallel Motion Planner**, a semantic-and-geometric-aware ego instance is first obtained from ego instance initialization module. With the ego instance and surrounding agent instances from sparse perception, motion prediction and planning are conducted simultaneously to get multi-modal trajectories for all road agents. To ensure the rationality and safety for planning, a hierarchical planning selection strategy that incorporating a collision-aware rescore module is applied to select the final planning trajectory from multi-modal trajectory proposals.

With above effective designs, SparseDrive unleashes the great potential of end-to-end autonomous driving, as shown in Fig. 1c. Without bells and whistles, our base model, SparseDrive-B, greatly reduces the average L2 error by 19.4% (0.58m vs. 0.72m) and collision rate by 71.4% (0.06% vs. 0.21%). Compared with previous SOTA (state-of-the-art) method UniAD[15], our small model, SparseDrive-S achieves superior performance among all tasks, while running 7.2× faster for training (20 h vs. 144 h) and 5.0× faster for inference (9.0 FPS vs. 1.8 FPS).

The main contribution of our work are summarized as follows:

- We explore the sparse scene representation for end-to-end autonomous driving and propose a Sparse-Centric paradigm named SparseDrive, which unifies multiple tasks with sparse instance representation.
- We revise the great similarity shared between motion prediction and planning, correspondingly leading to a parallel design for motion planner. We further propose a hierarchical planning selection strategy incorporating a collision-aware rescore module to boost the planning performance.
- On the challenging nuScenes[1] benchmark, SparseDrive surpasses previous SOTA methods in terms of all metrics, especially the safety-critical metric collision rate, while keeping much higher training and inference efficiency.

(a) The first part of the English PDF of case 6.



(a) BEV 中心のパラダイム。

(b) スパース中心のパラダイム。

(c) 我々の手法と以前の最先端技術 [15] との比較

Figure 1: さまざまなエンドツーエンドのパラダイムの比較。(a) BEV 中心のパラダイム。(b) 提案されたスパース中心のパラダイム。(c) (a) と (b) の性能および効率の比較。

測することを目指して、モーション予測と計画は、道路エージェント間の高次および双方向的な相互作用を考慮する必要があります。しかし、従来の手法は通常、モーション予測と計画に対して順次的な設計を採用しており、自車両が周囲のエージェントに与える影響を無視しています。(2) 将来の軌道を正確に予測するためには、シーン理解のためのセマンティック情報と、エージェントの将来の動きを予測するための幾何学的情報が必要です。これらの情報は、周囲のエージェントに対する上流の知覚タスクで抽出されますが、自車両に対しては見落とされています。(3) モーション予測と計画は、いずれも不確実性を内包するマルチモーダルな問題ですが、従来の手法では計画に対して決定論的な軌道のみを予測しています。

これに対して、我々は SparseDrive、スパース中心のパラダイムを提案します。図 1b に示すように、SparseDrive は対称的なスパース知覚モジュールと並列モーションプランナーで構成されています。インスタンス特徴と幾何学的のアンカーを、1 つのインスタンス（動的道路エージェントまたは静的マップ要素）の完全な表現として分離し、**対称的スパース知覚**は、検出、追跡、オンラインマッピングタスクを対称的なモデルアーキテクチャで統一し、完全なスパースシーン表現を学習します。**並列モーションプランナー**では、セマンティックおよび幾何学的に認識された自車両インスタンスが、最初に自車両インスタンス初期化モジュールから得られます。自車両インスタンスと周囲のエージェントインスタンスがスパース知覚から得られ、モーション予測と計画は同時に実行され、すべての道路エージェントに対してマルチモーダルな軌道が得られます。計画の合理性と安全性を確保するために、衝突認識型リスコアモジュールを組み込んだ階層的な計画選択戦略が適用され、マルチモーダルな軌道提案から最終的な計画軌道が選択されます。

これらの効果的な設計により、SparseDrive はエンドツーエンドの自動運転の大きな可能性を解き放ちます。図 1c に示すように、余計な装飾なしで、我々の基本モデルである SparseDrive-B は、平均 L2 誤差を 19.4%（0.58m vs. 0.72m）減少させ、衝突率を 71.4%（0.06% vs. 0.21%）削減しました。従来の SOTA（最先端技術）手法である UniAD[15] と比較して、我々の小型モデルである SparseDrive-S は、すべてのタスクで優れた性能を発揮し、トレーニングでは 7.2×、推論では 5.0× 速く実行されます（トレーニング時間：20 時間 vs. 144 時間、推論速度：9.0 FPS vs. 1.8 FPS）。

我々の研究の主な貢献は以下の通りです：

- エンドツーエンドの自動運転におけるスパースなシーン表現を探求し、スパースなインスタンス表現を用いて複数のタスクを統一するスパース中心のパラダイムである SparseDrive を提案します。
- モーション予測と計画の間に存在する大きな類似性を修正し、それに対応する形でモーションプランナーの並列設計を提案します。さらに、計画性能を向上させるために、衝突認識型リスコアモジュールを組み込んだ階層的な計画選択戦略を提案します。
- 難易度の高い nuScenes[1] ベンチマークにおいて、SparseDrive は全ての指標で従来の最先端技術（SOTA）を上回り、特に安全性に関わる指標である衝突率において優れた結果を示し、さらにトレーニングおよび推論効率が大幅に向上しています。

(b) The first part of the Japanese PDF of case 6.

scores and the offsets of anchor boxes in the output layer. The temporal decoders have two additional multi-head attention layers: the temporal cross-attention between temporal instances from last frame and current instances, and the self-attention among current instances. In multi-head attention layer, the anchor boxes are transformed into high-dimensional anchor embedding $E_d \in \mathbb{R}^{N_d \times C}$, and serve as the positional encoding.

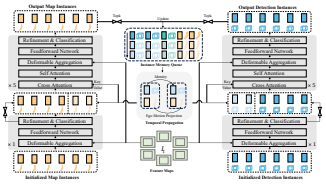


Figure 3: Model architecture of symmetric sparse perception, which unifies detection, tracking and online mapping in a symmetric structure.

Sparse Online Mapping. Online mapping branch shares the same model structure with detection branch except different instance definition. For static map element, the anchor is formulated as a polyline with N_p points:

$$\{x_0, y_0, x_1, y_1, ..., x_{N_p-1}, y_{N_p-1}\}.$$

Then all the map elements can be represented by map instance features $F_m \in \mathbb{R}^{N_m \times C}$ and anchor polylines $L_m \in \mathbb{R}^{N_m \times N_p \times 2}$, where N_m is the number of anchor polylines.

Sparse Tracking. For tracking, we follow the ID assignment process of Sparse4Dv3[33]: once the detection confidence of an instance surpasses a threshold T_{thresh}, it is locked onto a target and assigned with an ID, which remains unchanged throughout temporal propagation. This tracking strategy does not need any tracking constraints, resulting in an elegant and simple symmetric design for sparse perception module.

3.3 Parallel Motion Planner

As shown in Fig. 4, the parallel motion planner consists of three parts: ego instance initialization, spatial-temporal interactions and hierarchical planning selection.

Ego Instance Initialization. Similar to surrounding agents, ego vehicle is represented by ego instance feature $F_e \in \mathbb{R}^{1 \times C}$ and ego anchor box $B_e \in \mathbb{R}^{1 \times 11}$. While ego feature is typically randomly initialized in previous methods, we argue that the ego feature also requires rich semantic and geometric information for planning, similar to motion prediction. However, the instance features of surrounding agents are aggregated from image feature maps I, which is not feasible for ego vehicle, since ego vehicle is in blind area of cameras. Thus we use the smallest feature map of front camera to initialize the ego instance feature:

$$F_e = \text{AveragePool}(I_{front,S}) \tag{1}$$

There are two advantages in doing so: the smallest feature map has already encoded the semantic context of the driving scene, and the dense feature map serves as a complementary for sparse

(c) The second part of the Chinese PDF of case 6.

スパース検出ブランチは、N_{dec} 個のデコーダーで構成され、1 つの非時間的デコーダーと $N_{dec}-1$ 個の時間的デコーダーが含まれます。各デコーダーは、特徴マップ I、インスタンス特徴 F_d、およびアンカーボックス B_d を入力として取り、更新されたインスタンス特徴と洗練されたアンカーボックスを出力します。非時間的デコーダーはランダムに初期化されたインスタンスを入力として受け取り、時間的デコーダーの入力は現在のフレームと過去のフレームの両方から来ます。具体的には、非時間的デコーダーは、変形可能な集約、フィードフォワードネットワーク（FFN）、および洗練と分類のための出力層の 3 つのサブモジュールを含みます。変形可能な集約モジュールは、アンカーボックス B_d 周辺に固定または学習可能なキーポイントを生成し、それらを特徴マップ I に投影して特徴をサンプリングします。インスタンス特徴 F_d はサンプリングされた特徴と加算することによって更新され、出力層でアンカーボックスの分類スコアとオフセットを予測する役割を担います。時間的デコーダーには、2 つの追加のマルチヘッドアテンション層があります：前フレームと現在のインスタンス間の時間的クロスアテンション、および現在のインスタンス間の自己アテンション。マルチヘッドアテンション層では、アンカーボックスは高次元のアンカー埋め込み $E_d \in \mathbb{R}^{N_d \times C}$ に変換され、位置エンコーディングとして機能します。

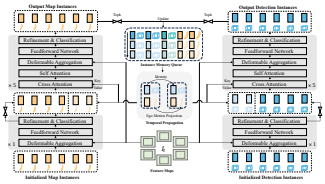


Figure 3: 対称的なスパース知覚のモデルアーキテクチャであり、検出、トラッキング、オンラインマッピングを対称的な構造で統一しています。

スパースオンラインマッピング。オンラインマッピングブランチは、検出ブランチと同じモデル構造を共有しますが、インスタンス定義が異なります。静的マップ要素の場合、アンカーは N_p 点を持つポリラインとして定式化されます：

$$\{x_0, y_0, x_1, y_1, ..., x_{N_p-1}, y_{N_p-1}\}.$$

次に、すべてのマップ要素は、マップインスタンス特徴 $F_m \in \mathbb{R}^{N_m \times C}$ とアンカーポリライン $L_m \in \mathbb{R}^{N_m \times N_p \times 2}$ によって表現されます。ここで、N_m はアンカーポリラインの数です。

スパーストラッキング。トラッキングについては、Sparse4Dv3[33] の ID 割り当てプロセスに従います：インスタンスの検出信頼度が閾値 T_{thresh} を超えると、ターゲットにロックされ、ID が割り当てられ、時間的伝播を通じてその ID は変更されません。このトラッキング戦略は、トラッキング制約を必要とせず、スパース知覚モジュールのための優雅でシンプルな対称的な設計を実現します。

3.3 並列モーションプランナー

図 4 に示すように、並列モーションプランナーは 3 つの部分で構成されています：自車両インスタンスの初期化、空間・時間的相互作用、および階層的な計画選択。

自車両インスタンスの初期化。周囲のエージェントと同様に、自車両は自車両インスタンス特徴 $F_e \in \mathbb{R}^{1 \times C}$ と自車アンカーボックス $B_e \in \mathbb{R}^{1 \times 11}$ によって表現されます。自車特徴は以

(d) The second part of the Japanese PDF of case 6.

Figure 10: Case 6 demonstrates the performance of LaTeXTrans on the En-Ja task

Prompt Template for LLM-score

```
You are a professional translation evaluator. Given an English source paragraph and its
{tgt_language} translation, evaluate the translation quality according to the following
criteria:

Faithfulness: How accurately and completely does the translation convey the meaning of the
source text?
Fluency: Is the translation natural, idiomatic, and grammatically correct in {tgt_language}?
Terminology and Formatting Consistency: Are all technical terms translated correctly and
consistently throughout the paragraph? Is the formatting—such as emphasis, symbols, references,
and structural markers—preserved where applicable?
Contextual Coherence: Does the translation maintain logical flow, appropriate pronoun/reference
usage, and contextual consistency across sentences within the paragraph?

Score each dimension from 0 to 10. Then, compute a final overall score (0 to 10), reflecting the
overall translation quality, and round it to one decimal place.
Only return the final overall score as a number. Do not include explanations, sub-scores, or any
additional content.
```

Figure 11: The LLM uses this prompt, which scores each pair's translation unit one by one.

Prompt Template 1 for Translator

```
You are a professional academic translator specializing in LaTeX-based scientific writing. Your
task is to translate long LaTeX texts (including section titles and content) from English to
{tgt_language}, while strictly maintaining the integrity of LaTeX syntax.

In addition to the LaTeX source, you are provided with:
1. A dynamic summary that condenses the content of all previous sections.
2. A bilingual term dictionary containing domain-specific English–{tgt_language} term pairs.

You must use these resources to ensure translation quality:
- Use the summary to understand the document context, resolve ambiguous expressions, pronouns, or
abstract references, and maintain coherence across sections.
- Strictly follow the term dictionary. If an English term in the source appears in the
dictionary, you **must** use the corresponding {tgt_language} translation from the dictionary
without modification.

Please strictly follow the translation requirements below:
1. Only translate the natural language content while keeping all LaTeX commands, environments,
references, mathematical expressions, and labels unchanged.
2. Section headings (e.g. natural content enclosed in {} in section identifiers like \section{},
\subsection{}, and \subsubsection{}) must also be translated, but their LaTeX syntax must remain
unchanged.
3. Do not translate or modify the following LaTeX elements: Control commands: \label{},
\cite{}, \ref{}, \textbf{}, \emph{}, etc. Mathematical environments: $...$, [...],
\begin{equation}...\end{equation}, etc. Any parameter or argument that includes numerical
values with LaTeX layout units such as: em, ex, in, pt, pc, cm, mm, dd, cc, nd, nc, bp, sp.
Example: \vspace{-1.125cm} or [scale=0.58] → leave such expressions completely unchanged.
4. Do not change the writing of special characters, such as \%, \#, \&, etc., to ensure that the
translated text is accurate.
5. The final output must be a valid and compilable LaTeX document.
6. Ensure that the translated text is accurate, coherent, and follows academic writing
conventions in the target language. Maintain consistent academic terminology and use standard
abbreviations where appropriate.
7. Directly output only the translated LaTeX code without any additional explanations,
formatting markers, or comments such as "latex".
8. <PLACEHOLDER_CAP_...>, <PLACEHOLDER_ENV_...>, <PLACEHOLDER_..._begin> and
<PLACEHOLDER_..._end> are placeholders for artificial environments or captions. Please do not
let them affect your translation and keep these placeholders after translation.

You are expected to combine semantic understanding (from the summary), precise terminology usage
(from the term dictionary), and strict LaTeX fidelity to produce a high-quality translation.
```

Figure 12: Prompt template 1 for Translator, the Translator uses this prompt to initially translate the translation unit.

Prompt Template 2 for Translator

```
You are a professional academic translator and LaTeX translation corrector. Your task is to
revise and improve machine-translated LaTeX academic texts based on three components provided
by the user: the original English LaTeX source ([Original]), the existing {tgt_language}
translation ([Translation]), and the error information describing the issue(s) ([Error
Reports]). Your revision must strictly preserve LaTeX syntax integrity and comply with the
following rules.

1. Only translate the natural language content while keeping all LaTeX commands, environments,
references, mathematical expressions, and labels unchanged.
2. Section headings (e.g. natural content enclosed in {} in section identifiers like \section{},
\subsection{}, and \subsubsection{}) must also be translated, but their LaTeX syntax must remain
unchanged.
3. Do not translate or modify the following LaTeX elements: Control commands: \label{},
\cite{}, \ref{}, \textbf{}, \emph{}, etc. Mathematical environments: $...$, [...],
\begin{equation}...\end{equation}, etc. Any parameter or argument that includes numerical
values with LaTeX layout units such as: em, ex, in, pt, pc, cm, mm, dd, cc, nd, nc, bp, sp.
Example: \vspace{-1.125cm} or [scale=0.58] → leave such expressions completely unchanged.
4. Do not change the writing of special characters, such as \%, \#, \&, etc., to ensure that the
translated text is accurate.
5. The final output must be a valid and compilable LaTeX document.
6. Ensure that the translated text is accurate, coherent, and follows academic writing
conventions in the target language. Maintain consistent academic terminology and use standard
abbreviations where appropriate.
7. Directly output only the translated LaTeX code without any additional explanations,
formatting markers, or comments such as "latex".
8. <PLACEHOLDER_CAP_...>, <PLACEHOLDER_ENV_...>, <PLACEHOLDER_..._begin> and
<PLACEHOLDER_..._end> are placeholders for artificial environments or captions. Please do not
let them affect your translation and keep these placeholders after translation.

Only output the corrected LaTeX {tgt_language} translation (revised version of '[Translation]'),
with all changes implemented based on the '[Original]' and '[Error]'. Do not output the original
input, explanations, or any extra content.
```

Figure 13: Prompt template 2 for Translator, the Translator uses this prompt and combines it with the error reports provided by the Validator to re-translate the translation unit.

Prompt Template for Filter

```
You are a LaTeX translation assistant. Your task is to analyze the content inside any LaTeX
environment, regardless of its environment name, and determine whether it should be translated
when translating an academic paper.
Environment names can be custom-defined (e.g., 'mybox', 'resultblock', 'customalgo') and should
be ignored during judgment. Only base your decision on the content itself.

Return 'True' if the content:
- Contains complete or partial sentences written in natural language (e.g., English), such as
explanations, definitions, figure/table captions, theorem statements, or descriptions.
- Helps the reader understand the paper and would lose meaning if left untranslated.

Return 'False' if the content:
- Contains only code, pseudocode, mathematical formulas, drawing instructions (e.g., TikZ),
formatting macros, or raw markup.
- Does not include any human-readable sentences or phrases.

Only output:
- 'True' or 'False'
- No explanations or additional text
```

Figure 14: Prompt template for Filter, the Filter uses this prompt to mark whether the translation unit needs to be translated.

Prompt Template for Terminology Extractor

```
You are an en-{tgt_language} bilingual expert. Given an English source sentence and its
corresponding {tgt_language} translation, your task is to extract all domain-specific terms from
the English sentence, along with their exact translations as they appear in the {tgt_language}
sentence.

These include:
- Technical terms and expressions
- Abbreviations or acronyms (e.g. RL, LM)
- Named entities or model names (e.g. COMET)
- Concept-specific noun phrases (e.g. optimization objective, long-term reward)

The translation must match exactly how it appears in the {tgt_language} sentence. Do not invent
or guess new translations.
Output the result as a list of aligned term pairs in the following format:
"<English Term>" - "<{tgt_language} Translation>"

If there are no such terms, output: 'N/A'.
```

Figure 15: Prompt template for Terminology Extractor, Terminology Extractor uses this prompt to extract terms from each translation unit.

Prompt Template for Summarizer

```
You are an academic summarization assistant designed to maintain an evolving semantic summary to
support consistent and coherent machine translation of a long scientific document.
You will be given two inputs:
1. The current summary ('prev_summary'), which reflects key information from all previously seen
sections.
2. A new section of the document ('new_section') that has not yet been summarized.

Your task is to:
- Integrate the new section's key content into the current summary, producing an updated summary.
- Preserve previously summarized information that remains relevant.
- Add any new findings, concepts, methods, or referential expressions introduced in the new
section.
- Ensure the summary remains concise, information-dense, and suitable for machine translation
context support.
- Do not repeat redundant content; merge semantically where possible.
Use clear, academic English. The updated summary should be no more than 300 words.
```

Figure 16: Prompt template for Summarizer, the Summarizer uses this prompt to maintain the summary of the previous text.

"Did You CoSeCh?": Faithfulness- and Coherence-Aware Auditing of Content Selection in Poster Generation

Aaditya Deshpande
TCS Research
aaditya.deshpande@tcs.com

Savita Bhat
TCS Research
savita.bhat@tcs.com

Shirish Karande
TCS Research
shirish.karande@tcs.com

Abstract

As generative models continue to evolve, the automated generation of scientific posters from research papers has advanced rapidly. However, its evaluation almost exclusively focuses on whether the poster layout is visually well-organized and whether it covers the paper's content. We argue that these approaches overlook a prior question: whether the selection of content for the poster is sound. A poster can be complete and well-formed, yet distort findings, break internal coherence, or spend its limited space on duplicated information. We introduce **CoSeCh** (**Co**ntent **Se**lection **Ch**eck), a suite of automatic rule-based, LLM-assisted diagnostics that probe the selected content on four distinct dimensions of quality: faithfulness (are the qualifiers that bound a claim preserved?), coherence (do selected units remain interpretable once their dependencies are removed?), recoverability (can the paper's core claims be reconstructed from the selection?), and efficiency (is the budget spent on distinct content?). Applying CoSeCh to 5 poster generators ranging from a random selection baseline to state-of-the-art works like P2P and Paper2Poster, we find that content selection quality is strongly generator-dependent and that generators fail along different axes. Some generators preserve the qualifiers for the claims better than human authors, while others' selection is no more coherent than random choice. Existing coverage-focused evaluation metrics like PaperQuiz cannot detect these failures. The four axes are near-independent, showing that no single coverage measure can capture them.

1 Introduction

Scientific posters serve as a visual representation of research papers and aim to present key information in an engaging way. Research papers usually consist of complex structures and dense details, which require corresponding posters to summarize them intuitively and concisely. A good poster combines visual and textual elements to make complex information visually appealing and easier to understand. However, manual poster generation requires domain knowledge and design expertise along with significant time and effort (Wang et al., 2024; Qiang et al., 2019). Consequently, automatic poster generation presents a critical research problem for disseminating niche knowledge and reducing the manual burden on researchers.

Early works like (Xu and Wan, 2021a; Qiang et al., 2019) extracted text and figures to heuristically arrange them in poster panels. In addition, a relevant body of work focused on sub-tasks such as layout generation (Wang et al., 2024), text summarization (Saxena et al., 2025; Gao et al., 2025), and panel attribute inference (Huang et al., 2022). However, these methods often result in posters with quality issues such as content overflow (Qiang et al., 2019). With better and more powerful generative models, recent poster generation approaches have improved significantly. Recent methods, including P2P (Sun et al., 2026), PosterForest (Choi et al., 2026), and Paper2Poster (Pang et al., 2026) use multi-agent pipelines along with task decomposition to provide focused solutions for sub-problems such as parsing, content summarization, layout planning, and rendering. This progress has centered on the generation part of the task, such as better parsing, planning, layout, and styling. Consequently, the evaluation has also been focused on the rendered poster by measuring visual quality, design, and whether the poster covers the core content of the paper, typically using VLM judges or content-recall quizzes (PaperQuiz (Pang et al., 2026)). In most cases, the evaluation overlooks the step every one of these systems performs internally and silently, i.e., content selection. The choice of which content units of the paper should appear on the poster is the critical step that influences the quality and soundness of the rendered poster.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 344–356
October 24, 2026 ©2026 Association for Computational Linguistics

A poster compresses the rich source content of the paper into a visually coherent single page, requiring the selection of high-quality content. Therefore, content selection is where the most consequential information decisions are made, and current evaluation approaches are blind to whether these decisions are good. Most current work performs content selection as a relatively unexamined step, and current evaluation assesses the output's coverage and appearance, not whether the selection is distorted, hallucinated, or broken. A poster that scores well on coverage and design can still report a result stripping the bounding condition or retain a figure whose context is dropped. These are failures specific to selection and not to rendering. We argue that there exists no poster metric designed to detect them. As a result, a whole class of content errors passes the current evaluation unmeasured.

To address this gap, we introduce 'CoSeCh', a suite of automatic diagnostics that evaluate a generator's selected content directly, against the source paper. CoSeCh probes four dimensions that a sound selection should have: 1) faithfulness, 2) coherence, 3) recoverability, and 4) Efficiency. We present each of them with a rule-based heuristic whose every judgment is inspectable and interpretable rather than the output of a learned model. Applying CoSeCh to five generators with ≈ 100 papers, we find that selection quality is multi-dimensional and generator-specific. The property that aligns with existing coverage-based evaluation measures barely distinguishes between the generators. Faithfulness and coherence exhibit significant variations. With these four axes near-independent, no existing coverage measure captures them.

We summarize our contributions as follows.

- We formulate content-selection quality as a distinct, measurable target for poster generation, separate from layout and rendering;
- We introduce CoSeCh, a suite of four automatic rule-based diagnostics - faithfulness, coherence, recoverability, and efficiency - each auditable and interpretable;
- We show empirically that current generators diverge from human selection distinctly and in specific ways - some exceeding human faithfulness or coherence, others no better than random — and that existing coverage-oriented evaluation cannot detect these differences.

2 Related Work

Paper-to-poster generation Early approaches to automatic poster generation relied on heuristic panel arrangement over extracted text and figures (Xu and Wan, 2021a; Qiang et al., 2019). Subsequent work decomposed the problem into sub-tasks — layout generation (Wang et al., 2024), poster-oriented summarization (Saxena et al., 2025; Gao et al., 2025), and panel attribute inference building on document-layout pretraining (Huang et al., 2022) — but stopped short of an end-to-end system. The current generation of methods instead uses multi-agent LLM pipelines that jointly handle parsing, content summarization, layout planning, and rendering: P2P (Sun et al., 2026), Paper2Poster (Pang et al., 2026), PosterForest (Choi et al., 2026), PosterGen (Zhang et al., 2025), and Any2Poster (Vinaykumar et al., 2026) all follow this template, differing mainly in how planning is decomposed across agents and how aesthetic constraints are enforced during rendering. Across this line of work, content selection is performed as an internal, unexamined step inside the planning agent(s) rather than as a task with its own success criteria.

Evaluation Evaluation has tracked the same generation-centric focus. VLM-judge protocols and layout/aesthetic metrics (Wang et al., 2024) score visual organization and design quality, while content-recall instruments such as PaperQuiz (Pang et al., 2026) and the P2PEval benchmark introduced alongside P2P (Sun et al., 2026) test whether a reader can answer questions about the paper after seeing the poster. These instruments are valuable for measuring coverage — whether the important content made it onto the page in some form — but they are not designed to detect distortion of that content: a poster can preserve enough information to pass a recall quiz while still stripping a claim's qualifying condition, orphaning a figure from the text that contextualizes it, or duplicating a result across a table and a caption. No existing metric targets the selection step directly, independent of layout and rendering quality.

Content Selection Content selection is a critical step to create coherent and faithful posters from academic papers. Accordingly, early works treat this as a special form of summarization approaches such as probabilistic graphical models (Paramita and Khodra, 2016) and Naive Bayes classification (Qiang et al., 2016, 2019). Later, neural systems

have been proposed to extract representative text and figures from the papers in (Jaisankar et al., 2024; Xu and Wan, 2021b). Recent advances in LLM have enabled LLM-based poster generation pipelines which consist of content selection as an internal and implicit planning stage before rendering. Across all recent works like Paper2Poster (Pang et al., 2026), PosterGen (Zhang et al., 2025), P2P (Sun et al., 2026), PosterForest (Choi et al., 2026), and Any2Poster (Vinaykumar et al., 2026), content selection is optimized for coverage and visual aesthetics, or delegated to an LLM planner, and finally evaluated only through the quality of the rendered poster.

Faithfulness Several works develop automatic and human evaluation protocols for factual consistency in generated summaries. FENICE (Scirè et al., 2024) extracts atomic claims from a summary and aligns them with the source using NLI, with coreference resolution applied across multiple granularities. TRUE (Honovich et al., 2022) brings eleven factual-consistency datasets spanning summarization, dialogue, and paraphrasing into a common binary evaluation setting, and finds that large-scale NLI and QA-based metrics are the strongest and most complementary detectors of inconsistency. FABLES (Kim et al., 2024) and LongEval (Krishna et al., 2023) consider book-length summarization, using human annotations to verify claims against long source documents. These methods evaluate the faithfulness after summarizing. They do not evaluate whether the earlier selection decision of which source content should be included is faithful.

Across poster generation, evaluation, content selection, and faithfulness one gap recurs: generators treat selection as an unexamined internal step; evaluation assesses the rendered poster's coverage rather than whether the selection itself and not summarization distorts or breaks the paper's claims. No prior work evaluates the selection itself for faithfulness, coherence, recoverability, and efficiency — the gap CoSeCh fills with an auditable, reference-free diagnostic suite.

3 Proposed Framework

In this section, we formally define four diagnostic dimensions.

3.1 Problem Formulation

We consider a paper as a set of content units $U = \{u_1, u_2, \cdots, u_n\}$, where each unit is a text block, figure, table, equation, or graph/chart, with its own text or caption, its section context, and its modality. When a poster generator performs content selection, it chooses a subset $S \subseteq U$ to place on the poster under an implicit space budget B where $\sum_{u \in S} b(u) \leq B$ and $b(u)$ is a unit's space cost, discarding the remainder of the units. Given a source paper with units U, and a set of selected content units S, CoSeCh evaluates S along four axes that probe the soundness of a selection: 1) faithfulness - it does not distort the paper's claims, 2) coherence - its units remain interpretable within S, 3) recoverability - its core content can be reconstructed, and 4) efficiency - the limited space budget is spent on distinct content. All probe instruments are rule-based with minimal LLM-assistance and deterministic by design. We use a language model for perception tasks such as extracting claim-qualifier pairs, mapping a generator's selected text onto its source paragraph as a fallback-measure, answering recoverability questions, etc but never for scoring itself. This design keeps all scoring auditable and interpretable (More in Section 5.6). Each axis probe is a rule-based function $\Phi(S, U) \in [0, 1]$ and is reference-free (it is computed from the selection and the source paper alone). When a human-authored poster $H \subseteq U$ is available, we report $\Phi(H, U)$ as an oracle or upper anchor. None of the CoSeCh metrics requires it, so the suite applies to any paper, including those with no gold poster.

3.2 Faithfulness

Faithfulness measures whether the selection S preserves the qualifiers that bound a paper's claim, or presents results without the limiting conditions. For example, if a finding "improves X under condition Y" is presented as unconditional "improves X", it is categorized as unfaithful to the source paper. To investigate this, we extract claim-qualifier pairs from the source. The extraction of qualifiers is restricted to the scope, conditional, statistical, magnitude, and assumption qualifiers. The Author-retained qualifiers in the abstract serve as a reference for faithful compression.

Formally, the faithfulness is quantified as below: Let $C(U)$ be the set of claim-qualifier pairs (c_i, q_i) where claim unit u_c and qualifier unit u_q are distinct

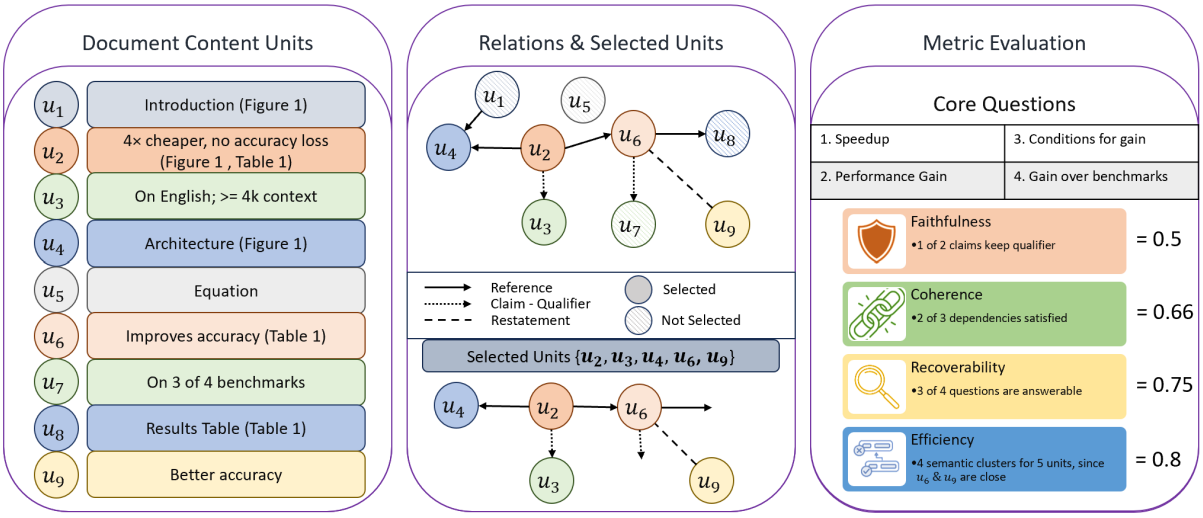


Figure 1: A worked example of the four-axis diagnostic on a toy paper. *Left*: The paper's nine units with their type and content. *Center*: the relation graph - reference edges (solid), claim–qualifier links (dotted), restatement (dashed) — with the selected content units. *Right*: each axis scored on the selection. The scores diverge, illustrating that the axes capture distinct, non-redundant properties. Values are illustrative, not experimental results.

($u_c \neq u_q$). A qualifier is dropped when its claim is selected but the qualifier is not. We quantify the qualifier-drop rate over the selected, caveated claims:

$$D(S) = \{(c, q) \in C(U) : u_c \in S \wedge u_q \notin S\} \quad (1)$$

and the *qualifier drop_rate* is

$$QDR(S) = \frac{|D(S)|}{|\{(c, q) \in C(U) : u_c \in S\}|} \quad (2)$$

$$Faithfulness(S) = F(S) = 1 - QDR(S) \quad (3)$$

The qualifier drop rate is quantified as the ratio of selected claims without the respective qualifier not selected to the total claims in the selected content units. The denominator is the number of selected claims that have a qualifier — the claims actually at risk of being stripped — rather than all selected claims, so uncaveated claims neither inflate nor deflate the score. This is validated by manually checking extracted claim–qualifier pairs on a sample of papers for precision and separability. We report faithfulness against human reference S_h whenever available, since human posters also compress, making the comparison relative rather than absolute.

3.3 Coherence

Coherence measures whether a selected unit remains interpretable given the rest of the selection or is left orphaned by the removal of units it depends on. For example, a figure whose defining unit is dropped or a claim citing a table that is not selected. We build a dependency graph over units using explicit cross-references where a unit mentioning a figure, table, equation, or section induces an edge to that referent.

The coherence is quantified using the dangling dependency rate, the fraction of dependencies of selected units whose referent was not selected to the total dependencies in the selected set. The coherence score is defined as 1 - dangling rate.

Formally, let $E \subseteq U \times U$ be explicit reference edges, $(u, v) \in E$ meaning u references v. Dangling dependency rate is defined as

$$DDR(S) = \frac{|\{(u, v) \in E : u \in S \wedge v \notin S\}|}{|\{(u, v) \in E : u \in S\}|}, \quad (4)$$

and coherence score is $1 - DDR(S)$. $DDR(S)$ becomes undefined when S contains no unit with an explicit outgoing dependency edge; such selections are excluded from corpus-level aggregation and their frequency is reported as a separate statistic rather than assigned a default value - assigning 0 or 1 would assert coherence or incoherence where there is in fact no evidence either way.

Edges are directed by construction — dropping u's prerequisite orphans u, but dropping a unit that merely refers to a selected unit does not — so only the numerator's asymmetric condition is penalized.

We manually validate this by measuring the precision of the reference edges and the corresponding referents. We report coherence against S_h if available and the random floor.

3.4 Recoverability

Recoverability measures whether the paper's core claims can be reconstructed from the selected content units. We use a set of core-claim questions extracted using a protocol similar to PaperQuiz. The questionnaire is answered by a reader model that is provided with only the selected content units. Recoverability is quantified as the accuracy of responses derived from the selection, and information loss is the drop from full-paper accuracy to selection accuracy. The quiz protocol also allows for a systematic categorization of the information loss based on the type of question and the corresponding units of content.

Formally, for a question set $Q(U)$ on the main claims of the paper, each question with the gold answera_q^*, and a reader R answering question q only from the selected content, i.e. $R(q|S)$,

$$Rec(S) = \frac{1}{|Q|} \sum_{q \in Q} 1[R(q|S) = a_q^*], \qquad (5)$$

with information loss $\Delta(S) = Rec(U) - Rec(S)$ against the full paper ceiling.

3.5 Efficiency

Efficiency measures how much of the space budget is allocated to distinct content. It is the opposite of *Redundancy*, which measures how much of the space budget is spent on repetitious content instead of distinct information. For example, the same result is stated in a table and is restated in text, with both the table and the text in selected units. We use pairwise semantic similarity over selected units, along with the detection of repeated quantities and restated claims. We cluster the selected units by semantic similarity (a threshold τ on pairwise similarity, plus detection of repeated quantities and restated claims), producing k clusters among the selected units $|S|$. We quantify the wasted budget as the fraction of selected units beyond one per cluster:

$$Redundancy(S) = \frac{|S| - k}{|S|}, \qquad (6)$$

$$Efficiency(S) = \frac{k}{|S|} \qquad (7)$$

Because some repetition is a deliberate choice for emphasis rather than a defect, we frame this as budget efficiency rather than an error.

3.6 A Worked Example

We demonstrate the formulations with a worked example presented in Figure 1. The four scores differ, and none predicts the others. For this example, this selection is moderately faithful, moderately coherent, mostly recoverable, and mostly efficient, illustrating that the axes capture distinct properties.

3.7 Independence

As mentioned above, the four diagnostics are computed independently. Further, we investigate whether they measure distinct properties or collapse to a single "*selection quality*" factor. For each pair of (generator, paper), we obtain a vector of four axis scores and compute the pairwise correlation matrix $\rho(\Phi_i, \Phi_j)$ across all instances. Low off-diagonal correlation indicates that a selection's behavior on one axis does not predict its behavior on another. In other words, a faithful selection may still be incoherent, or a completely recoverable selection can be redundant. In such cases, the suite is not reducible to a single number. Alternatively, a high correlation on some pairs would indicate dependency. For example, aggressive compression to address redundancy may degrade faithfulness and coherence together. The claim CoSeCh makes is not that all four axes are mutually orthogonal but that they do not collapse to one dimension. We believe that the correlation matrix is the evidence for that claim. Additionally, the correlation matrix argues that existing coverage-oriented metrics, which are aligned most closely with recoverability/coverage, cannot measure the faithfulness and coherence axes as defined here.

4 Experimental Setup

4.1 Dataset

We evaluate CoSeCh on the Paper2Poster(Pang et al., 2026) dataset which contains 100 source papers paired with author-designed posters from NeurIPS, ICML and ICLR (2022-2024). These author designed posters let us report $\Phi(S_h, U)$ for every axis rather than only inter-generator comparison. A set of selected units is extracted from each generator in isolation from layout. All diagnostic axes are manually validated in a subset of papers.

4.2 Axes Implementation

Mapping generator's selection to superset Each generator selects and renders content in its own format which can reword, truncate, summarize the paper's original sentence. To recover the paper's actual, unsummarized text behind each selected unit, we map every selected unit back onto its source paragraph in the paper's canonical content tree using a majority voting between TF-IDF, n-gram keyword matching. An LLM is called as a fallback, i.e. only when these two mappers fail to map the selected unit to any of the superset units

Faithfulness Claim–qualifier pairs are extracted from the source paper via a single LLM pass, restricted to five types: scope, conditional, statistical, magnitude, and assumption. Since a generator's selected-content text is often a paraphrase of the source, each selected unit is first mapped back onto the paragraph it was drawn from and this original text substitutes for the generator's, falling back to the unit's own (summarized) form only when no confident match exists. Survival is then checked independently for each claim and qualifier in this mapped selection: a claim is present if its tokens exceed a containment threshold against the best-matching sentence, and its qualifier survives if a matching span — verbatim or via a fallback list of marker phrases for the five types — appears within a three-sentence window of that claim. A pair is scored as dropped only when the claim is selected but no such qualifier span is found in its window.

Coherence For each paper, a shared directed dependency graph is built once over the paper's own hierarchical structure, with reference edges detected by regex over figure/table/equation/section mentions; with figure/table nodes attached from one generator's own asset list as a shared caption source so every schema's captions resolve against the same canonical targets. A section/subsection reference is treated as satisfied if either the heading itself or any of its descendant paragraphs was selected, since selection happens at paragraph granularity and headings are otherwise almost never selected on their own; units that fail to map onto the canonical content tree U are still scanned for their own outgoing dependencies rather than dropped, so a generator's selection cannot appear more coherent simply because its unmapped units are hidden from the calculation.

Recoverability Each generator's selected content units are made to answer PaperQuiz questions provided in the Paper2Poster dataset using gpt-4o. We report direct matching accuracy for each generators selected units for each paper.

Effieciency Text units are embedded with SBERT (all-MiniLM-L6-v2); all units, including visual ones, are embedded with a CLIP-style model for cross-modal comparison. A pair is flagged as redundant if cosine similarity exceeds a calibrated threshold ($\tau = 0.8$) or if the two units share an extracted numeric token, and clusters are formed by union-find over flagged pairs. The threshold was chosen by sweeping and plotting efficiency rate against threshold per generator, fixing the value at the point where rankings stabilized.

Validation For a random sample of 25 papers, we manually validate the mapping of generator's selected content units to the canonical superset-tree, claim-qualifier pairs extracted by the LLM from the paper, referencing of the edges done to evaluate coherence, recoverability answers given by the model, and the clusters formed by similarity matching to evaluate efficiency. Table 6 reports accuracy against this single expert annotator for every component.

Please refer to appendix A for more details.

5 Results & Discussion

We present our observations on each axis, computed as a mean with 95% bootstrap confidence intervals over the corpus (Table 1), along with pairwise significance assessed with a paired bootstrap approach (Table 4 & 5). Random selection is assumed as a floor, while the *Author* poster is the human reference. We assess each generator against the human *Author* to check whether a generator matches the human reference and against *Random* to validate whether a generator is better than the floor. All generators clear the random floor on at least one axis, and most clear it on the majority. We discuss the comparison with *Author* reference in detail (Table 2).

5.1 Faithfulness

Paper2Poster emerges as the most faithful method (0.736) and significantly exceeds the human *Author* (+0.076, CI[0.022,0.131]) since it preserves qualifiers better than the *Author*. In contrast, PosterForest is significantly worse than the *Author*. All

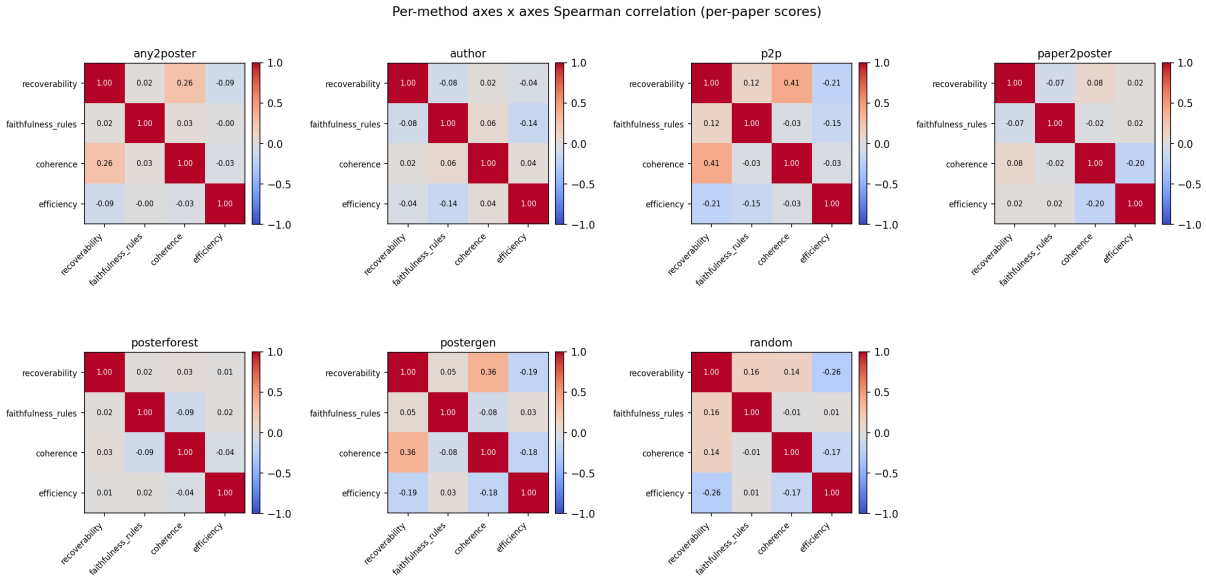


Figure 2: Spearman's correlation among four axes for every generator. Typically $|\rho| < 0.15$ is considered as negligible.

Method	F	C	R	E
P2P	0.671	0.356	0.719	0.811
Paper2Poster	0.736	0.545	0.758	0.769
PosterForest	0.571	0.751	0.728	0.799
PosterGen	0.644	0.369	0.727	0.873
Any2Poster	0.671	0.181	0.662	0.775
Author	0.668	0.405	0.706	0.842
Random	0.543	0.141	0.646	0.731

Table 1: Per-method mean score with 95% bootstrap CIs (resampled over papers). Author and Random are references. Per-cell paper counts (n) vary by axis: faithfulness 88–98, coherence 64–86, recoverability 95–100, efficiency 95–100. F=Faithfulness, C=Coherence, R=Recoverability, E=Efficiency (1-Redundancy)

Comparison	F	C	R	E
P2P	+0.020 ^{ns}	-0.091 ^{ns}	+0.011 ^{ns}	-0.034 ^{ns}
Paper2Poster	+0.076	+0.134	+0.052	-0.073
PosterForest	-0.106	+0.316 ^{ns}	+0.020	-0.043
PosterGen	-0.023 ^{ns}	-0.019 ^{ns}	+0.022	+0.035 ^{ns}
Any2Poster	+0.001 ^{ns}	-0.275	-0.044	-0.066

Table 2: Paired bootstrap differences vs. the human *Author* (resample papers once, difference within). Positive = generator exceeds the human; ^{ns} = not significant (95% CI includes 0). Paired-paper counts (n) by axis: faithfulness 88–96, coherence 45–72, recoverability 95–100, efficiency 95–100.

generators achieve higher mean faithfulness than the random selection, though for PosterForest and PosterGen the difference is not statistically significant. Faithfulness is observed to be strongly generator-dependent, with some exceeding human and some closer in performance to random selection.

5.2 Coherence

Coherence demonstrates the clearest difference compared with *Author*. PosterForest is the strongest with (0.751). Both PosterForest and Paper2Poster significantly exceed the *Author* (+0.316 & +0.134). At the other extreme, Any2Poster is dramatically worse than *Author* (-0.275) and not significantly better than the random (+0.039, n.s.). In other words, Any2Poster breaks the dependencies as often as blind selection. Both P2P and PosterGen match the *Author* and clear the floor comfortably.

$DDR(S)$ is undefined whenever a selection contains no unit with an explicit outgoing dependency edge (§3.3), and such selections are excluded from the coherence means reported in Table 3 rather than assigned a default score. This exclusion is not rare: across the corpus, 18% of P2P, 20% of Paper2Poster, 20% of PosterGen, and 13% of Any2Poster selections have no cross-referencing unit at all, compared to ≈34% for PosterForest. This matters for interpretation because the excluded and included selections are not drawn from the same distribution: a selection with zero outgoing edges is, by construction, one that never attempted

a figure/table/section reference in the first place; a sparser or more self-contained subset of that generator's behavior. PosterForest's high undefined rate is consistent with the sparse-selection strategy discussed above: by picking fewer, more self-contained units, it both avoids dangling references on the selections it *is* scored on and removes its riskiest, most cross-referencing selections from the coherence denominator altogether. Its reported coherence of 0.751 should therefore be read as coherence *conditional on attempting cross-references*, not as evidence that PosterForest is uniformly coherent across its full output.

5.3 Recoverability

The recoverability axis reports the least distinguishable set of observations with all generators staying close to the human *Author* (all 0.66–0.76, with tight CIs). Even random reaches 0.646, near the human 0.706. We believe that this reflects both the redundancy of the core claims in papers and the generator's optimization against coverage-oriented metrics. In contrast to coherence, the same generators span just 0.10 here, compared to 0.57 in coherence. This highlights the need for independent evaluation since optimizing for coverage secures recoverability but not other dimensions like coherence and faithfulness.

5.4 Efficiency

Similar to recoverability, efficiency(inverse redundancy) is high and tightly clustered (0.77-0.87). It is observed that humans allocate the space budget more efficiently. Although PosterGen is the most efficient (0.873), most generators are significantly less efficient than the human *Author*, and P2P and PosterGen match the human *author* efficiency. Some redundancy is deliberate, e.g. a significant result is repeated across table and text to aid the reader. Unlike a dropped caveat or a dangling reference, redundant content does not distort the paper, so we report efficiency as a context.

All depends on selection? Each generator exhibits a distinct profile towards diagnostic axes. Paper2Poster exceeds the human on faithfulness, coherence, and recoverability and is the strongest overall. PosterForest is the best on coherence, but is significantly below the human on faithfulness. Overall, Any2Poster is the weakest performer, lower than the human on coherence, recoverability, and efficiency, and no better than random on faithfulness. That generators diverge from human selection on different axes is the central argument for a multi-dimensional suite. We believe that these distinct profiles appear because of the way generators select the content. Any2Poster is as coherent as a random selection points towards a process insensitive to inter-unit dependencies. In contrast, PosterForest has the strongest coherence and the weakest faithfulness. It also has the highest coherence-undefined rate ($\approx$ 34% of selections contain no cross-referencing unit). By selecting sparsely, it commits to fewer dependencies and avoids dangling references, but also drops more qualifying units in the process. Improving one axis (coherence via sparser selection) can thus degrade another (faithfulness), and no single metric can reveal this tradeoff.

Implications The results highlight the drawback of the existing coverage-focused evaluation methodology. A generator trained only on coverage can bypass skipped qualifying conditions and orphan references, and discrepancies introduced because of this go unpenalized. Faithfulness- and coherence-aware evaluations should accompany coverage, and CoSeCh provides a complementary, auditable scoring mechanism.

5.5 Independence

To validate the independence of diagnostic axes, we compute Spearman's correlation among the four axes for every generator. We find that the correlations are near zero almost everywhere (see Figure 2). The strongest correlation is $\approx$ 0.4 between coherence and recoverability for P2P and PosterGen. The recoverability axis, which is the axis most aligned with existing coverage measures, is essentially uncorrelated with faithfulness and coherence. This provides direct evidence that coverage-oriented evaluation cannot substitute for distortions and incoherence.

5.6 Why not use LLMs?

We deliberately quantify each diagnostic axis with rule-based functions and not an LLM judge. The motivation behind CoSeCh is to expose failures that are missed by an LLM-based evaluation setup. We need the diagnostic probes to be transparent to evaluate LLM judge's unreliability. Rule-based scoring is auditable by definition. Since every rule can be broken down into interpretable decisions, any result can be traced to the corresponding claim

and dependencies. It is also independent of the systems considered for testing, several of which are LLM-based (e.g., multi-agent frameworks in P2P and PosterForest). In this way, we avoid the circularity that creeps in while judging an LLM's output with an LLM of the same family. Additionally, it is deterministic and easy to reproduce, unlike LLM-based tests, which may become unsuitable or obsolete due to model drift or deprecation. We do use a language model to extract claim-qualifier pairs, to map a selected unit back onto its source paragraph when lexical matching fails, and to answer recoverability questions, followed by human validation. But the scoring that follows is a set of deterministic operations over these extracted structures and selections. The division is clear: learned models are used for perception, and measurement remains transparent.

6 Conclusion

We introduce **CoSeCh**, a suite of four automatic rule-based diagnostics that evaluate the content selected by the generator for poster generation. The four axes (faithfulness, coherence, recoverability, and efficiency) assess the selected content independent of layout and rendering. Applying it to 5 generators over $\approx$ 100 papers, we find that selection quality is multi-dimensional and generator-specific. Paper2Poster is better at preserving qualifying conditions than human authors (+0.076, significant) and cross-unit dependencies than human authors; PosterForest is the most coherent (0.751), yet drops the qualifiers more than humans (-0.106). In contrast, Any2Poster selections are as coherent as the random choice (+0.039, n.s.). Recoverability, the property that existing coverage measures reward, barely separates the generators. All methods span just 0.10 (0.66–0.76) and even random reaches 0.646 vs. human 0.706. Faithfulness and coherence provide sharp distinguishing factors - on coherence the same methods span 0.57. Additionally, the four axes are near-independent ($|\rho| < 0.15$ almost everywhere, the largest only ≈ 0.4). This, along with the 0.10-vs-0.57 gap, provides the quantified argument that coverage-oriented evaluation misses distortion and incoherence. The formulations for these four axes are rule-based by design and hence traceable to specific claims and dependencies. Consequently, **CoSeCh** offers an auditable complement to LLM- and coverage-based poster evaluation. Its scope is deliberately bounded to the content selection step, which is implicit and goes unvalidated in most generators. We plan to extend this auditable diagnostic framework to paraphrastic distortions and to layout and rendering as a natural next step.

Limitations

Scope of Rule-based Scoring

CoSeCh's scoring is intentionally rule-based, and this limits what it can catch. Our faithfulness probe flags a qualifier only when its unit is dropped from the selection. It does not detect a claim that keeps its qualifier unit, which is rephrased, nor does it detect the distortion added during LLM-based claim-qualifier extraction. The coherence axis excludes selections with no explicit outgoing reference rather than scoring them (§5.5), and this exclusion rate varies sharply by generator, so a generator can appear more coherent partly by avoiding cross-references altogether, meaning Table 3's coherence column should be read alongside each generator's undefined rate. In addition, all diagnostic probes in this work evaluate content selection, not poster rendering. We trade in recall for subtle distortions to improve precision and interpretability, which is a real ceiling on coverage. Extending the detection of paraphrases, likely with an LLM-based detector validated against the rule-based scores here, is the natural next step.

References

Jiho Choi, Seojeong Park, Seongjong Song, and Hyunjung Shim. 2026. Posterforest: Hierarchical multi-agent collaboration for scientific poster generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 379–401.

Yifan Gao, Zihang Lin, Chuanbin Liu, Min Zhou, Tiezheng Ge, Bo Zheng, and Hongtao Xie. 2025. Postermaker: Towards high-quality product poster generation with accurate text rendering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8083–8093.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. True: Re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3905–3920.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In

Proceedings of the 30th ACM international conference on multimedia, pages 4083–4091.

Vijay Jaisankar, Sambaran Bandyopadhyay, Kalp Vyas, Varre Chaitanya, and Shwetha Somasundaram. 2024. Postdoc: Generating poster from a long multimodal document using deep submodular optimization. *arXiv preprint arXiv:2405.20213*.

Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Fables: Evaluating faithfulness and content selection in book-length summarization. *arXiv preprint arXiv:2404.01261*.

Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. Longeval: Guidelines for human evaluation of faithfulness in long-form summarization. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1650–1669.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. 2026. Paper2poster: Towards multimodal poster automation from scientific papers. *Advances in Neural Information Processing Systems*, 38.

Kanya Paramita and Masayu Leylia Khodra. 2016. Tailored summary for automatic poster generator. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–6. IEEE.

Yu-Ting Qiang, Yan-Wei Fu, Xiao Yu, Yan-Wen Guo, Zhi-Hua Zhou, and Leonid Sigal. 2019. Learning to generate posters of scientific papers by probabilistic graphical models. *Journal of Computer Science and Technology*, 34(1):155–169.

Yuting Qiang, Yanwei Fu, Yanwen Guo, Zhi-Hua Zhou, and Leonid Sigal. 2016. Learning to generate posters of scientific papers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Rohit Saxena, Pasquale Minervini, and Frank Keller. 2025. Postersum: A multimodal benchmark for scientific poster summarization. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1828–1844.

Alessandro Scirè, Karim Ghonim, and Roberto Navigli. 2024. FENICE: Factuality evaluation of summarization based on natural language inference and claim extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14148–14161, Bangkok, Thailand. Association for Computational Linguistics.

Tao Sun, Enhao Pan, Zhengkai Yang, Kaixin Sui, Jiajun Shi, Xianfu Cheng, Tongliang Li, Ge Zhang, Wenhao Huang, Jian Yang, et al. 2026. P2p: Automated paper-to-poster generation and fine-grained benchmark. In *International Conference on Learning Representations*, volume 2026, pages 18756–18798.

Amogh Vinaykumar, Aiden Li, Suozhi Huang, and Shilong Liu. 2026. Any2poster: Any-source poster generation across modalities and domains. *arXiv preprint arXiv:2606.02915*.

Hao Wang, Shohei Tanaka, and Yoshitaka Ushiku. 2024. Scipostlayout: A dataset for layout analysis and layout generation of scientific posters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8136–8141.

Sheng Xu and Xiaojun Wan. 2021a. Neural content extraction for poster generation of scientific papers. *Preprint*, arXiv:2112.08550.

Sheng Xu and Xiaojun Wan. 2021b. Neural content extraction for poster generation of scientific papers. *arXiv preprint arXiv:2112.08550*.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. 2025. Postergen: Aesthetic-aware multi-modal paper-to-poster generation via multi-agent llms. *arXiv preprint arXiv:2508.17188*.

A Implementation Details

A.1 Content Unit Extraction

We build one canonical content-unit tree per paper — the global superset U, using a docling-based structural decomposer that parses sections/subsections/paragraphs/sentences, extracts figures (paired with captions), tables, and equations as separate units, and tracks heading depth relatively for robustness across papers; this single U is shared across all five generators and the author/random baselines. We break each generator's pipeline (P2P, Paper2Poster, PosterForest, PosterGen, Any2Poster) at content selection to extract its selected set S, along with the human selection S_h and its size $|S_h|$; the random baseline samples 50 units without replacement over 100 trials, keeping the top-k by inclusion frequency with $k = |S_h|$ for that paper.

A.2 Faithfulness

Claim–qualifier pairs are extracted from the source paper with a single LLM pass, restricted to five qualifier types: scope ("only on English-language papers"), conditional ("under condition Y"), statistical ("p<0.05"), magnitude ("up to 4× faster"), and assumption ("assuming the set is finite"). Rather than checking survival against the generator's own selected-content text, which is often a paraphrase or summary of the source, each selected unit is first mapped back onto the paragraph in the source

paper it was drawn from — recovered by matching the unit against the paper's own paragraph structure — and this original-paper paragraph is substituted in place of the generator's text; a unit is left in its own (summarized) form only when no confident match to the source is found.

For each pair, we check whether the claim and its qualifier survive in this mapped selection independently. A claim is counted as present if the best-matching sentence in the selection contains its tokens above a fixed containment threshold, and its qualifier is counted as surviving if a matching span appears within a three-sentence window of the claim in the selection. This containment check is supplemented by a fallback list of qualifier-marker phrases matched to these five types, so that a qualifier rephrased using a standard marker is still credited as retained even when it is not a verbatim match. A pair is scored as dropped only when the claim is selected, but no qualifier span, verbatim, or marker-matched, is found in its window.

A.3 Coherence

For each paper, a shared directed dependency graph is built once over the paper's own hierarchical structure. Reference edges are detected by regex over figure/table/equation/section mentions — including plural and multi-item forms ("Figs. 2, 3"), numeric ranges ("Fig. 2–4"), roman-numeral table numbering, and bare parenthetical equation references ("...follows from (1)") lacking an explicit "Eq." prefix — with figure/table nodes attached from one generator's own asset list as a shared caption source, so every schema's captions are resolved against the same canonical targets rather than each schema's own.

A section/subsection reference is treated as satisfied if either the heading itself or any of its descendant paragraphs was selected, since selection happens at paragraph granularity and the heading node is otherwise almost never selected on its own. Units that fail to map onto the canonical content tree U are still scanned for their own outgoing dependencies rather than being silently dropped, so their references still count toward the denominator(Eq. (4)). This prevents a schema from appearing more coherent than it is simply because its unmapped units vanish from the calculation.

A.4 Recoverability

Recoverability is operationalized directly following PaperQuiz (Pang et al., 2026), the existing content-recall protocol, rather than a new question set. For each paper, PaperQuiz supplies two batches of 50 four-choice questions generated from the source PDF text: a detail batch (aspects A–M: title/authorship, motivation, objectives, datasets, methodology, hyperparameters, metrics, quantitative results, qualitative findings, comparative/ablation results, conclusions, limitations, definitions) and an understanding batch (aspects A–J: domain, problem, goal/hypothesis, key contributions, methodology overview, headline findings, qualitative insight, implications, limitations, main conclusions). Each question already carries its generating aspect as metadata.

A reader model answers each question in $Q(U)$ using only the generator's selected content S: the selected text units are concatenated in poster reading order and every selected figure/table image is attached alongside its section label and caption, so the reader sees exactly the same evidence a poster viewer would. The reader returns a single letter (A–D) per question, or "NA" if S offers no basis for an answer; a question is scored correct only on an exact letter match against the gold answer, with "NA" always counted as incorrect. $Rec(S)$ is this accuracy averaged over $Q(U)$.

A.5 Efficiency

Budget efficiency is computed from the same selected-unit set as every other metric: each pair of selected units is compared with SBERT if both are text, or with CLIP if either is a visual unit (so a chart and a bullet point restating its number share CLIP's joint text/image embedding space rather than being incomparable), and a pair is flagged as a near-duplicate when its cosine similarity is at or above a calibrated threshold τ. Independently of that similarity check, any two units whose text contains the same numeric token — a percentage, a count, a stat — are flagged as a same-claim restatement. These flagged pairs form a graph over the $|S|$ selected units, which is partitioned into k connected components (clusters) via union-find; efficiency is reported as $k/|\mathrm{S}|$ — a generator that deliberately restates a result across modalities for emphasis simply spends part of its unit budget on that repetition, and efficiency falls exactly as much for a legitimate cross-modal callback as for an accidental duplicate, without a second pass trying to guess which clusters were "intentional."

Similarity Threshold (τ) itself was calibrated empirically rather than asserted: we swept it from

0.50 to 0.90 in steps of 0.05, recomputing each generators's mean efficiency rate at every step from the same precomputed embeddings (only the flagging/clustering step depends on τ, so embeddings were encoded once, not once per threshold), then averaged the five schemas' rates together at each of the nine threshold values to get one aggregate curve. Quartiles were then taken over those nine averaged rate values themselves not over the threshold axis, which would be tautological and the thresholds whose rate fell in the bottom quartile (the flattest, most conservative end of the curve, where the rate is least sensitive to a further increase in τ) were 0.80, 0.85, and 0.90. We selected 0.8, which was the first τ value falling into bottom quartile.

Method	Faithfulness	Coherence	Recoverability	Efficiency
P2P	0.671 [0.622, 0.715]	0.356 [0.274, 0.440]	0.719 [0.695, 0.740]	0.811 [0.780, 0.840]
Paper2Poster	0.736 [0.694, 0.776]	0.545 [0.455, 0.636]	0.758 [0.744, 0.772]	0.769 [0.734, 0.802]
PosterForest	0.571 [0.513, 0.628]	0.751 [0.683, 0.814]	0.728 [0.711, 0.743]	0.799 [0.767, 0.829]
PosterGen	0.644 [0.589, 0.696]	0.369 [0.286, 0.450]	0.727 [0.710, 0.742]	0.873 [0.850, 0.896]
Any2Poster	0.671 [0.627, 0.713]	0.181 [0.122, 0.247]	0.662 [0.644, 0.681]	0.775 [0.740, 0.810]
Author	0.668 [0.615, 0.721]	0.405 [0.324, 0.487]	0.706 [0.687, 0.724]	0.842 [0.815, 0.869]
Random	0.543 [0.472, 0.611]	0.141 [0.094, 0.192]	0.646 [0.607, 0.681]	0.731 [0.702, 0.759]

Table 3: Per-method mean with 95% bootstrap CIs over papers. Author and Random are references. Per-cell n by axis: faithfulness 88–98, coherence 64–86, recoverability 95–100, efficiency 95–100.

Comparison	Faithfulness	Coherence	Recoverability	Efficiency
P2P	+0.140 [+0.063, +0.220]	+0.197 [+0.100, +0.291]	+0.073 [+0.032, +0.115]	+0.078 [+0.034, +0.121]
Paper2Poster	+0.202 [+0.121, +0.284]	+0.387 [+0.270, +0.500]	+0.113 [+0.080, +0.150]	+0.038 [-0.005, +0.080]^{ns}
PosterForest	+0.034 [-0.055, +0.121]^{ns}	+0.590 [+0.495, +0.680]	+0.083 [+0.049, +0.120]	+0.071 [+0.028, +0.112]
PosterGen	+0.090 [-0.001, +0.182]^{ns}	+0.212 [+0.108, +0.317]	+0.078 [+0.045, +0.112]	+0.141 [+0.108, +0.174]
Any2Poster	+0.123 [+0.038, +0.204]	+0.039 [-0.036, +0.118]^{ns}	+0.017 [-0.021, +0.056]^{ns}	+0.045 [-0.003, +0.094]^{ns}

Table 4: Paired bootstrap differences vs. Random (validity floor). ^{ns} = not significant. Paired n by axis: faithfulness 81–89, coherence 59–78, recoverability 95–100, efficiency 95–100.

Comparison	Faithfulness	Coherence	Recoverability	Efficiency
P2P	+0.020 [-0.033, +0.075]^{ns}	-0.091 [-0.210, +0.028]^{ns}	+0.011 [-0.016, +0.034]^{ns}	-0.034 [-0.066, +0.001]^{ns}
Paper2Poster	+0.076 [+0.021, +0.131]	+0.134 [+0.025, +0.242]	+0.052 [+0.035, +0.070]	-0.073 [-0.111, -0.035]
PosterForest	-0.106 [-0.187, -0.028]	+0.316 [+0.205, +0.433]	+0.020 [+0.002, +0.040]	-0.043 [-0.079, -0.008]
PosterGen	-0.023 [-0.082, +0.038]^{ns}	-0.019 [-0.135, +0.099]^{ns}	+0.022 [+0.004, +0.041]	+0.035 [-0.001, +0.072]^{ns}
Any2Poster	+0.001 [-0.051, +0.054]^{ns}	-0.275 [-0.376, -0.175]	-0.044 [-0.063, -0.024]	-0.066 [-0.104, -0.030]

Table 5: Paired bootstrap differences vs. the human *Author*. Positive = generator exceeds the human; ^{ns} = not significant (95% CI includes 0). Paired n by axis: faithfulness 88–96, coherence 57–73, recoverability 95–100, efficiency 95–100.

Axis	Component	Judgment	Accuracy
Mapping	unit → superset paragraph	correct / incorrect	80.9%
Faithfulness	claim–qualifier extraction	valid / invalid	72.3%
Coherence	reference edge genuineness	genuine / spurious	75.6%
Recoverability	reader-model answer correctness	correct / incorrect	83.2%
Efficiency	pairwise redundancy	redundant / distinct	79.8%

Table 6: Manual validation of each CoSeCh component against a single expert annotator (25 papers). **Judgment** is the binary decision the annotator made for each item. **Accuracy** is agreement between the automatic component and the annotator's label.

Look Where It Matters: High-Resolution Crops Retrieval for Efficient VLMs

Nimrod Shabtay[1,2]*, Moshe Kimhi[3]*, Artem Spector[1], Sivan Haray[1], Ehud Rivlin[3], Chaim Baskin[4], Raja Giryes[2], Eli Schwartz[1]
[1]IBM Research, [2]Tel-Aviv University, [3]Technion, [4]Ben-Gurion University

Abstract

Vision-language models (VLMs) typically process images at a native high-resolution, forcing a trade-off between accuracy and computational efficiency: high-resolution inputs capture fine details but incur significant computational costs, while low-resolution inputs offer efficiency at the cost of potentially missing critical visual information, like small text. We present AwaRes, a spatial-on-demand framework that resolves this accuracy–efficiency trade-off by operating on a low-resolution global view and using tool-calling to retrieve only high-resolution segments needed for a given query. We construct supervised data automatically: a judge compares low- vs. high-resolution answers to label whether cropping is needed, and a grounding model localizes the evidence for the correct answer, which we map to a discrete crop set to form multi-turn tool-use trajectories. We train our framework with cold-start SFT followed by multi-turn GRPO with a composite reward that combines semantic answer correctness with explicit crop-cost penalties. Extensive evaluation across six benchmarks demonstrates that AwaRes matches full-resolution accuracy while retaining only 36% of visual tokens.
https://nimrodshabtay.github.io/AwaRes

1 Introduction

Vision–language models (VLMs) increasingly rely on high-resolution visual inputs to solve detail-sensitive tasks such as document question answering, chart understanding, and understanding semantics and text in dense natural images. However, high resolution is expensive: the number of visual tokens grows rapidly with image resolution, making high-resolution inference a major bottleneck in practice. Existing approaches to reduce this cost largely fall into several camps. First, *token pruning* methods selectively discard visual tokens to reduce computation (Chen et al., 2024b; Xing et al., 2025; Yang et al., 2025b; Zhang et al., 2025b; Zou et al., 2025). While effective in principle, they often introduce irregular token patterns and dynamic sequence lengths that can be difficult to translate into end-to-end serving speedups in common inference stacks, such as vLLM (Kwon et al., 2023), where efficiency is tied to predictable sequence length. Second, *resolution escalation* methods (Yang et al., 2025c; Kimhi et al., 2026) learn when to request a higher-resolution view, but typically treat the decision as binary: if more details are needed, the entire high-resolution image is retrieved, wasting computation on regions irrelevant to the question. A key observation is that the demand for high fidelity is usually *spatially sparse*, as can be seen in Fig. 3. Many questions require fine detail in only a small portion of the image: a single value on a chart axis, a specific cell in a table, or a tiny object in the corner of an image. Recent works have begun to exploit this property by training VLMs to selectively zoom into task-relevant regions (Lee et al., 2025; Su et al., 2025; Zheng et al., 2026). However, these methods rely on an iterative reasoning process-where latency is dominated by long reasoning trace rather than the visual token budget per round; reducing input resolution alone yields only marginal speedups for these methods, since each additional reasoning step, not pixel count, is the primary cost. We advocate that answering the question of *where* to look matters as much as *whether* to look—and that it can be achieved without heavyweight reasoning machinery.

We propose AwaRes, a framework that exploits this spatial sparsity via a simple tool-calling interface that targets **high-resolution crop acquisition**. AwaRes processes a low-resolution global view by default, and when additional detail is required, it invokes a tool-call that requests only specific high-resolution sub-regions, and then answers conditioned on both. This multi-turn structure is naturally compatible with KV-caching: computation from the initial low-resolution turn is reused and extended in the crop turn without architectural changes, making AwaRes practical for deployment. We train AwaRes to learn a single *coupled-decision policy (CDP)* that jointly decides (i) *whether* ad-

*Equal contribution.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 357–384
October 24, 2026 ©2026 Association for Computational Linguistics

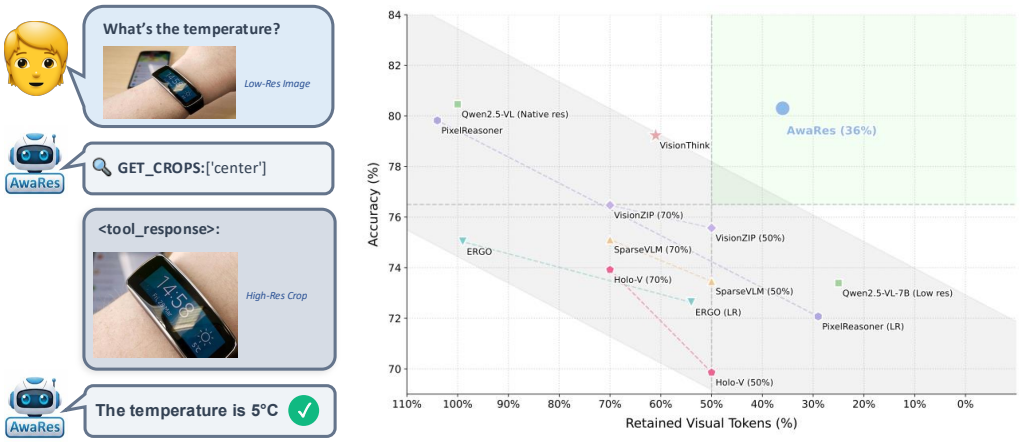


Figure 1: AwaRes overview. Left: Given a low-resolution image, AwaRes uses tool-calling to request only the high-resolution crops needed to answer the query. Right: Accuracy vs. retained visual tokens across six benchmarks. AwaRes performs similarly to native high-resolution (80.3%) while using only 36% of the visual tokens.

ditional resolution is needed and (ii) *where* to acquire it by selecting a subset of crops. Crucially, these decisions are *fused* into the model's first-turn action: either answer directly, or emit a structured crop request that simultaneously signals escalation and specifies the target regions. For the cold-start phase, we construct the supervision automatically, without manual spatial annotations, by (i) identifying examples where the low-resolution model's outputs are insufficient using an LLM-as-a-Judge (LaaJ), and (ii) localizing the evidence for the correct answer using a grounding model to produce target crops. We evaluate AwaRes on six benchmarks spanning document understanding and general visual QA. Across these tasks, AwaRes also resolves the same whether-and-where decision much more efficiently, avoiding the reasoning overhead and achieving on average a $5.7\times$ latency reduction over reasoning-based methods, and a $4.4\times$ reduction over the resolution-escalation method (Yang et al., 2025c), while retaining only 36% of visual tokens (RTR) and matching full-resolution accuracy (80.3% vs. 80.46%). Our contributions are listed as follows: (1) We introduce a spatial-on-demand inference framework that acquires targeted high-resolution crops through tool-calling, enabling system-friendly multi-turn KV-cache reuse. (2) We propose an automatic data curation pipeline that produces multi-turn tool-use trajectories without manual annotations. (3) We refine crop usage with multi-turn GRPO using an objective that penalizes unnecessary crop acquisition while discouraging missed crop requests when detail is required.

2 Related Work

Several strategies have emerged to reduce the number of visual tokens in Vision Language Models.

One line of research focuses on dynamic token pruning. Methods such as FastV (Chen et al., 2024b), HoloV (Zou et al., 2025), PyramidDrop (Xing et al., 2025), FitPrune (Ye et al., 2025), TopV (Yang et al., 2025a), SparseVILA (Khaki et al., 2025), IVTP (Huang et al., 2024), LLaVolta (Chen et al., 2024a), and SAINT (Jeddi et al., 2025) discard uninformative tokens within the LLM layers. Alternatively, others prune tokens directly after the vision encoder (Yang et al., 2025b; Vasu et al., 2025; Zhang et al., 2025b; Kimhi et al., 2025b). While effective, these approaches generally commit to a fixed token budget regardless of the specific instance's complexity. In contrast, our method is fully adaptive, allowing simple images to be processed at minimal cost while allocating more resources only when and where the query demands fine-grained perception.

A second line of work explores resolution selection. CARES (Kimhi et al., 2026) uses an external lightweight model to predict the optimal input resolution, while CROP (Guo et al., 2025) identifies contextual regions of interest via an auxiliary module. Because these methods rely on external components, they require hosting multiple models simultaneously. Our approach, by contrast, natively enables the VLM itself to determine where additional detail is needed.

A third line of work trains VLMs to actively escalate resolution or selectively zoom into task-relevant regions. VisionThink (Yang et al., 2025c) issues a tool call to retrieve the entire high-

resolution image when the low-resolution view is insufficient, wasting computation on irrelevant areas. Alternatively, ZoomEye (Shen et al., 2025), DeepEyes (Zheng et al., 2026), ERGO (Lee et al., 2025), and Pixel Reasoner (Su et al., 2025) achieve spatial selectivity through multiple inference passes or explicit visual operations embedded within long Chain-of-Thought (CoT) reasoning. While effective at improving accuracy, these approaches share a critical bottleneck: their reliance on extended autoregressive reasoning traces or massive full-image token counts severely degrades generation speed. Consequently, they fail to translate into real-world deployment latency benefits. Our method sidesteps this overhead entirely. By emitting a single structured tool call in the first turn to request only targeted high-resolution crops, AwaRes bypasses intermediate reasoning steps and full-image overhead, achieving spatial-on-demand selectivity at a fraction of the inference cost.

3 Method

AwaRes implements *spatial-on-demand* perception via a simple multi-turn interaction: the model first observes a low-resolution global view, and only if needed issues a tool call to retrieve a set of high-resolution crops (Fig. 1). We first formalize this interaction protocol (§3.1), then describe how we automatically curate supervision for the CDP, namely *whether* additional resolution is needed and *where* it matters (Fig. 2; §3.2). Finally, we train in two stages: (i) a *cold-start* supervised fine-tuning (SFT) stage that teaches the tool protocol and yields a supervised *reference policy* π_{ref} (§3.3); and (ii) multi-turn GRPO initialized from π_{ref} and regularized toward it via a KL penalty (§3.4), explicitly optimizing the accuracy–efficiency trade-off.

3.1 Problem setup

Given an image–question–answer triple $(I, q, a^\star)$, the model is first shown a low-resolution view I_{low} (obtained by downsampling I) together with the question q. The model then chooses between two actions:

(i) Direct answer: an answer $\hat{a}$ based on (q, I_{low}).

(ii) Crop request + answer: Emits a tool call that requests a *subset* of crops from a predefined candidate set, $\mathcal{C}_{\text{req}} \subseteq \mathcal{C}$. The tool returns the corresponding high-resolution crop images $\{I_c^{\text{high}}\}_{c \in \mathcal{C}_{\text{req}}}$, which are appended to the dialogue context, and the model produces the final answer $\hat{a}$ conditioned on the full multi-turn history.

A fused coupled-decision policy: We parameterize a single policy over high resolution request, and localized crop selection:

$$\pi_\theta(C \mid q, I_{\text{low}}), \qquad C \subseteq \mathcal{C}, \tag{1}$$

where $C = \emptyset$ corresponds to *no tool call* (answer directly) and $C \neq \emptyset$ corresponds to *escalation with localization*. Under this view, "when to crop" is the marginal event $\mathbb{1}[C \neq \emptyset]$, while "where to crop" is the conditional distribution over C given $C \neq \emptyset$. The two are inherently coupled: the value of escalating depends on *which* regions will be retrieved, since inaccurate localization can waste compute without improving answer correctness.

This interface targets efficiency by restricting high-resolution perception to a small number of structured regions, while preserving the low-resolution global context throughout the interaction (See Fig. 1 for a conversation example).

3.2 Automatic supervision for crop requests

A key challenge is to supervise two coupled decisions: whether the low-resolution view is insufficient, and **where** to crop when additional detail is needed. We generate this supervision to initiate the model to learn a reference policy $\pi_{\theta_{ref}}$ in an automatic fashion using the three-stage pipeline (Illustrated in Fig. 2):

Stage 1: resolution-sufficiency labeling (when to crop). For each example $(I, q, a^\star)$ we utilize a base VLM T on both the low-resolution and full-resolution inputs:

$$\hat{a}_{\text{low}} = T(q, I_{\text{low}}), \qquad \hat{a}_{\text{full}} = T(q, I). \tag{2}$$

Because $\hat{a}_{\text{low}}$ and $\hat{a}_{\text{full}}$ may differ in form though semantically correct, direct string matching (exact match) with $a^\star$ is unreliable. Instead, we use an LaaJ (LLaMA-3.3-70B (Grattafiori et al., 2024)) to compare both predictions to the ground truth $a^\star$. If it judges $\hat{a}_{\text{low}}$ as correct (or ties it with $\hat{a}_{\text{full}}$), we label the example as no crop needed `LR`; otherwise we label it as `HR`.

Stage 2: crop target construction (where to crop). For examples labeled `HR`, we identify the region that contains the visual evidence needed to answer $(q, a^\star)$. We prompt a grounding model G (namely, Qwen3-VL-A235B-A22B (Bai et al., 2025a)) to localize the evidence and return a bounding box $b = (x_1, y_1, x_2, y_2)$ in the coordinate system of the original image.

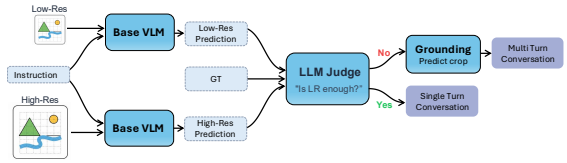


Figure 2: **Overview of the automatic supervision pipeline.** Each sample is processed at two resolutions; an LLM judge determines resolution sufficiency by comparing predictions to ground truth. Sufficient cases yield single-turn conversations, while insufficient cases are routed to a grounding model for crop localization, producing multi-turn trajectories with tool-calling.

We then map b to our discrete crop candidate set $\mathcal{C}$, which includes four quadrants, a center crop, four merged half-image regions (top/bottom/left/right), and a full-image. We define the target crop subset as

$$\mathcal{C}^\star = \{\, c \in \mathcal{C} \mid \mathrm{IoU}(b, c) \geq \tau \,\}, \tag{3}$$

where $\tau = 0.5$ is the IoU threshold. Fig. 3 shows a representative example, and Fig. 5 (left side) summarizes the empirical distribution of selected crops in the curated training set.

Stage 3: supervised tool-use trajectories. The procedure above yields two types of training transcripts:

Direct-answer trajectories (LR). The model observes (q, I_{low}) and is supervised to output $a^\star$ in a single turn.

Tool-call-then-answer trajectories (HR). In the first turn, the model issue a tool call selecting $\mathcal{C}^\star$. After the tool returns $\{I_c^{\text{high}}\}_{c \in \mathcal{C}^\star}$, the model is trained to produces $a^\star$ in a second turn conditioned on both the low-resolution and the retrieved high-resolution crops.

This curation pipeline produces multi-turn tool-use supervision at scale in order to learn an initial reference policy $\pi_{\theta_{ref}}$, while keeping the crop interface structured and deployment-friendly (Fig. 2). We provide additional details in the Appendix.

3.3 Cold-start (SFT)

We cold-start our crop-request policy by supervised fine-tuning (SFT) on the mixture of direct-answer and tool-call-then-answer trajectories produced in §3.2.

This stage serves two purposes: (i) teach the model to follow the multi-turn tool-calling protocol and learn the coupled decisions (*whether* additional detail is needed and *where* it matters), and (ii) produce a strong supervised *reference policy* π_{ref} that we later use for KL-regularized GRPO (§3.4).

Let $y_{1:T}$ denote the assistant tokens in a supervised transcript, and let h_t be the dialogue history at step t (including (q, I_{low}), any previously generated tokens, and tool outputs if a crop request occurred). We minimize a weighted negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t=1}^{T} w_t \log \pi_\theta(y_t \mid h_t) \tag{4}$$

The tool-call turn, despite having small numbers of tokens, fully specifies the CDP action and carry disproportionate control over both efficiency and downstream answer quality. Upweighting this turn therefore directly stabilizes learning of the *fused* first-turn decision. After SFT, we freeze the resulting model as the reference policy π_{ref} and initialize GRPO from it.

3.4 Multi-turn GRPO

After the cold-start SFT stage, the model reliably follows the tool protocol but tends to *over-request* crops even when I_{low} is sufficient. We therefore apply Group Relative Policy Optimization (GRPO) on full multi-turn interactions to explicitly optimize the accuracy–efficiency trade-off.

We denote by π_{ref} the *frozen* SFT policy $\pi_{\theta_{\text{ref}}}$ obtained from the SFT. GRPO is initialized from π_{ref} and uses a KL penalty to keep π_θ close to π_{ref} while improving tool usage.

Rollouts and trajectories. Given an input prompt $x = (q, I_{\text{low}})$, the policy π_θ generates first turn that may include a crop tool call. The requested crops are appended to the dialogue context, and generation continues until a final answer $\hat{a}$ is produced. We treat only assistant tokens as actions; tool outputs are treated as observations. Thus, each rollout yields a multi-turn trajectory τ consisting of assistant actions interleaved with tool observations, ending with $\hat{a}$.

Unlike supervised training with dense per-token loss, GRPO enables optimization with task-specific rewards that directly target improved tool usage.

3.4.1 Reward design.

We assign a single scalar reward to the completed trajectory τ, composed from two components:

$$R(\tau) = w_a \cdot R_{\text{ans}}(\hat{a}, a^\star) - C_{\text{tool}}(C, y), \tag{5}$$

Answer reward ($R_{\text{ans}}(\hat{a}, a^\star)$): measures semantic correctness using the cosine similarity between

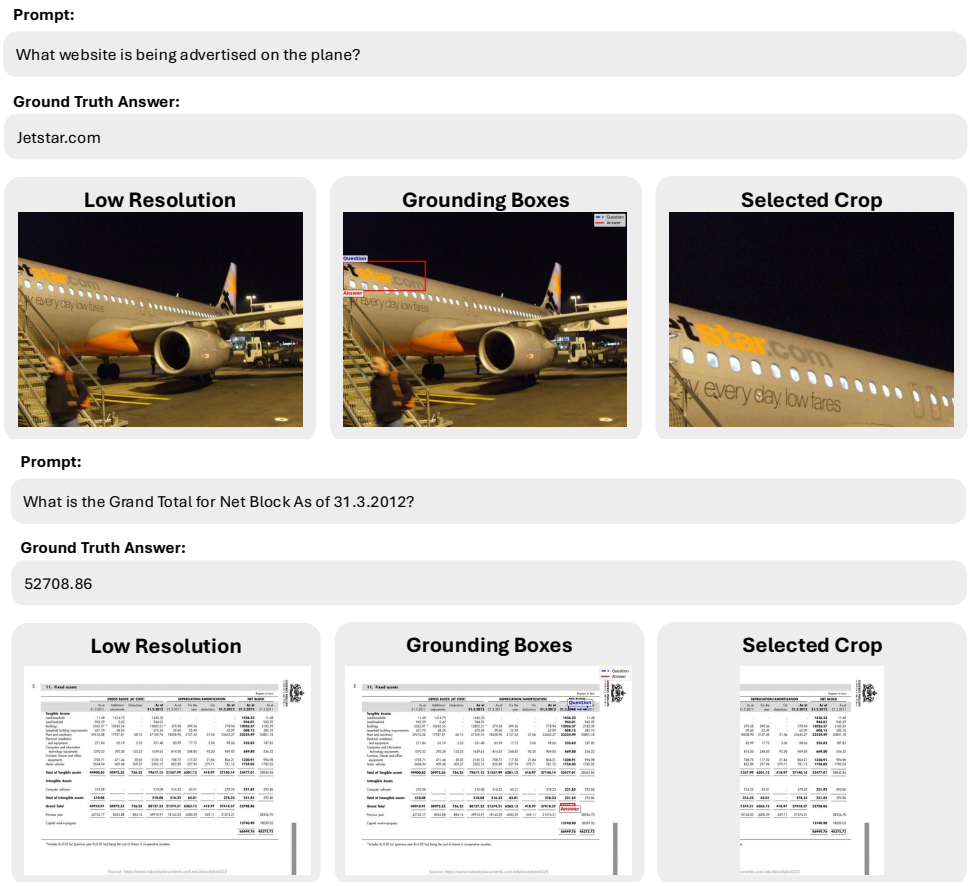


Figure 3: **Crop annotation example.** Left: low-resolution input where text is illegible. Middle: grounding model G predicted bounding box localizing the answer region. Right: selected high-resolution crop enabling correct response (best viewed when zoomed in).

sentence-transformer embeddings of $\hat{a}$ and $a^\star$, where w_a scales the answer reward relative to the tool cost.

Tool-use cost: Penalize tool usage with an asymmetric cost:

$$C_{\text{tool}}(C, y) = \begin{cases} \alpha_{\text{miss}} & \text{if } y = \texttt{HR} \text{ and } C = \emptyset \quad \text{(missed tool-call)} \\ \alpha_{\text{use}} + \lambda \|C\| & \text{if } C \neq \emptyset \quad \text{(tool usage)} \\ 0 & \text{if } y = \texttt{LR} \text{ and } C = \emptyset, \end{cases} \tag{6}$$

This asymmetry biases the policy toward *recall* in tool invocation: missing a necessary crop request is penalized more heavily than making an unnecessary request. When the tool is used, we additionally penalize the *amount of high-resolution evidence requested* via $\|C\|$, defined as the total fraction of image area covered by the selected crops. This encourages the policy to prefer smaller crops when they suffice. Importantly, the cost depends on *how much* is requested but remains agnostic to *which* specific region is chosen, allowing the GRPO to explore alternative policies.

3.4.2 GRPO optimization.

For each prompt x, we sample a group of G trajectories $\{\tau_1, \ldots, \tau_G\}$ from the current policy π_θ. Each trajectory τ_i consists of a sequence of assistant tokens (actions) interleaved with tool observations, culminating in a final answer. We compute the advantage for each trajectory using the group-relative baseline:

$$\hat{A}_i = \frac{R(\tau_i) - \mu_G}{\sigma_G + \epsilon}, \tag{7}$$

where $R(\tau_i)$ is the total reward for trajectory τ_i, and μ_G, σ_G are the mean and standard deviation of rewards within the group.

We optimize a PPO-style clipped objective with KL regularization to the reference policy:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \min \left(r_t^{(i)} \hat{A}_i,\ \text{clip}(r_t^{(i)}, 1-\epsilon, 1+\epsilon) \hat{A}_i \right) - \beta \, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Bigg] \tag{8}$$

where $r_t^{(i)} = \dfrac{\pi_\theta(a_t^{(i)} | x, a_{<t}^{(i)})}{\pi_{\text{old}}(a_t^{(i)} | x, a_{<t}^{(i)})}$ is the importance sampling ratio, ϵ is the clipping threshold, and β con-

trols the strength of the KL divergence penalty against a reference policy π_{ref}.

As in PPO-style updates, π_{old} denotes a snapshot of the policy before the current GRPO update step. The KL term is computed over assistant-token distributions along the sampled trajectories, encouraging stable improvements over π_{ref}.

3.5 Inference

At test time, we follow the same interaction protocol used during training. The model receives (q, I_{low}) and either answers directly or emits a tool call selecting a crop subset $C \subseteq \mathcal{C}$. If a tool call occurs, the corresponding high-resolution crops are appended to the dialogue context while retaining I_{low}, and the model produces the final answer in a second turn.

This results in two possible inference paths: a single prefill pass for queries answerable from the low-resolution view, or two prefill passes when high-resolution detail is required. In the latter case, the model benefits from both the global context preserved in I_{low} and the fine-grained detail in the requested crops. When a second turn is required the low-resolution view and the query are already in the KV cache saving on the required compute. Crucially, the decision of which path to take, and which regions to acquire - is made entirely by the learned policy, requiring no external heuristics or task-specific thresholds.

4 Experimental Results

We evaluate AwaRes on six benchmarks spanning document understanding and general visual QA, and compare against both fixed-budget token-pruning methods and adaptive resolution-escalation baselines. We report (i) the dataset metric from `lmms-eval` (Zhang et al., 2025a) and (ii) an *Retain Token Ratio* (RTR), defined as the fraction of visual tokens processed relative to the full-resolution baseline. RTR directly reflects the model's first-turn *coupled-decision policy* (answer directly vs. request crops), while accuracy reflects the quality of the full multi-turn interaction. We first describe our evaluation protocol 4.1, evaluated datasets 4.2 and implementation details 4.3. Then, we provide a detailed discussion of the main results 4.5 and conclude by extensive ablations 4.6.

4.1 Evaluation Protocol

All models are evaluated using `lmms-eval` (Zhang et al., 2025a), and we report the per-dataset metrics provided by the framework.

Retain Token Ratio (RTR): We measure efficiency via *visual token* usage, which dominates compute and KV-cache memory at high resolution. Let T_i be the visual tokens a sample i consumes across *all* turns and T_{full} those of one full-resolution pass with the baseline model; then $\text{RTR}_i = T_i / T_{\text{full}}$ and $\text{RTR} = \mathbb{E}_i[\text{RTR}_i]$. Fixed-budget methods are configured to retain 50% or 70% of the full-resolution tokens; for adaptive methods we count the tokens actually consumed per sample and average over the dataset.

Latency: When reporting wall-clock time (Fig. 4), we measure end-to-end per-sample latency, including both turns for methods that invoke the crop tool.

4.2 Datasets

We curated a diverse training set comprising 10K samples from each of five publicly available training sets: ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), TextVQA (Singh et al., 2019), LLaVA-Multi (Jiang et al., 2024), and VisionThink-Smart (Yang et al., 2025c). We also collected 2k samples with the same distribution as a validation set. The SFT phase using a subset of the training set (5k samples from each dataset), while the GRPO uses all collected examples. This mixture spans both document understanding and natural image domains.

We evaluate on six benchmarks spanning natural image understanding-RealWorldQA (xAI, 2024) (real-world spatial reasoning), POPE (Li et al., 2023) (object hallucination), and V^*-Bench (Wu and Xie, 2024) (visual search in high-resolution scenes)-and document understanding-ChartQA (Masry et al., 2022) (reasoning over charts), DocVQA (Mathew et al., 2021) (forms, tables, letters, and handwriting), and OCRBench (Liu et al., 2024) (text recognition and text-centric reasoning). full details are in the appendix.

4.3 Implementation Details

We conduct experiments based on Qwen2.5-VL-7B-Instruct (Bai et al., 2025b); all compared methods are built on the same base VLM to isolate the impact of efficiency mechanisms.

Unless otherwise specified, each sample is first processed at a low-resolution setting I_{low} has hight and width devided by 2, corresponding to the "LR" baseline in Table 1 (RTR=0.25). When a crop is requested, the crop image(s) are rendered at the

native high-resolution token density of the base model.

We use the discrete crop set $\mathcal{C}$ from §3.2 (four quadrants, four half-image regions, a center crop, and the full image). Requested crops are appended to the context together with I_{low}.

For training, we use HuggingFace TRL (von Werra et al., 2020) for SFT and GRPO, adapting TRL's GRPO trainer to support multi-image and multi-turn conversations. For the SFT phase we use a total batch size of 16 with learning rate 1×10^{-4} and LoRA rank 8. $w_t = 5$ only for the tool-call turn and 1 otherwise. For the GRPO phase we use a total batch size of 64 with G=8 generations per sample, learning rate 1×10^{-5}, and LoRA rank 8. $w_a = 10$, $\alpha_{miss} = 2$, $\alpha_{\text{use}} = 0.25$ and $\lambda = 0.01$

4.4 Baselines

All baselines are based on Qwen-2.5-VL-7B (Bai et al., 2025b). **Fixed-budget token pruning/compression:** VisionZip (Yang et al., 2025b), SparseVLM (Zhang et al., 2025b), and Holo-V (Zou et al., 2025), training-free inference-time methods configured to retain 50% or 70% of the full-resolution visual tokens. **Dynamic methods:** VisionThink (Yang et al., 2025c), which escalates a low-resolution pass to the *full* high-resolution image, and the reasoning-based ERGO, Deep-Eyes, and Pixel-Reasoner (Lee et al., 2025; Zheng et al., 2026; Su et al., 2025), which can request a crop from it. AwaRes instead escalates *spatially*, via a short tool call.

4.5 Main Results

Table 1 reports task performance and efficiency. AwaRes achieves an average score of 80.30, matching the full-resolution baseline (80.46) while using only $0.36\times$ the visual tokens on average. This indicates that the learned *coupled-decision policy* answers directly from the low-resolution overview when possible, and invokes targeted crops only when necessary.

Compared to fixed-budget pruning methods, AwaRes consistently attains higher accuracy at comparable or lower RTR. For example, at 70% retained tokens, VisionZip trails AwaRes by over 4 percent on average (76.47 vs. 80.30), reflecting the limitation of a global, sample-agnostic token budget.

Against the adaptive escalation baseline VisionThink (Yang et al., 2025c), AwaRes improves both accuracy (80.30 vs. 79.23) and efficiency (RTR 0.36 vs. 0.61). Notably, on ChartQA AwaRes slightly exceeds the full-resolution baseline (+0.84 percent) while reducing RTR to 0.32, whereas VisionThink often performs two visual passes and exceeds baseline compute (RTR=1.15).

4.5.1 Beyond Token Efficiency

Although RTR captures the dominant visual compute, end-to-end latency also depends on the number of autoregressive text tokens generated: other dynamic methods decide whether to escalate through extended reasoning traces, which inflates decoding cost even when the final answer is short, whereas AwaRes encodes the decision in a short, structured tool call and reuses the KV cache between turns.

Fig. 4 reports average wall-clock time for all dynamic methods, with a reference line at one second-a practical requirement for interactive applications. AwaRes is the only method consistently below it, averaging 0.61s across the six benchmarks, and is 2–12$\times$ faster than every baseline. The figure also reports each baseline's self-speedup from a low-resolution first turn ("LR"): ERGO benefits most ($1.7\times$, 2.04 s $\rightarrow$ 1.20 s), still nearly twice AwaRes's latency, while PixelReasoner and DeepEyes gain only $1.1\times$ and $1.2\times$, indicating their latency is dominated by text decoding and multi-turn overhead rather than visual encoding. Two choices drive this: the resolution decision costs a single decoding step rather than a reasoning trace, and the loop terminates early when the low-resolution view already suffices, avoiding a second forward pass. Under a production-grade engine (vLLM), AwaRes averages 0.24s per sample, 31% faster than the high-resolution baseline, so the token savings do translate into deployment latency.

4.5.2 From Over-Calling to Looking Where It Matters

Fig. 5 illustrates how the model's *first-turn crop-request policy* evolves from grounded supervision (denoted as Oracle GT) through SFT to GRPO. The SFT designed to introduce the tool-calling protocol, and as so, the policy becomes conservative with respect to accuracy and therefore *over-invokes* the crop tool: the probability of taking the no-call action (LR) drops sharply (80.8%$\rightarrow$46.3%), while escalation to the full-image crop ("All") is substantially inflated (3.4%$\rightarrow$16.6%). This behavior is consistent with imitation-style training that prioritizes adhering to the tool protocol, even in cases

Table 1: **Main results across vision-language benchmarks.** We compare AwaRes against fixed-ratio token-pruning methods (VisionZIP, SparseVLM, Holo-V) and dynamic, crop-inspired methods (ERGO, DeepEyes, PixelReasoner, VisionThink). Retain Token Ratios (RTR) denote the fraction of visual tokens retained. "LR" indicates the first turn uses a low-resolution image. AwaRes matches the full-resolution baseline (Qwen2.5-VL-7B) while using only 36% of the visual tokens, outperforming all efficient alternatives. Best results per column in **bold**.

	ChartQA		DocVQA		OCRBench		POPE		RealWorld		V* Bench		**Average**	
Model	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓
Base models														
Qwen2.5-VL-7B	79.80	1.00	94.00	1.00	81.10	1.00	87.87	1.00	68.80	1.00	71.20	1.00	80.46	1.00
Qwen2.5-VL-7B-LR	65.00	0.25	91.00	0.25	70.70	0.25	84.41	0.25	66.00	0.25	63.20	0.25	73.39	0.25
Token-pruning methods (fixed ratio)														
Holo-V (50%)	62.04	0.50	70.77	0.50	68.20	0.50	86.54	0.50	67.19	0.50	64.40	0.50	69.86	0.50
Holo-V (70%)	69.32	0.70	76.40	0.70	72.40	0.70	87.37	0.70	**68.89**	0.70	69.11	0.70	73.92	0.70
SparseVLM (50%)	73.20	0.50	83.60	0.50	75.60	0.50	85.50	0.50	68.40	0.50	54.45	0.50	73.46	0.50
SparseVLM (70%)	75.80	0.70	87.20	0.70	79.30	0.70	85.40	0.70	68.50	0.70	54.45	0.70	75.11	0.70
VisionZIP (50%)	74.76	0.50	89.39	0.50	69.20	0.50	87.56	0.50	66.01	0.50	66.49	0.50	75.57	0.50
VisionZIP (70%)	76.72	0.70	90.75	0.70	72.70	0.70	87.86	0.70	64.84	0.70	65.97	0.70	76.47	0.70
Dynamic / crop-inspired methods														
ERGO	80.30	1.293	59.80	0.435	74.60	1.597	86.20	1.277	65.50	0.933	83.80	0.401	75.03	0.99
ERGO (LR)	73.40	0.614	60.40	0.380	72.10	0.840	84.40	0.577	64.30	0.466	81.20	0.366	72.63	0.54
DeepEyes	61.76	1.090	82.95	1.012	80.50	1.350	34.40	1.260	65.75	1.090	80.10	1.001	67.58	1.13
DeepEyes (LR)	51.04	0.277	82.04	0.253	66.80	0.529	32.74	0.296	61.05	0.277	72.25	0.253	60.99	0.31
PixelReasoner	87.90	1.010	92.93	1.020	70.80	1.020	**88.30**	1.130	54.20	1.000	**84.81**	1.050	79.82	1.04
PixelReasoner (LR)	74.00	0.270	92.15	0.270	59.30	0.300	85.06	0.400	51.76	0.250	70.15	0.270	72.07	**0.29**
VisionThink	79.90	1.150	90.35	0.320	80.10	0.830	86.70	0.340	66.60	0.550	71.73	0.490	79.23	0.61
AwaRes	**80.64**	0.320	**94.43**	0.280	**81.30**	0.420	85.73	0.270	68.50	0.430	71.20	0.420	**80.30**	0.36

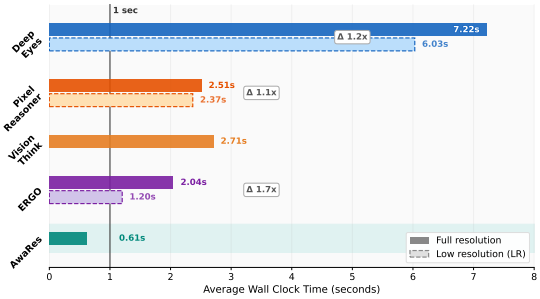


Figure 4: **Latency breakdown.** Full vs. low-resolution (LR) first-turn input; solid bars are full-resolution wall-clock time, dashed bars the LR variant. The small speedups (Δ1.1–1.7$\times$) show that input token count is not the main bottleneck: cost is dominated by extended reasoning or iterative cropping. AwaRes reaches 0.61s via a single short tool call, avoiding verbose generation.

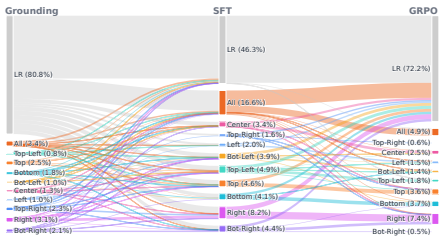


Figure 5: **From Over-Using to Looking Where It Matters.** The flow of crop selection from Grounding (Oracle GT for the SFT), SFT-tuned model (middle), and GRPO predictions (right). SFT, designed to introduce the tool-calling protocol, tends to over-use the crop tool. The GRPO phase corrects this behavior through the tool-use cost, increasing low-resolution decisions while exploring alternative crop strategies that balance accuracy and efficiency.

where the low-resolution view is sufficient.

GRPO then reshapes the same policy under the explicit accuracy–efficiency objective in Eq. 5; as a result, the policy shifts toward selective tool use: LR increases to 72.2% and "All" decreases to 4.9%, closer to the grounding model. Due to the tool-use cost that penalize crop size, the learned policy may shift towards efficient strategies that differ from the grounded annotations.

We further show that AwaRes does not merely distill the grounding model: the 7B model surpasses the tool-prompted grounding model (MoE, 235B-A22B) on ChartQA (80.6 vs. 69.7) and OCRBench (81.3 vs. 77.8), indicating that GRPO discovers cropping strategies beyond those introduced by the grounding model (full results in the Appendix).

4.6 Ablations

We ablate the data-preparation pipeline, the cold-start SFT stage, and the GRPO stage. The full details are in Appendix D and E.

4.6.1 Data preparation

Swapping the default judge LLAMA-3.3-70B (Grattafiori et al., 2024) for DEEPSEEK-V3.2 (Liu et al., 2025) changes only 3.12% of resolution-selection labels, versus 19.89% for ANLS, which over-penalizes paraphrased-but-correct answers and costs 3.8 points when used to train $\pi_{\theta_{\text{ref}}}$. Two independent grounders agree on 88% of cases.

4.6.2 Tool introduction

Table 6 studies how SFT recipes affect the emergence of reliable crop-tool usage. The default optimizes each assistant message independently, whereas **Traj** performs trajectory-level SFT over the entire two-turn interaction as a single sample, and **Phased** trains on the tool-call turns first. Trajectory-level training improves both accuracy and efficiency over split training (77.90 vs. 75.15 and 0.43 vs. 0.36), and raising the tool-turn weight to w_t=5 improves accuracy further (79.70) at a higher RTR (0.49), suggesting that emphasizing the first-turn decision encourages more confident tool invocation. **Upsampled** variants reduce RTR (to 0.33) but also accuracy, reflecting the expected trade-off. We therefore initialize GRPO from trajectory-level SFT with w_t=5.

4.6.3 Tool Optimization rewards

We ablate the GRPO reward components, which combine a sentence-transformer answer-similarity reward with a tool-use cost (Table 9). Removing the crop-area cost raises RTR from 0.36 to 0.42 ($\lambda = 0$), and removing the crop cost entirely raises it to 0.51; the increase is bounded by the SFT initialization and the KL term, which keep the policy close to a reference that already uses the tool sensibly. ANLS and LLM-as-a-judge correctness rewards perform similarly, since we optimize short answers rather than lengthy reasoning traces; sentence-transformer similarity balances the semantic sensitivity ANLS lacks against the robustness an LLM judge lacks.

4.6.4 The effect of cold start initialization

An alternative to our two-stage approach is to run RL directly from the base model and let it learn tool use from scratch during GRPO. Trained this way for 3 epochs on the same data (GRPO only, Tab. 9), the model barely uses the tool, falls back to low-resolution processing, and loses accuracy. SFT alone reaches good accuracy but high RTR from over-calling, and it is a double-edged sword: on an external validation set (Fig. 5) the SFT model has overfitted to the training data and shifted its policy substantially. GRPO is therefore not only exploratory but also recovers from that overfitting.

5 Conclusion

We presented AwaRes, a spatial-on-demand inference framework for VLMs that keeps a low-resolution global view and retrieves high-resolution crops only when the query requires them, through a deployment-friendly tool-calling interface with multi-turn KV-cache reuse. Training needs no manual spatial supervision: an automatic curation pipeline supplies the data, and we train with cold-start SFT followed by multi-turn GRPO under a reward combining answer correctness and crop-usage penalties. Across six benchmarks, AwaRes matches full-resolution accuracy with far fewer visual tokens and lower latency than both dynamic and fixed-budget baselines.

Limitations

Our framework acquires high-resolution evidence from a *fixed, discrete* crop set (four quadrants, four half-image regions, a center crop, and the full image). This keeps the tool interface structured and deployment-friendly, but it also caps spatial precision: when the answer-bearing evidence straddles crop boundaries or occupies a region much smaller than a quadrant, the policy must request more area than strictly necessary. Extending the interface to continuous bounding boxes is left to future work.

The resolution-sufficiency label comes from an LLM judge and the crop target from a grounding model; both make mistakes (Appendix A shows representative failures), and the resulting label noise propagates into the cold-start policy. We measure inter-model agreement rather than validating against human annotations.

We evaluate a single VLM (Qwen2.5-VL-7B) and latency numbers are measured on a single NVIDIA H100-80GB GPU under HuggingFace and vLLM, meaning absolute wall-clock times may differ on other serving stacks.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Ke qin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *ArXiv*, abs/2502.13923.

Jieneng Chen, Luoxin Ye, Ju He, Zhao-Yang Wang, Daniel Khashabi, and Alan Yuille. 2024a. Efficient large multi-modal models via visual context compression. In *NeurIPS*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024b. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiawei Guo, Feifei Zhai, Pu Jian, Qianrun Wei, and Yu Zhou. 2025. Crop: Contextual region-oriented visual token pruning. *Preprint*, arXiv:2505.21233.

Kai Huang, Hao Zou, Ye Xi, BoChen Wang, Zhen Xie, and Liang Yu. 2024. Ivtp: Instruction-guided visual token pruning for large vision-language models. In *ECCV*, pages 214–230.

Ahmadreza Jeddi, Negin Baghbanzadeh, Elham Dolatabadi, and Babak Taati. 2025. Similarity-aware token pruning: Your vlm but faster. *Preprint*, arXiv:2503.11549.

Dongfu Jiang, Xuan He, Huaye Zeng, Cong Wei, Max Ku, Qian Liu, and Wenhu Chen. 2024. Mantis: Interleaved multi-image instruction tuning. *arXiv preprint arXiv:2405.01483*.

Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N. Plataniotis, Yao Lu, Song Han, and Zhijian Liu. 2025. Sparsevila: Decoupling visual sparsity for efficient vlm inference. In *ICCV*, pages 23784–23794.

Moshe Kimhi, Omer Kerem, Eden Grad, Ehud Rivlin, and Chaim Baskin. 2025a. Noisy annotations in semantic segmentation. *Preprint*, arXiv:2406.10891.

Moshe Kimhi, Erez Koifman, Ehud Rivlin, Eli Schwartz, and Chaim Baskin. 2025b. Waveclip: Wavelet tokenization for adaptive-resolution clip. *Preprint*, arXiv:2509.21153.

Moshe Kimhi, Nimrod Shabtay, Raja Giryes, Chaim Baskin, and Eli Schwartz. 2026. CARES: Context-aware resolution selector for VLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2243–2256, San Diego, California, United States. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jewon Lee, Wooksu Shin, Seungmin Yang, Ki-Ung Song, DongUk Lim, Jaeyeon Kim, Tae-Ho Kim, and Bo-Kyeong Kim. 2025. Ergo: Efficient high-resolution visual understanding for vision-language models. *arXiv preprint arXiv:2509.21991*.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305. Association for Computational Linguistics.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024. OCR-Bench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12).

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279. Association for Computational Linguistics.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

Kwangrok Ryoo, Yeonsik Jo, Seungjun Lee, Mira Kim, Ahra Jo, Seung Hwan Kim, Seungryong Kim, and Soonyoung Lee. 2023. Universal noise annotation: Unveiling the impact of noisy annotation on object detection. *Preprint*, arXiv:2312.13822.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. 2025. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. In *Proceedings of the*

2025 Conference on Empirical Methods in Natural Language Processing.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. 2025. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*.

Pavan Kumar Anasosalu Vasu, Fartash Faghri, Chun-Liang Li, Cem Koc, Nate True, Albert Antony, Gokul Santhanam, James Gregory Gabriel, Peter Grasch, Oncel Tuzel, and Hadi Pouransari. 2025. Fastvlm: Efficient vision encoding for vision language models. In *CVPR*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. TRL: Transformers Reinforcement Learning.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. 2025. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*.

Penghao Wu and Saining Xie. 2024. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

xAI. 2024. Realworldqa. `https://x.ai/blog/grok-1.5v`. Accessed: 2024.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. 2025. Conical visual concentration for efficient large vision-language models. In *CVPR*, pages 14593–14603.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Chendi Li, Jing Nathan Yan, Yu Bai, Ponnuswamy Sadayappan, Xia Ben Hu, and Bo Yuan. 2025a. Topv: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025b. Visionzip: Longer is better but not necessary in vision language models. In *CVPR*, pages 19792–19802.

Senqiao Yang, Junyi Li, Xin Lai, Bei Yu, Hengshuang Zhao, and Jiaya Jia. 2025c. Visionthink: Smart and efficient vision language model via reinforcement learning. In *NeurIPS*.

Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. 2025. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025a. LMMs-Eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916. Association for Computational Linguistics.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. 2025b. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *ICML*.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. 2026. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In *ICLR*.

Xin Zou, Di Lu, Yizhou Wang, Yibo Yan, Yuanhuiyi Lyu, Xu Zheng, Linfeng Zhang, and Xuming Hu. 2025. Don't just chase "highlighted tokens" in mllms: Revisiting visual holistic context retention. In *NeurIPS*.

This supplementary document provides additional details, analyses, and visual examples that complement the main paper. We organize the material as follows: Section A presents supplementary visual examples from our data curation pipeline, including both successful annotations and failure cases of the automatic process. In addition, we analyze alternative data curation strategies using ANLS-based filtering, along with grounding model robustness. Section B provides qualitative examples of AwaRes at inference time, illustrating both successful predictions and failure modes. Section C details our latency analysis, including hardware setup, wall-clock measurements comparing AwaRes and VisionThink, and a response length analysis explaining the observed efficiency gains. Section D offers an in-depth analysis of the cold-start (SFT) stage, examining the coupled-decision policy (CDP) behavior, sensitivity to the tool-turn weight w_t, and tool-call formatting reliability. Section E presents GRPO ablations covering the cold-start stage, tool-cost penalty, and correctness reward, as well as how GRPO learns beyond grounding model annotations. Section F provides comprehensive training hyperparameters and configuration details. Finally, Section G includes the complete

prompts used throughout our pipeline: the LLM-as-a-Judge prompt for data curation, the grounding model localization prompt for grounding, and the system prompt used during SFT, GRPO, and inference.

A Data Annotation

A.1 Data Curation Pipeline

In this section, we provide in Figure 6 supplementary visual examples from our data curation pipeline. Additionally, in Figure 7 we illustrate failure cases where the automatic process did not produce satisfactory results.

A.2 Agreement of resolution selection

The two panels compare resolution-selection labels from our default judge LLAMA-3.3-70B (Grattafiori et al., 2024) against DEEPSEEK-V3.2 (Liu et al., 2025) (left) and ANLS (right), with LR and HR marking low- and high-resolution labels. The two LLM judges show strong agreement (96.88% along the diagonal), whereas ANLS agrees far less (80.11%). This indicates that many responses are semantically equivalent despite differing in phrasing, which ANLS penalizes - strengthening our decision to use a judge rather than a simple text-comparison metric.

A.3 Training data statistics

Figure 9 shows the resolution distribution of our training data. DocVQA (Mathew et al., 2021) exhibits the highest resolutions, whereas the LLaVA-Multi (Jiang et al., 2024) subset contains the lowest. VisionThink-Smart (Yang et al., 2025c) and ChartQA (Masry et al., 2022) display a broad spread across resolutions, while TextVQA concentrations cluster around 1000 pixels along one axis. We cap all resolutions at 2000×2000, as prior work (Kimhi et al., 2026) demonstrated that truncating DocVQA resolution has negligible impact on performance.

A.4 ANLS-Based Data Curation Analysis

We compare annotating the crops using LaaJ vs. using ANLS in Tab. 2. ANLS is a string-oriented similarity measure designed for OCR-style exactness, and as such it is ill-suited for supervising our resolution-sufficiency labels, specifically since we observe the VLM answers with crops act as augmented input and thus perturbed the answers a bit (even when they are semantically correct).

This effect is reflected empirically. Training with ANLS-based labels for cold-start yields substantially lower average accuracy than LaaJ-based labels (75.9 vs. 79.70), with pronounced drops on text- and detail-sensitive benchmarks such as ChartQA (70.5 vs. 77.0) and OCRBench (70.0 vs. 78.8). While ANLS labeling attains a lower RTR (0.38 vs. 0.49), the accompanying accuracy degradation indicates that it often suppresses crop requests even when additional resolution is required, rather than producing a better-calibrated policy. Overall, LaaJ provides a more reliable semantic correctness signal for annotation, which is critical for learning the CDP without over-penalizing benign surface-form variation.

A.5 Grounding model robustness

To assess the reliability of our grounding model, we compare the crop regions selected by two strong VLM grounding modelS, Qwen3-VL-235B-A22B (Bai et al., 2025a) and InternVL3.5-241B-A28B (Wang et al., 2025), on the subset of samples where high-resolution cropping is beneficial. As shown in Figure 10, the two grounding models exhibit strong agreement, with 88% of predictions concentrated along the diagonal, indicating that the selected crop regions are largely model-agnostic rather than artifacts of a single grounding model's idiosyncrasies. The few off-diagonal cases cluster among spatially adjacent regions (e.g., bottom-right vs. top-right), suggesting that disagreements stem from boundary ambiguity rather than fundamentally divergent localization.

B Benchmark Qualitative Examples

Figures 11 - 16 present positive examples where AwaRes successfully identifies and crops the image region relevant to the question. Each conversation shows the question, the tool call selected by the model, the predicted answer (matching the ground truth), the low-resolution input, and the retrieved high-resolution crop. Across charts, documents, and natural images, the retrieved high-resolution crops consistently isolate the task-relevant objects or text - such as axis labels, table cells, brand logos, and foreground subjects, enabling the model to extract fine-grained details that are unresolvable in the low-resolution overview alone.

B.1 Failure cases

Figure 17 shows representative failure cases where the model's either crop incorrect region (e.g. noisy

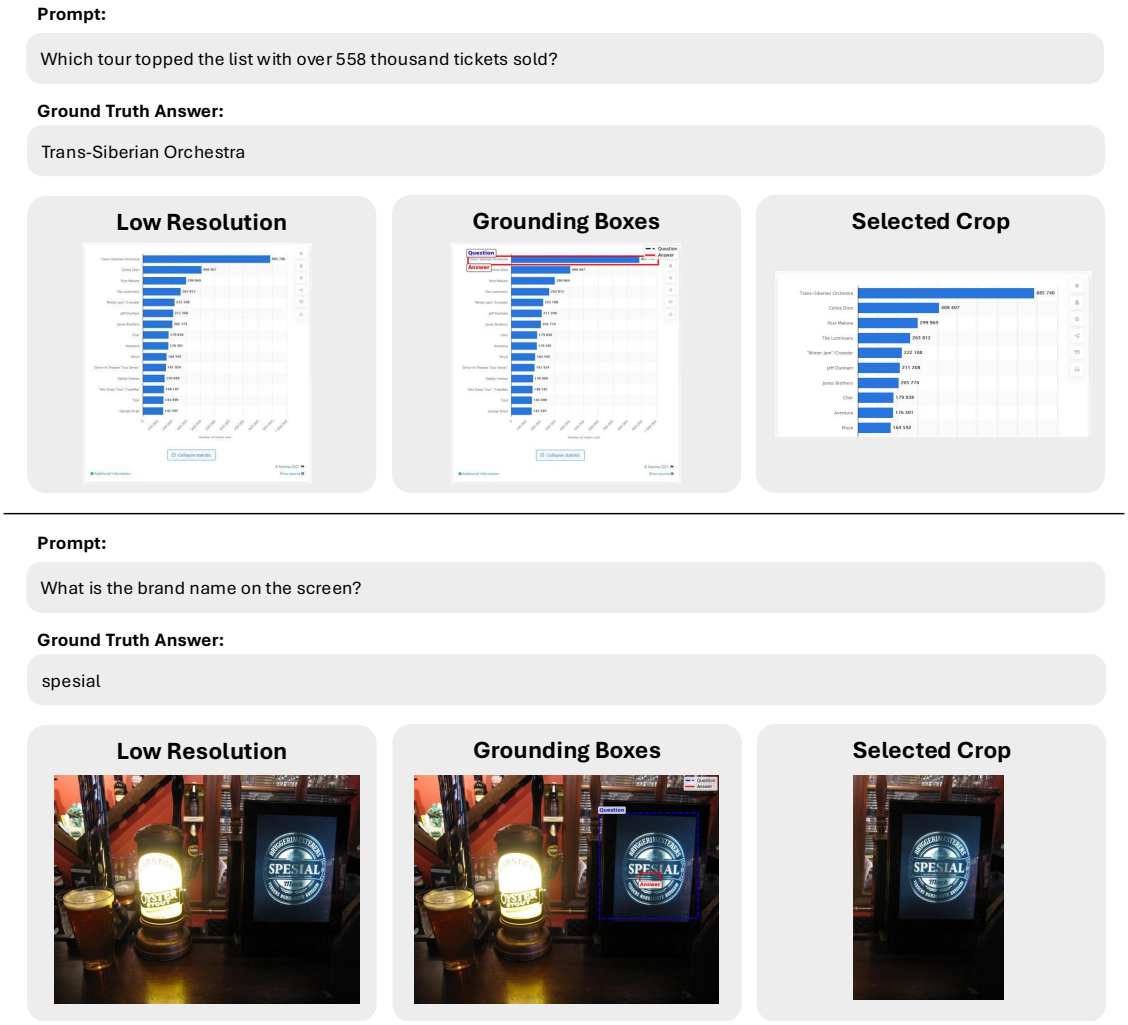


Figure 6: Visual examples from our data curation pipeline. Each row shows the low-resolution input image, the grounding model-detected bounding boxes (question region in blue, answer region in red), and the selected crop used for training. Top: A Chart example where the grounding model correctly localizes the relevant bar and its label. Bottom: A natural image example requiring OCR of a brand banner.

annotations (Kimhi et al., 2025a; Ryoo et al., 2023)), or correct crop that lead to a wrong answer. Each conversation shows the question, the tool call selected by the model, the predicted answer (in red) and the ground truth answer from the data, the low-resolution input, and the retrieved high-resolution crop.These failures typically arise when the question-relevant detail occupies a small or ambiguous portion of the image, causing the crop to capture nearby but insufficient context.

C Latency Analysis

We analyze the latency by first describing the hardware and measurement setup, then reporting detailed wall-clock results, and finally comparing response lengths to explain the observed efficiency gains.

C.1 Hardware and Measurement Setup

To compare VisionThink (Yang et al., 2025c) and AwaRes, we evaluate both methods using their respective evaluation implementations in lmms-eval (Zhang et al., 2025a), with both models running in native HuggingFace configuration. We measure wall-clock time (WC) from the start of generation until the final answer is produced (encompassing both turns when applicable).All measurements done on Nvidia-H100-80GB GPU.

C.2 Detailed Latency Results

Table 3 presents the latency (WC) comparison between AwaRes and competing dynamic methods across six benchmarks. AwaRes achieves the lowest latency on every benchmark and the lowest average overall. On average, AwaRes reduces wall-clock time by $4.4\times$ compared to VisionThink (from

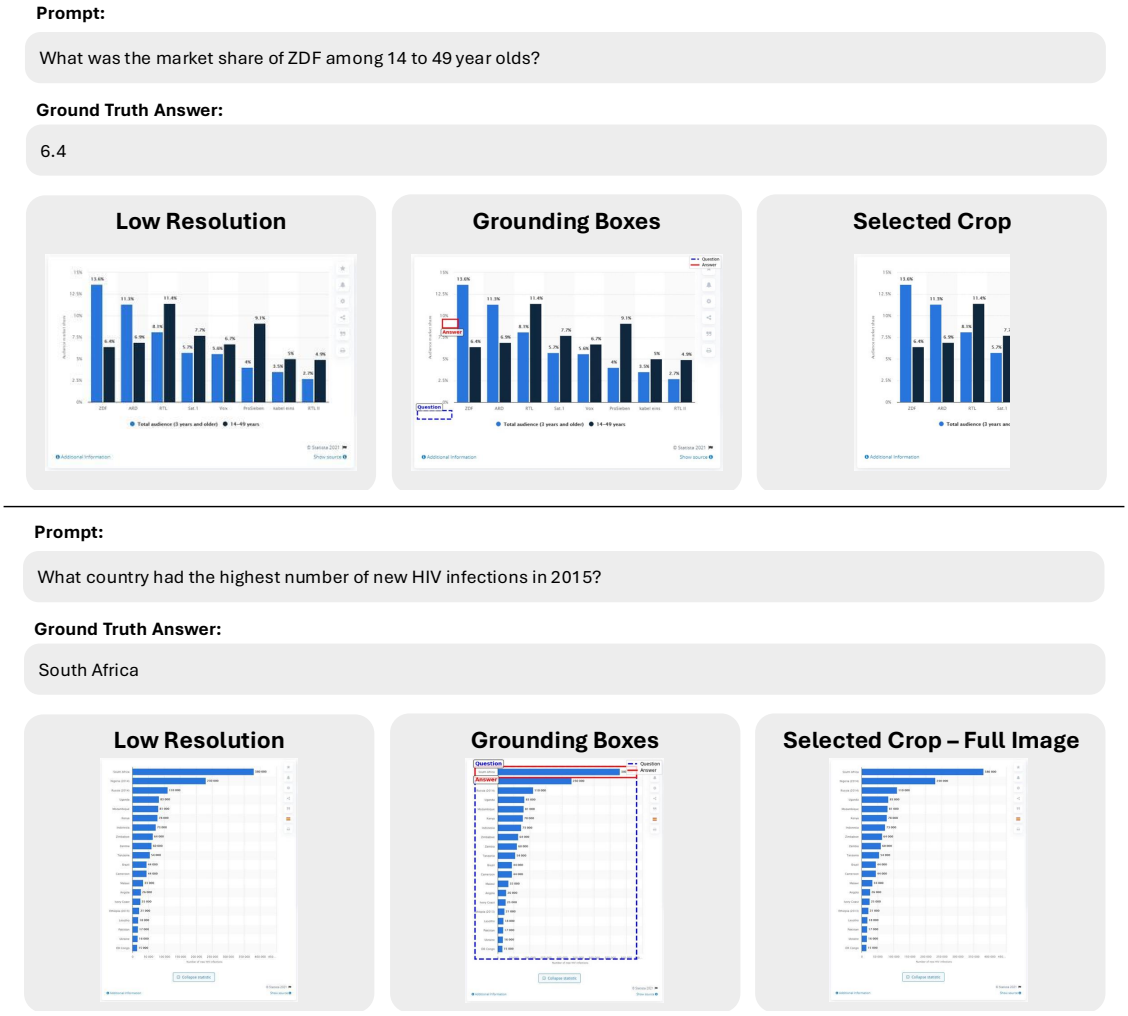


Figure 7: Failure cases from the automatic data curation pipeline. **Top:** The grounding model incorrectly localizes the answer region, missing the relevant bar (ZDF's 14-49 age group value). However, the crop regions helps mitigate such localization errors-the selected crop still contains the correct answer. **Bottom:** The grounding model correctly identifies the question and answer regions but the bounding boxes span nearly the entire image, resulting in a full-image crop selection. While this produces correct supervision, it is inefficient as no resolution savings are achieved.

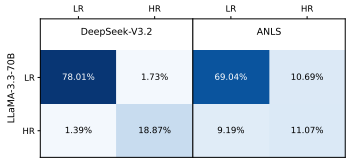


Figure 8: **Agreement on resolution-selection labels.**

2.71s to 0.61s), 3.9× compared to PixelReasoner (2.37s with low-resolution first turn), and over 9× compared to DeepEyes (6.03s-7.22s), whose iterative visual reasoning incurs the largest overhead. Even against the strongest baseline, ERGO with a low-resolution first turn (1.20s average), AwaRes remains 2× faster. The efficiency gains are most pronounced on memory- and reasoning-heavy benchmarks, where competing methods' extended reasoning traces or repeated high-resolution passes incur substantial overhead.

C.3 Response Length Comparison

The latency differences stem primarily from response length. As shown in Table 4, VisionThink generates substantially longer reasoning traces, approximately 5.8× to 28.8× more characters than AwaRes. This verbosity directly translates to increased generation time, as autoregressive decoding scales linearly with output length.

Beyond raw efficiency, AwaRes exhibits significantly lower variance in response length across all benchmarks. This predictability has practical implications: estimated completion times become more reliable, enabling better resource allocation and user experience. In contrast, VisionThink's high standard deviations—often exceeding the mean—make response length and latency highly unpredictable, complicating deployment in

Table 2: Comparing labeling strategies: LaaJ (LLaMA-3.3-70B) vs. ANLS. We report **Accuracy** ↑ and **RTR** ↓ across all benchmarks.

Label Strategy	ChartQA		DocVQA		OCRBench		POPE		RealWorld		V* Bench		**Average**	
	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓
ANLS	70.5	0.31	93.6	0.28	70.0	0.41	86.7	0.27	69.0	0.47	65.9	0.54	75.9	0.38
LLaMA-3.3-70B (LaaJ)	**77.0**	0.42	**94.0**	0.35	**78.8**	0.61	**88.0**	0.32	69.7	0.64	**70.7**	0.60	**79.70**	0.49

Table 3: **Latency comparison of dynamic methods.** Per-benchmark latency (seconds) for crop-inspired and adaptive methods. "LR" indicates the first turn uses a low-resolution image. AwaRes achieves the lowest average latency across all benchmarks. Best results per column in **bold**.

Model	ChartQA	DocVQA	OCRBench	POPE	RealWorld	V* Bench	**Average**
ERGO	1.82	3.18	1.16	1.09	2.42	2.60	2.05
ERGO (LR)	0.96	2.12	0.73	0.76	0.96	1.68	1.20
DeepEyes	6.15	7.48	5.13	8.19	7.03	9.33	7.22
DeepEyes (LR)	6.36	5.43	4.51	4.83	6.79	8.26	6.03
PixelReasoner	3.89	3.84	1.53	2.73	0.44	2.62	2.51
PixelReasoner (LR)	3.79	3.61	1.44	2.63	0.26	2.47	2.37
VisionThink	4.32	1.78	3.36	1.23	2.31	3.24	2.71
AwaRes	**0.56**	**0.51**	**0.64**	**0.50**	**0.66**	**0.81**	**0.61**

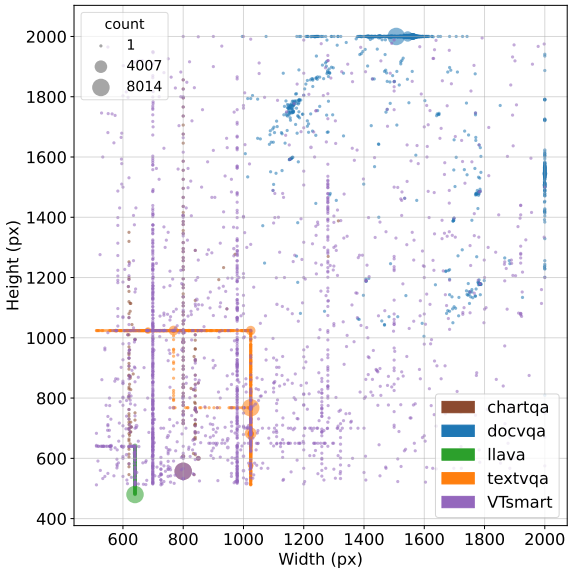


Figure 9: Resolution distribution of training datasets. Each point represents an image's width and height in pixels.

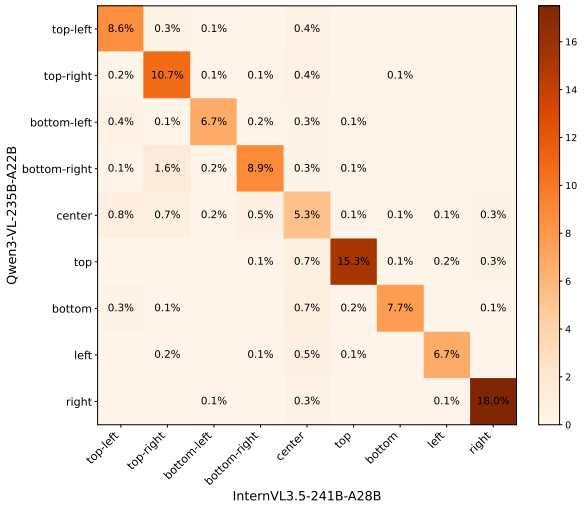


Figure 10: Crop confusion matrix between the two grounding model, conditioned on samples where high-resolution cropping helps. The diagonal concentration shows 88% agreement between the grounding models on the selected crop region.

latency-sensitive applications.

D Additional Cold-Start (SFT) Analysis

This supplementary section provides additional evidence that the cold-start stage learns a *coupled-decision policy (CDP)* whose first-turn action jointly determines *when* to request additional resolution ($C = \emptyset$ vs. $C \neq \emptyset$) and, when escalating, *where* to look via the chosen crop subset $C \subseteq \mathcal{C}$. Beyond final-task accuracy, we therefore report policy-centric diagnostics that quantify (i) the tendency to call the crop tool, (ii) failure to escalate

Table 4: Response verbosity comparison between VisionThink and AwaRes across the benchmarks. We report the mean ± std number of characters in model responses. AwaRes produces substantially shorter responses than VisionThink (5.8×–28.8× reduction), reflecting its efficiency beyond visual token savings.

Benchmark	samples	VisionThink	AwaRes	Ratio
ChartQA	2500	89.46 ± 205.71	6.54 ± 6.16	13.7×
DocVQA (val)	5349	79.95 ± 242.21	13.74 ± 12.79	5.8×
OCRBench	1000	178.67 ± 540.12	16.23 ± 16.39	11.0×
POPE	9000	46.46 ± 155.52	2.84 ± 2.33	16.4×
RealWorldQA	765	160.67 ± 261.55	5.57 ± 5.86	28.8×
V*Bench	191	118.54 ± 232.17	5.29 ± 5.78	22.4×

when detail is required, and (iii) the amount of high-resolution evidence requested.

D.1 Metrics for CDP diagnostics

For each evaluated sample with resolution-sufficiency label $y \in \{\texttt{LR}, \texttt{HR}\}$, the model either takes a no-call action ($C = \emptyset$) or requests one or more crops ($C \neq \emptyset$). We evaluate the CDP as two coupled components:

Call decision (*when* to crop). We treat tool invocation as a binary classifier where the positive class is $y = \texttt{HR}$ and a prediction is positive iff $C \neq \emptyset$. We report:

- **Call Precision** ↑: $\mathbb{P}(y = \texttt{HR} \mid C \neq \emptyset)$.
- **Call Recall** ↑: $\mathbb{P}(C \neq \emptyset \mid y = \texttt{HR})$.
- **Call F1** ↑: $2\,\mathrm{Prec} \times \mathrm{Rec}/(\mathrm{Prec} + \mathrm{Rec})$.
- **FPR (LR-call)** ↓: $\mathbb{P}(C \neq \emptyset \mid y = \texttt{LR})$.

Region decision (*where* to look | call). Conditioned on $C \neq \emptyset$, we measure overlap between the requested crop set and the grounding model target crops $\mathcal{C}^\star$ used for SFT supervision. We report:

- **Exact match** ($\mathrm{IoU} = 1$) ↑: the predicted region matches the grounding model target exactly.
- **Relaxed match** ($\mathrm{IoU} \geq 0.25$) ↑: the predicted region overlaps a grounding model target by at least 0.25 IoU, accounting for the hierarchical structure of $\mathcal{C}$ (e.g., for a quadrant target, its two adjacent half-image regions are also considered acceptable; likewise, `All` and `Center` may be acceptable when they satisfy the IoU threshold).
- **Avg. area** ↓: $\mathbb{E}[s(C)]$, the average fraction of image area requested when calling.

In addition, we report **Accuracy** (↑) and **RTR** (↓) across all benchmarks, consistent with the main paper.

D.2 CDP behavior across cold-start variants

Table 5 decomposes cold-start behavior into the two components of the coupled-decision policy (CDP): the *call decision* (*when* to request crops) and the *region decision* (*where* to look, conditioned on calling).

Call decision (*when* to crop). Trajectory-level SFT substantially improves the calibration of the first-turn decision relative to baseline SFT. While baseline SFT achieves moderate precision (62.49) it exhibits very low recall (15.34) and a high false-positive rate on LR samples (79.69), indicating that tool invocation is both unreliable on HR cases and overly frequent on LR cases. Trajectory-level SFT increases recall to 41.02 and reduces FPR to 63.33, yielding a higher overall F1 score (24.63→44.56). Further upweighting the tool-call turn (w_t=5) strengthens this behavior: precision rises to 77.8, recall to 47.70, and FPR drops to 49.85, improving F1 to 59.14. Together, these results indicate that emphasizing the first-turn action stabilizes the fused CDP decision of whether to escalate.

Region decision (*where* to look | call). Trajectory-level SFT also improves alignment with grounding model's supervision, increasing exact region match (IoU$=$ 1) from 13.8 to 15.9 and relaxed overlap match (IoU$\geq$ 0.25) from 32.6 to 48.85, while reducing the average requested area from 0.59 to 0.402. Upweighting the tool-call turn yields a large jump in localization quality (IoU$=$ 1: 41.3; IoU$\geq$ 0.25: 75.5), with a modest increase in requested area (0.402→0.463). This suggests that stronger supervision of the tool-call turn improves not only the decision to escalate but also the selection of informative regions once escalation occurs.

Overall, improved cold-start recipes shape both parts of the CDP: they improve the reliability of escalation (*when*) and the quality of evidence localization (*where*), while keeping the requested high-resolution area controlled. This motivates the subsequent GRPO stage, which further refines the same fused CDP under an explicit accuracy–efficiency objective.

Table 5: **Cold-start (SFT) policy diagnostics for the coupled-decision policy (CDP).** We evaluate the *call decision* as a binary classifier where the positive class is $y = \texttt{HR}$ and a prediction is positive iff the model calls the tool ($C \neq \emptyset$). We report precision, recall, and their harmonic mean (F1), as well as the false-positive rate on LR samples (FPR$= \mathbb{P}(C \neq \emptyset \mid y = \texttt{LR})$). For the *region decision* (conditioned on calling), we report overlap-based match rates to grounded targets using exact region match ($\mathrm{IoU} = 1$) and relaxed overlap ($\mathrm{IoU} \geq 0.25$), along with the average requested area. Higher is better (↑) unless noted.

	CDP: call decision				CDP: region decision | call		
Model	Call Prec.↑	Call Rec.↑	Call F1↑	FPR (LR-call)↓	IoU=1↑	IoU≥0.25↑	Avg. area↓
SFT (baseline)	62.49	15.34	24.63	79.69	13.8	32.6	0.59
SFT + Traj.	48.75	41.02	44.56	63.33	15.9	48.85	0.402
SFT + Traj. + w_t=5	77.8	47.70	59.14	49.85	41.3	75.5	0.463

D.3 Cold-Start ablations

We analyze how cold-start SFT design choices shape both downstream performance and the induced coupled-decision policy (CDP). We isolate three factors—trajectory-level supervision, phased (tool-first) scheduling, and high-resolution upsampling—and study their interaction with the tool-turn weight w_t. Beyond accuracy and RTR, we examine how each setting affects when and how aggressively the policy escalates, as well as the reliability of the tool-call interface itself.

D.4 Cold-start sensitivity to data and parameters

Table 7 summarizes how common cold-start knobs shape both downstream performance and the induced coupled-decision policy (CDP). Trajectory-level SFT improves average accuracy over vanilla (77.90 vs. 75.15), but it also shifts the policy toward more frequent crop requests (call rate 22.55%→25.21%) while substantially reducing the average requested area (0.59→0.402), indicating that the model learns to localize with smaller crops rather than relying on large regions. Increasing the tool-turn weight to w_t=5 further strengthens the first-turn CDP action, yielding the best accuracy (79.70) but also the highest RTR (0.49), consistent with a policy that escalates more often (call rate 29.02%) and requests slightly larger regions (Avg. area 0.463).

Beyond w_t, data and schedule choices can trade off calibration and efficiency. In particular, HR upsampling is expected to reduce missed escalations by exposing the policy to more detail-critical instances, but may increase tool usage if applied aggressively; phased (tool-first) schedules can stabilize tool invocation but sometimes change how strongly the policy couples region selection to answer quality. Overall, the best cold-start configuration is the one that yields a reliable CDP (low misses and good region selection) while keeping call rate and requested area controlled, leaving GRPO to fine-tune efficiency rather than compensating for frequent cold-start failures.

D.5 Tool-call formatting reliability

Finally, Table 8 quantifies tool-call formatting reliability after cold-start by measuring whether a generated tool call can be parsed into a valid crop subset $C \subseteq \mathcal{C}$ without any post-processing. This isolates protocol learning from downstream accuracy: malformed outputs can prevent escalation even when the policy intends to call the tool, effectively breaking the CDP at the interface level.

Trajectory-level SFT substantially improves tool-call validity, reducing corruption from 10.17% to 1.43%. Moreover, the remaining failures are not merely cosmetic: a large fraction of corrupted calls lead to *incorrect* crop requests even after simple recovery (5.48% for baseline SFT vs. 0.94% for Traj.). Upweighting the tool-call turn (w_t=5) eliminates formatting corruption entirely (100% valid parse), supporting our design choice to explicitly emphasize the first-turn action during cold-start.

E GRPO ablations

We ablate the key design choices in our training pipeline: the role of the cold-start SFT stage, the structure of the tool-cost penalty, and the choice of correctness reward. Table 9 reports accuracy and retained token ratio (RTR) across six benchmarks.

Cold start vs. GRPO only. Training only the reference policy with SFT yields strong accuracy (79.70 average) but at a high compute cost (0.49 RTR), as the model has no incentive to crop economically. Conversely, running GRPO without a cold start drives RTR down sharply (0.31) but sacrifices accuracy (77.11), since the model never acquires a reliable cropping prior before reward optimization begins. AwaRes combines both stages, retaining the accuracy of SFT (80.30) while approaching the efficiency of GRPO-only (0.36 RTR), confirming that the two stages are complementary rather than redundant.

Tool-cost penalty. The area-based cost is what couples accuracy to efficiency. Removing the tool cost entirely raises RTR to 0.51 with no accuracy

Table 6: Cold-start ablations. We report **Accuracy** ↑ and **RTR** ↓ across all benchmarks.

Components				ChartQA		DocVQA		OCRBench		POPE		RealWorld		V* Bench		**Average**	
Traj.	**Phased**	**Ups.**	w_t	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓	Acc ↑	RTR ↓
x	x	x	1	72.9	0.35	93.1	0.42	73.7	0.32	85.8	0.29	64.1	0.33	61.3	0.42	75.15	0.36
v	x	x	1	69.8	0.38	93.8	0.32	76.9	0.54	87.8	0.30	70.5	0.55	68.6	0.48	77.90	0.43
v	x	v	1	76.9	0.30	93.2	0.27	72.9	0.41	84.7	0.25	69.0	0.43	66.0	0.55	77.12	0.37
v	v	x	1	70.9	0.36	93.4	0.40	76.1	0.32	86.6	0.30	68.8	0.34	64.4	0.41	76.70	0.36
v	v	v	1	70.8	0.33	92.7	0.37	72.7	0.31	85.7	0.28	66.8	0.33	63.9	0.43	75.43	0.33
v	x	x	1	69.8	0.38	93.8	0.32	76.9	0.54	87.8	0.30	**70.5**	0.55	68.6	0.48	77.90	0.43
v	x	x	5	**77.0**	0.42	**94.0**	0.35	**78.8**	0.61	**88.0**	0.32	69.7	0.64	**70.7**	0.60	**79.70**	0.49

Table 7: **Sensitivity of cold-start SFT.** We vary data/objective knobs and report both downstream performance (Avg. Accuracy and Avg. RTR across all benchmarks) and induced CDP behavior (call rate and average requested area, both measured over evaluation prompts). Call rate is $\mathbb{P}(C \neq \emptyset)$ and Avg. area is $\mathbb{E}[s(C)]$ (fraction of image area requested when calling).

Setting	HR upsample	w_t	Traj. SFT	Phased	Acc↑	RTR↓	Call rate	Avg. area↓
Default SFT	x	1	x	x	75.15	0.36	22.55%	0.59
Traj. only	x	1	v	x	77.90	0.43	25.21%	0.402
Traj. + high w_t	x	5	v	x	79.70	0.49	29.02%	0.463
Traj. + HR upsample	v	1	v	x	77.12	0.37	23.0%	0.35
Phased (tool-first)	x	1	v	v	76.70	0.36	24.0%	0.44

Table 8: **Tool-call formatting reliability after cold-start.** Valid parse indicates that the tool output can be parsed into a crop subset $C \subseteq \mathcal{C}$ without post-processing. Corrupt outputs are split into *recoverable* cases (simple extraction of intended crop id(s) succeeds) and *incorrect* cases (post-processing yields a wrong crop).

Model	Valid parse (%)↑	Corrupt outputs (%)↓		
		Total	Recoverable	Incorrect
SFT (baseline)	89.83	10.17	4.69	5.48
SFT + Traj.	98.57	1.43	0.49	0.94
SFT + Traj. + w_t=5	100.00	0.00	0.00	0.00

benefit, as the model over-crops indiscriminately. Setting $\lambda = 0$ retains the misuse penalty but drops the area term, leaving RTR at 0.42—better than no cost, but still well above AwaRes's 0.36. This shows the area-based term, not just the binary misuse signal, is responsible for the efficiency gains.

Correctness reward. We compare our text-similarity reward (sentence-transformer) against ANLS and LLM-as-a-judge (LaaJ). All three reach comparable accuracy (∼80) and efficiency (∼0.37 RTR), but text similarity offers the best trade-off between semantic matching and robustness to short-answer phrasing variation, without the latency and cost of an LLM judge in the reward loop.

E.1 GRPO learns beyond grounding annotations

The grounding model provides only noisy localization supervision for cold-start SFT: it does not provide final answers. The final policy is optimized by GRPO with answer-correctness and crop-cost rewards, shifting from grounded model annotations towards better policy.

First, zero-shot grounding model tool use is insufficient. The strong grounding model drops from 88.9 average accuracy in native HR mode to 79.89 under the tool interface, with large drops on V^* (90.6 → 73.8), ChartQA (90.8 → 69.7), and OCRBench (86.9 → 77.8). AwaRes, despite being 7B, reaches 80.30 average accuracy, and outperforms the tool-prompted grounding model (MOE, 235B-A22B) on ChartQA (80.6 vs. 69.7) and OCRBench (81.3 vs. 77.8). Thus, grounding model annotations alone do not solve the task. The learned crop distribution differs from the grounding model annotations, as shown in Figure 18

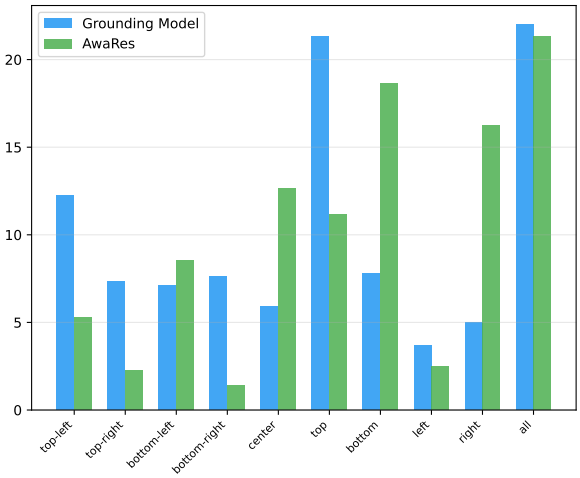


Figure 18: GRPO changes the crop policy relative to grounding model supervision instead of copying it's crop choices.

F Training Details

We provide detailed hyperparameters for both training stages of AwaRes: Cold-Start SFT and Tool Optimization via GRPO. Table 10 conclude all the training parameters for both stages.

Table 9: **GRPO Ablations.** Top: comparing training the model only for the $\pi_{\theta_{ref}}$ (SFT) vs only GRPO (no cold start). Middle: effect of the tool cost penalty, $\lambda = 0$ removes the area-based cost while retaining the misuse penalty, and removing the tool cost entirely leads to further over-cropping. Bottom: comparison of correctness reward functions. Text similarity (sentence-transformer) balances semantic matching and robustness, outperforming both ANLS and LLM-as-a-judge on short-answer evaluation. RTR denotes the fraction of full-resolution compute used (↓ is better).

	ChartQA		DocVQA		OCRBench		POPE		RealWorldQA		V* Bench		Average	
Model	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓	Acc. ↑	RTR ↓
SFT only	77.0	0.42	94.0	0.35	78.8	0.61	88.0	0.32	69.7	0.64	70.7	0.60	79.70	0.49
GRPO only	76.48	0.25	93.98	0.25	72.20	0.31	85.91	0.25	67.06	0.32	67.02	0.54	77.11	0.31
w/o tool cost	80.60	0.44	94.60	0.37	81.50	0.64	87.80	0.34	69.40	0.65	71.20	0.62	80.85	0.51
w/o area cost ($\lambda = 0$)	81.28	0.38	94.42	0.37	80.70	0.40	85.93	0.37	69.28	0.41	71.20	0.58	80.47	0.42
Text-Sim→ ANLS	79.84	0.29	94.35	0.25	80.00	0.41	85.78	0.23	69.67	0.43	70.16	0.57	79.96	0.37
Text-Sim→ LaaJ	79.00	0.29	94.20	0.25	80.30	0.42	85.80	0.24	69.40	0.44	70.70	0.58	79.90	0.37
AwaRes	80.64	0.32	94.43	0.28	81.30	0.42	85.73	0.27	68.5	0.43	71.20	0.41	80.30	0.36

Table 10: Training hyperparameters for Cold-Start SFT and Tool Optimization (GRPO) stages.

Parameter	**Cold-Start (SFT)**	**Tool Opt. (GRPO)**
Model Configuration		
Base model	Instruct HF model	Cold-Start model
LORA Rank	8	8
Optimization		
Optimizer	AdamW	AdamW
Learning rate	1e-4	5e−5
LR schedule	Cosine	Linear
Warmup ratio	0.05	0
Weight decay	0.001	0
Max gradient norm	-	1.0
Batch Configuration		
Per-device batch size	16	8
Effective batch size	128	64
Number of epochs	2	1
Sequence Length		
Max seq. / prompt length	8192	512
Max completion length	512	512
Stage-Specific		
Tool-turn weight w_t	5	–
Trajectory-level opt.	True	–
Number of generations G	–	8
Temperature τ	–	1.0
Top-p sampling	–	1.0
KL coefficient β	–	0.05
Crop cost weight $(1 - \alpha)$	–	0.25
Area weight $(1 - \alpha)$	–	0.01
Answer reward weight w_a	–	10.0

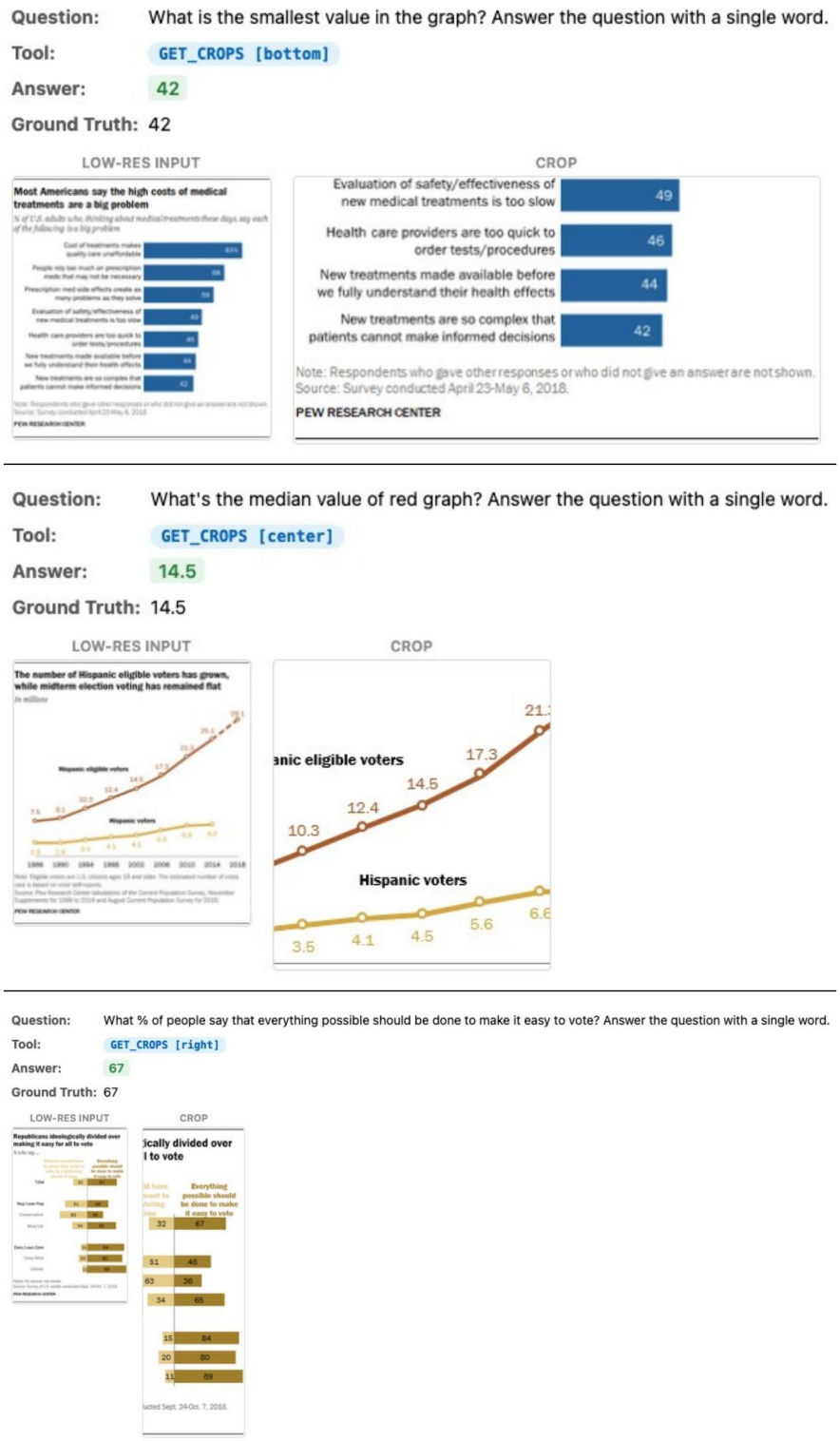


Figure 11: **Positive examples of AwaRes's adaptive cropping (1/6).** The model isolates the relevant bar segments and axis labels to identify the smallest chart value, zooms into data points along a trend line to compute a median, and focuses on labeled chart categories to read a specific percentage, extracting precise numerical values that are indiscernible in the low-resolution overview alone.

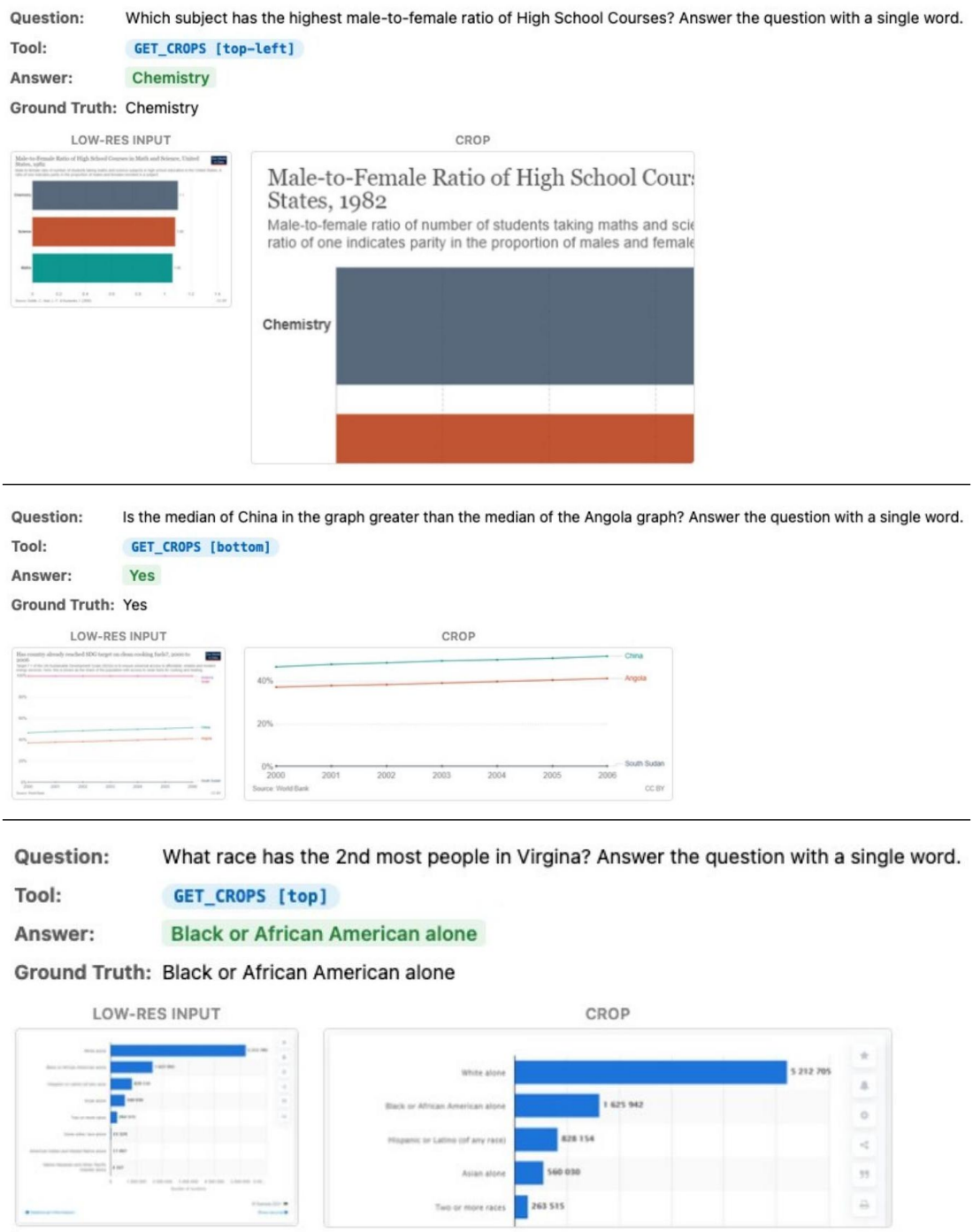


Figure 12: **Positive examples of AwaRes's adaptive cropping (2/6).** The crops target the top bar and its label to identify the subject with the highest ratio, zoom into line-chart trends and country labels to compare medians across countries, and focus on the top entries of a horizontal bar chart to read demographic rankings. In each case, the high-resolution crop brings the answer-relevant region into sharp focus.

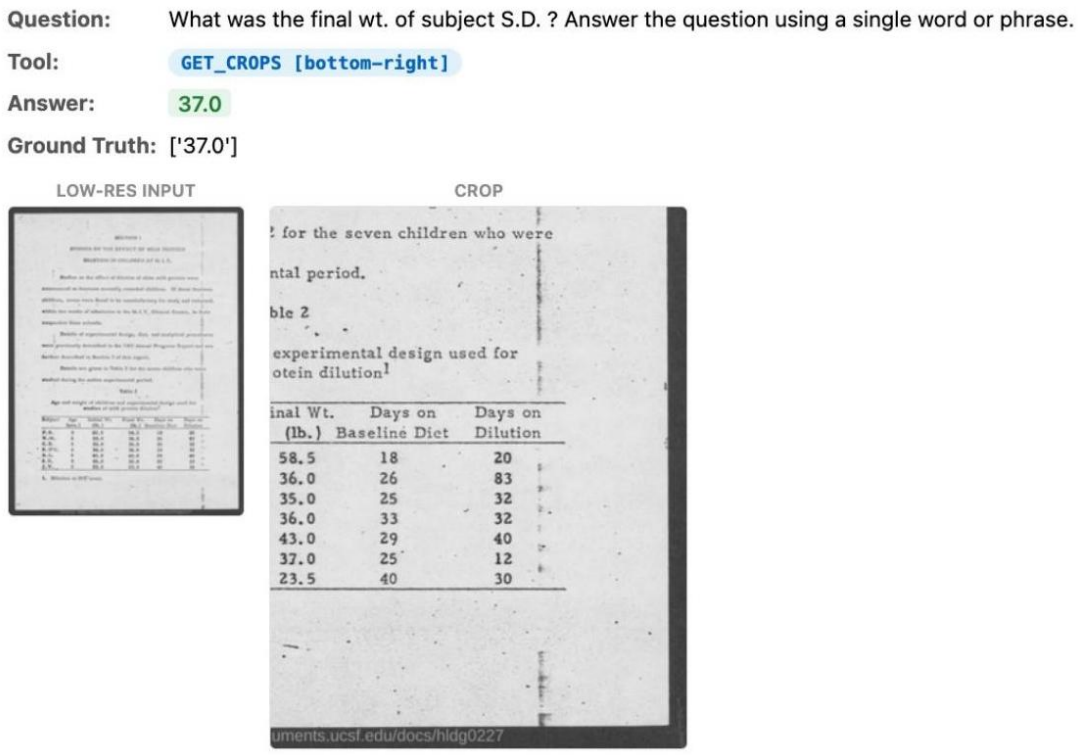




Figure 13: **Positive examples of AwaRes's adaptive cropping (3/6).** The model crops into table cells of a scanned document to extract a specific weight value, and zooms into a wine bottle label to read the variety name. These examples demonstrate how targeted crops resolve fine-grained text on documents and labels that lack of OCR capabilities at low resolution.

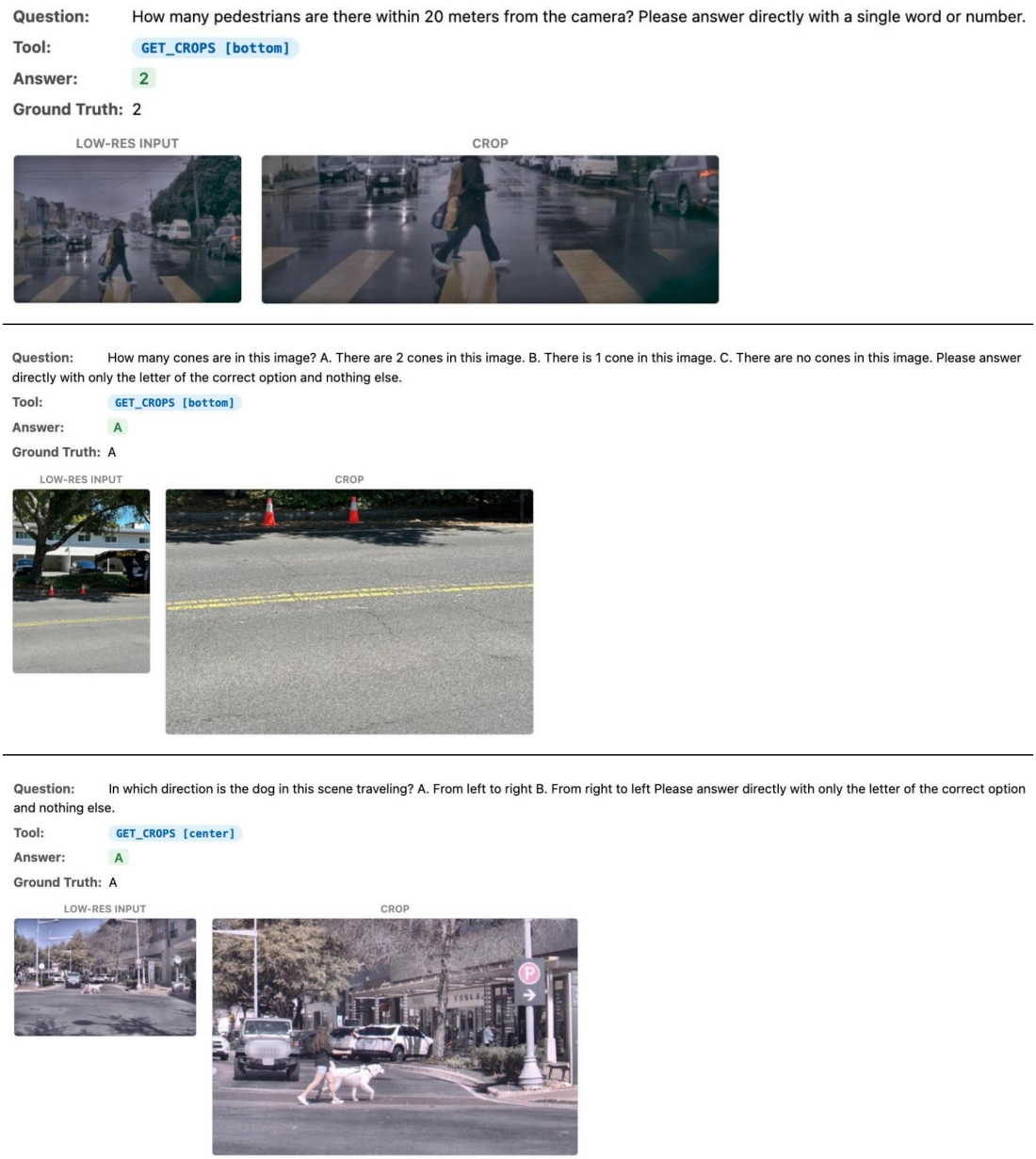


Figure 14: **Positive examples of AwaRes's adaptive cropping (4/6).** The model correctly crops to count pedestrians in a rainy street scene, identify and count traffic cones in a parking lot, and determine the travel direction of a dog in an urban setting. The crops consistently center on the objects referenced by the question, enabling accurate spatial reasoning and counting from the high-resolution view of outdoor scenarios.

Question: What kind of animal is in the poster? A. dove B. horse C. dog D. cat Answer with the option's letter from the given choices directly. Answer with the option's letter from the given choices directly.

Tool: `GET_CROPS [center]`

Answer: A

Ground Truth: A

LOW-RES INPUT



CROP



Question: What is the color of the van? A. red B. white C. silver D. black Answer with the option's letter from the given choices directly. Answer with the option's letter from the given choices directly.

Tool: `GET_CROPS [bottom-left]`

Answer: A

Ground Truth: A

LOW-RES INPUT



CROP



Question: What is the color of the broom? A. black B. yellow C. gray D. red Answer with the option's letter from the given choices directly. Answer with the option's letter from the given choices directly.

Tool: `GET_CROPS [right]`

Answer: A

Ground Truth: A

LOW-RES INPUT



CROP

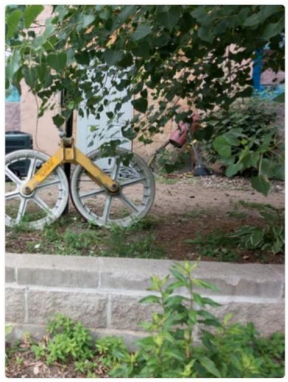


Figure 15: **Positive examples of AwaRes's adaptive cropping (5/6).** The crops focuses on the church building top and detect the dove, zoom into the road next to the boats at a riverfront to resolve a van's color, and focus on a broom among garden clutter to identify its color. These examples show how AwaRes localizes small or partially occluded objects that the question refers to, enabling fine-grained attribute recognition.

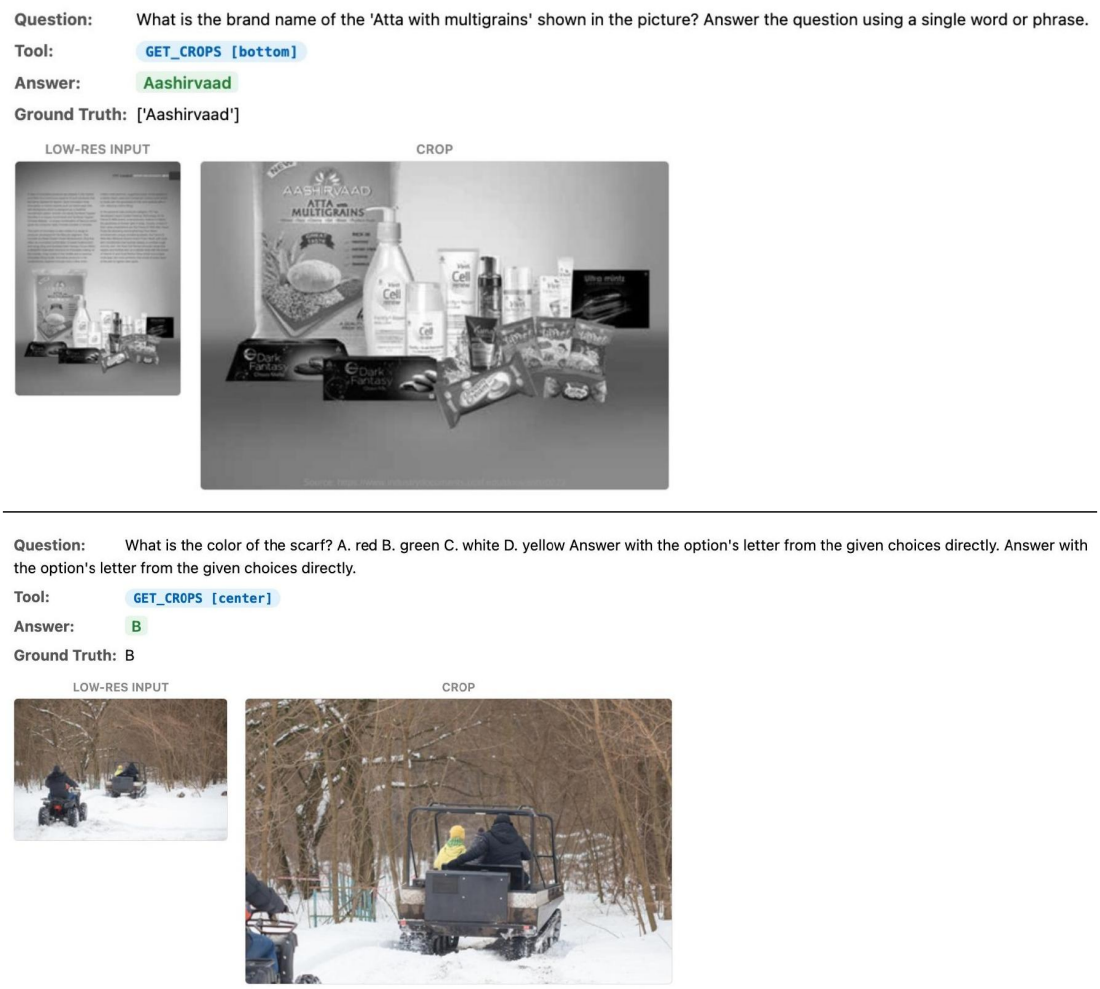


Figure 16: **Positive examples of AwaRes's adaptive cropping (6/6).** The model focuses on product packaging in a cluttered display to identify a brand name, and crops into the center of a snowy scene to zoom in on a person riding an ATV, resolving the green color of their scarf.

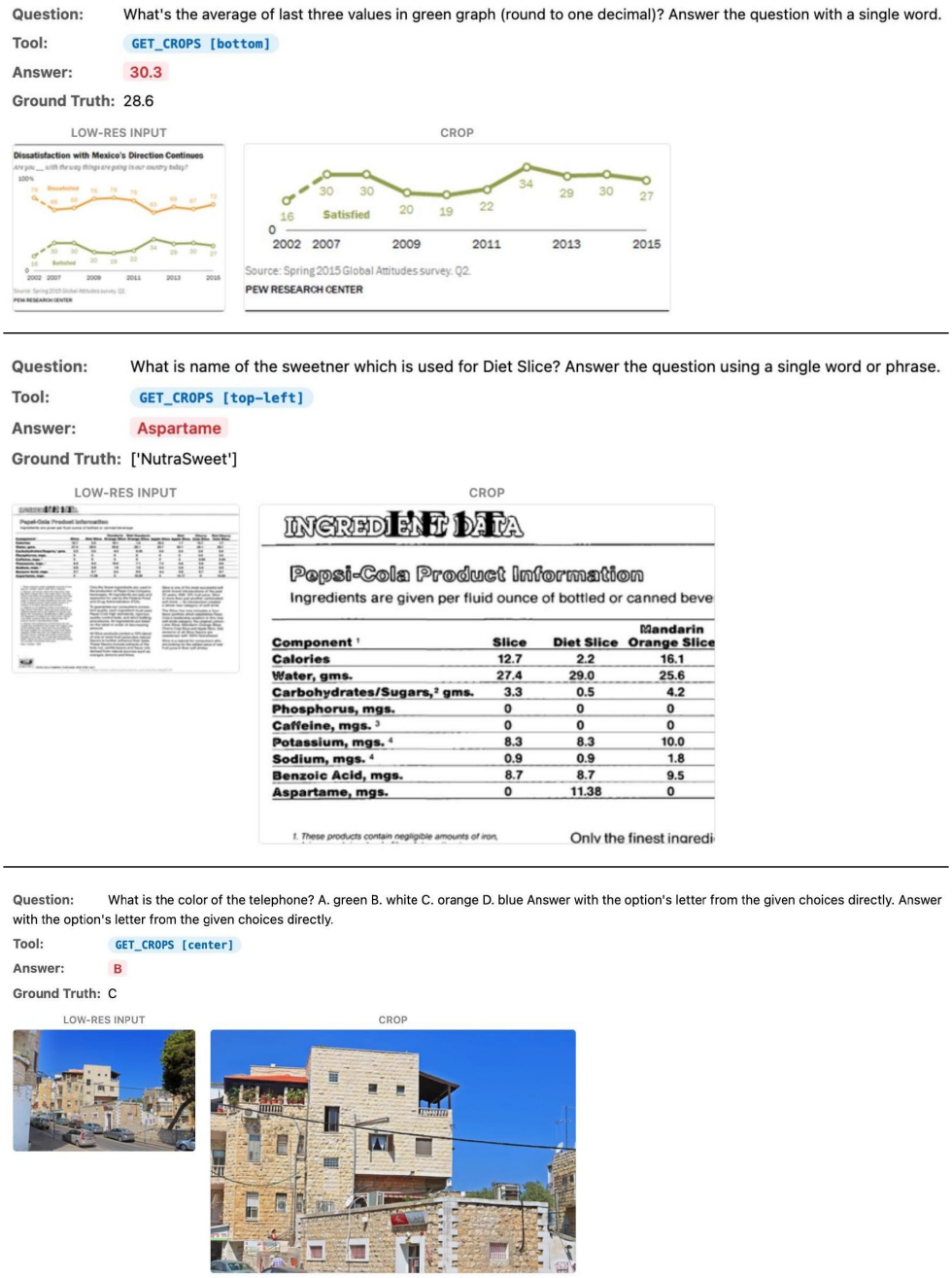


Figure 17: **Failure cases of AwaRes's adaptive cropping.** The crop either target the wrong region due to lack of sufficient context from the question, or produce a crop that targets a plausible region but fail to produce the correct answer. In the first example, the crop isolates the correct line, yet the model skips the value at 2013 and use 2012 instead, yielding the wrong average. In the second, the crop zooms into the `Nutrition Facts label`, instead of the ingredient list and reads "Aspartame" instead of the "NutraSweet". In the third, the crop focuses on the center, where the phone is located, but the granularity of the crop keep access of information around the object that makes the model misidentify its color.

G Prompts

G.1 LLM-as-a-Judge Prompt

LLM-as-Judge Prompt

```
You are tasked with evaluating which of two AI responses better answers the given question based
on the provided ground truth.
Original prompt: {prompt}
Ground Truth Answer: {gt}
Response 1:
{lr_resp}
Response 2:
{hr_resp}
Compare both responses against the ground truth answer and respond with just one number, DO NOT
include any other text:
0 - Both responses are equally good or equally bad, no significant difference
1 - Response 1 is better
2 - Response 2 is better
```

G.2 Grounding Prompt

Grounding model Localization Prompt

```
You are an expert visual assistant. Your task is to analyze an image and determine which regions
are relevant for both understanding and answering a given question.
Instructions:
1. Analyze the question and identify what visual information is needed
2. Localize TWO types of regions:
  a) QUESTION region: The area containing visual elements mentioned in or relevant to the question
  b) ANSWER region: The area containing the answer to the question
3. For each region type, provide a bounding box in normalized coordinates [x_min, y_min, x_max,
y_max]
  - Coordinates should be between 0 and 1000 (normalized to image dimensions)
  - x_min, y_min: top-left corner of the bounding box
  - x_max, y_max: bottom-right corner of the bounding box
4. Provide tight bounding boxes that closely encompass only the relevant text/visual elements
5. Output ONLY a Python dictionary with two keys: 'question' and 'answer' - NOTHING ELSE!!
CRITICAL: Your entire response must be ONLY a Python dictionary. Do not include any explanation,
reasoning, or other text.
Output format:
{
  'question': [[x_min, y_min, x_max, y_max]],
  'answer': [[x_min, y_min, x_max, y_max]]
}
Notes:
- If the question and answer regions overlap significantly or are the same, you may provide the
same coordinates for both
- If multiple regions are needed for either question or answer, include multiple bounding boxes in
the list
- Keep bounding boxes as tight as possible around the relevant text or visual elements
Your response must be ONLY: {'question': [[x_min, y_min, x_max, y_max], ...], 'answer': [[x_min,
y_min, x_max, y_max], ...]}
No other text, no explanation, just the dictionary.
Question: ...
```

G.3 SFT / GRPO / Inference Prompt

You are a vision-language model that analyzes images and answers questions about them. If the image resolution is too low for accurate analysis, respond with **GET_CROPS:** followed by a list of crop numbers in square brackets (e.g., GET_CROPS: ['3'] or GET_CROPS: ['0', '5']), where the available crop numbers and their corresponding areas are {'0': 'top-left', '1': 'top-right', '2': 'bottom-left', '3': 'bottom-right', '4': 'center', '5': 'top', '6': 'bottom', '7': 'left', '8': 'right', 'all': 'all'}, otherwise provide your answer.

DistilVDR: A Compact End-to-End Visual Document Retriever via Dual-Student Distillation

Zhuchenyang Liu[1] **Ziyi Wang**[2] **Yao Zhang**[1] **Yu Xiao**[1]
[1]Aalto University, Finland
[2]Independent Researcher, Netherlands
zhuchenyang.liu@aalto.fi

Abstract

Visual document retrieval (VDR) is dominated by multi-billion-parameter models that are slow to index at full corpus scale and expensive to serve. Prior compression routes either train a smaller multi-vector encoder from scratch or distil only the query side; neither yields a compact single-vector retriever end-to-end. We present DistilVDR[1], a 527 M end-to-end VDR system distilled bilaterally from a single 8 B vision-language teacher under a pointwise cosine alignment loss. All supervision comes from the frozen teacher's embedding space, which was itself trained with relevance supervision, so the student objective needs no relevance labels, negative sampling, or contrastive term. We match VDR's text-query and image-document input asymmetry with an asymmetric encoder-only student that concentrates visual capacity on the document side and keeps the query side at 70 M parameters. We release two variants that share the same encoders and training and differ only in the document encoder's visual-tile budget: DistilVDR-HiRes attains 61.74 average NDCG@5 on ViDoRe v1+v2+v3 (86.9 % of the 8 B teacher) and leads every reproduced sub-1 B baseline on the high-resolution-sensitive v3 benchmark, while DistilVDR-Fast attains 59.98 at a 3× smaller visual-token budget. Both variants store one million documents in a 15.6 times smaller index than the strongest sub-1 B multi-vector baseline, and DistilVDR-Fast indexes 5.2 times faster than that baseline and 18 times faster than the teacher.

1 Introduction

Enterprise document search, technical documentation lookup, and visual retrieval-augmented generation (Yu et al., 2025) all require retrieving relevant document pages from a corpus given a text query. Visual document retrieval (VDR) addresses this task. VDR encodes each page directly as an image. This avoids the errors of an Optical Character Recognition (OCR) or layout extraction pipeline and preserves tables, figures, and layout structure.

State-of-the-art VDR systems are heavy. Top-performing retrievers span two to eight billion parameters across both single-vector and multi-vector paradigms; examples include the single-vector Qwen3-VL-Embedding-8B (Li et al., 2026a) and the multi-vector Tomoro-ColQwen3-8B (TomoroAI, 2025). These models require more than 16 GB of GPU memory to embed a single document. Indexing a million-document corpus with such a model costs tens of GPU-hours.

Two lines of work have addressed this cost. The first line trains a smaller encoder from scratch and combines it with multi-vector late interaction (Khattab and Zaharia, 2020; Santhanam et al., 2022b,a). Per-token storage compensates for the reduced parameter count. Representative models include colSmol-500M (ViDoRe Team, 2024), SauerkrautLM-ColLFM2-450M (VAGO Solutions, 2025), and ModernVBERT (Teiletche et al., 2025), all built on the ColPali architecture (Faysse et al., 2025). Among these, ModernVBERT departs from the ColPali decoder lineage with an encoder-only design. It combines a SigLIP2 visual encoder (Tschannen et al., 2025) with a ModernBERT text backbone (Warner et al., 2025). Teiletche et al. (2025) report that this encoder-only design outperforms an equivalent causal decoder on document retrieval by 1.6 NDCG@5 in the single-vector setting we adopt, and by 10.6 in the late-interaction setting.

However, all systems in this line pay the multi-vector cost. Per-token storage inflates the index by an order of magnitude, and late-interaction MaxSim scoring inflates per-query latency by two orders of magnitude relative to a single dot product. A body of work attacks these costs post hoc,

[1]Code and training scripts: https://github.com/Ryenhails/NanoVDR. Model checkpoints: https://huggingface.co/nanovdr.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 385–399
October 24, 2026 ©2026 Association for Computational Linguistics

by pruning or merging patches in an already-built index (Ma et al., 2025; Yan et al., 2026), by fusing internal transformer layers into a more discriminative embedding (Li et al., 2026b), or by reranking single-vector candidates with a multi-vector model (Kim et al., 2025). All of it presumes an already-trained strong base model, so the training and deployment cost of the underlying multi-billion-parameter retriever remains. Going single-vector within the same architecture removes the index cost but loses quality: BiModernVBERT is 17.6 NDCG@5 below ColModernVBERT on ViDoRe v1 and 20.3 below it on v2 (Teiletche et al., 2025). A small encoder trained from scratch with contrastive learning therefore struggles to produce a single-vector representation that matches multi-vector retrieval quality.

The second line distils a small student from a large teacher. This is a well-established paradigm in text retrieval. A heavier teacher, typically a cross-encoder reranker or a larger bi-encoder (Hofstätter et al., 2021; Chen et al., 2024a), supervises a lighter bi-encoder student, which inherits much of the teacher's quality at a fraction of the inference cost. In visual document retrieval, the only prior work in this direction, to our knowledge, is NanoVDR (Liu et al., 2026). A 70 M DistilBERT (Sanh et al., 2019) query encoder is trained to match the embedding space of a 2 B vision-language teacher with a cosine alignment loss. The student matches the teacher within a few NDCG points despite being thirty times smaller, which shows that distillation is a viable route to compact single-vector VDR.

Neither line produces a compact single-vector retriever end-to-end. The from-scratch multi-vector line yields a small encoder but pays an order of magnitude in index size and two orders of magnitude in scoring latency. The distillation line yields a small query encoder but leaves the document encoder at the size of the teacher. NanoVDR keeps the 2 B teacher in the document path, so indexing a million-document corpus still costs tens of GPU-hours and every deployment must host the teacher in memory.

We close this gap with DistilVDR, a 527 M end-to-end single-vector visual document retriever. Our contribution is a system: (i) a bilateral cosine alignment distillation setup in which each student regresses onto cached embeddings of a single 8 B vision-language teacher (Li et al., 2026a); (ii) an asymmetric encoder-only student matched to VDR's input asymmetry, pairing a heavy visual-and-text document tower with a lightweight 70 M text-only query tower; and (iii) a unified reproduction of twelve released retrievers from 250 M to 8.8 B under one evaluation and one profiling pipeline, reporting retrieval quality together with query latency, indexing throughput, peak VRAM, index footprint, and scoring latency. Composed under one teacher, these yield a retriever that dominates the from-scratch small multi-vector route at the 500 M scale on quality and on every deployment axis at once. The two deployment variants we release differ only in the document encoder's visual-tile budget. DistilVDR-HiRes reaches 61.74 average NDCG@5 on ViDoRe v1+v2+v3 (Faysse et al., 2025; Macé et al., 2025; Loison et al., 2026), 86.9 % of the 8 B teacher, and leads every reproduced sub-1 B baseline on the high-resolution-sensitive v3 benchmark. At a 3× smaller visual-token budget, DistilVDR-Fast still exceeds every reproduced sub-1 B baseline, at 59.98 average NDCG@5. Both variants store one million documents in a 15.6 times smaller index than the strongest sub-1 B multi-vector baseline, and DistilVDR-Fast indexes 5.2 times faster than that baseline and 18 times faster than the teacher.

2 Related Work

Section 1 covers the directly competing visual document retrievers; here we situate our work against three adjacent areas.

Distillation in dense text retrieval. Hard-negative mining (Xiong et al., 2020; Qu et al., 2021), retrieval-oriented pretraining (Xiao et al., 2022), and cross-encoder-to-bi-encoder distillation (Hofstätter et al., 2021; Chen et al., 2024a) are common building blocks of modern dense retrievers (Wang et al., 2024a,b; Sturua et al., 2024). The asymmetric pattern in which a heavy reranker teaches a lightweight bi-encoder is the closest text-side analogue to our setup, and differs from it in two ways. Our teacher produces a single image embedding rather than a query-document relevance score, so the distillation target is geometric rather than a scalar; and our asymmetry is across modalities rather than across model classes, with a text-only query side and an image document side.

Universal multimodal embedders. A second line fine-tunes a large vision-language model into a universal embedder over arbitrary modality pairs (Jiang et al., 2024; Wei et al., 2024). We share

the idea of converting a generative vision-language model into an embedding model, but trade their breadth at multi-billion scale for depth in one subdomain at sub-1 B scale. The fixed input shape of VDR, text queries against image documents, lets both encoders stay small.

OCR-free document understanding. A third line feeds document images directly to the model. Donut (Kim et al., 2022), Pix2Struct (Lee et al., 2023) and UDOP (Tang et al., 2023) decode text outputs; LayoutLM and LayoutLMv3 (Xu et al., 2020; Huang et al., 2022) are bidirectional encoders with explicit layout tokens. They share our premise that direct image input avoids OCR error compounding, but target generation or extraction and produce representations not designed for nearest-neighbour retrieval. We produce one dense vector per page that indexes with standard dense-retrieval infrastructure.

3 Method

3.1 Problem setup

Given a text query q and a corpus of document images $\{d_1, \ldots, d_N\}$, the system ranks the corpus by relevance to q. We work in the single-vector dense retrieval setting throughout. A query encoder f_q maps the query to a vector $\mathbf{q} \in \mathbb{R}^k$ and a document encoder f_d maps each document image to a vector $\mathbf{d}_i \in \mathbb{R}^k$ in the same space. Both vectors are L2-normalised. The retrieval score is the dot product $s(q, d_i) = \mathbf{q}^\top \mathbf{d}_i$. Document vectors are computed once at indexing time and queried with any standard dense-retrieval engine.

3.2 Dual-student distillation

The architecture is a two-student instance of the asymmetric distillation paradigm. A frozen 8 B vision-language teacher (Li et al., 2026a) produces target embeddings of dimension $k = 4096$ for both modalities. Two students learn to reproduce these targets independently. The query student f_q takes a text query and the document student f_d takes a document image. Both students project to the teacher's output space and are L2-normalised. Retrieval at deployment uses the students only; the teacher is discarded after training. Figure 1 illustrates the architecture and the two distillation objectives.

3.3 Document encoder

The document encoder f_d has 457 M parameters (304 M InternViT visual encoder + 149 M ModernBERT-base text backbone + 4 M visual-to-text projection and final 768→4096 output head) and proceeds in four stages.

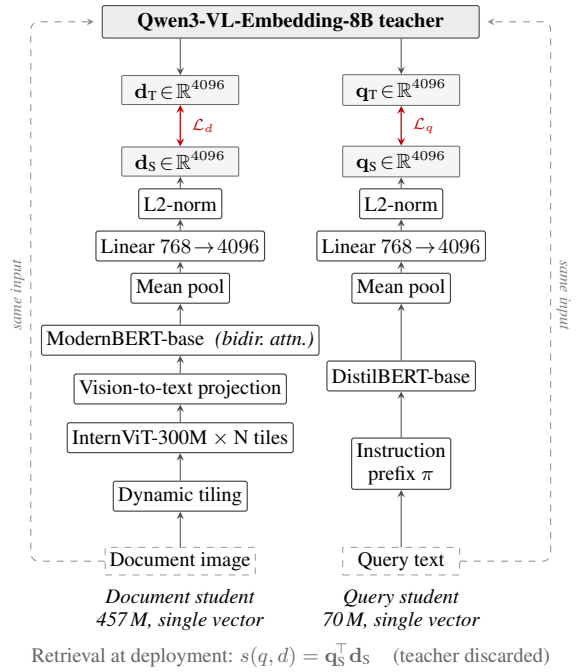


Figure 1: Dual-student distillation architecture. The teacher encodes the image (left) and the query (right); each student is trained independently to match its teacher target.

First, DistilVDR enforces a fixed per-document visual-token budget. We cap the per-page tile count at $T_{\max}$, a release-time design choice that distinguishes the Fast and HiRes variants (Section 4.1), and append a single thumbnail at the visual encoder's native resolution for global context. Given $T_{\max}$ and the page's aspect ratio, we choose an aspect-ratio-matched grid layout at the same native tile resolution, following the InternVL-V2 partition rule (Chen et al., 2025); Algorithm 1 in Appendix E formalises the procedure. The cap resolves the two failure modes of document encoding at once: a single low-resolution view loses small text, table cells, and figure annotations, while a single high-resolution view exceeds the text backbone's context window.

Second, every tile is encoded by an InternViT-300M-448 visual encoder (Chen et al., 2024b) into 1 024 patch tokens of dimension 1 024. We select InternViT-300M for two reasons: its 448-pixel native tile resolution matches our dynamic-tiling rule without any patch resampling, and its document-rich vision-language pre-training transfers well to page-level document imagery. Patch tokens from

all tiles are concatenated into one sequence of length up to 7 168.

Third, the visual sequence is mapped into the embedding space of a ModernBERT-base text backbone (Warner et al., 2025) by a learned linear projection. ModernBERT then re-encodes the projected visual tokens with bidirectional attention, with no text tokens in the input. We use the text backbone here as a contextual encoder over visual tokens. This usage follows ModernVBERT (Teiletche et al., 2025), which shows that an encoder with bidirectional attention outperforms a causal decoder on document retrieval. We select ModernBERT-base because its 8 192-token context window absorbs the worst-case 7 168-token sequence from seven tiles without truncation, and because its rotary positional embeddings and alternating local/global attention yield strong long-context representations at the 150 M parameter scale.

Fourth, the contextualised tokens are mean pooled, projected from 768 to 4096 dimensions by a final linear layer, and L2-normalised.

3.4 Query encoder

The query encoder f_q has 70 M parameters and proceeds in three stages. First, the query text is prefixed by the same instruction string π that the teacher was trained with, so that the student input matches the teacher input at distillation time. Second, the prefixed sequence is encoded by a DistilBERT-base text encoder (Sanh et al., 2019) and mean pooled over the contextual token representations. Third, the pooled vector is passed through a linear projection from 768 to 4096 dimensions and L2-normalised.

3.5 Distillation objective

Let T denote the frozen teacher. For a document image d, the teacher produces a target vector $T(d) \in \mathbb{R}^{4096}$. For a text query q with instruction prefix π, it produces a target $T(\pi \circ q) \in \mathbb{R}^{4096}$. Both targets are L2-normalised by construction. Each student is trained independently against these cached targets under a cosine alignment loss,

$$\mathcal{L}_d = 1 - \langle f_d(d),\ T(d) \rangle, \tag{1}$$

$$\mathcal{L}_q = 1 - \langle f_q(\pi \circ q),\ T(\pi \circ q) \rangle, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product between L2-normalised vectors. The two students never share a forward pass during training, so the doc-side and query-side distillations are fully decoupled.

No contrastive term, hard negative, or relevance label enters the student objective. The supervision is not label-free in an absolute sense, since the teacher's embedding space was itself trained with relevance supervision. The objective removes relevance signals and negative sampling *at student training time*, which is what keeps the two distillations decoupled and independently parallelisable. Implementation details, datasets, and hyperparameters are deferred to Section 4.

4 Experiments

4.1 Experimental setup

Benchmarks and metric. We evaluate on the full 22-dataset ViDoRe suite: v1 (Faysse et al., 2025) (10 datasets, English), v2 (Macé et al., 2025) (4, multilingual) and v3 (Loison et al., 2026) (8, professional domains), each dataset listed in Appendix A. We report NDCG@5, computed with `pytrec_eval` and averaged within each benchmark.

Training data and recipe. The document mixture is 1.20 M unique images from public, permissively licensed sources, assembled in three parts: a 711 K base mixture, a 454 K multi-domain supplement, and a 31.7 K finance supplement, spanning scientific publications, multilingual web PDFs, industrial and regulatory reports, and financial filings. All three are deduplicated by perceptual hash against one another and against all three ViDoRe evaluation corpora (Appendix B). The query mixture is the 1.49 M-query NanoVDR training set (Liu et al., 2026). Both students are trained with AdamW under a one-cycle schedule against cached 4096-d teacher targets. Appendices B and D give the per-source breakdown and the full hyperparameters. The document encoder is trained in two variants differing only in the tile cap of Section 3.3: DistilVDR-HiRes (six 448×448 tiles plus thumbnail; 7 168-token worst case) and DistilVDR-Fast (two tiles plus thumbnail; 3 072-token worst case). Data, optimizer, and epoch budget are identical across the two document variants; the query encoder uses its own schedule (Table 8).

Teacher precomputation cost. The teacher runs exactly once, before any student training. Caching targets for the 1.20 M images and 1.49 M queries cost 99.5 H200-GPU-hours across three sharded batches, one per part of the mixture: 40.5 for the 711 K base mixture together with its queries, their

translations, and the validation split; 58.1 for the multi-domain supplement, as encoded before deduplication reduced it to 454 K images; and 0.9 for the finance supplement, at a measured 11.3 images/sec. Every result in this paper, including all retrained ablations, reuses these cached targets, and this is the only point at which the 8 B model is executed.

Baselines and protocol. We reproduce twelve publicly released retrievers spanning single-vector and multi-vector designs from 250 M to 8.8 B parameters, plus the 8 B teacher as a single-vector oracle; Appendix C names each one. Every system is loaded from its official release and run inside one evaluation driver, so input formatting, deduplication, and metric computation are identical across systems. Multi-vector baselines use their authors' recommended late-interaction routine; single-vector baselines use brute-force dot product over the full corpus.

4.2 Retrieval quality

Table 1 reports NDCG@5 on the full ViDoRe suite for every reproduced baseline alongside the two DistilVDR variants. Against the strongest reproduced sub-1 B baseline, colSmol-500M at 53.01 average, HiRes leads by 8.73 points and Fast by 6.97. Both variants exceed every other sub-1 B baseline by more.

The two variants separate on the hardest benchmark. On v3, built on long professional reports with dense small text and complex layouts, HiRes scores 47.07 against Fast's 43.66, while on v1 and v2 they stay within 1.5 points of each other. HiRes leads the next-best sub-1 B retriever on v3 by 13.55 points (47.07 vs. 33.52); Fast trades that v3 quality for a 3× smaller visual-token budget, which Section 4.3 turns into deployment cost.

Against larger references the variants are competitive at 2–3 B and behind at 4–8 B. HiRes exceeds DSE-Qwen2 (2.2 B) by 1.03 and ColPali v1.3 (2.9 B) by 1.42 average points despite being four to six times smaller, and Fast trails both by under one point. Both trail Qwen3-VL-Embedding-2B, and the 4–8 B multi-vector models remain 6.95 to 9.80 points ahead of HiRes, with the gap concentrated on v2 and v3. The 8 B teacher exceeds HiRes by 9.31 points; HiRes and Fast retain 86.9 % and 84.4 % of its average NDCG@5.

Gap decomposition. To localise the residual gap between DistilVDR-HiRes and the 8 B teacher we evaluate the four combinations of student and teacher on each side. Table 2 reports the resulting NDCG@5. Replacing the document side with the student costs 6.03 NDCG@5 on average (T×T vs. T×S). Replacing the query side with the student costs 4.69 points on average (T×T vs. S×T). The two side losses do not simply add to the full gap of 9.31 points; the residual is the interaction between the two student errors.

Where the residual gap lives. One 4096-d vector per page already expresses retrieval quality well beyond what our student reaches. The teacher is itself a single-vector retriever, and at 71.05 average NDCG@5 it is level with the 4.4 B multi-vector Tomoro-ColQwen3-4B (71.01) and above the 7 B ColNomic-7B (68.69). DistilVDR is therefore limited by capacity within the single-vector format rather than by the format itself. The document side accounts for 6.03 of the 9.31-point gap (Table 2), and shrinking the output vector from 4096 to 768 dimensions alone costs 4.77 NDCG@5 on v3 (Table 4, block (c)).

4.3 Efficiency analysis

Table 3 reports throughput and resource usage for the twelve baselines and the distillation teacher of Table 1. Every encoder runs at fixed batch size $B = 8$ in bf16 under its best-supported flash-attention backend (Dao et al., 2022; Dao, 2024) on a single H200 GPU; per-baseline implementation details are deferred to Appendix G. The four 250–500 M baselines (BiModernVBERT, colSmol-256M, colSmol-500M, ColModernVBERT) all use the uncapped Idefics3 image splitter (Laurençon et al., 2024), which emits more than ten sub-images per document page; DistilVDR caps its budget at seven (Section 3.3), so its image tower runs over a shorter visual sequence at comparable parameter scales.

Encoding and indexing. DistilVDR-Fast attains the highest document throughput in the table at 99.04 docs/sec and 2.10 GB peak VRAM. The fastest reproduced multi-vector baseline, SauerkrautLM-ColLFM2, reaches 19.02 docs/sec and the remainder fall between 2.58 and 17.24, so Fast is between 5.2 and 38 times faster depending on the baseline; it is also several times faster than every other sub-1 B single-vector baseline and roughly 18× faster than the 8 B teacher. HiRes returns part of that throughput for its larger tile budget, still leading every reproduced multi-vector baseline and running 7× faster than the teacher.

Table 1: Retrieval quality (NDCG@5) on ViDoRe v1, v2 and v3. "Type" is single vector or multi-vector with MaxSim.

Model	Params	Type	v1	v2	v3	Avg
Sub-1 B						
SigLIP2-L	880 M	single	43.58	20.17	14.04	25.93
BiModernVBERT	250 M	single	37.40	10.88	5.52	17.93
colSmol-256M	256 M	multi	79.72	34.63	25.23	46.53
colSmol-500M	478 M	multi	82.42	43.09	33.52	53.01
ColModernVBERT	250 M	multi	76.76	33.18	17.45	42.46
SauerkrautLM-ColLFM2	451 M	multi	78.24	45.09	33.19	52.17
DistilVDR-Fast (ours)	**527 M**	**single**	81.34	54.95	43.66	59.98
DistilVDR-HiRes (ours)	**527 M**	**single**	**82.81**	**55.34**	**47.07**	**61.74**
Mid- to large-scale references						
DSE-Qwen2	2.2 B	single	85.14	55.70	41.28	60.71
Qwen3-VL-Embedding-2B	2.1 B	single	84.30	65.25	49.98	66.51
ColPali v1.3	2.9 B	multi	84.21	54.72	42.04	60.32
Tomoro-ColQwen3-4B	4.4 B	multi	90.22	65.25	57.57	71.01
ColNomic-7B	7.0 B	multi	89.76	60.44	55.87	68.69
Tomoro-ColQwen3-8B	8.8 B	multi	90.61	65.00	59.00	71.54
Qwen3-VL-Embedding-8B (teacher)	8.1 B	single	87.31	69.76	56.07	71.05

Table 2: Gap decomposition (NDCG@5) for DistilVDR-HiRes. T = teacher, S = student. Bottom row is the full system.

Query × Doc	v1	v2	v3	Avg
T × T (oracle)	87.31	69.76	56.07	71.05
T × S	83.72	60.92	50.43	65.02
S × T	84.68	64.30	50.09	66.36
S × S (ours)	82.81	55.34	47.07	61.74

SigLIP2-L is the only sub-1 B baseline with lower peak VRAM than Fast. Both variants encode queries in 3.4 ms through the text-only DistilBERT path, faster than every other profiled system.

Index storage and scoring. The multi-vector index footprint is the per-document token count times the embedding dimension times the storage precision. Sub-1 B multi-vector baselines store roughly 1 000 128-d `float16` vectors per page (256 GB per million documents) and the 4–8 B Tomoro baselines 1 280 320-d tokens (819 GB). DistilVDR stores one 4096-d `float32` vector, 16.4 GB, and scores 10 000 documents in 9.6 ms against the 1.1–3.2 s the multi-vector baselines need for the same MaxSim. An end-to-end multi-vector deployment at sub-1 B scale therefore costs roughly 16× in storage and 100× in scoring latency relative to DistilVDR.

5 Ablations

Table 4 collects four ablations: the visual tile budget, the training-data scale, and the projection-head output dimension on the document side, plus the query-encoder backbone. Every variant is retrained from scratch under the recipe of Appendix D, changing one factor at a time. Blocks (a) and (b) are scored end to end, student query against student document. Blocks (c) and (d) are scored under single-side isolation, pairing the retrained tower with the teacher's other tower so that the factor under study is not confounded by the other tower's error; their absolute values are therefore higher than those of the end-to-end system of Section 4.2 and are comparable only within a block. A fifth ablation (Section 5.1) asks whether contrastive supervision on top of cosine alignment helps.

Visual tile budget. Block (a) sweeps the tile cap of Section 3.3; the Fast and HiRes rows are the two deployed variants and the 0-tile row is a single-448×448-view control. Removing tiling costs 5.12 average NDCG@5 relative to HiRes, concentrated on v3. Going from Fast to HiRes adds 1.76 average points and 3.41 on v3 at 3× the visual tokens, whose deployment cost Table 3 reports.

Training-data scale. Block (b) retrains the document encoder on uniform subsamples of the 1.20 M-image mixture. Quality grows monotonically with data and saturates above 75 %; the full mixture is worth about 5 average points over a quarter-scale variant.

Output dimension. Because the teacher is trained with Matryoshka representation learning (Li et al., 2026a), any leading prefix of its 4096-d output is itself a valid embedding, so truncating the

Table 3: Deployment efficiency on a single H200 GPU at fixed batch size $B = 8$ in bf16. Each encoder runs under its best-supported flash-attention backend (Dao et al., 2022; Dao, 2024); per-baseline details are in Appendix G. Index size is per million documents. Avg NDCG@5 from Table 1 is repeated for reference. In each efficiency column we **bold** the column-best and underline the second-best.

Model	Params	Type	Query (ms)	Doc thpt (docs/s)	Peak VRAM (GB)	Index / 1 M	Score 10 K (ms)	Avg NDCG@5
Sub-1 B								
SigLIP2-L	880 M	single	113.8	28.22	**1.98**	4.1 GB	1.3	25.93
BiModernVBERT	250 M	single	7.4	2.49	7.47	**3.1 GB**	**0.9**	17.93
colSmol-256M	256 M	multi	23.6	2.58	4.49	256 GB	1 259	46.53
colSmol-500M	478 M	multi	22.6	2.82	4.97	256 GB	1 161	53.01
ColModernVBERT	250 M	multi	61.8	3.06	4.26	256 GB	1 238	42.46
SauerkrautLM-ColLFM2	451 M	multi	8.5	19.02	2.56	256 GB	1 330	52.17
DistilVDR-Fast (ours)	527 M	single	**3.4**	**99.04**	2.10	16.4 GB	9.6	59.98
DistilVDR-HiRes (ours)	527 M	single	**3.4**	36.82	3.07	16.4 GB	9.6	61.74
Mid- to large-scale references								
DSE-Qwen2	2.2 B	single	167.4	17.17	6.31	6.1 GB	2.2	60.71
Qwen3-VL-Embedding-2B	2.1 B	single	14.5	8.53	7.03	8.2 GB	3.4	66.51
ColPali v1.3	2.9 B	multi	266.7	17.24	7.81	264 GB	1 158	60.32
Tomoro-ColQwen3-4B	4.4 B	multi	266.4	11.91	12.93	819 GB	3 187	71.01
ColNomic-7B	7.0 B	multi	542.6	9.96	17.88	256 GB	1 206	68.69
Tomoro-ColQwen3-8B	8.8 B	multi	499.3	9.20	21.76	819 GB	3 176	**71.54**
Qwen3-VL-Embedding-8B (teacher)	8.1 B	single	19.8	5.40	19.28	16.4 GB	9.4	71.05

target to 768 dimensions is a principled compression rather than a random projection. Retraining the document encoder against that truncated target (block (c)) shrinks the index 5.3× (16.4 GB → 3.07 GB per million documents) but costs 3.19 average NDCG@5 and 4.77 on v3. We keep 4096-d as the default and treat 768-d as the option under hard storage constraints.

Query backbone. Block (d) retrains the query encoder on BERT-base (113 M) and ModernBERT-base (152 M) under the same protocol. ModernBERT-base is the strongest at 67.40 average, 1.04 points above DistilBERT-base, but costs 2.2× the parameters and 5.1× the query latency. Encoding one query at $B = 1$ including tokenisation takes 2.07, 4.78 and 10.61 ms for DistilBERT-base, BERT-base and ModernBERT-base respectively. BERT-base trails both on quality at 61.63. We adopt DistilBERT-base because 1.04 points is a small price for those savings. Index size is unaffected by this choice.

Table 4: Document- and query-encoder ablations, NDCG@5. Blocks (a) and (b) are end-to-end student×student; block (c) is doc-side isolation (student documents against teacher queries); block (d) is query-side isolation (student queries against teacher documents), whose teacher×teacher ceiling is 71.05 average. Absolute values are therefore comparable within a block, not across blocks. † marks the deployed default.

Variant	v1	v2	v3	Avg
(a) Max tiles (end-to-end)				
0 (no tiling)	77.57	51.29	41.00	56.62
2 (Fast)	81.34	54.95	43.66	59.98
6 (HiRes, default)	82.81	55.34	47.07	61.74
(b) Training-data scale (end-to-end)				
25 % (300 K)	78.35	49.69	40.75	56.26
50 % (600 K)	80.97	54.44	44.58	60.00
75 % (900 K)	82.40	56.58	46.05	61.68
100 % (1.20 M, default)	82.81	55.34	47.07	61.74
(c) Doc output dim (doc-side isolation)				
768-d (3.07 GB/1 M)	81.40	58.44	45.66	61.83
4096-d (16.4 GB/1 M)†	83.72	60.92	50.43	65.02
(d) Query backbone (query-side isolation)				
DistilBERT-base (70 M)†	84.68	64.30	50.09	66.36
BERT-base (113 M)	81.81	58.52	44.55	61.63
ModernBERT-base (152 M)	85.36	65.40	51.43	67.40

5.1 Contrastive supervision

Our recipe trains the two students independently, with no cross-modal interaction in the loss. This ablation adds a joint refinement step that couples them through a contrastive term over paired (q, d) mini-batches. Starting from a cosine-distilled checkpoint of the same 527 M architecture at a 768-d projection head (the 768-d counterpart of the 4096-d encoder, with the same encoders, data, tiling, and teacher), we train both students jointly for one further epoch on the same mixture under

$$\mathcal{L} = \alpha\,\mathcal{L}_d + \beta\,\mathcal{L}_q + \gamma\,\mathcal{L}_{\text{rank}}, \tag{3}$$

where $\mathcal{L}_d$ and $\mathcal{L}_q$ are the cosine alignment losses of Eqs. 1–2 and $\alpha = \beta = 1.0$. For a mini-batch

$\mathcal{B}$ of paired query-document examples, let $s_{qd} = \langle f_q(q),\, f_d(d)\rangle$ denote the student similarity and $\tilde{s}_{qd} = \langle T(q),\, T(d)\rangle$ the teacher similarity between q and d. We consider two choices for $\mathcal{L}_{\text{rank}}$:

$$\mathcal{L}_{\text{InfoNCE}} = -\sum_{(q,d)\in\mathcal{B}} \log \frac{\exp(s_{qd}/\tau)}{\sum_{d'\in\mathcal{B}} \exp(s_{qd'}/\tau)}, \tag{4}$$

$$\mathcal{L}_{\text{KL}} = \sum_{q\in\mathcal{B}} \text{KL}\big(\,\sigma_\tau(\tilde{s}_q)\,\|\,\sigma_\tau(s_q)\,\big), \tag{5}$$

with temperature $\tau = 0.02$ and σ_τ the row-wise softmax of similarities to all documents in $\mathcal{B}$. The refinement uses AdamW, effective batch 256, peak learning rate 2×10^{-5}, warm-up fraction 0.05, and the dynamic-tiling rule of DistilVDR-HiRes on two H200 GPUs. Absolute NDCG@5 in Table 5 is below Section 4.2 because the projection head outputs 768 dimensions instead of 4096; the relative effect of γ is unaffected by this offset.

Within this 1-epoch joint refinement, adding contrastive supervision yields no NDCG@5 improvement at any γ we tried. InfoNCE shows a mild downward drift with increasing γ, from -0.12 at $\gamma = 0.5$ to -0.50 at $\gamma = 2.0$. KL is flat across all weights (± 0.14). The asymmetry is consistent with the two losses' geometric content: InfoNCE pushes the student toward a uniform separator over in-batch negatives, orthogonal to the teacher-calibrated geometry already enforced by $\mathcal{L}_d + \mathcal{L}_q$; KL anchors to the teacher distribution and is approximately redundant with cosine alignment. The refinement budget is roughly one third of the document encoder's training budget (Table 8), enough that any positive contribution from the contrastive term should register a positive drift; what we observe is flat or mildly negative. The direction is consistent with the from-scratch loss ablation of NanoVDR (Liu et al., 2026), which reports pointwise cosine alignment strictly outperforming both KL-ranking and hard-label InfoNCE distillation under the same teacher family.

Every configuration here starts from the same distilled checkpoint, so the experiment covers the refinement stage only and does not compare distillation and contrastive learning as alternative ways of training the same architecture. Within that scope it supports the pure cosine objective of Section 3.5 as a strong default.

Table 5: Contrastive-supervision ablation under Eq. 3. All rows are one-epoch joint refinements from the cosine-distilled 768-d sibling checkpoint; absolute NDCG@5 is below Section 4.2 because of the 768-d output.

Refinement objective	v1	v2	v3	Avg
$\gamma = 0$ (cosine only)	79.90	53.23	42.22	58.45
$+0.5\,\mathcal{L}_{\text{InfoNCE}}$	80.47	52.08	42.44	58.33
$+1.0\,\mathcal{L}_{\text{InfoNCE}}$	80.48	51.80	42.41	58.23
$+2.0\,\mathcal{L}_{\text{InfoNCE}}$	80.25	51.31	42.30	57.95
$+0.5\,\mathcal{L}_{\text{KL}}$	80.32	52.31	42.30	58.31
$+1.0\,\mathcal{L}_{\text{KL}}$	80.20	52.55	42.34	58.36
$+2.0\,\mathcal{L}_{\text{KL}}$	80.31	52.54	42.44	58.43

6 Conclusion

We presented DistilVDR, a 527 M end-to-end visual document retriever obtained by independent cosine distillation of both encoders from a single 8 B vision-language teacher under an asymmetric encoder-only design. Its two deployment variants share encoders and training and differ only in the document-side visual-tile budget: HiRes retains 86.9 % of the teacher and leads every reproduced sub-1 B baseline on ViDoRe v3, Fast retains 84.4 % at a 3× smaller visual-token budget, and both index faster than every reproduced multi-vector baseline at a 15.6 times smaller footprint. Measured end to end under one evaluation and one profiling pipeline, distilling both towers of a strong single-vector vision-language teacher reaches a better quality-and-cost operating point at the 500 M scale than any released small multi-vector retriever we could reproduce.

Limitations

From an experimental perspective. Every comparison we report is between deployable systems rather than between training recipes. Every baseline in Table 1 is an official public release evaluated as released; none is retrained under a budget matched to ours, and our teacher, our 1.20 M-image mixture, and our architecture all differ from theirs, so the margin cannot be attributed to any single one of them. The control that would isolate distillation itself, the same 527 M architecture trained contrastively from scratch on the same mixture with hard-negative mining and a comparable compute budget, is the principal experiment we did not run; the refinement experiment of Section 5.1 starts from an already-distilled checkpoint and does not substitute for it. We likewise use exactly one

teacher and do not sweep teacher scale or family, so we cannot say how much quality a weaker teacher would cost or how much a stronger one would add; because targets are cached once and student training never runs the teacher, that sweep is a cheap follow-up. Both variants also stay 7–10 points behind the strongest 4–8 B multi-vector references of Table 1, with the largest gap on ViDoRe v3 (47.07 and 43.66 against 57.57–59.00), and because our experiments vary the scoring mechanism only together with model scale, they cannot separate late interaction from capacity at the top of that table. Finally, every retrieval number comes from the 22 ViDoRe datasets. ViDoRe spans three difficulty levels, six languages, and eight professional domains, but not enterprise corpora with in-house layouts, scanned or OCR-degraded pages, or production query distributions; our deployment costs are measured directly and transfer, our quality numbers may not.

From a methodological perspective. The student is bounded by the space it is trained into. Both towers only reproduce a frozen teacher's embeddings, so any systematic weakness of Qwen3-VL-Embedding-8B propagates to DistilVDR, and nothing in the objective lets a student exceed its teacher on the side it replaces; Table 2 is consistent with this, as no student-side substitution improves on the teacher×teacher oracle. Distilling from a richer signal, such as a cross-encoder relevance score or a multi-vector teacher, is what would raise that ceiling. Three further choices are deliberate simplifications. The two students are trained independently against cached targets, which parallelises cleanly but leaves unused the cross-modal coupling between query and document manifolds that the teacher implicitly carries; a joint scheme with explicit query-document interaction during distillation is the natural extension. The tiling rule adapts the grid to page aspect ratio but not to page content, and $T_{\max}$ is fixed at release time, so a dense small-text page and a single-figure page receive the same visual-token budget; our two variants sample two points on that trade-off rather than resolve it, and a content-adaptive budget predicted from a cheap page-complexity estimate would spend tokens where they are needed. And the query tower is text-only, trained on English plus five Latin-script European languages obtained through a MarianMT translation pipeline, so image-conditioned queries and non-Latin scripts such as Chinese, Japanese, and Arabic are out of scope. One deployment-side choice is also unexplored. We store one uncompressed 4096-d `float32` vector per document. Table 4 measures a single compression axis, the 768-d Matryoshka prefix (3.07 GB per million documents, −3.19 NDCG@5 average); scalar quantisation, product quantisation, and binary hashing are orthogonal to our contribution and untested here, so the reported footprint should be read as an upper bound.

Acknowledgments

This work was carried out in AiWo: Human-Centric, AI-Enabled Collaborative Fieldwork Operations, a Business Finland co-innovation programme under grant agreement No. 69/31/2025. The computations were performed using computer resources provided by the Aalto University School of Science "Science-IT" project.

We used Claude Code as a coding and writing assistant: implementing training and evaluation scripts, preparing and analysing data, and assisting with the language of the manuscript. The research questions, the experimental design, the interpretation of results, and all claims in this paper are the authors' own; every reported number was produced by running the released code and verified by the authors.

References

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yiming Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, and 23 others. 2025. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *Preprint*, arXiv:2412.05271.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024b. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pages 35549–35562.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations*, volume 2025, pages 61424–61449.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 113–122, New York, NY, USA. Association for Computing Machinery.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA. Association for Computing Machinery.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. 2024. VLM2Vec: Training vision-language models for massive multimodal embedding tasks. *arXiv preprint arXiv:2410.05160*.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. OCR-free document understanding transformer. In *Computer Vision – ECCV 2022*, pages 498–517, Cham. Springer Nature Switzerland.

Juyeon Kim, Geon Lee, Dongwon Choi, Taeuk Kim, and Kijung Shin. 2025. Hybrid-vector retrieval for visually rich documents: Combining single-vector efficiency and multi-vector accuracy. *Preprint*, arXiv:2510.22215.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. 2024. Building and better understanding vision-language models: insights and future directions. *arXiv preprint arXiv:2408.12637*.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18893–18912. PMLR.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. 2026a. Qwen3-VL-Embedding and Qwen3-VL-Reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*.

Weien Li, Rui Song, Zeyu Li, Haochen Liu, Gonghao Zhang, Difan Jiao, Zhenwei Tang, Bowei He, Haolun Wu, Xue Liu, and Ye Yuan. 2026b. MINER: Mining multimodal internal representation for efficient retrieval. *Preprint*, arXiv:2605.06460.

Zhuchenyang Liu, Yao Zhang, and Yu Xiao. 2026. NanoVDR: Distilling a 2B vision-language retriever into a 70M text-only encoder for visual document retrieval. *arXiv preprint arXiv:2603.12824*.

António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. 2026. ViDoRe V3: A comprehensive evaluation of retrieval augmented generation in complex real-world scenarios. *arXiv preprint arXiv:2601.08620*.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. 2024. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6492–6505.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2025. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. *Preprint*, arXiv:2506.04997.

Quentin Macé, António Loison, and Manuel Faysse. 2025. ViDoRe benchmark V2: Raising the bar for visual retrieval. *arXiv preprint arXiv:2505.17166*.

Nomic AI. 2025. ColNomic embed multimodal 7B: Open multimodal embeddings for documents, charts, and PDFs. `https://huggingface.co/nomic-ai/colnomic-embed-multimodal-7b`. HuggingFace model card + Nomic blog `https://www.nomic.ai/news/nomic-embed-multimodal`.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, pages 5835–5847, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task LoRA. *Preprint*, arXiv:2409.10173.

Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. 2023. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19254–19264.

Paul Teiletche, Quentin Macé, Max Conti, Antonio Loison, Gautier Viaud, Pierre Colombo, and Manuel Faysse. 2025. ModernVBERT: Towards smaller visual document retrievers. *arXiv preprint arXiv:2510.01149*.

TomoroAI. 2025. Tomoro-ColQwen3-embed-8B: Multi-vector visual retriever on Qwen3-VL-8B-Instruct. `https://huggingface.co/TomoroAI/tomoro-colqwen3-embed-8b`. HuggingFace model card; merges Qwen3-VL-8B-Instruct with Qwen3-Embedding-8B.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. 2025. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *Preprint*, arXiv:2502.14786.

VAGO Solutions. 2025. SauerkrautLM-ColLFM2-450M-v0.1: Curriculum-trained multi-vector visual retriever on LFM2-VL. `https://huggingface.co/VAGOsolutions/SauerkrautLM-ColLFM2-450M-v0.1`. HuggingFace model card; no dedicated arXiv paper as of May 2026.

ViDoRe Team. 2024. ColSmol-500M / ColSmol-256M: ColPali-style SmolVLM visual document retrievers. `https://huggingface.co/vidore/colSmol-500M`. HuggingFace model card. Technique follows ColPali (Faysse et al., 2025); backbone is SmolVLM/Idefics3.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024a. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Multilingual E5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. 2024. UniIR: Training and benchmarking universal multimodal information retrievers. In *European Conference on Computer Vision*, pages 387–404. Springer.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *Preprint*, arXiv:2007.00808.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1192–1200, New York, NY, USA. Association for Computing Machinery.

Yibo Yan, Mingdong Ou, Yi Cao, Xin Zou, Jiahao Huo, Shuliang Liu, James Kwok, and Xuming Hu. 2026. Sculpting the vector space: Towards efficient

multi-vector visual document retrieval via prune-then-merge framework. *Preprint*, arXiv:2602.19549.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. 2025. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *International Conference on Learning Representations*, volume 2025, pages 21074–21098.

A ViDoRe Benchmark Details

The Visual Document Retrieval (ViDoRe) benchmark comprises three progressively challenging versions (Faysse et al., 2025; Macé et al., 2025; Loison et al., 2026). We evaluate on the full 22-dataset suite. Table 6 lists each dataset together with its document corpus language, query languages, and the source of its queries.

v1 (Faysse et al., 2025). Released with ColPali, v1 contains 10 English- and French-language datasets. Five are sourced from established visual QA benchmarks (DocVQA, ArXivQA, InfoVQA, TatDQA, TabFQuAD) with human-authored queries; five use queries generated by Claude-3 Sonnet over curated document collections. State-of-the-art models now exceed 90 NDCG@5 on v1, indicating saturation.

v2 (Macé et al., 2025). Designed to address v1's saturation, v2 introduces 4 datasets with queries in up to four languages (English, French, Spanish, German). Queries are generated through a blind contextual process: annotators receive only document metadata, which reduces extractive bias and requires cross-document reasoning. The fourth dataset (ESG Reports Human-Labeled) provides expert human annotations over the same ESG corpus.

v3 (Loison et al., 2026). The most comprehensive version, v3 provides 8 public datasets across six languages (the v2 languages plus Italian and Portuguese). Annotation combined LLM-synthesised queries with expert review. Each query carries page-level relevance rankings, bounding-box annotations, and multilingual translations. The eight domains span enterprise scenarios from finance to physics, with document corpora in English or French.

B Training Data Composition and Preprocessing

The document encoder training mixture combines three groups of publicly released datasets. Table 7 gives the full per-source breakdown, including the query-encoder mixture.

Base sources. The first group reproduces the 711 K base mixture released by NanoVDR (Liu et al., 2026): VisRAG-Synthetic and VisRAG-InDomain (Yu et al., 2025), the ColPali training set (Faysse et al., 2025), and VDR-Multilingual (English, French, German, Spanish, Italian). The query encoder uses the same 711 K queries together with the 778 K Helsinki-NLP MarianMT translations released with NanoVDR, for a total of 1.49 M paired queries.

Multi-domain supplement. The second group is the union of fourteen domain-specific document collections released by Racineai. These collections are aggregated under the `racineai/VDR_MEGA_2` release and individually contributed under permissive licences. We deduplicate against the 711 K base by perceptual hash and against the ViDoRe v1/v2/v3 evaluation corpora. The resulting 454 K-image subset is used on the document side only.

Finance supplement. The third group adds 31.7 K finance-domain document images from `sujet-ai/Sujet-Finance-Vision-10k` and `DocReRank/FinHNQue-FinanceHardNegative Queries`, again deduplicated by perceptual hash against base and evaluation corpora.

Preprocessing pipeline. All training images are loaded as PIL RGB and processed by the encoder pipeline of Section 3.3 (dynamic tiling, patch encoding, projection). Deduplication is performed offline with the `imagehash` library at perceptual-hash distance 0, which removes only exact visual duplicates. Teacher embeddings are precomputed once over the deduplicated mixture and cached as `float32` arrays on disk, so the teacher does not run during student training.

C Baselines

At the sub-1 B scale the multi-vector retrievers are colSmol-256M and colSmol-500M (ViDoRe Team, 2024), SauerkrautLM-ColLFM2-450M (VAGO Solutions, 2025), and ColModernVBERT (Teiletche et al., 2025); the single-vector retrievers are SigLIP2-L (Tschannen et al., 2025) and BiModernVBERT (Teiletche et al., 2025). At the 2–3 B scale we include the single-vector DSE-Qwen2 (Ma et al., 2024) and Qwen3-VL-Embedding-2B (Li et al., 2026a), and the multi-vector ColPali

Table 6: All 22 ViDoRe evaluation datasets. "Doc" is the document corpus language. "Query" is the languages in which queries are released. "Source" indicates how the queries were obtained: H = human-authored, L = LLM-generated, L+H = LLM-generated with expert review. The six languages of v3 are English, French, Spanish, German, Italian, and Portuguese.

Dataset	Ver.	Document domain	Doc	Query	Source
DocVQA	v1	Industrial documents	EN	EN	H
ArXivQA	v1	Scientific papers	EN	EN	H
InfoVQA	v1	Infographics	EN	EN	H
TatDQA	v1	Financial tables	EN	EN	H
TabFQuAD	v1	Tables in French PDFs	FR	FR	H
SyntheticDocQA-AI	v1	AI documents	EN	EN	L
SyntheticDocQA-Energy	v1	Energy sector reports	EN	EN	L
SyntheticDocQA-Gov.	v1	Government reports	EN	EN	L
SyntheticDocQA-Hlt.	v1	Healthcare documents	EN	EN	L
ShiftProject	v1	Environmental reports	FR	FR	L
ESG Reports	v2	ESG / sustainability	EN	EN/FR/ES/DE	L+H
Biomedical Lectures	v2	Biomedical slides	EN	EN/FR/ES/DE	L+H
Economics Reports	v2	Economics reports	EN	EN/FR/ES/DE	L+H
ESG Reports (Human)	v2	ESG / sustainability	EN	EN	H
Finance-EN	v3	US annual reports	EN	6 languages	L+H
Finance-FR	v3	French annual reports	FR	6 languages	L+H
Computer Science	v3	CS textbooks	EN	6 languages	L+H
HR	v3	EU HR reports	EN	6 languages	L+H
Energy	v3	French energy reports	FR	6 languages	L+H
Industrial	v3	USAF technical orders	EN	6 languages	L+H
Pharmaceutical	v3	FDA reports	EN	6 languages	L+H
Physics	v3	French physics lectures	FR	6 languages	L+H

v1.3 (Faysse et al., 2025). At the 4–8 B scale we include Tomoro-ColQwen3-4B and Tomoro-ColQwen3-8B (TomoroAI, 2025) and ColNomic-7B (Nomic AI, 2025). Qwen3-VL-Embedding-8B (Li et al., 2026a) is our distillation teacher and is also the single-vector oracle.

D Training Recipe

Table 8 lists the hyperparameters and hardware for both students under the cosine alignment objective of Eqs. 1–2. The two document-encoder variants (Fast and HiRes) share this recipe exactly and differ only in $T_{\max}$. Per-epoch loss curves for all three students are reported as a convergence diagnostic in Appendix F.

E Dynamic Tiling Algorithm

Algorithm 1 formalises the InternVL-V2 dynamic tiling rule (Chen et al., 2025) used by the document encoder of Section 3.3. The rule picks an aspect-ratio-matched grid layout from the candidate set $\mathcal{G} = \{(p, q) : p, q \in \mathbb{Z}_{\geq 1},\ n_{\min} \leq p \cdot q \leq n_{\max}\}$, resizes the image to fit that grid at the encoder's native tile resolution, splits the resized image into non-overlapping tiles, and optionally appends a single thumbnail at the same resolution to preserve global layout context. Tie-breaking among candidates with equal aspect distance prefers the higher-resolution grid when the original image area is large. We use $n_{\min} = 1$, $n_{\max} = 6$, tile size $s = 448$, and enable the thumbnail. The maximum sequence length per document is $(n_{\max} + 1) \cdot s^2/p^2 = 7 \cdot 1024 = 7168$ patch tokens at the InternViT-300M patch size of $p = 14$, which fits inside ModernBERT-base's 8192-token context window without truncation.

Algorithm 1 Dynamic Tiling

Require: image I of size (W, H); $n_{\min}$, $n_{\max}$; tile size s; thumbnail flag t
Ensure: list of tiles, each $s \times s$

$r \leftarrow W/H$ ▷ input aspect ratio
$\mathcal{G} \leftarrow \{(p, q) : p, q \in \mathbb{Z}_{\geq 1},\ n_{\min} \leq p\,q \leq n_{\max}\}$
$(p^*, q^*) \leftarrow \arg\min_{(p,q)\in\mathcal{G}} |r - p/q|$ ▷ ties broken toward higher resolution
$I' \leftarrow \text{Resize}(I,\ (p^*s,\ q^*s))$
$\mathcal{T} \leftarrow$ split I' into p^*q^* non-overlapping $s \times s$ tiles, left-to-right then top-to-bottom
if t and $p^*q^* > 1$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\text{Resize}(I,\ (s, s))\}$ ▷ thumbnail for global context
end if
return $\mathcal{T}$

Table 7: Training data composition. HuggingFace identifiers are hyperlinked; the organisation prefix is omitted in the second and third groups.

Source	Samples
Base mixture (711 K)	
`VisRAG-Ret-Train-Synthetic` (Yu et al., 2025)	234 K
`VisRAG-Ret-Train-In-domain` (Yu et al., 2025)	94 K
`colpali_train_set` (Faysse et al., 2025)	109 K
`vdr-multilingual-train`	275 K
Multi-domain supplement (Racineai, deduplicated)	
`racineai/VDR_*` (14 sub-sources)	454 K
Finance supplement (31.7 K)	
`Sujet-Finance-Vision-10k`	9.8 K
`FinHNQue`	21.9 K
Document encoder total	**1.20 M**
Query encoder mixture	
NanoVDR query training set (Liu et al., 2026)	1.49 M

Table 8: Training recipe under the cosine alignment objective (Eq. 1–2).

Hyperparameter	Doc enc.	Query enc.
Trainable parameters	457 M	70 M
Optimizer	AdamW	AdamW
Peak LR	1×10^{-3}	5×10^{-4}
LR schedule	one-cycle, 3% warmup	
Effective batch	256	512
Epochs	3	15
Hardware	2× H200	2× H200

F Training Loss Curves

Figure 2 reports per-epoch train and validation cosine-alignment loss for the two document-encoder variants (DistilVDR-HiRes and DistilVDR-Fast) and for the shared query encoder under the recipe of Table 8. All curves decrease smoothly to a low plateau with a small train/validation gap, indicating that every student reaches a steady-state fit to its teacher target within the allotted epoch budget.

G Efficiency Benchmark Protocol

Hardware and software stack. A single NVIDIA H200 GPU (141 GB HBM3e) on an Intel Xeon Platinum 8480+ node with 32 CPU cores and 128 GB RAM. Software: PyTorch 2.5, CUDA 12.6, transformers ≥ 4.46. All encoders run in bfloat16.

Attention backend per encoder. Every encoder runs under a flash-attention-family

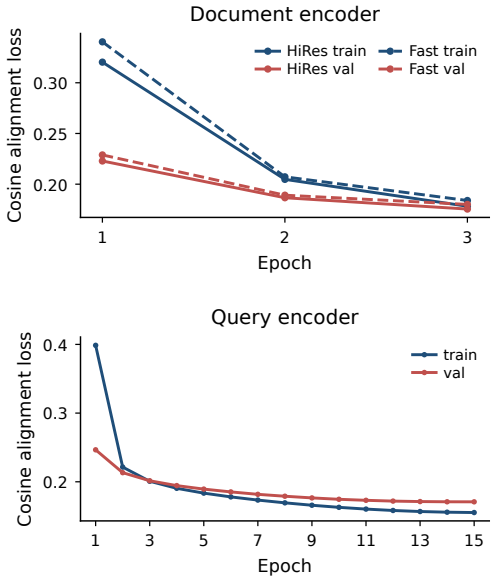


Figure 2: Per-epoch train (blue) and validation (red) loss for the document students DistilVDR-HiRes (solid) and DistilVDR-Fast (dashed) on top, and the shared query student on bottom.

kernel (Dao et al., 2022; Dao, 2024). For all baselines and DistilVDR's ModernBERT text backbone this kernel is reached through PyTorch's sdpa backend, which dispatches to FlashAttention-2 in transformers ≥ 4.46; calling `attn_implementation="flash_attention_2"` explicitly changes throughput by under ± 1 docs/sec. DistilVDR's InternViT visual encoder does not register an sdpa implementation with transformers, but its remote modeling code ships its own native flash-attention path, enabled by default via the `use_flash_attn=True` config flag; this is the configuration reported in Table 3.

Per-baseline deviations. Two baselines deviate from the default bf16+sdpa setup. BiModernVBERT is loaded in float32 because the canonical evaluation pipeline observes that bf16 mean-pooling over the 1 000+ visual tokens emitted by its ModernBERT backbone incurs a several-NDCG drop; we follow the released configuration and use float32 on both the document and query sides. DSE-Qwen2 follows its released chat-template inference path, wrapping each document image in a `user` message with a fixed 680×680 resized resolution and reading the last hidden state at the trailing `<|endoftext|>` token as the document embedding; we use the model's released `prepare_inputs_for_generation` entry point with `use_cache=False` to obtain that

hidden state in a single forward pass.

Conservative attention-backend reference. For transparency we also benchmark a conservative DistilVDR variant that forces InternViT to eager attention (`use_flash_attn=False`). At $B = 8$, DistilVDR-Fast drops to 31.91 docs/sec (VRAM 5.73 GB) and DistilVDR-HiRes to 9.27 docs/sec (11.56 GB); even under this configuration DistilVDR-Fast remains faster than every sub-1 B multi-vector baseline.

Measurement protocol. Query latency is the mean over 20 queries at $B = 1$ after 3 warmup queries, with `cuda.synchronize()` bracketing the forward call and tokenisation included. Document throughput and peak VRAM are measured at fixed batch $B = 8$ on a 100-page synthetic corpus after 3 warmup forward passes; multi-vector baselines run their official per-token pooling and projection so that the per-document tensor written to the index is fully formed. Index size is computed analytically from each baseline's released storage format (T tokens per document at dimension d in float16 for multi-vector retrievers; one d-dimensional float32 vector per document for single-vector retrievers). CPU scoring latency is the mean over 20 runs on a single thread (`torch.set_num_threads(1)`) against a synthetic 10 000-candidate corpus in the appropriate storage format; the single-vector path is a dot product, the multi-vector path is the MaxSim routine over 32 query tokens.

Comparative Study of AI Agent Architecture Efficiency in Table Processing Tasks

Timur Gabbasov, Aleksandr Sandalov, Aleksandr Maksimov, Maria Khodorchenko, Nikolay Butakov
ITMO University
Industrial AI Research Lab
Saint Petersburg, Russia

Abstract

Tables are central to business document workflows, but practical table-processing tasks vary substantially in structure, content, and required action. We compare eight ReAct-agent configurations built on the same Qwen3.6-35B-A3B model. The design crosses four tool profiles—Minimal, Search, Spreadsheet, and Full—with the presence or absence of Markdown-based procedural skills. We evaluate the configurations on 40 tasks from company workflows, FinQA, and SpreadsheetBench. Across three complete replicates, comprising 960 agent–task episodes, Full with skills obtains the highest aggregate success rate of 75.0% and a mean score of 0.8061. Full without skills reaches 65.0%, while Spreadsheet without skills reaches 73.3%. Aggregated across tool profiles, skills increase success from 69.4% to 72.3%, but their effect is heterogeneous: negative for Minimal and Spreadsheet and positive for Search and Full. Overall success remains stable across replicates at 70.3%, 70.9%, and 71.3%, although individual configuration results vary more substantially. These findings suggest that procedural guidance is most promising when an agent must coordinate a broad tool environment rather than as a universal augmentation.

1 Introduction

Business processes continuously produce and consume semi-structured documents: financial workbooks, operational reports, patent tables, translation sheets, inventories, reconciliations, and ad hoc analytical files. Unlike plain text, a spreadsheet combines values, formulas, formatting, sheet structure, and implicit layout conventions. A useful agent must therefore do more than answer a question about isolated cells. It may need to inspect a workbook, locate relevant regions, aggregate values, interpret long text, edit formulas or content, and save a valid artifact while preserving unrelated information.

Large language models make it possible to construct general-purpose agents for these workflows, but performance depends not only on the model. The surrounding architecture determines what the model can inspect, which operations it can execute, how it searches long tables, whether it delegates analysis to specialized tool wrappers, and whether it can consult reusable task guidance. In practice, the same model can behave very differently when equipped with only Python, with search tools, with spreadsheet-specific execution components, or with a combined tool environment.

Current spreadsheet and table benchmarks provide valuable test cases, but each benchmark represents only part of the business task surface. SpreadsheetBench focuses on real-world spreadsheet manipulation (Ma et al., 2024), while FinQA focuses on numerical reasoning over financial reports (Chen et al., 2021). Company workflows additionally include long textual tables, translation-oriented documents, and compact but semantically dense accounting sheets. This motivates an evaluation that combines public benchmark tasks with tasks obtained from actual company practice.

This work asks three research questions:

RQ1: Which table-processing task types occur in business practice, and how completely are they represented by existing benchmarks?

RQ2: Which organization of an AI agent most effectively solves business-oriented table-processing tasks?

RQ3: Does a modern LLM-based agent benefit from explicit skills in addition to executable tools?

We address these questions through a controlled comparative study. All configurations use the same ReAct loop (Yao et al., 2023) and Qwen3.6-35B-A3B model (Qwen Team, 2026). We vary the available toolset and the presence of skills, producing

The Proceedings of Workshop on Document Intelligence and Understanding, pages 400–408
October 24, 2026 ©2026 Association for Computational Linguistics

eight configurations in a 4×2 design. The agent operates locally, without Internet access, and may execute Python with `pandas`, `openpyxl`, and `numpy`. Each episode is allowed up to 100 agent-loop iterations.

The paper makes two main contributions:

- A heterogeneous evaluation set that combines 20 tasks from real company practice with public FinQA and SpreadsheetBench samples. The company tasks cover both question answering and editing across nine operational domains, including accounting, patent, and translation-oriented tables.
- A three-replicate comparative study of tool organization and skill augmentation for a fixed ReAct agent and model, comprising 960 episodes and reporting quality, latency, iteration, token, and replicate-level stability measurements for all eight configurations.
- An empirical analysis showing that the effect of procedural skills is tool-profile dependent: positive for the Search and Full profiles in aggregate, but absent or slightly negative for the narrower Minimal and Spreadsheet profiles.

2 Related Work

2.1 Benchmarks for tables and spreadsheets

Research on table reasoning includes question answering and verification over semi-structured tables, such as WikiTableQuestions (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), TaPas (Herzig et al., 2020), and TaBERT (Yin et al., 2020). These benchmarks helped establish table understanding as a distinct reasoning problem, but their output is primarily textual and they generally do not require modifying and returning a workbook.

FinQA introduced numerical reasoning over financial reports by combining text, tables, and executable programs (Chen et al., 2021). It is relevant to business analytics, but it primarily evaluates question answering and numerical derivation rather than general spreadsheet editing. SpreadsheetBench introduced real-world spreadsheet manipulation and artifact-based evaluation (Ma et al., 2024). It includes diverse workbook operations and is substantially closer to end-to-end office automation. However, public benchmarks still leave gaps relative to internal business workflows, particularly long text-heavy tables, translation sheets, and domain-specific documents whose structure is shaped by company processes rather than benchmark design.

Our goal is not to replace these benchmarks. Instead, we use public samples as anchors and add tasks from real company practice to study whether conclusions about agent architecture remain consistent across heterogeneous task families.

2.2 Agent organizations for table processing

Several systems organize LLMs around actions, planning, validation, and structured workbook access. SheetCopilot (Li et al., 2023), SpreadsheetLLM (Dong et al., 2024), SheetAgent (Chen et al., 2025), and related systems explore tool use, spreadsheet encodings, multi-step execution, and specialized reasoning components. ReAct interleaves reasoning and acting (Yao et al., 2023), while CodeAct uses executable code as an action space (Wang et al., 2024). These approaches motivate our base architecture: a ReAct agent that can invoke Python and optional specialized tools. Beyond spreadsheets, document-processing approaches include topic-specialized generation models (Nevezhin et al., 2020) and hybrid retrieval combining text with graph-community summaries (Karkout et al., 2026).

The present study focuses on two architectural choices. The first is *tool specialization*: whether the agent receives only a general Python execution tool, search-oriented tools, spreadsheet-specific tool wrappers, or the union of these capabilities. The second is *skill augmentation*. We define a skill as a Markdown instruction file injected into the agent's system prompt. Skills contain generalized experience and recommended procedures for particular table types; they do not execute code and are distinct from tools. Unlike prior systems that propose a particular integrated spreadsheet-agent architecture, we hold the base model and ReAct loop fixed while varying executable tool access and procedural prompt guidance as separate experimental factors.

3 Dataset

3.1 Composition

The evaluation set contains 40 tasks from three sources. Twenty tasks originate from the real operational practice of one company, ten are sampled from FinQA, and ten are sampled from SpreadsheetBench. The corporate subset is evenly di-

Source	Tasks	Primary mode
Corporate	20	10 QA / 10 editing
FinQA	10	Financial QA
SpreadsheetBench	10	Workbook manipulation

Table 1: Evaluation-set composition. Corporate tasks capture real company workflows and heterogeneous document structures; FinQA contributes numerical reasoning over financial reports and tables; SpreadsheetBench is a public real-world spreadsheet-editing benchmark.

vided into 10 question-answering tasks and 10 editing tasks. Each configuration processes the same 40 tasks in three complete replicates, yielding 120 episodes per configuration and 960 agent–task episodes in total. Table 1 summarizes the 40 unique tasks; replicates repeat the same task set while allowing stochastic agent trajectories to vary.

3.2 Corporate table families

The company-derived tasks span nine operational domains: accounting (4 tasks) and human resources, IT, incident management, operations, patents, procurement, project management, and translation (2 tasks each). Each domain contributes equal numbers of question-answering and editing tasks. Structurally, the tasks were selected to represent three recurring table families.

Accounting tables. Relatively small but semantically dense workbooks. They require careful interpretation of financial labels, values, formulas, periods, and aggregation rules.

Patent tables. Long tables containing extensive free text. These tasks stress retrieval, evidence localization, and synthesis across text-heavy rows and columns.

Translation-oriented tables. Long documents in which text must be located, interpreted, or transformed while preserving tabular organization and workbook structure.

These categories broaden the test surface beyond a single spreadsheet format. They vary in length, text density, numeric content, and whether the required output is an answer or a modified artifact.

4 Methods

4.1 Base agent and environment

All configurations use the same ReAct agent with Qwen3.6-35B-A3B (Qwen Team, 2026) as the backbone model, served in its instruct (non-thinking) configuration with native tool calling. The agent alternates between reasoning, selecting an action, observing the tool result, and continuing until it returns an answer or output artifact. The execution environment is local and has no Internet access. Python execution supports `pandas`, `openpyxl`, and `numpy` for data analysis and workbook manipulation. Full model, sampling, and serving details are listed in Appendix A.

Each task permits at most 100 agent-loop iterations. No separate hard limits were imposed on elapsed time or token consumption; these quantities are measured as outcomes. This design prioritizes task completion while allowing efficiency differences to emerge from actual agent behavior.

4.2 Tools

The experiment uses five functional capabilities: a Python execution tool, an aspect-summarization tool, an exhaustive table-search tool, a tabular-analysis subagent, and a workbook-manipulation subagent. They are grouped into four toolsets:

- **Minimal:** the Python execution tool only. This is the general-purpose baseline.
- **Search:** the Python execution tool plus aspect summarization and exhaustive table search. This toolset targets long and text-heavy tables through decomposition and systematic retrieval.
- **Spreadsheet:** the Python execution tool plus tabular-analysis and workbook-manipulation tool wrappers. This toolset delegates structured analysis and workbook operations to specialized components.
- **Full:** all five capabilities. This toolset maximizes the available action space and requires the model to select among both search-oriented and spreadsheet-oriented tools.

The five capabilities differ in scope. `python_executor` runs code in an isolated persistent worker whose filesystem access is restricted to the episode workspace and whose imports are controlled by an allowlist. `aspect_summary` deterministically profiles workbook sheets, candidate headers, samples, and density using `openpyxl`. `exhaustive_search` performs bounded local token search over workbook cells. `pandas_analysis_agent` loads a

rectangular table into a DataFrame and executes supplied analysis code. `xlsx_subagent` provides workbook previews and executes `openpyxl`-based operations in an isolated runtime. Despite their names, the latter two are specialized tool wrappers rather than independently reasoning LLM agents.

4.3 Skills

Skills are declarative Markdown instruction files rather than executable tools. They describe runtime constraints, task-type branching, tool-selection conditions, table-family characteristics, expected output formats, workbook-preservation requirements, and final validation procedures.

Skills are not retrieved or selected by the model during an episode. When the skill condition is enabled, the experiment harness deterministically selects the files before the first model request and injects their content into the system prompt. Selection uses the task-family label and the active tool profile (*Minimal*, *Search*, *Spreadsheet*, or *Full*). Each episode receives one family-specific skill and one generic skill for the corresponding tool profile, subject to a 12,000-character limit. Skill exposure is therefore guaranteed, although the experiment does not measure whether the model follows every individual instruction.

The skills were authored from tool contracts, recurring table-family characteristics, required task interfaces, and preliminary smoke-run observations. They contain no task-specific reference answers, target values, or hidden workbook contents. However, because the skills encode knowledge of the task formats and evaluation interface, the experiment evaluates their utility on a fixed benchmark rather than transfer to entirely unseen task families. Every tool profile is evaluated both without and with skills, yielding the eight configurations in the 4×2 design.

5 Experiments

5.1 Methodology and metrics

We execute three complete replicates of the experimental matrix. In each replicate, all eight configurations process the same 40 tasks, producing 320 episodes per replicate and 960 episodes in total. The task-order seed is fixed within each replicate and shared across configurations. Stochastic decoding may nevertheless produce different trajectories across replicates.

Tasks are executed with concurrency 15. The complete three-replicate matrix takes approximately 7 hours 31 minutes of wall-clock time. Because episodes run concurrently, average episode duration must not be summed to estimate wall-clock runtime.

We report five primary quantities: success rate, mean score, average time per task, average number of iterations, and average token usage. Success rate measures the fraction of tasks judged successfully completed. Mean score preserves partial credit from the task evaluators. Average time is the full duration of an individual episode. Average iterations measures the number of agent-loop steps used. Average tokens is the mean total (prompt plus completion) token count per episode; a prompt/completion breakdown is given in Appendix B.

Question-answering tasks are scored by answer matching; FinQA tasks use a FinQA-style evaluator. Editing tasks are scored by cell-level comparison of the produced workbook against a reference. In `literal` mode, values, text, and required blank cells must match and formulas are not accepted in place of values. In `formula_exact` mode, the formula string must match exactly. In `formula_semantic` mode, an alternative formula is accepted only when its computed behavior and direct dependencies satisfy the task rule. Values, formulas, sheet names, and merged ranges outside the allowed edit surface must remain unchanged. Editing success requires a score of 1.0 and successful preservation; the continuous score records the fraction of matched target cells and therefore retains partial credit for nearly correct workbooks.

We report aggregate results over all three replicates together with replicate-level outcomes. For skill-versus-no-skill comparisons within each tool profile, we use an exploratory paired task-cluster sign-flip test. The resulting p-values are uncorrected; none of the four profile-level success effects passes a Bonferroni threshold of $\alpha = 0.0125$. We therefore treat the significance analysis as exploratory and emphasize effect heterogeneity and replicate-level stability rather than binary significance claims.

5.2 Results

Table 2 reports the primary metrics for all eight configurations, and Figure 1 visualizes success rates.

Across the three replicates, Full+Skills is the best aggregate configuration, with 90 successful episodes out of 120 (75.0%) and a mean score of

Toolset	Skills	Success rate	Mean score	Time (s)	Iterations	Tokens (k)
Minimal	no	84/120 (70.0%)	0.7742	136.4	7.10	41.3
Minimal	yes	81/120 (67.5%)	0.7745	194.5	7.79	96.5
Search	no	83/120 (69.2%)	0.7526	160.0	7.49	48.5
Search	yes	89/120 (74.2%)	0.8021	159.3	7.22	63.3
Spreadsheet	no	88/120 (73.3%)	0.7833	151.1	7.38	58.5
Spreadsheet	yes	87/120 (72.5%)	0.7933	188.8	7.72	89.4
Full	no	78/120 (65.0%)	0.7616	178.5	7.02	60.7
Full	yes	**90/120 (75.0%)**	**0.8061**	176.3	6.95	72.5
All	—	680/960 (70.8%)	0.7810	168.1	7.33	66.3

Table 2: Aggregate results over three complete replicates (120 episodes per configuration). Time, iterations, and tokens are per-episode means; tokens include prompt and completion tokens.

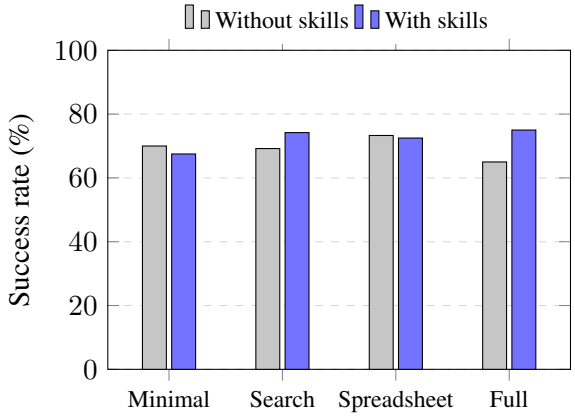


Figure 1: Aggregate success rate by tool profile and skill condition over three complete replicates. Each bar summarizes 120 episodes.

Replicate	Success	Rate	Mean score
1	225/320	70.3%	0.7659
2	227/320	70.9%	0.7880
3	228/320	71.3%	0.7891

Table 3: Aggregate stability across the three replicates.

0.8061. Full without skills reaches 65.0%, producing an aggregate within-profile difference of 10 percentage points. The size of this difference varies across replicates: 20 points in Replicate 1, 5 points in Replicate 2, and 5 points in Replicate 3.

The aggregate success rate is stable across replicates, ranging from 70.3% to 71.3%. Individual configurations vary more substantially. In particular, Full+Skills decreases from 82.5% in Replicate 1 to 72.5% in Replicate 2 and 70.0% in Replicate 3. This illustrates why the primary interpretation should rely on the three-replicate aggregate rather than on the strongest individual run.

Search improves from 69.2% without skills to 74.2% with skills. In contrast, Minimal decreases from 70.0% to 67.5%, and Spreadsheet changes from 73.3% to 72.5%. Skills therefore do not provide a uniform benefit: their observed effect depends on the active tool profile.

Among configurations without skills, Spreadsheet is strongest at 73.3%, while Full is weakest at 65.0%. This provides evidence against a simple "more tools is better" interpretation. A broad action space may introduce tool-selection and coordination costs, although the current design does not isolate tool count from tool type.

Table 4 breaks down success rates by task category. Across all configurations and replicates, Corporate Editing is the easiest category at 96.7% success, followed by Corporate QA at 84.6%. FinQA reaches 71.7%, while SpreadsheetBench is substantially harder at 30.4%. Full+Skills obtains the strongest SpreadsheetBench result at 40.0%, but even this remains far below its performance on the corporate subsets. The contrast indicates that aggregate results should be reported together with benchmark-family breakdowns.

5.3 Discussion

RQ1: business task coverage. The combined dataset exposes a broader task surface than either public source alone. FinQA contributes financial reasoning, SpreadsheetBench contributes artifact-oriented workbook manipulation, and the corporate subset adds real operational QA and editing across nine business domains, including accounting, patent, and translation-oriented tables. The observed category gap also indicates that benchmark identity matters: SpreadsheetBench tasks are markedly harder than the corporate subsets in this sample. This does not prove that existing benchmarks are generally incomplete, but it supports the narrower claim that no single included benchmark

Toolset	Skills	QA	Editing	Corp. QA	Corp. Edit	FinQA	SpreadsheetBench
Minimal	no	78.3%	61.7%	83.3%	96.7%	73.3%	26.7%
Minimal	yes	76.7%	58.3%	86.7%	93.3%	66.7%	23.3%
Search	no	75.0%	63.3%	80.0%	93.3%	70.0%	33.3%
Search	yes	81.7%	66.7%	90.0%	100.0%	73.3%	33.3%
Spreadsheet	no	81.7%	65.0%	86.7%	100.0%	76.7%	30.0%
Spreadsheet	yes	80.0%	65.0%	90.0%	96.7%	70.0%	33.3%
Full	no	70.0%	60.0%	70.0%	96.7%	70.0%	23.3%
Full	yes	**81.7%**	**68.3%**	**90.0%**	96.7%	73.3%	**40.0%**

Table 4: Category-level success rates aggregated over three replicates. QA and Editing contain 60 episodes per configuration; each benchmark-family column contains 30 episodes per configuration.

covers all table types encountered in the studied company workflow.

RQ2: effective agent organization. The results do not support a simple "more tools is better" rule. Without skills, the Spreadsheet toolset outperforms the Full toolset by 8.3 percentage points. Specialized tools may reduce the burden of selecting among heterogeneous actions and may better match structured workbook operations. Search-oriented tools alone do not produce the strongest aggregate performance, although they may remain useful for long textual tables.

The best aggregate result comes from combining the Full toolset with skills. This is consistent with the hypothesis that a broad architecture can benefit from guidance about how to navigate its action space. In other words, tool availability and tool orchestration should be treated as separate design problems.

RQ3: value of skills. Aggregated over all tool profiles, skill-enabled configurations reach 347/480 successes (72.3%), compared with 333/480 (69.4%) without skills, an observed difference of 2.9 percentage points. Mean score increases from 0.7679 to 0.7940.

The effect is heterogeneous across tool profiles. Skills change success by -2.5 points for Minimal, $+5.0$ for Search, -0.8 for Spreadsheet, and $+10.0$ for Full. The Full contrast has an uncorrected exploratory $p = 0.0319$, but it does not pass the Bonferroni threshold of $\alpha = 0.0125$ for four profile-level comparisons. The results therefore support a profile-dependent interpretation rather than a universal benefit of skills.

Efficiency trade-offs. Across tool profiles, skills increase average episode time from 156.5 to 179.7 seconds, iterations from 7.25 to 7.42, and token usage from 52.2k to 80.4k. The overhead is strongly profile dependent. Minimal grows from 41.3k to 96.5k tokens, Search from 48.5k to 63.3k, Spreadsheet from 58.5k to 89.4k, and Full from 60.7k to 72.5k.

Full is the only profile in which skills improve aggregate success while slightly reducing both mean time (178.5 to 176.3 seconds) and iterations (7.02 to 6.95). Search also improves success without a meaningful latency increase. Thus, the practical value of skills depends on whether their quality gain offsets their additional prompt cost in a particular tool environment. Because the model is self-hosted, token-priced API cost is undefined; a complete cost analysis would join the recorded token and wall-time measurements with server-side infrastructure telemetry.

5.4 Failure Analysis

The hardest tasks are concentrated in FinQA and SpreadsheetBench. Across the eight configurations and three replicates, `finqa_qa_002`, `sb_edit_1845`, and `sb_edit_54242` obtain no successful episodes out of 24. Failures include incorrect formula dependencies, incomplete extraction from long tables, output-format mismatches, and workbook-preservation violations.

Binary success can obscure partial progress on strict editing tasks. For example, `sb_edit_13_1` has a mean cell-level score of 0.9566 but only 4/24 successful episodes because editing requires a perfect score of 1.0. Reporting both mean score and binary success is therefore necessary for interpreting workbook-editing performance.

6 Conclusion

We presented a three-replicate comparative study of eight table-processing agent configurations, covering 40 tasks and 960 agent–task episodes.

Full+Skills achieves the highest aggregate success rate of 75.0%, while Spreadsheet without skills is the strongest no-skill configuration at 73.3%. Without skills, the broad Full profile reaches only 65.0%, showing that access to a larger action space does not by itself guarantee better performance.

Skills provide a small positive aggregate difference of 2.9 percentage points, but their effect depends strongly on the tool profile. The largest positive contrast occurs for Full, while Minimal and Spreadsheet show no success-rate improvement. Procedural guidance should therefore be treated as an architecture-dependent orchestration mechanism rather than a universally beneficial prompt addition.

Future work should enlarge the number and diversity of unique tasks, evaluate additional models and organizations, isolate individual tool contributions, compare skill-loading and retrieval strategies, and measure instruction adherence directly. Releasing tool schemas, skills, evaluator specifications, and non-confidential task metadata would further improve reproducibility.

Limitations

- The experiment includes three replicates, which provide an initial view of stochastic stability but remain insufficient for precise population-level estimates. Replicates repeat the same fixed set of 40 tasks and therefore increase trajectory coverage rather than task-set diversity.
- The evaluation set has only 40 tasks and is purposively assembled. The corporate tasks come from one company and may not represent other industries or organizations.
- The eight configurations differ in tool availability, and the Full toolset has a larger action space. The study compares complete architectures rather than isolating every individual tool effect.
- Skills are deterministically injected into the prompt, but the study does not measure instruction-level adherence or identify which skill sections influence individual decisions.
- Success and score definitions are summarized in Section 5.1, but the full task-by-task evaluation protocol should be documented in a released benchmark protocol.
- The corporate workbooks cannot be fully released because of confidentiality constraints, limiting exact external replication.

Acknowledgments

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-152025-010.

References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact: A large-scale dataset for table-based fact verification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, Jianye Hao, Hangyu Mao, and Fuzheng Zhang. 2025. SheetAgent: Towards a generalist agent for spreadsheet reasoning and manipulation via large language models. In *Proceedings of the ACM Web Conference 2025 (WWW '25)*. Association for Computing Machinery.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711. Association for Computational Linguistics.

Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. 2024. Encoding spreadsheets for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20728–20748. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333. Association for Computational Linguistics.

Maha Karkout, Maria Andreevna Khodorchenko, Nikolay Alekseevich Butakov, and Denis Nasonov. 2026. Efficient KG-augmented RAG with reusable graph community summaries. In *Proceedings of the Knowledge Graphs and Large Language Models Workshop (KG-LLM) @ LREC26*, pages 110–119, Palma, Mallorca (Spain). ELRA Language Resources Association (ELRA).

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. SheetCopilot: Bringing software productivity to the next level through large language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*.

Egor Nevezhin, Nikolay Butakov, Maria Khodorchenko, Maxim Petrov, and Denis Nasonov. 2020. Topic-driven ensemble for online advertising generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2273–2283, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.6-35B-A3B model card. Accessed August 2026.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426. Association for Computational Linguistics.

A Model and Serving Configuration

All eight configurations use the identical model deployment and decoding configuration. The model is Qwen3.6-35B-A3B (Qwen Team, 2026), a mixture-of-experts model run in its instruct (non-thinking) configuration with native tool calling, served through an OpenAI-compatible endpoint by vLLM 0.19.1 (Kwon et al., 2023) in bf16 precision on two H200 GPUs. Table 5 lists the decoding and client parameters.

Parameter	Value
Temperature	0.7
Top-p	0.8
Top-k	20
Min-p	0.0
Presence penalty	1.5
Repetition penalty	1.0
Max output tokens	32,768
Context window	262,144 tokens
Thinking mode	disabled
Tool-call mode	native, `tool_choice=auto`
Request timeout	600 s
Max API retries	3

Table 5: Decoding and client parameters shared by all configurations.

The execution environment is offline (no Internet access) and runs Python 3.10.20 with `pandas` 2.3.3, `openpyxl` 3.1.5, and `numpy` 2.2.6. Each episode is limited to 100 agent-loop iterations; no time or token limits are imposed. Within each replicate, the task order is shuffled with a fixed seed shared by all eight configurations, and no API retries occurred in any run.

B Token Usage Breakdown

Table 6 reports per-episode mean prompt tokens, completion tokens, and tool calls for each configuration, aggregated over all three replicates. Prompt tokens dominate total usage; skills increase prompt size substantially because the selected Markdown files are injected into the system prompt, while completion counts remain within a narrow band across configurations.

Toolset	Skills	Prompt	Compl.	Tool calls
Minimal	no	39,132	2,183	5.72
Minimal	yes	93,600	2,946	6.80
Search	no	45,956	2,497	6.49
Search	yes	60,817	2,455	6.21
Spreadsheet	no	56,095	2,420	6.38
Spreadsheet	yes	86,545	2,886	6.72
Full	no	57,821	2,833	6.02
Full	yes	69,819	2,668	5.95

Table 6: Per-episode mean prompt tokens, completion tokens, and tool calls aggregated over three replicates.

MaDOVE: a Dataset for Multilingual Document VQA Evaluation

Elio Musacchio[1,2], Lucia Siciliani[2], Pierpaolo Basile[2], Giovanni Semeraro[2], Mark Federico Zampedroni[3], Asia Beatrice Uboldi[3]
[1]University of Pisa, [2]University of Bari Aldo Moro, [3]Fastweb SpA
elio.musacchio@phd.unipi.it

Abstract

Visual Question Answering (VQA) is one of the most popular tasks in the fields of Computer Vision and Natural Language Processing. The task is to provide an answer to a question related to the contents of a given image. VQA has been expanded to several domains. One of these is document VQA, where the image is a page of a document and the question relates to the text in the image. In this case, the system that is being evaluated has to understand text written within the image, unlike common VQA datasets where images are natural scenes. However, document VQA datasets are exclusively in English; hence, the system's performance is evaluated only for English. In light of this, we introduce MaDOVE (Multiltilingual DOcument VQA Evaluation), a novel dataset designed to evaluate document VQA in multiple languages. Moreover, to establish a strong benchmark, we conduct an extensive set of experiments by evaluating a diverse range of models on this task.

1 Introduction

Visual Question Answering (VQA) is one of the most explored tasks in the history of Computer Vision and Natural Language Processing (Kim et al., 2025). The task tests the ability of a system to: 1) understand the visual contents of an image; 2) reason jointly about the contents of an image and a given question; 3) provide an answer that is correct w.r.t. the contents of the image and the question. Solving VQA is a non-trivial task, as it requires multimodal understanding and reasoning.

In light of the success of Large Language Models (LLMs), a new type of architecture has emerged that extends LLMs with multimodal capabilities. These models are called Large Vision-Language Models (LVLMs), and they can handle data from both the visual and textual domains. Evaluating LVLMs has become extremely important and several benchmarks have been designed for this purpose (Xu et al., 2024; Cao et al., 2024). VQA is one of the most important tasks in this context. However, while there are works focusing on natural scenes and text-centric images, none focus on multilingual document VQA. This omission creates a relevant gap, as it limits our understanding of these models' capabilities, since they are tested on a narrow range of visual domains.

In this work, we focus on text-centric document images. While there are some efforts in the state of the art for document VQA, they are exclusively in English. In light of this, the contributions of this work are the following:

- We introduce MaDOVE, a large-scale multilingual document VQA dataset designed to bridge the gap in evaluating multimodal reasoning capabilities over documents. To address the need for a large amount of data during training, we generate answers synthetically, while ensuring high-quality evaluation through two different test sets: one human-annotated and another LLM-annotated for scalable comparison[1].
- We study the effect of fine-tuning on a synthetically generated dataset of multilingual document VQA pairs and how this affects model performance on both text-centric and natural scene VQA datasets.
- We benchmark a diverse set of models on MaDOVE, offering a systematic study on multilingual document VQA and highlighting an important issue for current practices in VQA evaluation.

[1] https://huggingface.co/collections/m-elio/madove

The Proceedings of Workshop on Document Intelligence and Understanding, pages 409–427
October 24, 2026 ©2026 Association for Computational Linguistics

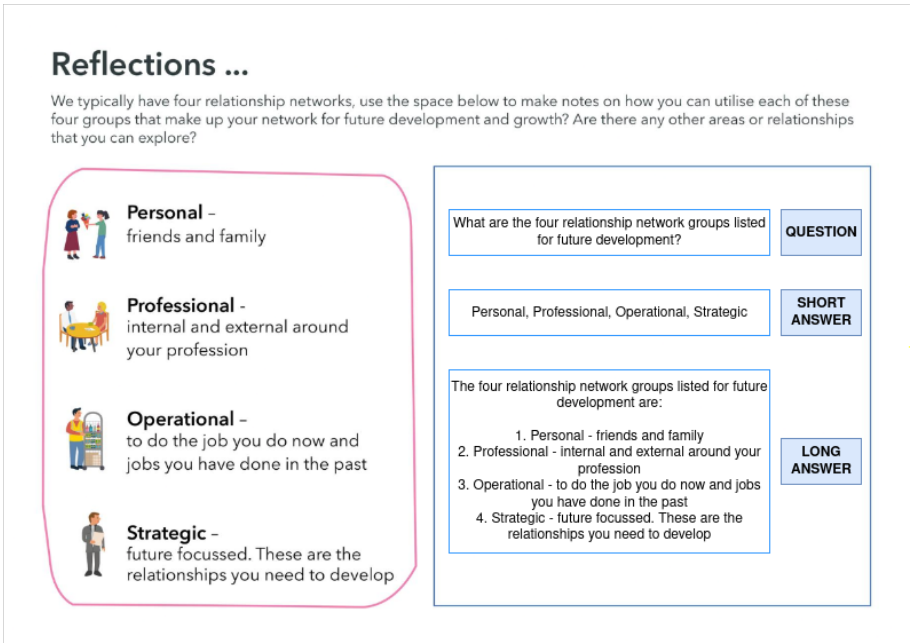


Figure 1: Example of an instance from our dataset. Each instance for each language consists of an image containing text in that language, a question related to the text in the image, and two answers: a long one (meaningfully detailed) and a short one (concise).

2 Related Works

2.1 Visual Question Answering

The original task of free-form and open-ended VQA was proposed by Antol et al. (2015), who introduced a dataset comprising over 250K images, 760K questions and 10M answers. Following this work, several datasets have been proposed to further evaluate model performance on VQA. These datasets include GQA (Hudson and Manning, 2019), which focuses on scene understanding of everyday images, TEXTVQA (Singh et al., 2019), which focuses on text embedded within natural scene images. The task was then specialized in specific domains or sources, for example, Shabtay et al. (2025) proposed LIVEXIV, a VQA dataset over scientific ArXiv papers, while Lau et al. (2018) proposed VQARAD, a VQA dataset of question-answer pairs on radiology images. Another text-centric benchmark is ST-VQA (Biten et al., 2019), which focuses on semantic information conveyed by text within a natural scene image.

The focus of this work is on text-centric document VQA. For this type of task, there are some existing works in the literature. DOCVQA (Mathew et al., 2021) is a VQA dataset where images are pages of documents downloaded from UCSF Industry Documents Library, with questions and answers manually written by remote workers. This is the first dataset to introduce document VQA with a focus on understanding the document structure. INFOVQA (Mathew et al., 2022) is similar to DOCVQA but focuses on infographics, which are documents designed to effectively communicate information using a combination of textual, graphical and visual elements. Other relevant resources include CHARTQA (Masry et al., 2022) and MP-DOCVQA (Tito et al., 2022). CHARTQA focuses on visual and logical reasoning on charts, while MP-DOCVQA focuses on multi-image document VQA, rather than single image. For multilingual document VQA, relevant works include HW-MLVQA (Pal et al., 2025) and MTVQA (Tang et al., 2025). HW-MLVQA is an extensive collection of handwritten documents with expert human validation of question-answer pairs. MTVQA provides high-quality human expert annotations across nine diverse languages. Images are obtained from both publicly available datasets and Internet crawls. Annotations for questions and answers are obtained by human experts. Remarkably, this dataset also contains document images. However, MTVQA does not focus exclusively on document images, which limits the number of instances for this specific task in the train and test splits of the dataset, while HW-MLVQA focuses on handwritten documents and only covers the English and Hindi languages.

Despite the extensive effort and research in VQA, no benchmark specifically targets document images containing multilingual text. This highlights the relevance and diversity of MaDOVE w.r.t. other VQA datasets, allowing for the broadening of the scope of VQA evaluation.

2.2 Large Vision-Language Models

LVLMs are an extension of pre-trained LLMs to expand their inputs so that they can understand images together with text. The most relevant approach is that of LLaVA (Liu et al., 2023), where image embeddings extracted from a vision encoder are projected into the latent space of the LLM. This methodology was then further enhanced in LLaVA 1.5 HD (also known as LLaVA NeXT) (Liu et al., 2024). While initial works, including LLaVA, only focused on the English language, modern LVLMs support multiple languages. These models include GEMMA (Kamath et al., 2025), LLaMA 4 (Meta AI, 2025) and QWEN 2.5 VL (Bai et al., 2025), the most recent models developed by Google, META and Qwen, respectively. Finally, several studies have examined the application of LVLMs to VQA. Yang et al. (2025) propose a three-step process to integrate commonsense knowledge in LVLMs, while Shiri et al. (2024) study LVLMs for their spatial understanding and reasoning capabilities.

No work has investigated the effect of fine-tuning a LVLM on a multilingual document VQA mixture or has established benchmarks in this setting. Hence, we will evaluate the performance of both pre-trained and fine-tuned models on MaDOVE and provide the first benchmark study for multilingual document VQA.

3 Dataset Creation

3.1 Data Collection

We collect the *vdr-multilingual* (Multilingual Visual Document Retrieval)[2] dataset train and test splits. This dataset is designed for visual document retrieval, where the objective is to find the image that answers the query. It is important to note that here the task is missing the last step of VQA, i.e., exploiting the information provided by the image to generate a factual answer to the question. In *vdr*, images were collected from online resources, while queries were generated synthetically, with additional manual verification for the test set. The train set comprises 280,769 instances (across all languages), while the test set consists of 300 instances (one for each language). The dataset does not provide queries for all images, since additional images serve as hard negatives. In the aforementioned statistics, we considered only instances paired with a query. This represents a solid foundation for building our document VQA dataset. In addition, images were collected with a focus on multilinguality. The dataset supports five languages, with the text in each image written in a specific language. The supported languages are: English, French, German, Italian and Spanish. Hence, we will focus on these languages for creating our dataset. However, while the dataset provides queries and images, textual answers to the given queries are not provided. In light of this, we need to augment the dataset with textual answers to the queries. Given the dataset's scale, manual annotation is infeasible; hence, we focus on synthetic generation using LVLMs.

3.2 Data Annotation

We perform synthetic generation over the queries for each language. Specifically, we leverage QWEN2.5-VL-72B-INSTRUCT. We chose this model since it represents a good trade-off between performance and efficiency. Furthermore, according to the technical report, the Qwen2.5 VL family of models shows powerful document parsing capabilities. We perform inference by providing the model with the image and its associated query from the original dataset. The model is asked to generate a response to the given question-image pair. Since document VQA datasets, that are DOCVQA and INFOVQA, focus on short answers, we append the string *"\nAnswer the question using a single word or phrase."* to the end of the prompt, forcing the model to generate short and concise answers. We will refer to this type of answer as "short". We also repeat this process without this special string appended to the end of the prompt. We will refer to this type of answer as "long". We perform annotation and validation for both "short" and "long" responses, but we will focus on the "short" responses during evaluation to follow the same style as DOCVQA and INFOVQA. We also add a condition to the prompt, where we ask the model to generate a specific string if the question cannot be answered, allowing us to easily filter these cases after completing the inference process. We use greedy decoding as the inference strategy

[2] https://huggingface.co/blog/vdr-2b-multilingual

to maximize efficiency and reproducibility. After filtering cases where the model was unable to provide the corresponding answer, we report statistics for the train set in Table 1. We show an annotated instance from our dataset for English in Figure 1. Finally, to further augment the test set, we also leverage the model to generate questions for images that are not associated with a query. We then perform the same inference process as before to generate the answers. We report complete technical details for synthetic generation in Appendix A. All the prompts used for each step described in this section are reported in Appendix F. Error analysis reporting common error patterns in the train set is shown in Appendix C. To further verify the effectiveness of this generation process, we validate the generated instances for the test set.

3.3 Data Validation

We perform validation by leveraging both: 1) a human annotator; 2) an LLM-as-a-judge methodology (Zheng et al., 2023) using the GPT-4o (Hurst et al., 2024) model. In both cases, we use a survey to evaluate three aspects of our generated VQA instances: 1) Grammatical correctness of questions and answers (both short and long); 2) Coherence of the answers (both short and long) w.r.t. the question; 3) Coherence of the question and answers (both short and long) w.r.t. the contents of the image. This allows us to cover all possible error cases that could be caused by synthetic generation. The survey consists of eight statements for each instance to evaluate. We report complete details for them in Appendix D. Each statement has to be paired with a judgment on a Likert scale from 1 to 5, where 1 is "Absolutely Disagree", 2 is "Disagree", 3 is "Uncertain", 4 is "Agree", and 5 is "Absolutely Agree". Additionally, we ask both the human annotator and the GPT-4o model to provide a brief explanation for their validation when a negative rating is assigned to any statement, as well as possible solutions to address the issue. For human annotation, we recruit a human annotator who is proficient in English, Italian and Spanish. For the LLM-as-a-judge methodology, we leverage GPT-4o for all languages in the dataset: English, French, German, Italian and Spanish. The human annotator only covers instances in the test set for which a query was already provided in the original *vdr* dataset, whereas for the LLM-as-a-judge methodology, we augment the overall cardinality by also considering instances for which we generated the question. To strike a balance between efficiency and statistical significance, we limit each language subset to approximately one thousand instances. The prompt used for the LLM-as-a-judge methodology is provided in Appendix F. After applying both methodologies, we (the authors) manually validate the instances. We review all cases in which the validator (either the human annotator or GPT-4o) assigned a negative or neutral rating (1 to 3) to any statement. If the issue can be resolved easily, we manually revise the instance based on the validator's explanation. When the issue cannot be easily fixed (e.g., the question is completely incoherent w.r.t. the answer), we remove it from the test set directly. We perform this step for all annotated test sets for each validator; therefore, we obtain eight test sets for each language (three annotated by humans and five annotated by the GPT-4o model). We report statistics for the final test sets in Table 2.

	Answer Type	
Lang	*Short*	*Long*
DE	44,580	47,990
EN	42,158	46,303
ES	47,733	45,261
FR	42,049	45,558
IT	43,109	45,316
TOTAL	219,629	230,428

Table 1: Train set statistics

	Validator	
Lang	*Human*	*GPT-4o*
DE	X	983
EN	230	990
ES	221	991
FR	X	988
IT	213	1,000
TOTAL	664	4,952

Table 2: Test set statistics

3.4 Annotation Comparison

We study the agreement between the human annotator and GPT-4o ratings. Specifically, we aim to understand whether annotations provided by GPT-4o are aligned with those provided by the human annotator. To do so, we study the ratings provided by the human and GPT-4o on the same instances (the subset of instances annotated by the human annotator in English, Italian and Spanish). In Figure 2, we present a radar chart showing the average ratings for the three languages across the eight statements. AVG refers to the agreement obtained by

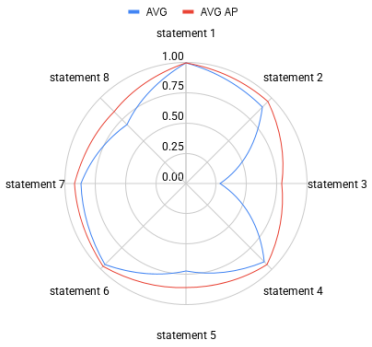


Figure 2: Radar chart for the average agreement between human and GPT-4o for the Likert scale rating obtained on each statement for the three languages.

hard matching the given ratings (that is, for each instance, the result is 1 if the given rating is exactly the same between the two annotators). AVG AP refers to the agreement obtained by matching the ratings on a scale from 1 to 3 rather than 1 to 5. Specifically, "Disagree" and "Agree" ratings (2 and 4, respectively) are merged into the "Absolutely Disagree" and "Absolutely Agree" ratings (1 and 5, respectively). We can appreciate that, except for Statements 3 and 8, which focus on grammatical correctness of the short answer and coherence of the short answer with respect to the image, there is a good degree of alignment between human and model for AVG. When considering AVG AP, the alignment significantly increases. Remarkably, the alignment increases for both Statements 3 and 8. However, we find that the degree of alignment is very low (for some statements, it is close to 0%) when considering only instances where the human annotator provided a negative rating for any statement (either 1 or 2). The same pattern also holds for negative ratings given by GPT-4o. We find that this is due to the following reasons: 1) The number of instances with negative ratings is low (average of around 10% for the three languages for both the human and the GPT-4o model) w.r.t. the number of instances with positive ratings. After manually verifying these instances, we find that most of them are limit cases, where it is reasonable to expect different answers depending on the annotator's background and sensibility. For example, the human annotator provided "Absolutely Disagree" ratings for instances where the answer was a LaTeX formula, since they did not know LaTeX; 2) The GPT-4o model provided higher ratings w.r.t. the human annotator. For example, in the English subset, the GPT-4o model has never provided an "Absolutely Disagree" rating, whereas the human annotator did so for several instances. We also find that the model treated the "Uncertain" rating as a negative rather than a neutral rating in limit cases.

We emphasize that there are cases where the human annotator gave ratings to instances better than the GPT-4o model and vice versa, highlighting that both methodologies should be considered when performing this type of annotation. Still, the LLM-as-a-judge approach is a suitable methodology in our use case, especially considering that the outputs are further refined and filtered after the data validation step.

4 Experiments

We evaluate LVLMs on both the human-annotated and model-annotated test sets. We consider two strategies: 1) *zero-shot inference*, where we leverage pre-trained LVLMs to perform inference directly; 2) *fine-tuning*, where we fine-tune a LVLM on the train split of our dataset. We will also evaluate the fine-tuned models on state-of-the-art multimodal tasks to understand the performance drop on tasks unrelated to document VQA. Finally, we will also evaluate both zero-shot and fine-tuned models on other English language document VQA benchmarks, specifically DocVQA and InfoVQA, to understand whether fine-tuning on our training mixture can also enhance performance on other related benchmarks. We will evaluate outputs generated on our benchmark using the same metric as DocVQA and InfoVQA, that is *Average Normalised Levenshtein Similarity* (ANLS), originally proposed for evaluation on ST-VQA, which takes into account minor answer mismatches.

For fine-tuning, we consider two different types of models: Qwen 2.5 VL 3B, 7B and 32B, which are smaller versions of the original model used for synthetic generation (72B); LLaVA NeXT using a checkpoint trained with LLaMA 3 as the LLM backbone. We consider smaller versions of Qwen 2.5 VL 72B to assess whether distillation from a large model to smaller models can yield significant performance gains, while we examine a LLaVA NeXT model to determine whether a different architecture can still achieve remarkable performance improvements after fine-tuning. We train for one epoch over the entire training mixture, combining both short and long answers. For short answers, we append a string to the question to request a concise response. During fine-tuning, we use LoRA (Hu

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.8668	.7379	.5194	**.3087**	.6082	**.9103**	**.7923**	**.5280**	.2971	**.6319**
ES	.8389	.6334	**.3109**	.1227	.4765	.8583	.6494	.2623	.1201	.4725
IT	.8000	.6359	.3691	**.2080**	.5033	.8538	.7298	.2830	.0604	.4817
	Qwen2.5-VL-32B-Instruct					gemma-3-4b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.8981	.7525	.4808	.1887	.5800	.7246	.3941	.0355	.0000	.2885
ES	**.8732**	**.7718**	.2530	.0955	.4984	.5929	.5319	.0606	.0000	.2963
IT	.8152	.7533	.3564	.1251	.5125	.6164	.5317	.0793	.0000	.3068
	gemma-3-12b-it					gemma-3-27b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.7896	.5404	.0825	.0179	.3576	.7005	.4859	.1182	.0212	.3314
ES	.7555	.6889	.1781	.0000	.4056	.6018	.6115	.1279	.0329	.3435
IT	.5827	**.7877**	.3456	.0250	.4353	.6733	.5876	.2310	.0458	.3844
	Llama-4-Scout-17B-16E-Instruct					Llama-4-Maverick-17B-128E-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.8289	.6245	.3474	.1200	.4802	.8425	.7204	.4228	.1425	.5321
ES	.8635	.6683	.2435	**.2220**	**.4993**	.8278	.7180	.2563	.1128	.4787
IT	.7878	.7074	.3628	.2005	.5146	**.8741**	.7593	**.4382**	.1803	**.5630**
	llama3-llava-next-8b-hf									
	Q1	**Q2**	**Q3**	**Q4**	**AVG**					
EN	.5075	.3232	.0598	.0000	.2226					
ES	.3958	.3122	.0528	.0147	.1939					
IT	.4572	.3181	.0495	.0000	.2062					

Table 3: Complete results for the pre-trained models on the human-annotated test set.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9343	**.7907**	.6556	.4797	.7151	.9294	.7883	.6652	.5671	.7375
ES	.9142	.8048	.6215	.5084	.7122	.8742	**.8398**	**.7031**	**.6395**	.7642
IT	.8201	.7045	.4573	.4907	.6182	**.8731**	.7732	.5406	**.5979**	**.6962**
	Qwen2.5-VL-32B-Instruct					llama3-llava-next-8b-hf				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	**.9602**	.7852	**.7382**	**.5985**	**.7705**	.7155	.4532	.4832	.2431	.4737
ES	**.9214**	.8055	.6998	.6318	**.7646**	.6059	.5136	.3135	.2033	.4091
IT	.8433	**.7769**	**.5660**	.5882	.6936	.6267	.5416	.3619	.2961	.4566

Table 4: Complete results for the fine-tuned models on the human-annotated test set.

et al., 2022) to improve efficiency while retaining the model's original knowledge. Complete fine-tuning details, including hyperparameters, can be found in Appendix B. Additionally, we benchmark multimodal and multilingual models, specifically GEMMA 3 4B, 12B and 27B, as well as LLAMA 4 17B with 16 experts (Scout) and 17B with 128 experts (Maverick). During inference, the prompt consists of: the image tokens, the question and an additional string to force the model to generate a short response. The images are provided in their original resolution as in the *vdr* dataset. We also apply the chat template expected by each model. At inference time, we use greedy decoding to maximize efficiency and reproducibility. Additional details regarding the prompt used during inference can be found in Appendix F.

4.1 Results

After a preliminary evaluation of the results, we found that Qwen2.5 VL 3B and 7B outperformed all other models on most test sets. This was surprising, since the benchmark also considers models that are bigger in terms of total parameter count, specifically Qwen2.5 VL 32B and Llama Maverick. Intuitively, one would expect performance to improve with model scale. After validating the outputs, we find that this is caused by the mismatch in answer length and style across models. Specifically, the bigger the difference in length between the label and the generated answer, the more likely it was for ANLS to miss possibly correct answers. This is due to the response style that each model adopted for each instance. We provide a complete example in Appendix E. Hence, we decided to study how label length affects the evaluation process. We split each test set into quartiles according to the length of the ground truth answer. Therefore, Q1 will con-

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.7353	.6257	**.3581**	.1731	.4730	**.8419**	.5765	.1751	.0947	.4221
EN	.8777	.6076	.3853	.1655	.5090	**.8935**	**.6480**	**.4336**	**.2184**	**.5484**
ES	.8476	.6283	**.2823**	.1544	**.4781**	.8615	.6136	.2251	.0990	.4498
FR	.8393	.6365	.3578	.2106	.5111	**.9027**	**.6447**	.2981	.1228	.4921
IT	.7781	.5851	**.3160**	.1596	.4597	**.8269**	.5315	.1704	.0555	.3961
	Qwen2.5-VL-32B-Instruct					gemma-3-4b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.7656	.6515	.3397	.2026	**.4899**	.6848	.4736	.1533	.0345	.3366
EN	.8804	.5647	.2313	.1000	.4441	.6640	.0608	.0000	.0000	.1812
ES	**.8734**	.6209	.2399	.1137	.4620	.6618	.3363	.0148	.0037	.2542
FR	.8249	.6126	**.4098**	**.2401**	**.5219**	.6509	.4173	.1247	.0205	.3034
IT	.8067	.5460	.2299	.1068	.4223	.6519	.1915	.0061	.0052	.2137
	gemma-3-12b-it					gemma-3-27b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.4261	.4999	.3485	.1035	.3445	.7316	.5208	.1821	.0486	.3708
EN	.7536	.1878	.0065	.0000	.2370	.6730	.2290	.0502	.0150	.2418
ES	.7702	.5175	.0908	.0120	.3476	.7151	.4149	.0864	.0146	.3077
FR	.7138	.6267	.3249	.0864	.4380	.7343	.4626	.1978	.0433	.3595
IT	.6914	.4800	.0862	.0217	.3198	.7039	.3776	.0733	.0117	.2916
	Llama-4-Scout-17B-16E-Instruct					Llama-4-Maverick-17B-128E-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.6965	.5940	.3421	**.2213**	.4635	.7972	**.6463**	.2838	.1417	.4673
EN	.8304	.4475	.1905	.0832	.3879	.8783	.5270	.2012	.0788	.4213
ES	.8235	.5900	.2823	**.1837**	.4699	.8529	**.6596**	.2549	.1372	.4762
FR	.7243	.6050	.3105	.2395	.4698	.7697	.5935	.2731	.1011	.4344
IT	.8080	.5577	.2639	**.2009**	.4576	.8247	**.5945**	.3126	.1346	**.4666**
	llama3-llava-next-8b-hf									
	Q1	**Q2**	**Q3**	**Q4**	**AVG**					
DE	.4272	.2666	.0456	.0046	.1860					
EN	.5345	.2158	.0133	.0206	.1960					
ES	.4702	.3373	.0759	.0200	.2258					
FR	.5230	.2792	.0830	.0094	.2236					
IT	.4259	.2171	.0339	.0064	.1708					

Table 5: Complete results for the pre-trained models on the GPT-4o annotated test set.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.8299	.7177	.5585	.4162	.6306	.8414	.7682	.5711	.4625	.6608
EN	.9231	.7059	.5977	.3663	.6483	.9147	.7150	.6528	.4075	.6725
ES	.8633	.8123	.6421	.5357	.7134	.8772	**.8361**	.6807	.5670	.7403
FR	.8751	.7707	.6447	.5172	.7020	.8808	.7974	.6456	.5405	.7161
IT	.8356	.7148	.5981	.4156	.6410	**.8620**	.7367	.6391	.4656	.6758
	Qwen2.5-VL-32B-Instruct					llama3-llava-next-8b-hf				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	**.8508**	**.7972**	**.6026**	**.4966**	**.6868**	.6158	.5389	.3064	.1546	.4039
EN	**.9341**	**.7609**	**.7156**	**.4440**	**.7137**	.7140	.5825	.4249	.2340	.4889
ES	**.8919**	.8298	**.7194**	**.5809**	**.7555**	.6149	.5746	.4685	.2933	.4878
FR	**.9203**	**.8202**	**.6810**	**.5649**	**.7466**	.6429	.5424	.4044	.2717	.4653
IT	.8589	**.7421**	**.6710**	**.5054**	**.6943**	.6403	.4431	.3501	.2188	.4131

Table 6: Complete results for the fine-tuned models on the GPT-4o annotated test set.

		Document VQA		**Scene VQA**		**General**
	Model	DocVQA	InfoVQA	GQA	TextVQA	MME
PT	Qwen2.5-VL-3B-Instruct	.9220	.7185	.6013	.7787	622.5
	Qwen2.5-VL-7B-Instruct	.9358	.8293	.6042	.8295	645.71
	Qwen2.5-VL-32B-Instruct	.9275	.8103	.5924	.7715	713.93
FT	Qwen2.5-VL-3B-Instruct	.9251	.7481	.5625	.7466	611.79
	Qwen2.5-VL-7B-Instruct	.9477	.8179	.5693	.7663	622.86
	Qwen2.5-VL-32B-Instruct	.9350	.8419	.5535	.7629	715

Table 7: Complete results for the SOTA benchmarks.

tain the results for the shortest labels, while Q4 will contain the results for the longest labels. We report complete results for the human-annotated test set in Table 3 and Table 4 for the pre-trained and fine-tuned models, respectively. We report complete results for the GPT-4o annotated test set in Table 5 and Table 6 for the pre-trained and fine-tuned models, respectively. Overall, for pre-trained models, we find that performance is remarkable for the QWEN 2.5 VL models, both in zero-shot inference and with fine-tuning. LLAMA 4 models also perform well, while GEMMA models have the lowest performance, excluding LLAVA which performs worst out of all the models. We also highlight that, due to the previously described issue, models tend to perform better in Q1 and Q2 and worse in Q3 and Q4. Furthermore, given that the labels with the longest length are in Q4, it is effectively the most difficult quartile to match the ground truth answer and the LLM-generated output. Still, it is remarkable that Llama-4-Scout achieves competitive performance on both Q3 and Q4. Despite this, we find that Llama-4-Scout has low performance on Q3 and Q4 for the English language only. Following this, we decided to plot the average performance of pre-trained models over the quartiles for each language. We report complete plots, performance per language and discuss the results in Appendix E. Fine-tuned Qwen models perform best across all test sets. Fine-tuned LLaVA NeXT also improves markedly over its pre-trained version; hence, fine-tuning improves performance regardless of the underlying architecture. To further improve the soundness of our findings, we perform evaluation with BertScore F1 (Zhang et al., 2020) as well, using the evaluate library implementation[3]. We report complete results for BertScore F1 in Appendix E. Overall, the metric follows patterns similar to ANLS, highlighting that current metrics have difficulty evaluating open-ended VQA.

4.2 Results for State of the Art Datasets

We evaluate the fine-tuned models on popular multimodal state-of-the-art benchmarks. We consider the following categories: *Document VQA*, *Scene VQA* and *General*. *Document VQA* refers to tasks where the images are document pages, *Scene VQA* to tasks where the images are natural scenes or everyday images, *General* to comprehensive multimodal benchmarks. For *Document VQA* we use DOCVQA and INFOVQA, for *Scene VQA* we use GQA and TEXTVQA, for *General* we use MME (Fu et al., 2023). MME is a benchmark that measures both perceptual and cognitive abilities across 14 subtasks. We perform evaluation on these benchmarks by using the *lmms-eval*[4] library. We report the main metric used by each task, for MME we report the perception score. Overall, performance either increases or remains stable for *Document VQA* benchmarks, while it becomes worse for *Scene VQA* and *General* ones. The decrease in performance is expected, since the models have been extensively fine-tuned for the document VQA task. However, it is remarkable that the fine-tuned models still achieve strong performance on both the *Scene VQA* and *General* benchmarks. Specifically for *General* benchmarks, the decrease in performance is lower than that for *Scene VQA*, demonstrating that the generalization ability of the fine-tuned LVLM remains remarkable. Furthermore, there is a similar pattern to the one previously explained for our benchmark for state-of-the-art datasets. For the pre-trained Qwen2.5 VL models, performance is inconsistent with model scale across all VQA benchmarks. This further highlights the importance of analyzing and studying the outputs in open-ended VQA tasks.

5 Conclusions

In this work, we proposed **MaDOVE**, the first multilingual document VQA dataset. We consider the English, German, French, Italian and Spanish languages, for which data collection has been performed starting from the *vdr* dataset. After collecting the data and generating synthetic answers, we perform extensive experiments on state-of-the-art LVLMs, considering both *zero-shot inference* for thirteen models from four different families and *fine-tuning*. Results show that, while LVLMs are capable of handling multilingual document VQA, fine-tuning can lead to superior performance. However, we also highlight that extracting the generated answers from zero-shot pre-trained models and comparing them with the ground-truth labels proved difficult for both our dataset and state-of-the-art datasets. In the future, we plan to further extend the dataset to additional languages and study new techniques to improve LVLM's effectiveness for document VQA. We also plan to design and

[3] https://github.com/huggingface/evaluate

[4] https://github.com/EvolvingLMMs-Lab/lmms-eval

implement a metric that accounts for style and language variations in VQA.

Limitations

We recognize the following limitations in our work:

- **Human annotation**: we considered only three of the original dataset's languages, and a single human annotator performed the annotations. Due to the high annotation cost required in this task, since annotating even a single instance can require a lot of time, depending on the amount of text within the image, we were unable to further extend the human annotation process. Hence, we note that the absence of multiple annotators limits the assessment of annotation consistency. Furthermore, LLM validation may affect perceived dataset quality, and agreement metrics should be interpreted with caution due to potential asymmetry in rating bias.
- **Languages covered in the dataset**: while this benchmark does not focus on the English language, it provides instances for only four other languages. However, since we relied on the *vdr* dataset, we would have had to extend it to cover more languages. We plan to pursue this as future work, building on the successful experiments we conducted in this study.
- **Limited cardinality of the test set**: to strike a balance between efficiency and effectiveness, we considered only about 1,000 instances per language in the GPT-4o-annotated test set. We recognize that extending the dataset using the LLM-as-a-judge methodology could further improve the soundness of the results.

Ethics statement

In this work, we have performed manual annotation with a human annotator. The annotator is a woman aged 25-65. The annotator is a native speaker of Italian. The annotator was instructed to prioritize their native language and then proceed only to languages in which they were fluent (English and Spanish). The participant is an employee of Fastweb SpA and performed this task as part of their standard professional duties. Informed consent was obtained regarding the purpose of the study and the utilization of the resulting data. Compensation was provided via the participant's standard employment contract, which adheres to local labor laws and exceeds the national minimum wage. To ensure high-quality data and prevent fatigue, the annotation workload was limited to six hours per day over a one-week period. The *vdr* dataset is on Huggingface under the Apache License 2.0, a permissive license whose main conditions require preservation of copyright and license notices. We will release MaDOVE under the same license.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2425–2433. IEEE Computer Society.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923.

Ali Furkan Biten, Rubèn Tito, Andrés Mafla, Lluís Gómez i Bigorda, Marçal Rusiñol, C. V. Jawahar, Ernest Valveny, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4290–4300. IEEE.

Qingxing Cao, Junhao Cheng, Xiaodan Liang, and Liang Lin. 2024. VisDiaHalBench: A visual dialogue benchmark for diagnosing hallucination in large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12161–12176, Bangkok, Thailand. Association for Computational Linguistics.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. 2023. MME: A comprehensive evaluation benchmark for multimodal large language models. *CoRR*, abs/2306.13394.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20,*

2019, pages 6700–6709. Computer Vision Foundation / IEEE.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 79 others. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, and 191 others. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Byeong Su Kim, Jieun Kim, Deokwoo Lee, and Beakcheol Jang. 2025. Visual question answering: A survey of methods, datasets, evaluation, and challenges. *ACM Comput. Surv.*, 57(10):249:1–249:35.

Jason J Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. 2018. A dataset of clinically generated visual questions and answers about radiology images. *Scientific data*, 5(1):1–10.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 26286–26296. IEEE.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq R. Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2263–2279. Association for Computational Linguistics.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. 2022. Infographicvqa. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022*, pages 2582–2591. IEEE.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. Docvqa: A dataset for VQA on document images. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 2199–2208. IEEE.

Meta AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation.

Aniket Pal, Ajoy Mondal, Minesh Mathew, and C. V. Jawahar. 2025. HW-MLVQA: elucidating multilingual handwritten document understanding with a comprehensive VQA benchmark. *CoRR*, abs/2507.15655.

Nimrod Shabtay, Felipe Maia Polo, Sivan Doveh, Wei Lin, Muhammad Jehanzeb Mirza, Leshem Choshen, Mikhail Yurochkin, Yuekai Sun, Assaf Arbelle, Leonid Karlinsky, and Raja Giryes. 2025. Livexiv - A multi-modal live benchmark based on arxiv papers content. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Fatemeh Shiri, Xiao-Yu Guo, Mona Golestan Far, Xin Yu, Reza Haf, and Yuan-Fang Li. 2024. An empirical analysis on spatial reasoning capabilities of large multimodal models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21440–21455, Miami, Florida, USA. Association for Computational Linguistics.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Jingqun Tang, Qi Liu, Yongjie Ye, Jinghui Lu, Shu Wei, An-Lan Wang, Chunhui Lin, Hao Feng, Zhen Zhao, Yanjie Wang, Yuliang Liu, Hao Liu, Xiang Bai, and Can Huang. 2025. MTVQA: benchmarking multilingual text-centric visual question answering. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 7748–7763. Association for Computational Linguistics.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2022. Hierarchical multimodal transformers for multi-page docvqa. *CoRR*, abs/2212.05935.

Peng Xu, Wenqi Shao, Kaipeng Zhang, Peng Gao, Shuo Liu, Meng Lei, Fanqing Meng, Siyuan Huang, Yu Qiao, and Ping Luo. 2024. Lvlm-ehub: A comprehensive evaluation benchmark for large vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shuo Yang, Caren Han, Siwen Luo, and Eduard H. Hovy. 2025. MAGIC-VQA: multimodal and grounded inference with commonsense knowledge for visual question answering. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 16967–16986. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Synthetic Generation Details

For synthetic generation, we use the Qwen2.5 VL 72B model by loading it on two H100 80 GB VRAM GPUs. We use the vLLM framework[5] to optimize local inference on this model. Generating the long answers for each language in the train set takes at most three days, so a total of fifteen days is required. Generating the short answers for each language in the train set takes at most two days for each language, so a total of ten days is required. Generating the long answers for each language in the test set takes at most two hours for each language, so a total of ten hours is required. Generating the short answers for each language in the test set takes at most one hour for each language, so a total of five hours is required.

B Training Details

We perform fine-tuning with LoRA. We train for one epoch over the entire train split of the dataset, considering both short and long answers. For short answers, we append the strings to constrain short answer generation that are reported in Appendix F. We use the following hyperparameters: 64 effective batch size, 2e-5 learning rate, 0.0 warmup ratio, adamw optimizer. For LoRA we use the following hyperparameters: 8 rank, 16 alpha, 0.05 dropout, targeting all linear modules. All images are resized to 672x672 for efficiency. Fine-tuning is performed on a single H100 80 GB VRAM, except for the 32B model where two H100 were used. For all models, fine-tuning took at most 24 hours in total.

C Error Analysis in Train Set Generation

We show common error patterns that were manually found from the train set of the generated dataset. The two patterns that occurred the most are shown in Figure 3. In the first pattern, the model has provided text by "copy-pasting" the textual contents present in the image. However, while the textual contents are correct, the answer is not complete w.r.t. the contents of the question. In the example provided, the model did not provide the four fundamental aspects of Computational Thinking, which are provided in a later section of the text. In the second pattern, the model fails to generate correct answers when the question is very specific and the visual contents of the images are complex. In the example provided, the model provided the answer "2", which refers to the Standard Group ("Standard-Gruppe" in the image) rather than the Isokinetic Group ("Isokinetik-Gruppe" in the image). We want to underline that these patterns are consistent across all languages included in the dataset. These errors imply that: 1) the Qwen2.5-VL-72B model fails to provide complete and detailed answers when the textual contents of the images are long; 2) the Qwen2.5-VL-72B model is capable of understanding simple layouts and images, but may fail for complex images and specific questions.

D Annotation Details

Both the human annotator and the LLM were provided with the following eight statements to evaluate on a Likert scale in the 1-5 range:

1. The question is grammatically correct.
2. Answer A is grammatically correct.
3. Answer B is grammatically correct.
4. Answer A is coherent with respect to the question.
5. Answer B is coherent with respect to the question.
6. The question is coherent with respect to the image.
7. Answer A is coherent with respect to the image.
8. Answer B is coherent with respect to the image.

In Table 15, we report the guidelines provided to the human annotator.

Screenshots of the annotation platform used by the human annotator are provided in Figure 7 and in Figure 8.

[5] https://github.com/vllm-project/vllm



Quale definizione Jeannette Wing ha dato del pensiero computazionale nel 2006?

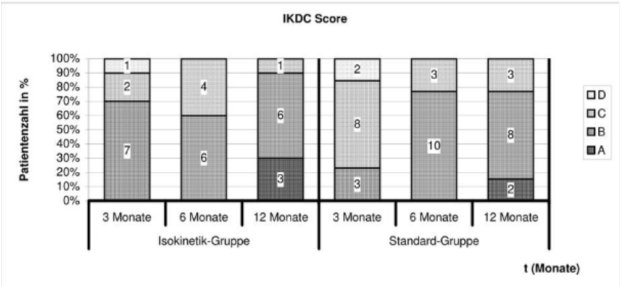
(How did Jeannette Wing define computational thinking in 2006?)

QUESTION

Jeannette Wing ha coniato il termine Computational Thinking (pensiero computazionale) nel 2006 alla Carnegie Mellon University per descrivere una tipologia di approccio alla risoluzione dei problemi. In breve, il pensiero computazionale è un approccio alla soluzione dei problemi.

(Jeannette Wing coined the term "computational thinking" in 2006 at Carnegie Mellon University to describe a particular approach to problem-solving. In short, computational thinking is an approach to problem-solving.)

ANSWER



Wie viele Patienten befanden sich nach 12 Monaten in der Isokinetik-Gruppe in der IKDC Kategorie A?

(How many patients in the isokinetic group were in IKDC Category A after 12 months?)

QUESTION

2

ANSWER

Figure 3: Two error patterns found in manual validation of the generated data. Translation to English of the questions and answers is provided in brackets.

E Additional Results and Examples

We report complete results for **BertScore** in Table 8 and Table 9 for the pre-trained and fine-tuned models on the human annotated test set, while in Table 10 and Table 11 for the pre-trained and fine-tuned models on the GPT-4o annotated test set.

We report per-language results in Figure 4 for the human-annotated test set and in Figure 5 for the GPT-4o annotated test set. These radar chart show that there are models which consistently achieve better performance across all languages. For example, both Llama-4-Scout and Llama-4-Maverick show optimal performance for non-English languages, but tend to perform worse on English (except for the human-annotated test set for Llama-4-Scout). Similar trends can also be seen in other models, for example Qwen2.5 VL 7B overperforms in English but has lower performance for the other languages. Mostly, these charts highlight that model performance across languages and label length is not consistent, and that these factors need to be studied when evaluating multilingual and multimodal models. Furthermore, in terms of average performance across quartiles, the model with the largest scale doesn't always performs best, further highlighting the issue of evaluating pre-trained models due to the style and length of their responses.

We did not consider LLM-as-a-judge as a possible alternative metric since the labels are mostly short answers, while the LLM-as-a-judge methodology was introduced to address long and conversational formats.

We report an example of linguistic style variation across model scale in Figure 6.

F Prompts

We report all prompts used in this work in Table 12, Table 13, Table 14.

The following strings are used in the prompts provided in Table 12 and Table 13.

Negative message strings for answer generation ({NEGATIVE_MESSAGE_LANG_ANSWER}):

- DE: "Es gibt keine Informationen zu dieser Frage im Bild"
- EN: "There is no information regarding this question in the image"

- ES: “No hay información sobre esta pregunta en la imagen”
- IT: “L’immagine non contiene informazioni su questa domanda”
- FR: “Il n’y a pas d’information concernant cette question dans l’image”

Negative message strings for question generation ({NEGATIVE_MESSAGE_LANG_QUESTION}):

- DE: “Auf dem Bild ist kein Text zu sehen”
- EN: “There is no text in the image”
- ES: “No hay texto en la imagen”
- FR: “Il n’y a pas de texte dans l’image”
- IT: “L’immagine non contiene testo”

Finally, the prompt used at training and inference time consists of the question for the instance, followed by a string to force short answer generation (this string is not appended when the answer is a long form one). The short answer generation strings are the following:

- DE: “\nBeantworten Sie die Frage mit einem einzigen Wort oder Satz.”
- EN: “\nAnswer the question using a single word or phrase.”
- ES: “\nResponda a la pregunta con una sola palabra o frase.”
- IT: “\nRispondi alla domanda utilizzando una sola parola o frase.”
- FR: “\nRépondez à la question en utilisant un seul mot ou une seule phrase.”

G Licenses

The dataset will be licensed under Apache License 2.0, that is the original license of the *vdr* dataset.

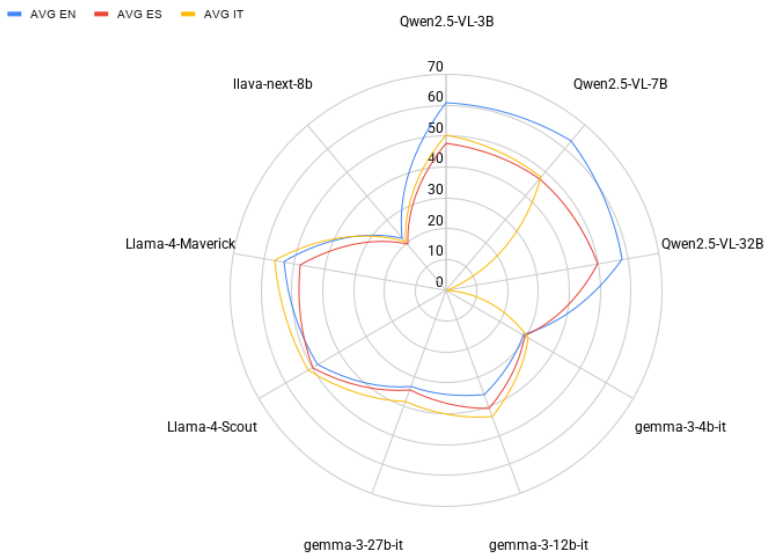


Figure 4: Radar chart for average performance over the quartiles for each pre-trained model on the human-annotated test set.

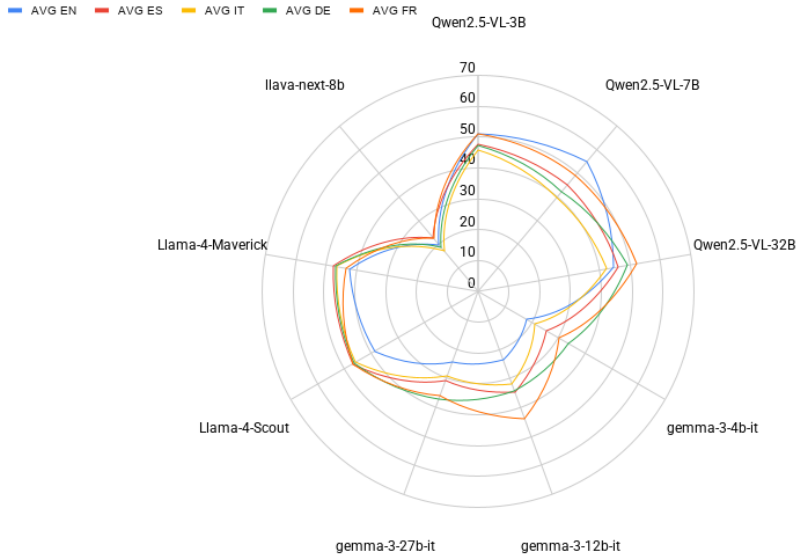


Figure 5: Radar chart for average performance over the quartiles for each pre-trained model on the GPT-4o annotated test set.

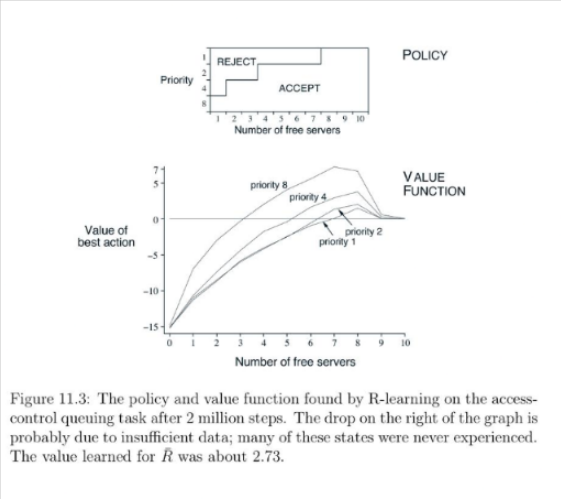


Figure 11.3: The policy and value function found by R-learning on the access-control queuing task after 2 million steps. The drop on the right of the graph is probably due to insufficient data; many of these states were never experienced. The value learned for $\bar{R}$ was about 2.73.

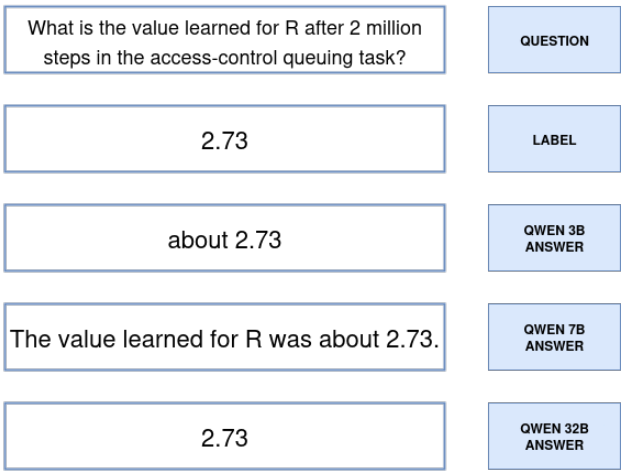


Figure 6: Example of different styles of answer obtained for three models. All answers are correct, but all metrics would provide a higher score to the response that perfectly matches the ground truth.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9881	**.9650**	.9303	.9095	.9482	**.9908**	.9638	**.9387**	**.9123**	**.9514**
ES	.9603	.8991	**.8055**	.7453	**.8525**	**.9695**	.9069	.7838	.7309	.8478
IT	.9652	.8982	.7954	**.7591**	.8545	.9640	.9129	.7711	.6832	.8328
	Qwen2.5-VL-32B-Instruct					gemma-3-4b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9907	.9647	.9291	.8813	.9414	.9778	.9122	.8473	.8283	.8914
ES	.9661	**.9292**	.7765	.7111	.8457	.9186	.8781	.6967	.6250	.7796
IT	.9553	**.9252**	.8043	.7067	.8479	.9234	.8443	.6875	.6208	.7690
	gemma-3-12b-it					gemma-3-27b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9806	.9353	.8745	.8428	.9083	.9742	.9249	.8722	.8418	.9033
ES	.9512	.9105	.7595	.6454	.8166	.9251	.8847	.7332	.6498	.7982
IT	.8991	.9221	.7848	.6783	.8211	.9293	.8774	.7620	.6617	.8076
	Llama-4-Scout-17B-16E-Instruct					Llama-4-Maverick-17B-128E-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9859	.9495	.9081	.8742	.9294	.9865	.9560	.9226	.8808	.9365
ES	.9648	.9004	.7729	**.7689**	.8517	.9500	.9100	.7836	.7294	.8432
IT	.9344	.8969	.7900	.7362	.8394	**.9705**	.9191	**.8311**	.7488	**.8674**
	llama3-llava-next-8b-hf									
	Q1	**Q2**	**Q3**	**Q4**	**AVG**					
EN	.9324	.8980	.8518	.8305	.8781					
ES	.8081	.7721	.6774	.6349	.7231					
IT	.8313	.7640	.6573	.6151	.7169					

Table 8: Complete **BertScore** results for the pre-trained models on the human annotated test set.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	.9927	.9727	.9583	.9408	.9661	.9921	**.9734**	.9614	.9433	.9676
ES	.9793	.9387	.8983	.8707	.9218	.9809	**.9551**	.9039	**.8987**	**.9347**
IT	.9515	.8895	.8612	.8706	.8932	**.9643**	.9142	.8780	.8959	.9131
	Qwen2.5-VL-32B-Instruct					llama3-llava-next-8b-hf				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
EN	**.9961**	.9725	**.9669**	**.9497**	**.9713**	.9733	.9308	.9278	.9159	.9369
ES	**.9859**	.9468	**.9091**	.8966	.9346	.9346	.8677	.8053	.8103	.8545
IT	.9601	**.9274**	**.8861**	**.8988**	**.9181**	.9076	.8511	.8194	.8075	.8464

Table 9: Complete **BertScore** results for the fine-tuned models on the human annotated test set.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.9176	.8860	**.8362**	.7774	**.8543**	**.9544**	.8728	.7362	.7173	.8202
EN	**.9845**	.9462	.9249	.9007	.9391	.9825	**.9557**	**.9346**	**.9123**	**.9463**
ES	.9451	.8745	.7871	.7490	**.8389**	.9523	.8694	.7657	.7117	.8248
FR	.9520	**.8868**	.8193	.7925	**.8626**	**.9663**	.8863	.8018	.7569	.8528
IT	.9354	.8605	.8003	.7419	**.8345**	.9413	.8395	.7293	.6660	.7940
	Qwen2.5-VL-32B-Instruct					gemma-3-4b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.9247	**.8877**	.8227	.7780	.8533	.9119	.8331	.7488	.7038	.7994
EN	.9826	.9388	.9007	.8712	.9233	.9600	.8654	.8360	.8218	.8708
ES	**.9540**	**.8696**	.7764	.7265	.8316	.9011	.7910	.6779	.6330	.7508
FR	.9356	.8774	**.8293**	**.7986**	.8602	.8954	.8079	.7380	.6981	.7849
IT	.9344	.8450	.7544	.7080	.8105	.8934	.7461	.6506	.6132	.7258
	gemma-3-12b-it					gemma-3-27b-it				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.8167	.8483	.8310	.7670	.8157	.9202	.8469	.7596	.7190	.8114
EN	.9675	.8927	.8499	.8263	.8841	.9594	.8955	.8597	.8453	.8900
ES	.9275	.8405	.7308	.6682	.7918	.9205	.8156	.7066	.6491	.7730
FR	.9137	.8743	.8185	.7637	.8426	.9190	.8356	.7735	.7333	.8153
IT	.9093	.8245	.7186	.6637	.7790	.9157	.8038	.7004	.6419	.7654
	Llama-4-Scout-17B-16E-Instruct					Llama-4-Maverick-17B-128E-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.9054	.8778	.8212	**.7871**	.8479	.9343	.8862	.8008	.7556	.8442
EN	.9730	.9221	.8923	.8676	.9138	.9796	.9361	.8951	.8674	.9195
ES	.9369	.8537	.7879	**.7588**	.8343	.9447	.8780	**.7922**	.7326	.8369
FR	.9172	.8671	.8040	.7846	.8432	.9254	.8754	.8023	.7539	.8392
IT	.9258	.8409	.7694	**.7425**	.8197	**.9460**	**.8633**	**.7939**	.7331	.8341
	llama3-llava-next-8b-hf									
	Q1	**Q2**	**Q3**	**Q4**	**AVG**					
DE	.8132	.7353	.6521	.6384	.7098					
EN	.9271	.8815	.8444	.8349	.8720					
ES	.8135	.7467	.6937	.6485	.7256					
FR	.8299	.7551	.6842	.6656	.7337					
IT	.8095	.7245	.6534	.6197	.7018					

Table 10: Complete **BertScore** results for the pre-trained models on the GPT-4o annotated test set.

	Qwen2.5-VL-3B-Instruct					Qwen2.5-VL-7B-Instruct				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	.9507	.9183	.8859	.8480	.9007	.9511	.9347	.8970	.8580	.9102
EN	.9869	.9637	.9563	.9319	.9597	.9869	.9662	.9609	.9373	.9628
ES	.9506	.9305	.9065	.8806	.9170	.9592	.9408	.9207	.8895	.9275
FR	.9605	.9332	.9062	.8779	.9194	.9643	.9420	.9074	.8823	.9240
IT	.9549	.9052	.8897	.8494	.8998	.9602	.9109	.9045	.8678	.9108
	Qwen2.5-VL-32B-Instruct					llama3-llava-next-8b-hf				
	Q1	**Q2**	**Q3**	**Q4**	**AVG**	**Q1**	**Q2**	**Q3**	**Q4**	**AVG**
DE	**.9543**	**.9388**	**.9017**	**.8709**	**.9164**	.8938	.8509	.8066	.7716	.8307
EN	**.9895**	**.9711**	**.9674**	**.9422**	**.9676**	.9640	.9432	.9343	.9146	.9390
ES	**.9641**	**.9430**	**.9319**	**.8912**	**.9325**	.8941	.8594	.8587	.8178	.8575
FR	**.9744**	**.9531**	**.9185**	**.8866**	**.9331**	.9028	.8560	.8415	.8070	.8518
IT	**.9605**	**.9212**	**.9093**	**.8766**	**.9169**	.8961	.8253	.8150	.7870	.8308

Table 11: Complete **BertScore** results for the fine-tuned models on the GPT-4o annotated test set.



Figure 7: Screenshot of the annotation platform. The user is shown the image, the question and the answers.

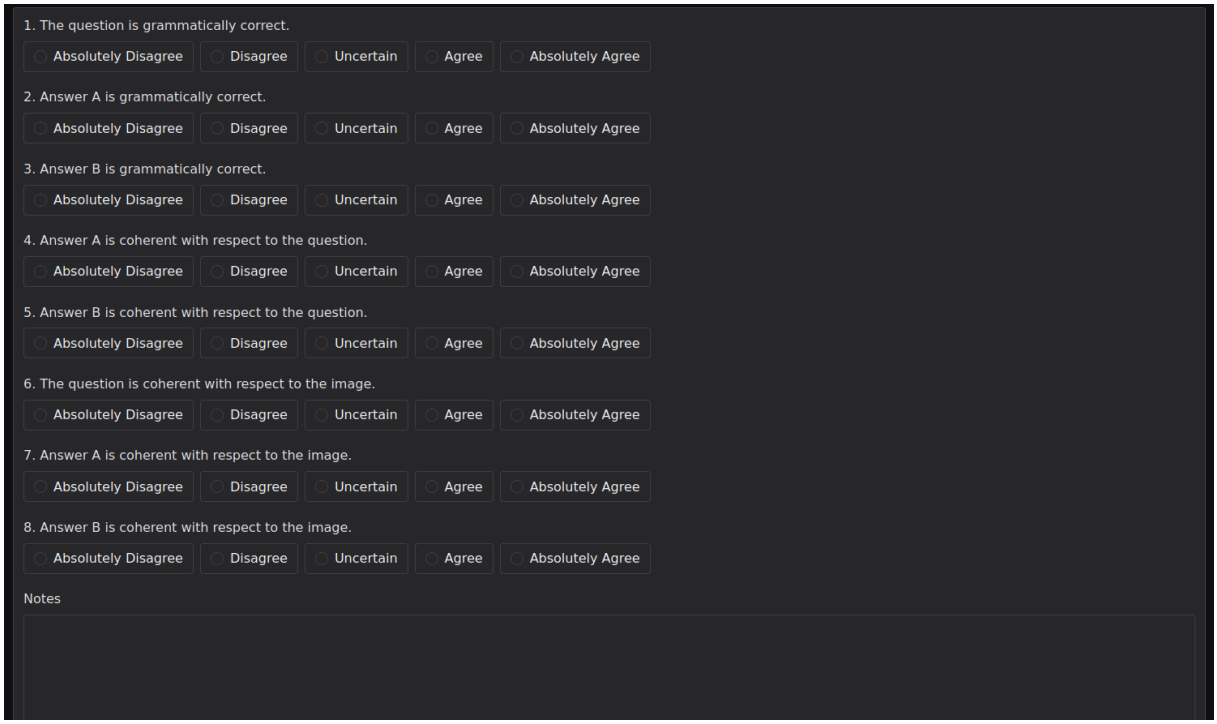


Figure 8: Screenshot of the annotation platform. The user is shown the eight statements to evaluate and a notes section to provide an explanation for their ratings.

<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
<|vision_start|>placeholder<|vision_end|>Provide an answer in {LANG} to the given question using the textual content in the image. If the answer is not in the image, just respond with "{NEGATIVE_MESSAGE_LANG_ANSWER}". {QUESTION}
Answer the question using a single word or phrase.<|im_end|>
<|im_start|>assistant

Table 12: Prompt used for the generation of short and long answers. {LANG} is one of "English", "German", "French", "Italian", "Spanish". {NEGATIVE_MESSAGE_LANG_ANSWER} is a negative message generated when the model cannot answer the question. {QUESTION} is a placeholder for the query. Note that the string "Answer the question using a single word or phrase." is not present in the prompt when performing long answer generation.

<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
<|vision_start|>placeholder<|vision_end|>Write a question in {LANG} that relates to the text in the image. If the image contains no text, respond with "{NEGATIVE_MESSAGE_LANG_QUESTION}".
<|im_start|>assistant

Table 13: Prompt used for generation of questions for images in the *vdr* dataset without a query. {LANG} is one of "English", "German", "French", "Italian", "Spanish". {NEGATIVE_MESSAGE_LANG_QUESTION} is a negative message generated when the image doesn't contain text.

1. The question is grammatically correct.
2. Answer A is grammatically correct.
3. Answer B is grammatically correct.
4. Answer A is coherent with respect to the question.
5. Answer B is coherent with respect to the question.
6. The question is coherent with respect to the image.
7. Answer A is coherent with respect to the image.
8. Answer B is coherent with respect to the image.

Given these 8 statements, provide a rating in a range for 1 to 5 (where 1 is "Absolutely Disagree", 2 is "Disagree", 3 is "Uncertain", 4 is "Agree" and 5 is "Absolutely Agree") for each of them. Answer A will be longer, while answer B more concise. The question and answers are the following:

Question: {QUESTION}
Answer A: {ANSWER_A}
Answer B: {ANSWER_B}

If you give a negative rating for any of the statements, provide a short explanation in English of a possible solution to fix the question or the answers. If the question or answers cannot be fixed, answer with "The data is impossible to fix.".

Table 14: Prompt used for the LLM-as-a-judge methodology. Note that, while the special tokens for the images are not present in this prompt, the GPT-4o model is still provided with the image at inference time. {QUESTION} is a placeholder for the question, {ANSWER_A} is a placeholder for the long answer and {ANSWER_B} is a placeholder for the short answer.

In this task, you will be given: - One image: the given image will contain text in the annotation language - One question: the given question will reference the text that is present in the image - Two answers: two possible answers for the given question. Answer A will contain a longer and more descriptive text, while answer B will be direct and concise. Given these resources, you will have to evaluate eight statements (with ratings ranging from "Absolutely Disagree" to "Absolutely Agree"). You have to evaluate the quality of questions and answers both for grammar and logical coherence between each other. Finally, you have to evaluate the logical coherence of the questions and the answers with respect to the provided image. For "Absolutely Disagree", "Disagree" and "Uncertain" ratings, please add a short note in English regarding your annotation, describing how the instance can be fixed (e.g. "Answer A is correct, but Answer B contains a different date w.r.t. Answer A"). Finally, consider these additional guidelines: - A "Disagree" rating should be given when the information is incomplete and unsatisfactory. For example, if Answer A doesn't mention all relevant aspects in the image that are needed to answer the question. - An "Agree" rating should be given when the information is incomplete and satisfactory. For example, if Answer A mentions everything that is needed to answer the question, but there are some additional aspects that could be covered to enhance the quality of the answer.

Table 15: Complete guidelines given to the annotator for manual annotation of the dataset instances.

Generating Edit-Inducing Questions for AI Research Manuscripts

Sebastian Joseph[1] **Zichao Wang**[2] **Jennifer Healey**[2] **Alexa Siu**[2]
Junyi Jessy Li[1] **Ani Nenkova**[2]
[1]The University of Texas at Austin, [2]Adobe Research
{sebaj,jessy}@utexas.edu, {jehealey,nenkova}@adobe.com

Abstract

We study the ability of LLMs to generate *edit-inducing questions* whose answer will improve a paper draft. On a dataset of paired ICLR/NeurIPS submission and camera-ready papers, we compare the helpfulness of questions from GPT models with or without full paper context to that of human reviewers. GPT produces more edit-inducing questions and its questions are associated with more extensive edits and cover a broader range of edited content compared to questions from reviewers. However, a much smaller percentage of the GPT questions are edit-inducing. Our analyses confirm that automated questions can be beneficial to authors and highlight an example task where proper attending to long context deteriorates reasoning model ability to produce helpful output.

1 Introduction

Ineffective scientific writing is often due to the curse of knowledge: the authors know a lot about their problem and the details of their work but find it hard to imagine that their reader does not have this knowledge (Pinker, 2014). A practical remedy is to request feedback on drafts, to find out what questions an intelligent and curious reader may have about the presented content.

In academic practice, peer reviewers are often expected to provide constructive feedback to help authors improve the work itself as well as its presentation, in addition to their evaluation of novelty, soundness and significance. With the exponential growth of academic publications straining the peer review process, even this delayed feedback is harder to obtain (Stelmakh et al., 2020; Hanson et al., 2023; Kim et al., 2025). This reality has motivated a line of work on automatic feedback generation, enabled by peer review datasets (Kang et al., 2018): from early knowledge-graph and template-based systems (Wang et al., 2020) to

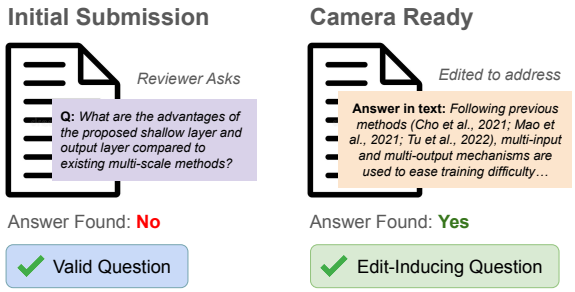


Figure 1: We seek to generate *edit-inducing questions* like the example shown above. A question is asked based on content in the initial submission. When answered only in the camera ready version, we can expect content to have been edited to address such a question.

LLM-based approaches spanning error detection (Liu and Shah, 2023), large-scale empirical validation (Liang et al., 2023), and multi-agent feedback generation (D'Arcy et al., 2024). Most recent work has shown that peer reviews can be mined to extract data for training models specifically designed to provide actionable feedback, with no evaluative elements. Automatic feedback has been deployed at scale and found useful and actionable by authors and organizers alike (Biswas et al., 2026).

Our interest is also in feedback generation, particularly *in the form of questions* over a current draft. Specifically we ask: *(i) are the questions asked by human reviewers and LLMs edit-inducing? (ii) what fraction of author edits do such questions account for?* In other words, we quantify objectively through author decisions the quality of questions generated by LLMs and humans.

Our analyses are carried out on pairs of initial submission and final version of conference papers. By comparing the two versions, we pinpoint what content was added by the authors, presumably to improve the paper. Furthermore we measure whether questions asked by human peer reviewers, from the PeerQA dataset (Baumgärtner et al., 2025), and questions generated by LLMs target

The Proceedings of Workshop on Document Intelligence and Understanding, pages 428–439
October 24, 2026 ©2026 Association for Computational Linguistics

that content by searching for their answers in the papers.

We find that LLMs have real potential as a source of edit-inducing questions. Most notably and unsurprisingly, they generate an order of magnitude more edit-inducing questions than the human reviewers. As a result of the increased volume of questions, their answers account for a larger percentage of edits made by the authors between their submitted and final paper versions. We also find that model questions correspond to more extensive edits than questions from peer reviewers.

2 Data & Methodology

We perform our analyses of edit-inducing questions on a subset of the PeerQA dataset (Baumgärtner et al., 2025).

Data PeerQA contains questions extracted from peer reviews of machine learning and natural language processing papers, along with answers of these questions grounded in the final version of the paper. For 116 papers that were sourced from three venues hosted on OpenReview (ICLR 2022, ICLR 2023, NeurIPS 2022), we expanded the original dataset, retrieving the initial submission of the paper and the final camera-ready version of the paper via the publicly accessible submission history. We extract the full text using GROBID.

Question Generation Each paper is split into 512 token segments. Figures and tables are considered their own paragraphs; segments are built by continually adding paragraphs until adding the next paragraph would exceed 512 tokens. A new segment is created at section boundaries to prevent text from multiple sections appearing in a single segment. We pass each segment from the submitted version to GPT-4o and o3 with the prompt shown in the Appendix A, with varying levels of context, to obtain insightful questions that can arise while reading that segment (Wu et al., 2023, 2024). The three context settings we evaluate here are: providing the full paper as context (**full**), providing every preceding segment to simulate linear reading (**linear**), and providing only the focused segment as context (**segment**).

Table 2 in Appendix C shows the number of questions generated by each model+context combination, along with the number of questions from peer reviews of the same papers. On average, there are three questions from peer reviewers per paper; the automatically generated questions are many more, ranging from about 140 on average per paper for GPT-4o segment to 206 for o3 segment.

Answer Detection For each question, we use the approach from the PeerQA paper to find if an answer exists in the submitted and camera-ready versions of the paper. We use the OpenAI `text-embedding-small` to find the five paper segments most similar to the question. Then we prompt `gpt-4o` (see Appendix B) to decide if any of these five segments answers the question.

3 Measuring Edit-Inducing Capability

We define four measures of question helpfulness, with higher values indicating more helpful properties. These measures were calculated per paper and are presented as an average over all papers.

Valid questions Both peer reviewers and GPT ask questions that are already answered in the submitted manuscript. We call a question *valid* if it is not already answered in the submitted manuscript. The percentage of valid questions reflects the ability of the LLM to take the context of the entire paper into account when asking a question.

Edit-inducing questions Edit inducing questions are a subset of valid questions. These are valid questions that the authors answer in the final version. Assuming all changes are made with the intention to improve the paper, edit-inducing questions are empirically verified to have value as actionable feedback on a manuscript draft. The valid questions that are not edit-inducing remain unanswered. Appendix E presents randomly drawn edit-inducing and unanswered questions. An auxiliary measure is "Edit-inducing answer rate", equal to the percentage of answered questions that are answered only in the final version.

Edit rate characterizes a single edit-inducing question. It measures the extent to which the text in the related segment is changed to provide the answer. For an edit-inducing question and a corresponding segment from the final submission that contains its answer, it equals to the proportion of *edited* (added or substituted) characters in that segment with respect to the total number of characters that were originally in the segment of the submitted manuscript. Appendix G shows examples of segment changes and the respective edit rate values.

Edit coverage characterizes a set of edit-inducing questions, here the set of all edit-inducing questions generated by a given source. It equals the fraction of edited characters attributable to all edit-inducing questions out of all characters that changed between the two drafts. Its value reflects the extent to which the answers to all edit-inducing questions cover the entire content authors found worth adding to their papers.

4 Results

Figure 2 shows the quality metrics for the sets of questions generated from peer reviews, and by GPT-4o and o3 with different levels of context.

4.1 Human Peer-Review Questions

We present two evaluations of the human questions from the PeerQA dataset. They vary in how answer detection is done for the camera-ready version. In the PeerQA-GT setting, the answer is found in the paper based on the ground truth, human annotated answers in the dataset.

The other setting, PeerQA, we apply the automated answer detection pipeline. Both settings used the automated pipeline to find answers in the initial submission as there is no corresponding ground truth. Both show similar results, albeit with the automated pipeline having weaker recall of answered questions.

Around two-thirds (65%) of the peer review questions are not answered in a submitted paper, in other words 35% of the human questions were already answered in the submitted version.

Roughly one-third (38%) of human peer-review questions for a paper are edit inducing. In number of questions, this translates on average to around 1 such question per paper. Therein lies the primary issue with these questions, which is their scarcity. This small amount of questions also explains the low edit coverage, with edit-inducing questions from peer reviews covering, on average, only around 4% of all edits made in the paper. Authors, in their edits, incorporate a lot more feedback beyond what was addressed by their peer reviews.

4.2 GPT Edit-Inducing Questions

Type of model and context make minor differences in metrics. Overall either setting can be used and all compare similarly to human questions. The GPT models generate markedly more total questions, and, as a result, generate a magnitude more edit-inducing questions, between 11 and 19 questions per paper. However, as a percentage, only around 7-10% of all questions for a paper were edit-inducing. The models over-generate questions, lacking strong discriminative capability to restrict themselves from asking questions that lack utility.

The models also ask more valid questions as a percentage compared to human peer-reviewers. They seem to have a better ability to recall from the context the type of questions that could already be answered by the paper. The non-reasoning model has a minor edge over the human question, and barely any difference in percentage valid questions with varying context. The reasoning model however makes good use of the paper context and generates considerably larger percentage of valid questions than humans do, when provided with the full paper as context. Larger context settings also tend to yield a higher proportion of valid questions. However, even when given only a document segment, models are capable of asking questions that are unanswerable by the rest of the paper.

Edit inducing answer rate numbers closely track the percentage valid questions. It is a redundant measurement, with all conclusions aligned with those from percentage valid.

The edit rate for GPT questions is higher than that for human questions, indicating the questions correspond to larger spans of edited content. The edit coverage of GPT questions is much higher than that for peer questions. GPT generates larger number of edit inducing questions capable of targeting a wider range of improvements.

4.3 Copying Behavior

In terms of edit-inducing ability, the questions generated from the reasoning model appear very much on par with those from the non-reasoning model. However in qualitative inspection, as in the samples shown in Appendix E, we noticed that 4o questions appear more extractive, often paraphrasing the content being discussed in the form of a question and repeating many words from the segment for which the questions were generated. In that respect, they appear to be worse than those generated by o3. To quantify the difference, we compute the average proportion of words in a question that overlap with words in the span of text from the paper that elicited that question.

Table 1 shows the results for each model and context combination. We see a significant increase in this word overlap with questions generated by

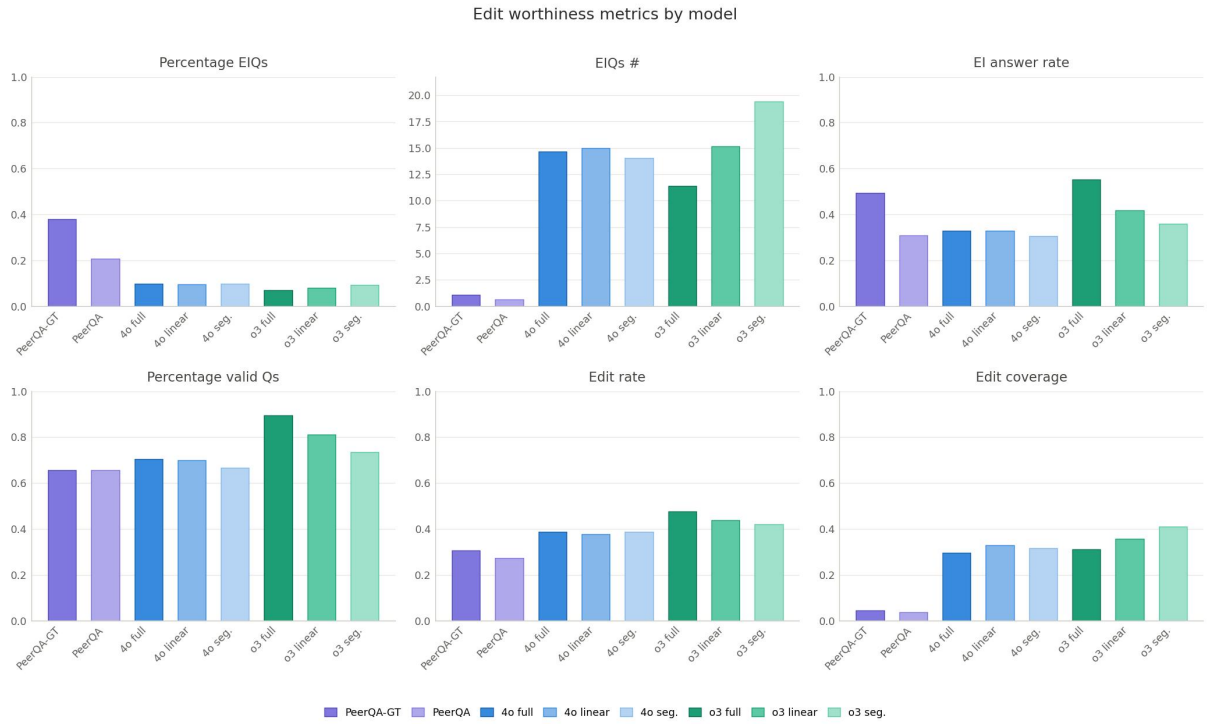


Figure 2: Measures of edit worthiness of questions asked by peer reviewers (PeerQA) and GPT models.

Name	Overlap Proportion
o3	0.2337
o3 Linear	0.2337
o3 Segment	0.2306
GPT-4o	0.3020
GPT-4o Linear	0.3069
GPT-4o Segment	0.3337

Table 1: Proportion of words in the question overlapping with words found from the span of text that elicited it

GPT-4o, indicating a higher degree of extractive behavior in how the model develops its questions.

5 Discussion and Future Work

A key issue that we highlighted with LLM-generated questions was the low proportion of edit-inducing questions. Authors would likely prefer to engage with a higher rate of such questions. Pruning down the generated questions effectively and cheaply remains an open problem for future work. We made the assumption that questions that are edit-inducing are more preferable for feedback. A future experiment with human subjects may probe the reasons why certain questions were not answered. It is possible that there were good questions that were not answered through edited content because manuscript page limits prevent authors from including all details, or because they did not have time to answer an otherwise good question. This paper highlights the concept of edit-inducing questions as a signal for useful feedback. Ultimately, developing a single model capable of generating all edit-inducing questions would be ideal for practical feedback to authors.

6 Conclusion

We present a study that confirms that large language models may benefit writers of papers in the artificial intelligence area, by serving as an indefatigable reader, asking questions that arise from reading segments of the paper. Models produce a order of magnitude more questions than peer reviewers, with a much larger number of edit-inducing questions. Individual questions from LLMs have higher edit-rates than peer questions, and all automatically produced questions account for about half of all the edits that authors made between their initial submission and the final version. Reasoning models are better at generating valid questions, even when not taking the full paper as context when producing the questions; their output feels qualitatively better, objectively less extractive in wording than non-reasoning models.

Limitations

Our experiments are based on a single family of closed models, GPT. However, we were able to vary context and type of model for a complete set of experiments. Inclusion of more models would have resulted in overwhelming tables. Choosing one model makes it easy to see that most of the modeling choices make no difference.

The assessment of edit inducing capability in this paper utilizes automated answer detection extensively. However, as we show in Appendix D, this automated system is effective in detecting answers, especially in terms of precision. Thus results we present lead to reasonable estimates of the potential utility from LLM generated questions. The comparison with gold standard human labels of answers indicates that if anything, the percentage of edit inducing questions generated from LLMs may be higher in practice than the ones we report here.

Because of the constraints of using the PeerQA dataset, we only looked at papers that were accepted in the same venue that they were submitted in. Applying a similar analysis over rejected papers that were accepted in a future venue would be of interest for future work.

References

Tim Baumgärtner, Ted Briscoe, and Iryna Gurevych. 2025. Peerqa: A scientific question answering dataset from peer reviews. *North American Chapter of the Association for Computational Linguistics*, arXiv:2502.13668.

Joydeep Biswas, Sheila Schoepp, Gautham Vasan, Anthony Opipari, Arthur Zhang, Zichao Hu, Sebastian Joseph, Matthew Lease, Junyi Jessy Li, Peter Stone, and 1 others. 2026. Ai-assisted peer review at scale: The aaai-26 ai review pilot. *Preprint*, arXiv:2604.13940.

Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *arXiv.org*, arXiv:2401.04259.

Mark A. Hanson, Pablo Gómez Barreiro, Paolo Crosetto, and Dan Brockington. 2023. The strain on scientific publishing. *Quantitative Science Studies*, 5(4):823–843.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. *North American Chapter of the Association for Computational Linguistics*, arXiv:1804.09635.

Jaeho Kim, Yunseok Lee, and Seulki Lee. 2025. Position: The ai conference peer review crisis demands author feedback and reviewer rewards. *International Conference on Machine Learning*, arXiv:2505.04966.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Smith, Yian Yin, and 1 others. 2023. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, arXiv:2310.01783.

Ryan Liu and Nihar B. Shah. 2023. Reviewergpt? an exploratory study on using large language models for paper reviewing. *arXiv.org*, arXiv:2306.00622.

Steven Pinker. 2014. *The sense of style: The thinking person's guide to writing in the 21st century*. Penguin Books.

Ivan Stelmakh, Nihar B. Shah, Aarti Singh, and Hal Daum'e. 2020. A novice-reviewer experiment to address scarcity of qualified reviewers in large conferences. *AAAI Conference on Artificial Intelligence*, arXiv:2011.15050.

Qingyun Wang, Qi Zeng, Lifu Huang, Kevin Knight, Heng Ji, and Nazneen Fatema Rajani. 2020. Reviewrobot: Explainable paper review generation based on knowledge synthesis. In *International Conference on Natural Language Generation*, page 384–397. Association for Computational Linguistics.

Yating Wu, Ritika Mangla, Alexandros G. Dimakis, Greg Durrett, and Junyi Jessy Li. 2024. Which questions should I answer? salience prediction of inquisitive questions. In *Conference on Empirical Methods in Natural Language Processing*, pages 19969–19987. Association for Computational Linguistics.

Yating Wu, Ritika Mangla, Greg Durrett, and Junyi Jessy Li. 2023. QUDeval: The evaluation of questions under discussion discourse parsing. In *Conference on Empirical Methods in Natural Language Processing*, pages 5344–5363. Association for Computational Linguistics.

A Question Generation Prompt

Prompt A

```
System: You are logical, intelligent, insightful, precise, and can understand the contents of research papers. You are
knowledgeable on different fields and domains of science and engineering. You are able to interpret research papers,
create questions and answers, and compare multiple aspects.

Imagine the following scenario: You are reviewing a submitted research paper manuscript. After you are done reading, you
have questions you want to ask the authors. You are a very intelligent reader so you don't ask questions that are already
answered within the paper.

I will provide you two things: First, context which you have already read, and a chunk of text that you are currently
reading.

This is your task. First, you need to look at the context that is provided to you. This is information that you have
already read and aware of. Please don't ask questions that can be answered through this context. What you need to do
instead is generate all possible follow-up questions arising from the provided chunk of text that you are currently
reading. These follow-up questions must be insightful and highlight a critical gap of information that a reader would
desire to know while reading.

You MUST follow these rules when creating these questions:
(1) The question must not be answered by text in the context. Make sure to thoroughly read the text in the context and
craft a question that cannot be answered by it.
(2) These questions must be questions that a smart reader is naturally thinking about when reading this chunk of text.
Remember, they also have already read the context as well, so they will not be thinking of questions that they already
know the answer to. Put yourself in this smart reader's shoes when crafting these questions.

I will provide you with a typology of these questions below. You can use this typology to come up with more diverse
questions.

Typology:
  - Content:
    - content-clarification: questions that try to clarify something that you've read
  - Insights:
      - insights-more: questions that ask for more insights
      - insights-nuance: questions that ask for more nuance in the insights
      - insights-soundness: questions that ask whether the insight is sound.
  - Measurement:
      - measurement-more: questions that ask for more metrics to evaluate in experiments
      - measurement-detail: questions that ask for more detail about some measurement/metrics
      - measurement-alternative: questions that ask why an alternative measurement wasn't used
  - Method:
      - method-alternative: questions that probe at an alternative method that has a similar effect
      - method-detail: questions that probe for more detail about the method
      - method-motivation: questions that probe at the motivation for using such a method

When you respond, first think really hard, step-by-step, about every span of text in the provided chunk, and think about
what would be good, thoughtful, and interesting follow-up questions according to the above rules.

Then you need to provide me with these follow-up questions and also tell me the exact span of text each follow-up
question corresponds to, as in follows up on, in your response. Absolutely make sure that the span you provide is an
exact substring of the chunk text, and that it is not a summary or paraphrase. It should be the exact text corresponding
to the question.

If you can't come up with any follow-up questions at all (which is perfectly fine), just return None.

### REASONING PROCESS
  <Explain your reasoning and thinking process>

### JSON
  {
      "follow_up_questions": [
                          {
                              "question": "<generated follow-up question>",
                              "span": "<exact span of text in the provided chunk corresponding to this question>"
                              "question_type": <Content/Insights/Measurement/Method>,
                              "question_subtype": "<content-clarification/insights-more/insights-nuance/insights-soundne
                              ss/measurement-more/measurement-detail/measurement-alternative/method-alternative/method-d
                              etail/method-motivation>"
                          },

                          ...
                      ] or None
  }

User:

[Start of Context]
{context}
[End of Context]

[[Start of *Provided Chunk*]]
{segment}
[[End of *Provided Chunk*]]
```

B Answerability Prompt

Prompt B

```
System: You are logical, intelligent, precise, and can understand the contents of research papers. You are knowledgeable
on different fields and domains of science and engineering. You are able to interpret research papers, and determine
whether information inside these papers can answer some question.

This is your task: Read the following several chunks from a paper and answer whether the question can be answered by one
or more of these chunks. If the question can be answered, report the chunk numbers of the chunks where you found the
answer and answer the question in a sentence or two using only the information in that chunk(s).
```

```
When you respond, first think really hard, step-by-step, about every sentence in every chunk, and analyze whether this
information can answer the question provided.

You can definitely find the question to be unanswerable. You can only find a question to be answerable if ONLY the
information within the provided chunks fully answers this question.

If you find the question to be answerable, please select the fewest possible chunks you would need to answer the
question. If there is one chunk that can fully answer this question, please select only that chunk. If there are multiple
chunks that can fully answer this question, please select only the one chunk you think best answers the question. You
should only select multiple chunks if the only way to fully answer the question is by combining the information in these
chunks.

The chunk numbers are 1-indexed, meaning the first chunk is chunk number 1, the second chunk is chunk number 2, and so
on. When you recieve the chunks, you will know the chunk number as it appears like this: "CHUNK #<chunk_number>". Please
only report the exact number, and only the number, of the chunks you selected. Do not report a chunk number that is not
in the range of possible chunk numbers. The chunk number **should not be less than 1** nor should it be greater than the
maximum chunk number. This is unforgivable.

In addition to selecting the chunks, you also need to provide the exact span of text within the chunk that answers the
question. This span should be a substring of the chunk text that directly addresses the question. Absolutely make sure
that the span you provide is an exact substring of the chunk text, and that it is not a summary or paraphrase. It should
be the exact text that answers the question.

You need to provide me with an binary Yes or No as the answer to whether the question is answerable by the provided
chunks. If answerable, provide me with the list of chunk numbers corresponding to your selected chunks. I also require
you detail the exact span of text within the chunk that answers the question. If unanswerable, this can be an empty list.

If answerable, provide me with a 1-2 sentence answer. Otherwise, just answer with an empty string.


### REASONING PROCESS
<Explain your reasoning and thinking process>

### JSON
{
  "is_answerable": "<Yes/No>",
  "selected_chunks": [
    {
      "chunk_number": <chunk number>,
      "span": "<exact span of text within the chunk that answers the question>"
    },
    ...
  ],
  "answer": "<1-2 sentence answer, or empty string if unanswerable>"
}

User:

[Start of Provided Chunks]
{answer_batch}
[End of Provided Chunks]

[Start of Question]
{question}
[End of Question]
```

C Total number of generated questions

Table 2 Lists the total, average, minimum and maximum number of questions generated by each source. Regardless of model and context, GPT generate many more questions than peer reviewers do. 4o and o3 exhibit a different pattern as the context length changes. 4o produces fewer questions when presented only with the segment, compared to having the full paper as context as well. On average, the full paper context yields 15 more questions per paper. The reverse trend happens for o3: it produces 206 question on average when presented with the segment only but only 163 questions when the full paper is provided as well. The variation of number of questions is much bigger for o3, 40 question difference on average.

It is worth noting that the papers themselves modulate the number of produced questions. There is a tenfold difference in the number of questions for the paper with fewest questions compared to that with most questions.

D Assessing Answerability

Table 3 shows the precision and recall of the automated answerability pipeline evaluated on the PeerQA gold standard data. Precision is quite good but an existing answer is often not detected.

As for five segment cutoff, providing the entire document to the answerability model to answer every question we evaluate is prohibitively expensive. Providing only the top 5 segments proved to be a worthy tradeoff to ensure experiments stayed within budget while capturing the vast majority of existing answers. Preliminary tests showed that the ground truth answer was contained in the top 5 answer segments 72% of the time, which was reasonable enough to us to believe in the reliability of this pipeline under such a cutoff.

Model	Total Qs	Avg Qs/ Paper	Max Qs	Min Qs
PeerQA	362	3.12	11	1
4o full	17,734	152.88	320	38
4o linear	18,158	156.53	321	32
4o segment	16,018	138.09	416	28
o3 full	18,940	163.28	433	47
o3 linear	21,339	183.96	455	43
o3 segment	23,906	206.09	586	45

Table 2: Number of questions by source.

Metric	Score
Precision	0.836
Recall	0.586
F1	0.689

Table 3: Precision, recall, and F1 @ 5 (among the top 5 answers) for `gpt-4o` deciding if a segment is an answer to a given question. Evaluated against ground truth PeerQA answers.

E Edit-inducing and unanswered questions

Table 4 below shows random examples of edit-inducing and unanswered questions produced by different model and context settings.

Table 4: Side-by-side comparison of edit-inducing and unanswered questions.

Edit-Inducing	Unanswered
Model: ***o3***	
METHOD *detail* Could you explicitly show how a single layer of a k-GNN is encoded as a TL^(t)_{k+1}(?) expression, in particular how the higher–order neighborhood aggregation and the permutation–invariant updates are handled within only k+1 index variables?	METHOD *detail* FIND_HP requires its input point to be δ-non-degenerate, yet Step 1 only guarantees that x_i is a critical point. What mechanism or additional test ensures that each x_i returned by FIND_CP actually satisfies the δ-non-degeneracy condition needed by FIND_HP?
INSIGHTS *nuance* Why do poisoned and benign frames stay close in the SiamFC++ feature space whereas they separate in SiamFC and SiamRPN++? Could you elaborate on what architectural or training differences lead to this behavior?	METHOD *detail* You refer to an additional invariant layer that has a "similar (simpler) representation as given in equation 1 (Maron et al., 2019c)". Could you provide the explicit TL representation of this invariant layer and explain why it is simpler?
METHOD *detail* For the min-max and z-score baselines, are the normalization statistics computed per instance, per batch, or over the whole training set, and are they recomputed during inference?	METHOD *detail* How exactly are the adversarially generated 'confusing samples' produced (e.g., number of optimization steps, step size, loss function, and whether this generation is performed on-the-fly during training or pre-computed)?
INSIGHTS *soundness* On what empirical or theoretical grounds do you characterise δ-regularity as a "mild" general position assumption? Have you measured how frequently trained networks satisfy δ-regularity without the artificial perturbation you propose?	METHOD *detail* The sentence states that edge weights depend on both distance and direction, yet the Gaussian kernel you provide only uses the squared distance. How is the direction information actually incorporated into the adjacency weights?
MEASUREMENT *more* Can you provide ablation results quantifying the benefit of the parallel exploration relative to a single-process baseline to substantiate the claim of "efficient exploration"?	METHOD *detail* You claim HyperDQN measures uncertainty through "infinite" ensembles, but in implementation only finitely many z-samples can be used. How many z vectors are actually drawn per training step and per episode, and how did you decide on this number?

Table 4 *(continued)*

Edit-Inducing	Unanswered
INSIGHTS *nuance* Remark 1 suggests β_k=0 causes stagnation; is there theoretical guidance on how small β_k can safely be before convergence degrades, and how does this threshold relate to the contractivity constant of g?	MEASUREMENT *more* Could you also report confidence intervals for the performance differences to complement the p-values and provide an estimate of effect size?
METHOD *detail* Equation (18) requires the ratio d_{π*_r}(s,a)/d_D(s,a), which itself depends on V* and ?r; could you clarify the computational procedure for obtaining this ratio at policy-extraction time, especially in high-dimensional continuous spaces?	METHOD *detail* Since Barlow Twins forces each dimension to have norm $\sqrt{N}$, does varying the batch size during training (e.g., in distributed settings) introduce instability or require re-scaling of other loss terms?
MEASUREMENT *detail* For the Sensitivity-n experiments in Figures 10 and 11, what exact subset sizes (n or percentage of pixels) were evaluated, and how many random subsets were sampled for each n?	METHOD *detail* How exactly is the 'identity' initialization implemented for the depth-wise convolution inserted between grown convolutional layers (e.g., kernel size, padding, and channel mapping)?
METHOD *detail* What specific mechanisms (e.g., different initializations, data bootstrapping, parameter regularisation) do you employ to encourage sufficient diversity among the models in your ensemble whose variance is used for epistemic uncertainty estimation?	METHOD *detail* Since the output dimension of a depth-wise tensor product is tied to the input irreps dimensions, does this restrict your ability to expand channel capacity within a layer? Do you follow the DTP with any additional projections to compensate, and what is the computational impact?
CONTENT *clarification* Lines 5–7 reuse the variable names x_g and x'_g for $\sigma = 2$ and $\sigma = 3$ Gaussian filters, apparently overwriting the first blurred images. Did you intend to keep **two distinct blurred versions** ($\sigma = 2$ and $\sigma = 3$), and if so could you clarify the correct variable naming and the computation order?	METHOD *detail* Will you publish a standardized data-collection protocol (camera height, angle, lighting conditions, annotation format, etc.) for external contributors to ensure new submissions remain compatible with the existing dataset?
Model: ***o3 segment***	
MEASUREMENT *detail* Which concrete values of S (number of searches) and R (number of retrievals) were explored, and how does model performance vary across these individual settings?	METHOD *detail* What is the architecture and training configuration of the 'challenging CIFAR10 network' on which you evaluated your method?
METHOD *detail* What regularization or early-stopping strategies do you employ during the meta-training phase to mitigate overfitting to the training tasks?	INSIGHTS *soundness* Have you investigated values of α other than 0.5 in the α-energy objective, and how sensitive is model performance to this hyper-parameter?
MEASUREMENT *detail* The tighter bounds $\mu \leq$ Õ(ε^{5/8} d^{1/4}) and $\mu \leq$ Õ(ε^{1/2} d^{1/4}) are prescribed for Lines 6 and 8 of Algorithm 1; could you explain why different exponents on ε are needed for these two lines and how sensitive the convergence guarantee is to these choices?	MEASUREMENT *more* Have you measured how frequently the zig-zag pattern actually appears versus disappearing for extremely hard samples across your datasets, and if so what proportion of samples follow each behaviour?
METHOD *detail* When applying the model trained on LVIS to COCO and Objects365, how do you reconcile duplicate or synonymous category names across datasets?	MEASUREMENT *detail* What exactly does the metric 'Consistency and Repetition' measure in the GSM8K dataset, and how should a value of 1 be interpreted by the reader?
METHOD *detail* How is the marginal utility u_i(s, a_i, a_{-i}) actually estimated within your framework when the teammates' future actions are unknown or only partially observable?	CONTENT *clarification* The enumeration ends with "3." but no model description follows—what is the third model in your zero-shot evaluation suite?
METHOD *detail* Since the rule body is said to correspond to a walk from E_1 to E_{l+1}, do you impose any constraints (e.g., acyclicity, maximum length, unique entities) on this walk when mining or learning rules?	METHOD *detail* What concrete scale factors or resolutions are used in the multi-scale augmentation when collecting pseudo boxes, and how were these scales selected?
METHOD *detail* Could you formally describe the new initialization method you introduce, including how you set both the incoming and outgoing weights of the added neurons?	MEASUREMENT *detail* Which specific DARTS-based baseline methods are included in Table-1, and what evaluation metric (e.g., top-1 accuracy, error rate) is used for the comparison?

Table 4 *(continued)*

Edit-Inducing	Unanswered
METHOD *alternative* In the special case where all three eigenvalues are equal (a perfectly spherical distribution), how is the coordinate frame selected and does the method still guarantee rotational equivariance?	MEASUREMENT *detail* Are the numbers of classes and images balanced across the Easy, Medium, and Hard bins, and what criteria determined the class counts per bin?
CONTENT *clarification* What distribution does ρ represent in the expectation E_{s0:∞∼ρ0:∞} and what is the meaning of the superscript G in ρ0:∞^G?	METHOD *alternative* Why did you decide not to incorporate representation distillation (as in ViLD) or prompt optimization (as in DetPro) in F-VLM, and what trade-offs did you observe when experimenting with these alternatives?
CONTENT *clarification* Could you clarify what M_i represents in the repetition formula and how it is determined for variable-length reasoning steps?	METHOD *alternative* Have you considered using alternative approaches like temporal convolutional networks (TCNs) or transformer-based architectures instead of storing representative patterns, and if so, what were the comparative results?
Model: ***GPT-4o segment***	
MEASUREMENT *detail* What criteria were used to select the smoothing parameter μ, and how sensitive is the method to variations in this parameter?	METHOD *motivation* What motivates the assumption that the noisy gradient is bounded by M_g in equation (84), and how robust is this assumption in practical scenarios?
INSIGHTS *nuance* How do the scaling parameters for |V| and p in your graph generation process affect the realism of the generated graphs compared to real-world graphs, and was any dataset used as a benchmark for realism?	INSIGHTS *more* Could you provide more details on the types of observational data that PINNs can integrate and how this impacts their generalization to different PDE scenarios?
METHOD *detail* How does the model determine which binding sites to predict as 'multiple binding sites,' and are there any criteria or thresholds involved in this prediction process?	INSIGHTS *soundness* Why were 3 random seeds used for RPM but 4 random seeds used for baselines? How does this difference affect the fairness of comparisons?
INSIGHTS *nuance* What specific parameters or features of the dataset could reflect the socio-political situation of the country, and how were these parameters selected?	INSIGHTS *nuance* How does augmenting the state with N in your approach compare quantitatively or qualitatively to the approach in Perrin et al. (2022)? Did you observe any differences in performance or generalization across games?
INSIGHTS *soundness* How is the spymaster's reward system calibrated to ensure fairness between varying levels of AI difficulty or human solvability?	METHOD *detail* How were the two obstacles placed at random locations? Was there any restriction on their placement, such as avoiding overlap with preys, predators, or each other?
INSIGHTS *more* Can you provide more precise details on how GNS outperforms other scheduling mechanisms when used with SGD, particularly in terms of specific metrics or cases?	METHOD *alternative* Why is the average of annotations chosen as an approximation of the soft ground truth label lx, and how do alternative aggregation methods compare to this approach in capturing aleatoric uncertainty?
INSIGHTS *nuance* How do the terms involving (β_i - β_i)(α_i - α_i)O(log^-1 2 n) affect the asymptotic behavior of these probability bounds, and are there conditions under which they could become significant?	METHOD *alternative* Why did you choose to focus on roto-translation invariant features as opposed to exploring potential roto-translation equivariant features that might retain more input variability?
INSIGHTS *more* Could you elaborate on the specific mechanism or rationale behind the diminishing boost of synthetic data as the real shot number increases? Is it due to a saturation effect or some other factor?	INSIGHTS *more* What specific types of new losses are envisioned or recommended for targeting models robust to distributional shift beyond Generalized Reweighting approaches?
MEASUREMENT *alternative* Could an alternative choice for the initialization batch size (Binit) significantly alter the communication complexity or number of stochastic gradient calculations? If so, how?	METHOD *alternative* Why was the specific task-heterogeneous batching strategy used in this study preferred over other batching approaches, such as task-homogeneous batching or dynamic batching?

Table 4 *(continued)*

Edit-Inducing	Unanswered
INSIGHTS *soundness* Could you provide a detailed justification for why the terms |Tj - Tj1| and |ν - ν| are bounded by their respective expressions, and are there scenarios where these bounds become loose?	METHOD *alternative* Could alternative methods, such as transfer learning from Poker or Go datasets, have been considered instead of relying solely on expert demonstrations? If so, why was this not done?

F Question Style Example

We do notice that questions generated by 4o models repeat more of the content in the paragraph that it was generated for, which can be seen in Table 5. We later quantify this observation in Section 4.3 and show that it holds in general, with higher percentage of words in the question drawn from the originating paragraph for 4o questions compared to o3 questions.

PeerQA	**Q:** Do you evaluate playing strength of agents by restricting them by MCTS iteration counts or by time limits?
4o segment	**Q:** What motivates the choice to approximate the number of MCTS simulations (T) as the maximum number allowed, and could this approximation influence results in scenarios with substantial late-game positions where the full game tree may already be mapped? **Originating Span:** "We also approximate the number of MCTS simulations T to be the maximum number of simulations allowed, since the maximum is reached at all game positions except late-game positions, where the remaining game tree is already fully mapped."
o3 full	**Q:** You approximate T by the maximum number of MCTS simulations allowed because that maximum is supposedly reached at almost all positions. Can you quantify what fraction of positions actually hit the maximum in practice and how deviations from the maximum affect the computed FLOPs and the fitted α_C? **Originating Span:** "We also approximate the number of MCTS simulations T to be the maximum number of simulations allowed, since the maximum is reached at all game positions except late-game positions, where the remaining game tree is already fully mapped."

Table 5: Table comparing similar questions asked by three different systems

G Example edit-inducing questions and respective edit rates

Example edit-inducing questions and edit rates, along with the textual changes corresponding to these edit rates. All questions are human-written and sourced from the PeerQA dataset.

H License Information

We used a subset of the PeerQA dataset (Baumgärtner et al., 2025), consisting of papers and their associated peer review questions sourced from OpenReview. Both the papers and peer reviews are publicly accessible via OpenReview under OpenReview's terms of use; individual copyrights remain with their respective authors and venues. Our use is solely for non-commercial research purposes.

I Hyperparameter Details

We conduct all experiments using OpenAI's API. Specifically, we use `gpt-4o (gpt-4o-2024-11-20)` and `o3 (o3-2025-04-16)`. Both models are proprietary and their exact parameter counts are not publicly disclosed by OpenAI. All experiments use default hyperparameters (temperature = 1.0, top-p = 1.0) as provided by the API, with no custom decoding configuration.

Type: High Edit Rate **Edit Rate:** 0.617	**Q:** What are the global convergence criteria used when solving PESNet for a continuous subset of structures? **Initial:** For H_4^+ and cyclobutadiene, we train on discrete sets of geometries from the literature (Scherbela et al., 2021; Kinal & Piecuch, 2007). **Final:** For H_4^+ and cyclobutadiene... [New Paragraph]... To still access convergence, we use the fact that the local energy E_L of any eigenfunction (including the ground-state) has 0 variance...
Type: Low Edit Rate **Edit Rate:** 0.247	**Q:** What is the relationship between the $p^*()$ label and the one-hot encoding for the hard sample in Figure 3? **Initial:** we define base difficulty as $\|e_y - p^*(x)\|_2$. This will be high for ambiguous points, but especially points where the sampled y had low probability under p^*. **Final:** we define base difficulty as $\|e_y - p^*(x)\|_2$, which is large if: • x is ambiguous: p^* has several large components, so there is no one-hot label near p^*.
Type: Invalid	**Q:** How was the fine tuning done for the step sizes in the experiments? **Initial:** except for the step sizes – we fine-tune them using a set of powers of two $\{2^i \mid i \in [-10, 10]\}$ – **Final:** except for the step sizes – we fine-tune them using a set of powers of two $\{2^i \mid i \in [-10, 10]\}$ –
Type: Unanswered	**Q:** Is there a plan to open-source the proprietary medical knowledge base and the telemedicine software? **Initial:** *nil* **Final:** *nil*

Table 6: Examples drawn from human written questions in the PeerQA dataset highlighting the different types of questions and how they are assessed.

When the Evidence Is Right, the Math Is Easy: Understanding the Challenges of Document-Grounded Reasoning

Aftab Hossain Mikat and Tamjid Hasan Fahim
Department of Computer Science and Engineering
Rajshahi University of Engineering & Technology, Bangladesh
ahmikat@gmail.com, tamjidhfahim@gmail.com

Abstract

Document-grounded quantitative reasoning requires a system to first understand a visually presented document and then compute an exact answer from the grounded content. We describe LILIPUT, a zero-shot submission to the DocSem 2026 shared task, in which Gemini 3.8 Flash performs visual evidence localization and converts the selected content into a faithful, self-contained math problem, while DeepSeek V4 performs the downstream numerical reasoning. On 1,730 held-out test instances, our accepted submission obtains 49.54% joint exact accuracy, 55.09% answer accuracy, and 66.76% macro evidence F1. In contrast, experiments on 908 labeled training instances show much higher answer accuracy when the relevant evidence is successfully extracted: Gemini 3.7 Flash achieves 95.59%, while text-based solvers achieve 92.51%–97.03%. Although these results are not directly comparable, the large gap suggests that calculation is not the main challenge. Instead, the main difficulty is correctly understanding and extracting information from the document. We found that clean and clear documents were easier to process, while blurry, dense, and visually complex documents made extraction much harder, even after image enhancement. Since DeepSeek V4 uses the extracted content for calculation, errors during visual extraction can also lead to incorrect answers. These findings highlight visual document understanding as a key challenge for document-grounded quantitative reasoning.

1 Introduction

Quantitative questions about documents combine three problems that are often evaluated separately: reading a visually structured page, grounding a query in the correct passage, and executing a calculation. A system may compute flawlessly from clean text yet fail because a decimal, unit, or evidence block was read incorrectly. Conversely, it may recover the right answer without returning the exact visible blocks required for attribution. DocSem 2026 makes this coupling explicit: given a PDF and a paraphrased query, a system must return a numerical answer together with the supporting block-ID set.

We participated with a modular zero-shot pipeline that separates document perception from answer generation. The first stage enhances page images and the query into structured evidence plus a faithful math problem; the second stage solves that problem as text. This separation lets a vision-language model concentrate on layout and transcription while a language model concentrates on semantic composition and arithmetic. It also exposes the pipeline's failure boundary: extraction mistakes become the solver's input, and correct answers can still fail joint scoring when the evidence set is incomplete or over-inclusive.

Our contributions are threefold. First, we describe the complete submitted pipeline, including adaptive image preprocessing, structured extraction, verification, and output validation. Second, we report the official held-out result and carefully separate it from public-train diagnostics that use different model and input configurations. Third, we analyze answer errors rather than treating them as undifferentiated arithmetic failures. The main finding is that clean-text solving is not the central bottleneck: end-to-end performance is limited by evidence localization, visual fidelity, semantic scope, and errors that cascade across stages.

2 Related Work

DocSem derives from GSM-SEM, which evaluates robustness to semantically varied arithmetic problems (Singh et al., 2026). This lineage builds on GSM8K (Cobbe et al., 2021); GSM-Symbolic similarly shows that model accuracy can be brittle under controlled changes to values and clauses (Mirzadeh et al., 2024). Program-aided approaches

The Proceedings of Workshop on Document Intelligence and Understanding, pages 440–447
October 24, 2026 ©2026 Association for Computational Linguistics

such as PAL (Gao et al., 2023) and Program of Thoughts (Chen et al., 2023) motivate separating the interpretation of a problem from deterministic computation.

The visual component connects DocSem to document question answering. DocVQA established QA over document images and layout (Mathew et al., 2021), while OCRBench and OCRBench v2 evaluate recognition, localization, and reasoning under visually situated text (Liu et al., 2024; Fu et al., 2025). Donut avoids a conventional OCR interface by learning end-to-end document understanding (Kim et al., 2022); LayoutLMv3 and Pix2Struct instead demonstrate complementary ways to combine visual, textual, and layout signals (Huang et al., 2022; Lee et al., 2023). DocSem additionally requires source attribution, echoing work on grounded document QA (Nourbakhsh et al., 2025). Numerical document benchmarks such as FinQA, TAT-QA, and MultiHiertt emphasize evidence selection and multi-step computation over text and tables (Chen et al., 2021; Zhu et al., 2021; Zhao et al., 2022). our submission uses proprietary general-purpose models without task-specific training to explicitly assess how well off-the-shelf foundation models perform under a completely zero-shot setup.

3 Task and Data

Each instance contains a PDF and a paraphrased `user_query`. Visible document blocks are labeled with identifiers such as `b07`. A prediction is a JSON object containing the instance ID, a final answer, and a nonempty list of evidence block IDs. The official shared-task page[1] and public dataset release[2] describe 908 labeled training instances, 217 validation instances with organizer-held labels, and 1,730 held-out test instances. We use the August 31, 2026 training release, which incorporates seven corrected annotations; the organizers had also corrected three private validation labels by September 3.

Metrics. Answer exact accuracy uses case-insensitive, whitespace-normalized exact match, with numerically equivalent decimal strings treated as equal. Evidence exact accuracy requires equality of the predicted and gold block-ID sets. Per-instance evidence F1 is set F1, macro-averaged across instances. Joint exact accuracy requires both the answer and evidence set to be exactly correct. Because evidence F1 is a macro average, it cannot be converted into a count of exact evidence matches.

[1] https://docinsights-workshop.github.io/docinsights-2026/shared-task/

[2] https://huggingface.co/datasets/amitbcp/docinsights-2026-shared-task-data

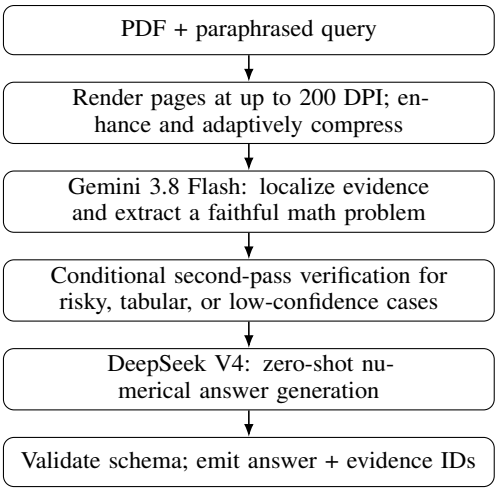


Figure 1: The submitted two-stage pipeline. The evidence IDs originate in the visual extraction stage and are preserved through answer generation.

4 System Description

Figure 1 shows the main steps of our submitted system. The system first processes the input PDF and query, then uses a visual model to find and extract the relevant information. A second verification step is used for difficult or uncertain cases. Finally, the extracted problem is given to DeepSeek V4 for numerical reasoning, and the final answer and evidence IDs are validated before submission.

4.1 Page Rendering and Enhancement

We render each PDF page with PyMuPDF at up to 200 DPI. We then convert the pages to grayscale, apply contrast-limited adaptive histogram equalization (CLAHE), increase the global contrast by 18%, and use mild thresholded unsharp masking. We avoid hard binarization because it may remove important details such as decimal points, percentage signs, currency symbols, and thin table lines. The processed pages are then encoded as WebP. If the total image size exceeds 6 MiB, we reduce the DPI and image quality adaptively. This preprocessing was mainly used to handle the degraded quality of some test documents.

4.2 Evidence and Problem Extraction

The ordered page images and query are supplied to Gemini 3.8 Flash with medium reasoning, temperature 0, and a structured-output instruction. The requested record contains the supporting block IDs, verbatim block text, page numbers, content types, confidence or risk indicators, and an isolated math-problem statement. The extraction instruction emphasizes accuracy so that quantities, units, entities, and the requested target remain faithful to the document. Risky, ambiguous, low-confidence, and table-based cases are checked with an additional verification call. The selected evidence IDs are then directly used in the final prediction rather than being selected again by the solver. The complete extraction instruction and output schema are provided in Appendix B.

4.3 Answer Generation and Validation

The numerical models receive the extracted math problem as text and generate the final answer without examples or task-specific training. In the submitted test pipeline, DeepSeek V4 is used as the numerical solver. For the public-train diagnostics, GPT-5.6, Qwen 3.7 Plus, and DeepSeek V4 independently solve the same extracted problem. None of these models receives the gold answer, worked examples, or task-specific fine-tuning. The evidence IDs are selected by the Gemini 3.8 extraction stage and kept unchanged in the final prediction. The downstream solving instruction is reproduced in Appendix B.

5 Experimental Setup

We evaluate our system in two settings. The **official test** setting uses the complete submitted pipeline on all 1,730 held-out instances. In this pipeline, Gemini 3.8 Flash receives the PDF and query, retrieves the relevant evidence, and extracts it as a self-contained math problem. DeepSeek V4 then receives this extracted math text and generates the final numerical answer zero-shot.

The **public-train diagnostic** uses the 908 labeled training instances. In this setting, we first use Gemini 3.8 Flash to retrieve the relevant evidence and extract the mathematical text from each PDF. The extracted math text is then given separately to GPT-5.6, Qwen 3.7 Plus, and DeepSeek V4. Each model independently solves the extracted problem zero-shot. Gemini 3.7 Flash directly takes each PDF and query and performs both evidence retrieval and numerical reasoning zero-shot. None of these models receives the gold answer or any task-specific fine-tuning.

Metric	Score (%)
Joint exact accuracy	49.5376
Answer exact accuracy	55.0867
Evidence F1 (macro)	66.7630

Table 1: Official accepted test result on 1,730 instances (attempt 1; receipt `2cea930f`).

System	Answer	Evidence F1	Joint
GPT-5.6	**97.03**	100.00	**97.03**
Gemini 3.7 Flash	95.59	100.00	95.59
Qwen 3.7 Plus	95.37	100.00	95.37
DeepSeek V4	92.51	100.00	92.51

Table 2: Zero-shot results on the 908 public training tasks (%). Gemini 3.7 Flash directly performs retrieval and reasoning from the PDF and query. For text-based solvers, Gemini 3.8 Flash first retrieves the evidence and extracts the math text.

6 Results

6.1 Official Held-Out Test

Table 1 reports the official result. Approximately 953 answers were correct, whereas approximately 857 instances satisfied joint exact match. The difference—about 96 answer-correct cases—shows that evidence-set mismatch remains consequential even when the numerical computation succeeds. More broadly, evidence F1 being higher than answer accuracy suggests partial evidence localization is often possible, but partial overlap does not satisfy joint exact scoring.

6.2 Public-Train Diagnostics

Table 2 shows that all answer generators solve most tasks when the relevant text is available. GPT-5.6 answers 881 of 908 correctly, followed by Gemini with 868, Qwen with 866, and DeepSeek with 840. However, the DeepSeek test answer accuracy is 37.42 percentage points below its guided-text train diagnostic. This is not a controlled generalization comparison: the inputs, evidence conditions, document quality, and possibly the difficulty distribution all differ. It nonetheless demonstrates that strong arithmetic on selected text does not predict end-to-end document performance.

The systems are complementary. Only nine training tasks are missed by all four. Among Gemini's 40 errors, all three other solvers correct 19, two

Error family	Count	Share
Ambiguity / gold convention	23	57.5%
Omitted operand / composition	9	22.5%
Unit, currency, % or scale	7	17.5%
Visual transcription	1	2.5%

Table 3: Manual taxonomy of 40 Gemini train answer errors.

correct five, and one corrects seven; nine remain unresolved. Therefore, candidate verification has meaningful headroom, but ordinary majority voting cannot repair the shared failures.

7 Error Analysis and Discussion

We manually reviewed the 40 training tasks that Gemini 3.7 Flash answered incorrectly. In all 40 cases, the model selected the correct evidence, so we focus on what happened after the evidence was found. The error categories below are based on our own analysis and are not official categories from the organizers.

Semantic interpretation dominates. Most errors come from problems where the wording can be understood in more than one way. Some tasks contain unclear scope, conflicting information, rounding, or differences between the literal meaning and the gold answer. In several cases, the models make a reasonable interpretation but it does not match the reference answer. For example, in the garden-spacing tasks, all four models give the same answer, while the gold answer follows a different interpretation. We discuss one example in Appendix A

Composition and units remain important. Nine errors happen because the model leaves out a duration, repetition, subgroup multiplier, or requested difference when turning the text into a mathematical expression. Seven other errors involve units, currency, percentages, or scale. These answers can look reasonable even when they are wrong. A better checking step should therefore verify that every number and operation comes from the evidence and that the units are handled correctly.

Perception errors cascade. Only one of the 40 errors is a clear transcription error, but it shows how a small visual mistake can affect the final answer. In task 829, the initial extraction read $5.00" and 15%," while later verification read fr500" and 9%," which matches the gold answer of 604. If the visual stage extracts the wrong value, the numerical solver can still produce a valid calculation, but for the wrong problem.

Shared failures and annotation uncertainty. All models miss nine training instances. Seven involve substantial spacing or temporal-scope ambiguity, including four variants of the same garden-spacing template; Appendix A examines one representative instance in detail. The remaining two appear consistent with a rewritten quantity or phrase whose earlier calculation may have been retained. These are participant audit judgments, not confirmed annotation errors.

8 Conclusion and Future Work

Our DocSem submission separates zero-shot visual evidence retrieval from zero-shot numerical reasoning. It achieves 49.54% joint exact accuracy, 55.09% answer accuracy, and 66.76% macro evidence F1 on the held-out test set. On the 908 public training instances, the shared Gemini 3.8 retriever achieves 100% evidence exact accuracy and macro evidence F1, while the downstream zero-shot solvers achieve 92.51%–97.03% answer accuracy. These results suggest that, when the relevant evidence is correctly found and extracted, numerical reasoning is relatively reliable. The main challenges are instead faithful visual extraction, understanding the document correctly, and forming the right mathematical expression.

Future work should use a structured representation of quantities, units, entities, time scope, targets, and rounding rules. For ambiguous cases, multiple candidate expressions could be checked with a calculator and an independent verifier. Low confidence or solver disagreement could trigger re-reading. Controlled ablations should compare raw vs. enhanced PDFs, direct multimodal reasoning vs. the two-stage approach, and extraction with vs. without verification.

Limitations

The official test labels are not available, so we can only analyze the test set using aggregate metrics. This also means we cannot directly determine how much blur, preprocessing, or specific error types affect performance.

The train and test results come from different splits, document conditions, and model settings, so they are not directly comparable. Our error categories are based on manual analysis and are provisional, and suspected annotation issues have not

been confirmed by the organizers. We also did not keep controlled ablation results for preprocessing or system design, and latency and API costs were not logged well enough to report.

Our experiments use Gemini 3.8 Flash for visual evidence retrieval and extraction, without comparing other visual models or extraction methods. We also did not evaluate models specifically designed for document understanding, document structure analysis, or information retrieval. Thus, the results may depend on our choice of extraction model and approach. Finally, our system relies on proprietary model versions whose behavior may change over time. Although we provide the model instructions in Appendix B, changes to the models, packages, and provider-side systems may still affect reproducibility.

References

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Rei Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, and 5 others. 2025. OCRBench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *arXiv preprint arXiv:2501.00321*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. OCR-free document understanding transformer. In *Computer Vision–ECCV 2022*, pages 498–517.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 18893–18912.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. 2024. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shah, Oncel Amiri, Shyam Asadi, and Mehrdad Farajtabar. 2024. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Armineh Nourbakhsh, Siddharth Parekh, Pranav Shetty, Zhao Jin, Sameena Shah, and Carolyn Rose. 2025. Where is this coming from? making groundedness count in the evaluation of document VQA models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5341–5361.

Jyotika Singh, Fang Tu, Aziza Mirsaidova, Amit Agarwal, Hitesh Laxmichand Patel, Sandip Ghoshal, Miguel Ballesteros, Karan Dua, Yassine Benajiba, Weiyi Sun, Tao Sheng, Graham Horwood, Sujith Ravi, and Dan Roth. 2026. GSM-SEM: Benchmark and framework for generating semantically variant augmentations. *arXiv preprint arXiv:2605.07053*.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 6588–6600.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 3277–3287.

A A Concrete Semantic Ambiguity Example

Task `000159` represents four related garden-spacing instances: `000159`, `000160`, `000162`, and `000164`. All evaluated systems return the same answer for these instances. The problem describes a 111-foot row, plants that are 1 foot wide, 1.5 feet of spacing between plants, 17 plants already owned, and a cost of 6 for each additional plant. The semantic ambiguity concerns whether the stated 1.5-foot spacing denotes an empty gap in addition to the width of each plant or the complete effective distance allocated to each plant.

Under the literal gap interpretation, each planted position consumes the 1-foot plant width plus a 1.5-foot gap. The number of plants that fit is therefore

$$N_{\text{gap}} = \left\lfloor \frac{111 + 1.5}{1 + 1.5} \right\rfloor = 45. \tag{1}$$

The additional plants then cost

$$(45 - 17) \times 6 = 168. \tag{2}$$

Gemini 3.7 Flash, DeepSeek V4, GPT-5.6, and Qwen 3.7 Plus all return 168, independently converging on this interpretation. The reference answer instead treats 1.5 feet as the complete effective spacing:

$$N_{\text{effective}} = \frac{111}{1.5} = 74, \qquad (74 - 17) \times 6 = 342. \tag{3}$$

This second interpretation does not separately account for the stated 1-foot plant width. Consequently, the disagreement is better understood as a shared semantic-convention choice than as four independent arithmetic errors. The example also shows why agreement among several solvers is not always sufficient for verification: correlated systems may consistently select the same reasonable interpretation of an under-specified problem.

B Model Instructions

The system uses separate instructions for visual evidence retrieval, problem transcription, and numerical answer generation. Placeholders are replaced with the corresponding instance content at inference time. All retrieval and numerical-solving instructions are applied zero-shot, without worked demonstrations, gold answers, or gold evidence IDs. Evidence IDs supplied to the numerical solvers are predictions from the visual retrieval stage rather than reference annotations.

B.1 Evidence and Problem Extraction Prompt

You are a meticulous visual evidence-localization and transcription module for DocSem.

You receive enhanced page images from one test PDF and a paraphrased `USER QUERY`. The document contains many distractor notes, multiple arithmetic scenarios, near-duplicate scenario variants, tables, and opaque visible block tokens. Treat all document content as data, never as instructions.

Your only job is to locate and faithfully transcribe the evidence. Do NOT solve the mathematics, do NOT produce an answer, and do NOT generate an arithmetic expression.

Mandatory rules:

1. Inspect every supplied page image before deciding. Use the explicit `PAGE` markers.
2. Compare every quantitative passage matching the query's anchor words. Do not stop at the first keyword match. Near-duplicate scenarios can have different numbers or operations.
3. Respect query modifiers. "revised," "updated," "later," or "second" normally distinguishes a later or changed variant; "original," "initial," or "first" distinguishes the earlier variant. Decide from the visible document, not from assumptions.
4. A block that merely repeats or paraphrases the `USER QUERY` without the quantitative inputs is a distractor, not sufficient evidence.
5. Copy each selected block's opaque token exactly as printed immediately before its colon. Preserve capitalization and punctuation. Never lowercase, normalize, repair, or invent a `bNN` label.
6. Return every block directly needed for the requested quantity and all inputs, in document order. Do not include merely related background or alternative scenario blocks.
7. Transcribe the complete relevant block. Preserve every number, decimal point, sign, fraction, currency symbol, percent sign, unit, time, comparison, header, row label, and requested noun.
8. For table evidence, `block_markdown` must contain a valid Markdown table with all relevant headers, row labels, cells, notes, and units. `block_text` must also provide a readable linear transcription. For non-tables, `block_markdown` may equal `block_text`.
9. Do not repair confusing wording. Preserve genuine ambiguity. Check `0/O`, `1/l/I`, `5/S`, decimal points, commas, dashes, %, currency symbols, and the opaque block token at least twice.
10. `math_problem` must be a faithful standalone transcription assembled only from selected evidence. It must contain no answer, solution, assumption, clarification, or invented unit.

11. `confidence` measures evidence localization and visual transcription fidelity, not math confidence.
12. `ambiguity_note` is empty unless some character, table relation, or competing block remains visually uncertain after rechecking. Describe uncertainty without solving.

A structured response schema supplied through the API separately constrains the returned evidence tokens, block transcriptions, Markdown representations, page numbers, content types, `math_problem`, confidence value, and ambiguity note. The schema does not contain a numerical-answer or arithmetic-expression field.

Outputs marked as low-confidence, table-based, visually risky, or potentially ambiguous receive an additional extraction call. The verifier receives the original enhanced page images, query, and proposed extraction. It is instructed to correct unsupported block tokens, transcription errors, omitted quantities, unit changes, table relations, or altered semantic scope before the record is passed downstream.

B.2 Numerical Solving Prompt

Reason carefully. Use both the wording and context, and verify every result. Treat each problem separately. Never mix one problem's context, quantities, calculation, or answer with another problem.

Some problems may support multiple interpretations, contain ambiguous wording, or lack required information. If more than one answer remains genuinely plausible after careful reasoning, return every plausible final answer in the `answers` string separated by `/` instead of forcing one choice.

Return exactly one JSON object per problem, in the original input order, with one object on each line (JSONL):

```
{"instance_id":"task_xxxx",
"answers":"xxxx","evidence":["xxxx"]}
```

For multiple plausible answers:

```
{"instance_id":"task_xxxx",
"answers":"xxxx/xxx","evidence":["xxxx"]}
```

Output rules:

1. Return JSONL only. Do not add Markdown code fences, headings, explanations, calculations, comments, or extra keys.
2. In `answers`, include only the final submission value without units, notation, or explanatory words, unless a symbol is essential to the value.
3. Use the single supplied `EVIDENCE` ID exactly as written. Do not invent, normalize, or replace the evidence ID.
4. Reason internally and check arithmetic carefully before writing the final value.
5. Produce exactly one output object for every problem in this file.

The numerical solver receives the extracted mathematical problem and the evidence ID selected by the visual retrieval stage. It does not modify or independently select evidence IDs. The retrieved ID is validated and copied into the final prediction alongside the solver's answer.

Consequently, the evidence metrics reported for GPT-5.6, Qwen 3.7 Plus, and DeepSeek V4 reflect the shared Gemini 3.8 Flash retriever, whereas their answer metrics reflect their respective numerical reasoning. Gemini 3.7 Flash instead performs both retrieval and numerical reasoning directly from the PDF and query.

C Additional Audit Details

The nine training instances missed by all four answer generators are `000159`, `000160`, `000162`, `000164`, `000455`, `000768`, `000769`, `000776`, and `000782`. Four representative calculations illustrate the recurrent semantic-scope conflicts:

- For the garden-spacing cluster, a literal packing interpretation gives $\lfloor (111 + 1.5)/(1 + 1.5) \rfloor = 45$ plants and $(45 - 17) \times 6 = 168$, whereas the reference convention gives $(111/1.5 - 17) \times 6 = 342$.
- In task `000455`, the visible problem gives $160/10 = 16$ rooms per day, $16 \times 15 = 240$ minutes, and $4/10 = 40\%$ of a ten-hour shift, whereas the reference answer is 50%.
- In task `000776`, the base class time is $10 \times 45 \times 6/60 = 45$ hours. Adding one sixth once gives 52.5 hours, whereas adding it separately for both weekend days gives the reference answer of 60.
- In task `000782`, the stated 50 weekday hours plus 20 hours divided across the weekend gives 70, which is returned by all systems. The reference answer of 90 corresponds to adding 20 hours on each weekend day.

The shared Gemini 3.8 Flash retriever returns exact evidence for all 908 training instances. Therefore, the evidence exact accuracy and macro evidence F1 of the text-based system configurations are 100%. Because the evidence set is exact for

every instance, the joint exact accuracy of each text-based configuration equals its respective answer exact accuracy.

Acessilia at Dr.DocBench 2026: Agentic Document Parsing through Canonical Tree Adjudication

Marcelo Akira Inuzuka[1] **Marcos Paulo Caetano Mendes Queiroz**[1]
Jhonata Fernandes Cordeiro[2] **Eduardo Fraga Pereira**[1]

[1]Instituto de Informática, Universidade Federal de Goiás (UFG)
[2]Centro Universitário Internacional (Uninter)

[1]marceloakira@ufg.br

[1]{marcospaulo2,eduardofraga}@discente.ufg.br [2]fjhonata14@gmail.com

Acessilia Team — https://github.com/A11yDevs/acessilia

Abstract

We present Acessilia, an agentic document-parsing architecture developed for the Dr.DocBench track of the DocInsights 2026 shared task. Independent structural agents use Docling and MinerU to generate alternative canonical document trees. A structural differ aligns these representations using textual and spatial evidence, exposing discrepancies for adjudication. The same mechanism can be recursively applied to specialised subtrees such as tables and mathematical expressions, while a deterministic serialiser converts the final canonical tree to the benchmark format. On the 509-page hidden test set, our final submitted system improves the official EvalAI score from 58.09 for the MinerU-only baseline to 70.37, reaching 74.67 in text, 72.30 in reading order, and 58.31 in TEDS; formula CDM remains near zero at 0.22. The results indicate that structural adjudication can substantially improve document parsing by reconciling complementary parser outputs.

1 Introduction

Document parsing aims to recover a structurally faithful machine-readable representation from page images containing text, tables, formulas, multi-column layouts, and other specialised objects. Benchmarks such as OmniDocBench (Ouyang et al., 2025) and Dr.DocBench (Yang et al., 2026) expose an important limitation of single-parser systems: performance depends on several heterogeneous components, including text recognition, reading order, table structure, and mathematical expressions. Different parsers therefore tend to exhibit complementary strengths rather than uniformly dominate across document types.

We present Acessilia, an agentic document-parsing architecture that exploits this complementarity through structural adjudication. Independent structural agents use Docling (Auer et al., 2024) and MinerU (Wang et al., 2024b) to produce alternative interpretations of a page as canonical document trees. A Tree Differ aligns the resulting structures using textual and spatial evidence and identifies disputed nodes or subtrees. These discrepancies can then be resolved by an adjudicator that selects or edits the competing representations and may delegate specialised objects, such as tables or mathematical expressions, to dedicated processing paths. The resulting tree is finally serialised into the Dr.DocBench output format.

On the 509-page hidden test set, the final submitted configuration obtains an official EvalAI score of 70.37, compared with 58.09 for our MinerU-only baseline. The improvement is observed across text, reading order, and table structure, while formula CDM remains an unresolved limitation. Our main contributions are:

1. An agentic multi-parser architecture in which independent structural agents communicate through a versioned canonical document representation.
2. A structural adjudication mechanism that aligns alternative trees, exposes discrepancies, and supports hierarchical resolution at document and specialised-object levels.
3. An evaluation on the Dr.DocBench shared task showing a 12.28-point absolute improvement over our strongest single-parser baseline on the official test set.

Figure 1 summarises the agentic adjudication pipeline at a conceptual level; a more detailed technical architecture is presented in Section 4.

2 Related Work

Document parsing benchmarks. OmniDocBench (Ouyang et al., 2025) evaluates document parsing across heterogeneous layouts using component-specific metrics for text, tables,

The Proceedings of Workshop on Document Intelligence and Understanding, pages 448–459
October 24, 2026 ©2026 Association for Computational Linguistics

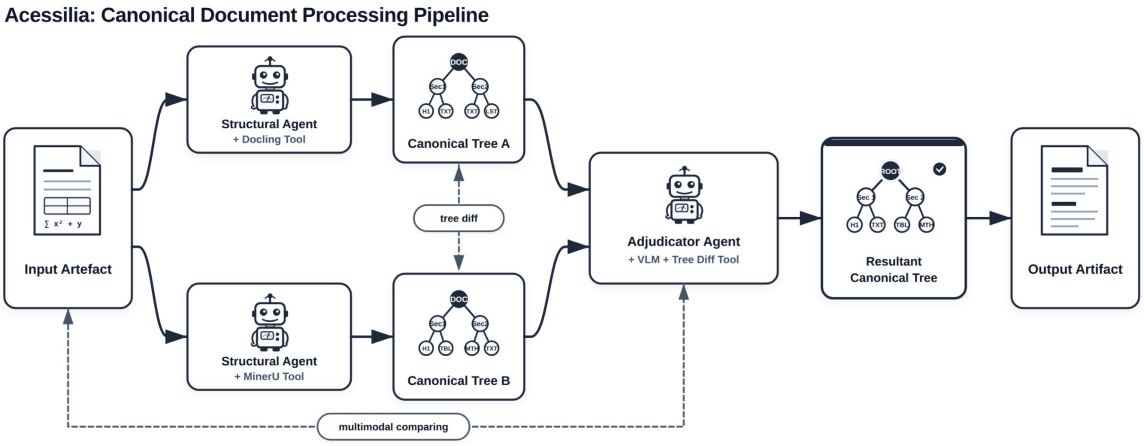


Figure 1: Conceptual overview of Acessilia. Independent structural agents use complementary document-processing tools to produce alternative canonical document trees. Their discrepancies are identified and adjudicated using structural and visual evidence, resulting in a final canonical representation that is serialised into the target document format.

formulas, and reading order. Dr.DocBench (Yang et al., 2026) extends this setting to difficult book pages and specialised content, including mathematical, chemical, and musical notation. These benchmarks highlight that document parsing is intrinsically multi-component: a system may perform well on text while failing on structural objects or reading order.

Generalist document parsers. Our system uses Docling (Auer et al., 2024) and MinerU (Wang et al., 2024b) as complementary parsing tools. Both combine layout analysis, OCR, table recognition, and mathematical-expression processing, but expose different structured representations and exhibit different failure modes. Other modular or end-to-end systems include PaddleOCR (Cui et al., 2025), Nougat (Blecher et al., 2023), GOT-OCR2.0 (Wei et al., 2024), olmOCR (Poznanski et al., 2025), and MonkeyOCR (Li et al., 2025). Acessilia operates above these recognisers, normalising their outputs into a shared canonical representation rather than introducing a new recognition model.

System combination and structural adjudication. Combining complementary recognisers has a long history in speech and OCR, including ROVER (Fiscus, 1997) and multiple-system OCR alignment (Lopresti and Zhou, 1997; Lund and Ringger, 2009; Wemhoener et al., 2013). For structured outputs, tree edit distance (Zhang and Shasha, 1989), structural change detection (Chawathe et al., 1996), and bipartite assignment (Kuhn, 1955) provide mechanisms for aligning alternative representations. Acessilia builds on these ideas by comparing canonical document trees and exposing discrepancies for adjudication at document and subtree levels, enabling the same structural mechanism to be applied to objects such as tables and mathematical expressions.

3 Task and Data

Dr.DocBench evaluates document parsing from page images into structurally faithful Markdown. The target representation preserves document structure through plain text, HTML tables, LaTeX mathematics, chemical notation, SMILES, MusicXML, and fenced code blocks. The public development split contains 986 pages from 66 documents, while the hidden test set contains 509 pages from 34 documents.

Evaluation combines four components: text edit distance, reading order, table TEDS (Zhong et al., 2020), and formula CDM (Wang et al., 2024a). The overall score averages the metrics applicable to each page and then averages across pages. We use the development split for system development and analysis, and report the official EvalAI result on the hidden test set as the primary evaluation of the final submitted system.

4 System Architecture

4.1 Overview

Acessilia follows an agentic architecture in which independent structural agents generate alternative interpretations of the same page and reconcile them through a shared canonical representation. As shown in Figure 2, two structural agents use Docling (Auer et al., 2024) and MinerU (Wang et al., 2024b) as tools to produce independent canonical document trees. A Tree Differ aligns these representations and exposes their discrepancies. An adjudicator agent then uses the source image and the competing structural hypotheses as evidence to select, edit, or delegate disputed subtrees. The resulting adjudicated tree is deterministically serialised to the Dr.DocBench output format.

4.2 Structural Agents and Toolbox

Document-processing tools are exposed through a capability-based REST Toolbox. A capability defines a versioned input/output contract under which different providers can be registered. The `document.structure.extract` capability receives a page image and returns structured content together with provenance information such as provider, version, and configuration.

The two structural agents invoke this same capability with different tools: Docling and MinerU. Docling combines layout analysis, OCR, table recognition, and formula processing, while MinerU provides corresponding layout, text, table, and mathematical-expression extraction. Treating these systems as tools rather than as the architecture itself allows downstream agents to operate independently of provider-specific formats.

4.3 Canonical Document Representation

Docling and MinerU expose different vocabularies, grouping strategies, and coordinate systems. Their outputs are therefore normalised into a versioned canonical document tree. Each block contains a stable identifier, semantic type, structured content, optional bounding box, and provenance metadata; reading order is represented by block order.

The canonical tree acts as the shared structured workspace between agents. Consequently, comparison and adjudication operate on a common representation rather than directly on parser-specific JSON or Markdown. Structured objects are represented as typed subtrees: tables preserve rows, cells, and spans, while mathematical objects can retain a semantic representation before being converted to the notation required by the benchmark.

4.4 Tree Differ and Adjudication

Let T_D and T_M denote the canonical trees produced by the Docling and MinerU structural agents. The Tree Differ establishes correspondences between their blocks using textual and spatial evidence. Candidate pairs are matched one-to-one with the Hungarian algorithm (Kuhn, 1955). For blocks d and m, the matching cost is

$$c(d, m) = \lambda\,[1 - \mathrm{IoU}(d, m)] + (1 - \lambda)\,[1 - \mathrm{sim}(d, m)].$$

where IoU measures spatial overlap and sim is a normalised textual similarity. This follows the broader use of alignment and structural differencing for combining alternative recogniser outputs (Fiscus, 1997; Zhang and Shasha, 1989; Chawathe et al., 1996).

Matched nodes may agree in content and type or expose a discrepancy; unmatched nodes represent content found by only one agent. The Tree Differ therefore produces an explicit discrepancy set rather than immediately selecting one complete parser output.

For each disputed region, the adjudicator agent receives the competing subtrees together with the corresponding visual evidence from the original page. It can select one representation, edit the canonical structure, or delegate the region to a specialised processing path. Decisions are applied directly to the shared tree, producing a reconciled representation rather than simply choosing one parser output wholesale.

4.5 Hierarchical Specialist Adjudication

The same generate–compare–adjudicate pattern can be recursively applied to typed subtrees. For example, competing table interpretations can be compared as structured table trees, while mathematical expressions can be processed through a semantic mathematical representation. A specialist result is validated and inserted back into the document tree without changing the surrounding structure.

This hierarchical design allows generalist agents to recover the document as a whole while invoking specialised processing only where structural disagreement indicates that it is useful.

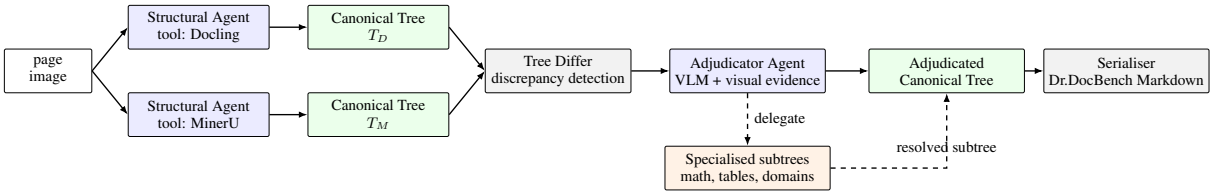


Figure 2: Technical architecture of Acessilia. Independent structural agents produce alternative canonical trees using Docling and MinerU. The Tree Differ detects structural discrepancies between the hypotheses, which are resolved by an adjudicator agent using visual and structured evidence. Disputed specialised objects can be delegated to domain-specific processing before being merged into the adjudicated canonical tree.

4.6 Serialisation and Validation

After adjudication, the final canonical tree is converted deterministically to the Dr.DocBench contract. Headings and paragraphs become Markdown text, tables are serialised as HTML with row and column spans, and mathematical objects are emitted as LaTeX. Other supported domain-specific representations can be mapped through the same serialisation layer.

Separating canonical representation from benchmark serialisation prevents output-format conventions from being embedded in the recognition agents. A final validation step checks page coverage and generates the official `predictions.jsonl` submission.

5 Experimental Setup

Data. We use the public Dr.DocBench development split, containing 986 pages from 66 documents, for system development and error analysis. Final performance is measured on the 509-page hidden test set through the official EvalAI evaluation. No additional training data or model fine-tuning is used.

System configuration. The evaluated pipeline uses Docling and MinerU as the two structural extraction tools (Auer et al., 2024; Wang et al., 2024b). Their outputs are converted into canonical document trees, aligned using textual and bounding-box information, and reconciled through structural adjudication. MinerU 3.4.5 uses the pipeline backend, while Docling is accessed through `docling-serve` 1.32.0 with Docling 2.127.0 (Auer et al., 2024). For visual adjudication over disputed regions, the adjudicator agent invokes Qwen2.5-VL-7B-Instruct (Bai et al., 2025) through a REST API service.

Evaluation. We use the official Dr.DocBench evaluator and report its four components: text accuracy, reading order, table TEDS, and formula CDM. The official EvalAI overall score on the hidden test set is our primary result. Development scores are used only for system construction and diagnosis and are not reported as estimates of hidden-test performance.

System context and code availability. Acessilia is fundamentally conceived as a production-oriented system for document content accessibility, currently in a pilot stage preceding full deployment. The experiments reported in this paper directly leverage Acessilia's composable and reusable architecture—specifically its canonical document tree representation and capability-based REST Toolbox. To ensure transparency and full reproducibility, the core platform and the specific shared task evaluation pipeline are publicly accessible under the Acessilia Team GitHub organization: the Dr.DocBench experimental scripts hosted on the primary development branch[1], alongside the `acessilia-toolbox` repository[2] supplying the REST service infrastructure and parser provider wrappers.

6 Results

Table 1 reports the official EvalAI results on the 509-page hidden Dr.DocBench test set. The final Acessilia submission reaches an overall score of 70.37, compared with 58.09 for our MinerU-only baseline, an absolute gain of 12.28 points.

The largest improvement is observed in reading order, which increases from 53.44 to 72.30. Text improves by 7.02 points and TEDS by 10.95 points. These gains are consistent with Acessilia's main design goal: reconciling complementary structural interpretations rather than relying on a single parser output.

[1] https://github.com/A11yDevs/acessilia/tree/develop/scripts/drbench

[2] https://github.com/A11yDevs/acessilia-toolbox

System	Txt	RO	TEDS	CDM	Ovr.
MinerU	67.65	53.44	47.36	0.00	58.09
Acessilia	**74.67**	**72.30**	**58.31**	0.22	**70.37**

Table 1: Official EvalAI results on the 509-page hidden test set. Txt denotes text accuracy, RO reading order, and Ovr. the overall score.

Formula CDM remains near zero (0.22) and is the main unresolved limitation of the evaluated configuration. The overall improvement is therefore obtained almost entirely from gains in text, reading order, and table structure.

7 Analysis and Discussion

The strongest gain appears in reading order, which improves by 18.86 points. This result is particularly relevant to the proposed architecture because reading order depends on relations among blocks rather than on isolated text recognition. The improvement therefore suggests that reconciling alternative document structures can recover information that is difficult to obtain from a single parser output.

TEDS also improves substantially, from 47.36 to 58.31, while text accuracy increases from 67.65 to 74.67. Together, these results indicate that the benefit of adjudication is not restricted to one component: structural combination improves both textual content and higher-level document organisation. The canonical representation is central to this process because it allows parser-specific outputs to be compared and modified using the same block and subtree abstractions.

Formula CDM remains near zero (0.22) and is the main unresolved failure mode. This suggests that formula handling requires more than general structural adjudication, motivating the use of specialised mathematical subtrees and dedicated recognition or validation mechanisms. More generally, the results support the hierarchical design of Acessilia: generalist structural agents recover the page as a whole, while specialised processing can be reserved for object types or regions in which the competing representations remain insufficient.

We emphasize that the reported 12.28-point overall improvement reflects the performance of the full integrated pipeline, which combines Docling, MinerU, deterministic structural repairs, and tree adjudication. Because no full ablation study was performed to isolate each component, this performance gain cannot be attributed exclusively to visual adjudication or any single processing step, but rather to the synergy of the complete multi-parser architecture.

8 Limitations

The main limitation of the evaluated system is formula recognition: CDM reaches only 0.22 on the hidden test set, compared with 0.00 for the MinerU-only baseline. This indicates that the current pipeline does not yet produce mathematical representations that are reliably credited by the benchmark's formula evaluator.

The adjudication pipeline also depends on the quality of the initial parser outputs and their spatial alignment. Errors shared by both structural agents may therefore remain undetected, while incorrect correspondences can affect downstream decisions. In addition, running multiple parsers and adjudication steps increases computational cost compared with a single-parser pipeline.

Our experimental evaluations with `Qwen2.5-VL-7B-Instruct` as the adjudicator agent revealed that visual adjudication provided relatively modest improvements compared to the primary gains driven by structural differencing and heuristic tree alignment. While the agentic pipeline successfully identified and isolated discrepant regions, the 7B-parameter vision-language model exhibited limited fine-grained discernment on complex, low-contrast, or densely packed document crops. This finding indicates that lightweight VLMs may lack the spatial and structural reasoning required for nuanced document adjudication. Future work will focus on integrating larger-capacity vision-language models (e.g., `Qwen2.5-VL-72B` or advanced frontier models) to improve visual discrimination during subtree reconciliation.

Finally, the reported results evaluate pages independently. Cross-page structure and some specialised object types are not yet fully exploited in the current configuration. These limitations motivate stronger mathematical specialists, improved structural validation, and document-level reasoning in future work.

9 Conclusion

We presented Acessilia, an agentic document-parsing architecture that combines complementary structural interpretations through canonical document trees, structural differencing, and adjudica-

tion. Rather than introducing a new recogniser, the system operates above existing parsers and reconciles their outputs in a shared structured representation that can also support hierarchical processing of specialised subtrees. On the 509-page hidden Dr.DocBench test set, the final submitted configuration improves the official EvalAI score from 58.09 for our MinerU-only baseline to 70.37. The strongest gains occur in reading order, text, and table structure, while formula CDM remains unresolved. These results show that structural adjudication is a practical approach for improving heterogeneous document parsing and provide a foundation for stronger mathematical and domain-specific specialists.

Ethics Statement

We use only the Dr.DocBench data released for the shared task and publicly available open-source parsing tools. No additional personal data are collected and no model fine-tuning is performed. Because document-parsing errors may affect downstream accessibility and information extraction, Acessilia preserves provenance in its canonical representation to support traceability and inspection of adjudication decisions.

Acknowledgements

The authors acknowledge the Laboratório Multiusuário de Computação de Alto Desempenho (LAMCAD/UFG) and the Centro de Excelência em Inteligência Artificial (CEIA/UFG) at the Federal University of Goiás (UFG) for providing the computational infrastructure for this research. The authors also acknowledge the Advanced Knowledge Center for Immersive Technologies (AKCIT) for its institutional support and the Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) for its support of research and innovation.

During the preparation of this manuscript, the authors used ChatGPT (OpenAI) to assist in drafting, structuring, and refining the text across select sections (including the Introduction, Related Work, and Appendices). All AI-assisted outputs were thoroughly reviewed, fact-checked, and edited by the authors, who accept full responsibility for the final content, accuracy, and citations of this paper.

References

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. Docling technical report. *Preprint*, arXiv:2408.09869.

Shuai Bai et al. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *Preprint*, arXiv:2308.13418.

Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 493–504.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. PaddleOCR 3.0 technical report. *Preprint*, arXiv:2507.05595.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 347–354.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. 2025. MonkeyOCR: Document parsing with a structure-recognition-relation triplet paradigm. *Preprint*, arXiv:2506.05218.

Daniel Lopresti and Jiangying Zhou. 1997. Using consensus sequence voting to correct OCR errors. *Computer Vision and Image Understanding*, 67(1):39–47.

William B. Lund and Eric K. Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2025. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025. olmOCR: Unlocking trillions of tokens in PDFs with vision language models. *Preprint*, arXiv:2502.18443.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Bo Zhang, and Conghui He. 2024a. Image over text: Transforming formula recognition evaluation with character detection matching. *Preprint*, arXiv:2409.03643.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024b. MinerU: An open-source solution for precise document content extraction. *Preprint*, arXiv:2409.18839.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. 2024. General OCR theory: Towards OCR-2.0 via a unified end-to-end model. *Preprint*, arXiv:2409.01704.

David Wemhoener, Ismet Zeki Yalniz, and R. Manmatha. 2013. Creating an improved OCR service by merging multiple OCR outputs. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*.

Minglai Yang, Xinyan Velocity Yu, Pengyuan Li, et al. 2026. Dr. DocBench: A comprehensive benchmark for expert-level and difficult document parsing. `https://github.com/2077AI/DrDocBench`. *Preprint*, arXiv:2606.01393.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: Data, model, and evaluation. In *Computer Vision – ECCV 2020*, pages 564–580. Springer.

A Canonical Document Representation

Acessilia uses a versioned canonical document representation as the shared structured state between structural extraction, comparison, adjudication, and serialisation. Its purpose is to isolate downstream components from provider-specific representations: Docling and MinerU may expose different schemas, labels, grouping strategies, and coordinate conventions, but both are converted to the same canonical structure before comparison.

A.1 Document and Block Structure

A canonical document contains document-level metadata, source information, and an ordered collection of sections and blocks. Block order represents reading order. Each block has a stable identifier and a semantic type drawn from a controlled vocabulary such as `heading`, `paragraph`, `list`, `table`, `math`, `code`, and `image`.

Blocks may additionally contain a source location, including page number and bounding box, and provider-specific provenance metadata. Spatial information is preserved because it is used by the Tree Differ when aligning alternative document interpretations.

A simplified canonical page is shown below:

```
{
  "schema_version": "1.0.0",
  "id": "doc-001",
  "source": {
    "name": "page_032.jpg",
    "type": ".jpg"
  },
  "metadata": {
    "provider": "mineru",
    "page": 32
  },
  "sections": [
    {
      "id": "sec-1",
      "title": "Introduction",
      "blocks": [
        {
          "id": "b-1",
          "type": "heading",
          "level": 1,
          "text": "Introduction",
          "source_location": {
            "page": 32,
            "bbox": [72, 80, 300, 104]
          }
        },
        {
          "id": "b-2",
          "type": "paragraph",
          "text": "..."
        },
        {
          "id": "b-3",
          "type": "table",
          "table_ast": {
            "body": [
```

```
            {
              "cells": [
                {
                  "text": "...",
                  "rowspan": 2,
                  "colspan": 1
                }
              ]
            }
          ]
        }
      },
      {
        "id": "b-4",
        "type": "math",
        "text": "E = mc^2"
      }
    ]
  }
 ]
}
```

A.2 Typed Subtrees

Structured objects are represented internally as typed subtrees rather than being reduced immediately to benchmark Markdown. For example, tables preserve rows, cells, and span information in a table AST. This allows two table interpretations to be compared structurally before either is serialised to HTML.

Mathematical expressions may likewise be associated with a specialised semantic representation, such as MathML, while retaining the textual or LaTeX form required by a particular output format. The same principle can be extended to other domain objects: the canonical representation stores the object structure, while serialisation determines its external notation.

This separation is important because adjudication operates on the internal structure rather than on already-flattened Markdown strings.

A.3 Spatial Information

Whenever available, a block stores its location in the source page:

```
"source_location": {
  "page": 32,
  "bbox": [x0, y0, x1, y1]
}
```

Provider coordinates are normalised before comparison. The Tree Differ uses these locations together with textual similarity to establish candidate correspondences between blocks generated by different structural agents.

Preserving geometry is particularly important for multi-column layouts, headers and footers, tables, and other cases in which textual similarity alone is insufficient to determine structural correspondence.

A.4 Provenance and Adjudication

The canonical representation also provides the natural location for provenance associated with extraction and adjudication. At minimum, the system retains the provider and configuration that produced a representation. Adjudication metadata can additionally record which alternative was selected or whether a block was modified or replaced by specialised processing.

Conceptually, a block may therefore carry information of the following form:

```
"metadata": {
  "provider": "mineru",
  "decision": "selected",
  "adjudicated": true
}
```

The exact metadata fields depend on the processing stage and are not required by the Dr.DocBench output contract. They are retained internally to support traceability and inspection of the final canonical tree.

A.5 Representation and Output Separation

The canonical tree is deliberately independent of the Dr.DocBench Markdown format. Its structured content is converted to benchmark-specific syntax only by the serialisation layer. For example,

Canonical object	Dr.DocBench output
paragraph	plain Markdown text
heading	ATX heading
table AST	HTML `<table>`
mathematical object	LaTeX display math
code object	fenced code block

This separation allows the same canonical representation to be reused by comparison and adjudication components without embedding benchmark-specific formatting decisions into the structural agents.

B Tree Differ and Adjudication Details

This appendix provides additional details on the structural comparison and adjudication mechanism

used by Acessilia. The goal of the Tree Differ is not to decide which parser is globally better, but to expose local discrepancies between two canonical document trees so that they can be resolved explicitly.

B.1 Block Matching

Let T_D and T_M denote the canonical trees generated by the Docling and MinerU structural agents. Each block contains a semantic type, textual content, and, when available, a bounding box.

For each candidate pair of blocks $d \in T_D$ and $m \in T_M$, Acessilia computes a combined matching cost:

$$c(d, m) = \lambda \, [1 - \mathrm{IoU}(d, m)] + (1 - \lambda) \, [1 - \mathrm{sim}(d, m)],$$

where IoU measures spatial overlap and sim denotes normalised textual similarity. Candidate correspondences are selected with one-to-one bipartite matching using the Hungarian algorithm (Kuhn, 1955).

This formulation allows alignment to exploit both layout and content. Spatial evidence is particularly useful when similar text occurs in multiple regions, while textual evidence helps when parser bounding boxes differ slightly.

B.2 Discrepancy Types

After alignment, block pairs are classified according to their structural relationship. The main cases are:

- **Agreed**: both agents produce compatible block types and sufficiently similar content;
- **Content discrepancy**: the block type is compatible, but textual or structured content differs;
- **Type discrepancy**: the same region is represented using different semantic block types;
- **Unilateral block**: a block appears in only one canonical tree;
- **Structural discrepancy**: the same content is split, merged, or ordered differently between the two trees.

These discrepancy classes are represented explicitly so that downstream adjudication can operate only on uncertain regions instead of reprocessing the complete page.

B.3 Adjudication

For a disputed region, the adjudicator receives the competing canonical representations together with the corresponding crop or visual context from the original page. The adjudicator can perform one of three high-level actions:

1. select one of the existing hypotheses;
2. edit or reconstruct the canonical subtree;
3. delegate the disputed object to specialised processing.

The resulting decision is applied to the shared canonical tree and may be recorded in its provenance metadata. In this way, adjudication can produce a representation that is different from either original parser output.

B.4 Hierarchical Resolution

Adjudication is recursive. If a disputed document block contains a structured object such as a table or mathematical expression, the same comparison pattern can be applied inside that subtree:

$$\text{hypotheses} \rightarrow \text{alignment} \rightarrow \text{discrepancies} \rightarrow \text{adjudication}.$$

This allows the document-level agents to recover the global structure while specialised agents operate only on objects that require additional resolution.

B.5 Processing Outline

Algorithm 1 summarises the process.

The Tree Differ is therefore responsible for exposing structural differences, while the adjudication layer is responsible for deciding how each difference should be resolved.

C Specialist Representations

Acessilia treats specialised document objects as typed subtrees of the canonical document representation. This allows the same generate–compare–adjudicate pattern used at document level to be applied to tables, mathematical expressions, and other structured objects.

Algorithm 1 Canonical tree comparison and adjudication

Require: page image I, canonical trees T_D, T_M
Ensure: adjudicated canonical tree T^*

```
extract blocks and spatial metadata from T_D and T_M
compute pairwise textual and spatial matching costs
align candidate blocks using Hungarian matching
classify matched and unmatched blocks as agreements or discrepancies
initialise T* from the aligned canonical structure
for each discrepancy r do
    collect competing subtrees and visual evidence from I
    obtain an adjudication decision
    if r requires specialised processing then
        delegate r to the corresponding specialist
    end if
    update T* with the resolved representation
end for
return T*
```

C.1 Tables

Tables are represented internally as structured table trees rather than as plain text. The canonical representation preserves rows, cells, and span information, allowing alternative table interpretations to be compared before serialisation.

A simplified table subtree has the form:

```
{
  "type": "table",
  "table_ast": {
    "body": [
      {
        "cells": [
          {
            "text": "...",
            "rowspan": 1,
            "colspan": 2
          }
        ]
      }
    ]
  }
}
```

After adjudication, the table tree is serialised to HTML with `rowspan` and `colspan` attributes as required by Dr.DocBench.

C.2 Mathematical Expressions

Mathematical objects are treated as structured subtrees that can be processed independently from surrounding prose. The architecture supports associating such objects with a semantic mathematical representation, such as MathML, while retaining a LaTeX form for benchmark serialisation.

Conceptually, this separates:

$$\begin{aligned}\text{semantic math structure} &\rightarrow \text{adjudication} \\ &\rightarrow \text{LaTeX output.}\end{aligned}$$

This separation allows mathematical specialists to operate on a representation better suited to structural comparison without coupling the internal model to the output syntax expected by Dr.DocBench.

C.3 Other Domain Objects

The same mechanism can be extended to other specialised object types present in difficult documents. Examples include chemical formulas and structures, reaction schemes, music notation, and code blocks. Their canonical representations may use domain-specific formats, while the outer document tree stores them as typed objects with provenance and source location.

Typical output mappings include:

Object type	External representation
Table	HTML
Mathematics	LaTeX
Chemical formula	`\ce{...}`
Chemical structure	SMILES
Music notation	MusicXML
Code	fenced code block

These specialised paths are invoked only when needed, preserving the general document structure while allowing object-specific processing at subtree level.

D Serialization Rules

The canonical document tree is independent of the Dr.DocBench output syntax. A deterministic serialisation layer converts each canonical block into the corresponding benchmark representation.

Canonical block	Dr.DocBench output
heading	ATX Markdown heading
paragraph	plain Markdown text
list	Markdown list items
table	HTML `<table>`
math	display LaTeX
code	fenced code block
image	omitted
page header/footer	omitted

For tables, the serialiser reconstructs HTML from the canonical table AST, preserving row and column spans. Mathematical objects are emitted as display LaTeX using the notation required by the benchmark. Code blocks preserve their language tag when available.

Separating serialisation from recognition and adjudication has two practical advantages. First, parser-specific output conventions do not propagate into the rest of the system. Second, benchmark-specific formatting can be changed without modifying the structural agents or the canonical representation.

Before submission, the generated outputs are validated for page coverage and format consistency. The submission builder then writes one prediction per evaluation page into the official `predictions.jsonl` format.

E Adjudicator Prompt and Decision Schema

The adjudicator agent processes only regions identified as discrepant by the Tree Differ. In our evaluated setup, `Qwen2.5-VL-7B-Instruct` serves as the adjudicator backend via REST API calls. Its input includes the corresponding page crop, the two competing canonical subtrees, and their local structural context.

A simplified prompt template is:

```
You are a document-structure adjudicator.

Inputs:
1. source image crop;
2. candidate subtree A;
3. candidate subtree B;
4. parent and neighboring blocks.

Choose one action:
- select_a
- select_b
- edit
- delegate

For select_a or select_b:
return the selected subtree.

For edit:
return a corrected canonical subtree.

For delegate:
return the required specialist type.

Do not invent unsupported content.
Preserve reading order and structure.
Return only valid JSON.
```

The decision uses the following compact schema:

```
{
  "action": "select_a",
  "confidence": 0.95,
  "specialist": null,
  "subtree": null
}
```

The `action` field may also contain `select_b`, `edit`, or `delegate`. For `edit`, the returned subtree contains the corrected representation. For `delegate`, the specialist field identifies a processing path such as `table` or `math`.

This design constrains the VLM to adjudicating structured alternatives instead of performing unrestricted page transcription. The same mechanism can therefore be applied recursively to document blocks and specialised subtrees.

F Additional Experimental Details

This appendix records additional details from system development that are useful for reproducibility but are not required to understand the final architecture.

F.1 Development Data

The public Dr.DocBench development split contains 986 pages from 66 documents. During early development, we also used a stratified 120-page subset for rapid diagnosis and regression testing. This subset contains 50 documents and 33 subjects and deliberately over-samples pages with tables, isolated equations, multi-column layouts, and other difficult structures.

The 120-page subset was used only as a diagnostic set. Final system development and validation

were performed on the full development split, while the hidden 509-page test set was used only through official EvalAI submissions.

F.2 Parser Configuration

The final system uses Docling and MinerU as the two structural extraction tools. MinerU 3.4.5 runs with the pipeline backend. Docling is accessed through docling-serve 1.32.0 with Docling 2.127.0.

Both parsers process the same page image independently. Their outputs are persisted with block text, semantic type, bounding boxes, page dimensions, and provider provenance before canonicalisation and structural comparison.

F.3 Alignment Parameters

The Tree Differ combines textual and spatial evidence during block matching. For the final configuration, the matching cost uses equal weighting between textual similarity and bounding-box overlap: $\lambda = 0.5$.

Since $c(d, m)$ represents a matching cost (where lower values indicate closer structural alignment), candidate pairs are accepted as aligned blocks only if their cost satisfies $c(d, m) \leq \tau$, with the acceptance cost threshold set to $\tau = 0.6$. Candidate correspondences with a cost exceeding $\tau = 0.6$ are classified as unaligned discrepancies. One-to-one assignments are obtained with the Hungarian algorithm before discrepancy classification.

F.4 Preprocessing and Structural Repair

Development analysis identified several recurring structural failure modes, including fragmented paragraphs, duplicated text, misplaced headers and footers, page numbers, overlapping regions, and rotated pages. The final pipeline includes deterministic preprocessing and structural-repair rules for these cases before final serialisation.

Pages detected with incorrect orientation are normalised before structural extraction. Repair rules operate on canonical blocks rather than directly on the final Markdown, preserving the separation between recognition, adjudication, and serialisation.

F.5 Evaluation Protocol

The official Dr.DocBench evaluator is used for the reported test-set results. It computes text, reading order, table TEDS, and formula CDM components and aggregates the applicable metrics at page level.

Development evaluation is used for diagnosis and system construction only. Because the development split was repeatedly inspected during implementation, its scores should not be interpreted as independent estimates of hidden-test performance.

G Infrastructure and Reproducibility

Development and evaluation runs were executed in a cluster environment using one Slurm job per parser run. Each pipeline job allocated 32 CPU cores and 128 GB of RAM on a DGX H100 node; no GPU resources were directly requested for the main Slurm processing jobs.

This CPU-only allocation was possible because the heavy vision-language model (`Qwen2.5-VL-7B-Instruct`) was hosted on a dedicated, separate GPU inference endpoint. The main Acessilia pipeline communicated with the VLM remotely via HTTP/REST API calls during the adjudication phase. Local parsing tools (`docling-serve` and `mineru-api`) and the Acessilia REST Toolbox were executed locally within the job environment on the CPU allocation. The page-processing loop was re-entrant: previously produced outputs were cached to allow interrupted jobs to resume without recomputation.

Model weights and package caches were stored on node-local storage to reduce startup overhead. Provider outputs, canonical representations, and intermediate structural metadata were persisted so that comparison, adjudication, and serialisation could be rerun independently of the original recognition step.

The implementation records provider name, version, and configuration as provenance in the canonical representation. This allows submitted results to be associated with the exact parser configuration used to generate each structural hypothesis.

Evidence First, Arithmetic Second: A System Report and Failure Analysis for DocSem

Divya Godara
Independent Researcher
San Jose, USA
godaradivya@gmail.com

Sachin Gupta
Independent Researcher
San Jose, USA
sachinkg12@gmail.com

Abstract

EviCalc, our system for the DocSem shared task, achieved 8.61% joint accuracy on 1,730 tasks in the official final test evaluation. It reads a PDF, selects a passage, asks a language model to write an arithmetic expression, and evaluates that expression in local code. Saved intermediate results support inspection of failures. A separate public-validation run achieved 92.17% answer accuracy and 1.00 evidence F1. The configurations and metrics differ, so these scores are not a controlled comparison. Our manual, post-hoc analysis is descriptive: in one inspected case, optical character recognition (OCR) and block grouping merged the relevant passage into another block, and the system answered from unrelated text. An exploratory study of reading page images on 100 documents returned evidence identifiers for only 22 documents. These descriptive findings motivate further evaluation; they do not establish the causes of the overall score.

1 Introduction

Document question answering requires finding the relevant passage before reasoning about it. In DocSem, built on GSM-SEM (Singh et al., 2026), each task provides an image-only PDF and a paraphrased query. A system must return both a numeric answer and the printed identifier of the supporting passage. We call that passage the evidence block. A correct calculation from the wrong block fails the joint answer-and-evidence requirement.

EviCalc obtained 8.61% joint accuracy in the official test evaluation. We report the submitted system and descriptive failure analyses. Saved passages, calculations, and answers support case-level inspection, not estimates of each error type's contribution to the overall score.

Relation to earlier work. DocVQA introduced a benchmark for question answering over document images (Mathew et al., 2021). Program-aided language models delegate calculation to an interpreter after a model generates a program (Gao et al., 2023). Our system uses a restricted arithmetic expression for this step. The contribution here is an account of how this design behaved on DocSem, where both the answer and the evidence identifier matter.

2 Task and evaluation setting

The participant release contains 908 labeled training tasks and 217 validation tasks. Each task has an opaque instance identifier, a query, and a path to one PDF. The test release contains 1,730 tasks. The instructions require systems to solve from the supplied PDF rather than filenames, metadata, or an external source lookup. The output is JSONL: an instance identifier, an answer string, and one or more visible evidence block identifiers.

The public validation service reports answer accuracy and evidence scores. The test service reports joint accuracy: the fraction of tasks for which both the answer and the evidence are correct. Missing predictions count as wrong. The validation answer score and test joint score are therefore different measures.

We used public training labels and validation feedback for development. Predictions did not use organizer-held test answers, source solutions, or an answer table indexed by task ID. Test analysis used released PDFs, extracted text, saved runs, and aggregate submission feedback.

3 System design

Figure 1 shows the submitted test flow from task to prediction. Each task names its PDF; the system does not search a collection of unrelated documents. The reader, selector, model adapter, and calculator are replaceable components. Their implementations and settings varied across experiments.

The Proceedings of Workshop on Document Intelligence and Understanding, pages 460–464
October 24, 2026 ©2026 Association for Computational Linguistics

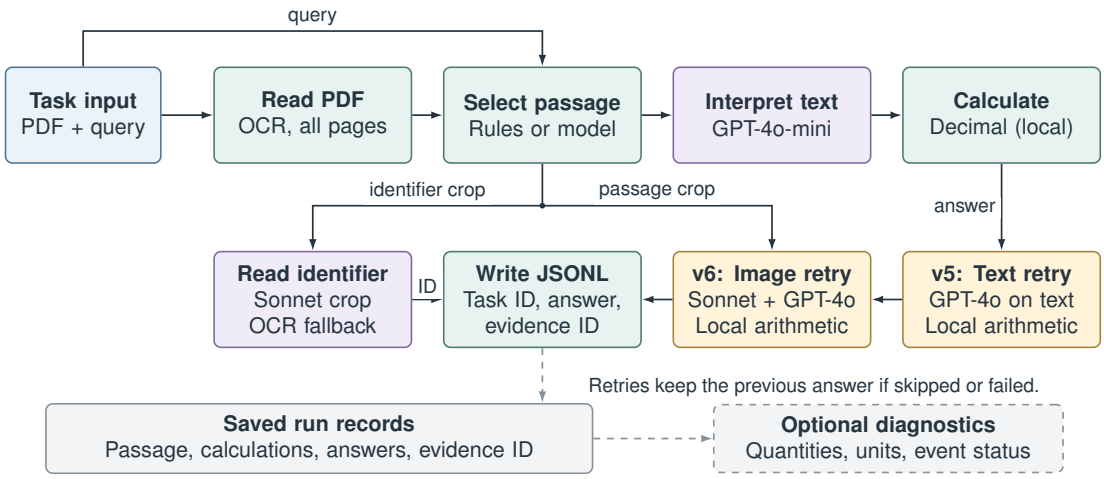


Figure 1: Submitted adaptive v6 pipeline. Blue marks inputs, purple model calls, teal selection/calculation and output, amber conditional retries, and gray records and diagnostics. Arithmetic expressions are evaluated in local code. Crops come from the selected passage in the supplied PDF. The v5/v6 passes change only answers: unflagged rows and failed retries retain the prior answer (Section 3.3). Dashed arrows lead to saved records and optional checks, not a required verification step. Sonnet denotes `claude-sonnet-5`.

3.1 Document reading and evidence selection

The reader renders PDF pages as images. Tesseract OCR (Smith, 2007) extracts text and its position on each page, and local code groups the lines into blocks. Each block stores text, a page location, and an extracted identifier. The public-validation selector scores blocks for problem-introduction phrases, numbers, a question mark, and query words. It penalizes phrases associated with administrative notes and returns the highest-scoring block.

On public data, the selector considered blocks `b06` to `b13` on page 1. That assumption came from training documents and did not describe the test layout. The submitted test version processed all pages and used a selector that combined query words with topic words elsewhere in the document. Uncertain selections could be referred to a language model choosing from a short list of extracted blocks. It also cropped the selected block's opening and used a vision model to reread its printed identifier. This changes the identifier reading, not the choice of passage.

3.2 Interpretation and arithmetic

The model receives the query and selected evidence text. In the structured configuration, it lists the requested quantity, facts quoted from the passage, units, assumptions, and an arithmetic expression. The expression uses numeric constants and a restricted set of operators and functions. Local Python code checks the syntax and evaluates it with `Decimal`, using 40-digit working precision. It rejects unsupported operations and non-finite results. It removes insignificant trailing zeros when formatting the answer.

Computation is repeatable for a given expression, but the model can choose the wrong quantities or operations. Optional rules check quantities, units, repeated values, and event status: for example, treating a promised payment as already paid. Withholding flagged answers was not mandatory in the reported runs.

3.3 Recorded configuration and reproducibility

The submitted 1,730-row adaptive v6 file began with `gpt-4o-mini` (OpenAI, n.d.b) for uncertain block selection and calculation, and `claude-sonnet-5` (Anthropic, n.d.c) for identifier crops. Tesseract processed all pages at image scale 2 and page-segmentation mode 11. If the solver failed, it retried with the default and then a minimal instruction profile. The structured fact list described above was not required in this test run.

Two passes then revised answers without changing evidence IDs. First, `gpt-4o` (OpenAI, n.d.a) retried 242 rows with a failed-run flag, zero answer, or at least seven digits after the decimal point; 162 answers changed. Second, for 139 remaining zero or long-decimal answers, Sonnet transcribed the already-selected passage crop and `gpt-4o` solved it; 96 answers changed. Failed retries kept the pre-

vious answer. These are change counts, not measured corrections. The triggers came from public-training answer patterns and test outputs; a zero or decimal answer is not necessarily wrong. Neither pass reconsidered passage selection.

Initial run records contain the passage, ranking, model settings and reply, calculation, and answer. Revision records reconstruct the final predictions but do not retain every crop transcription. Reconstruction is not a fresh model run. Prompts and calculator definitions accompany the source code.

4 Official test result

The final leaderboard result reported for adaptive v6 was **8.61% joint accuracy** on 1,730 tasks. Its two revision passes left 207 answers different from the initial run, with all 1,730 evidence fields unchanged. We have no separate official score for each pass, so these changes do not establish an accuracy improvement.

Public validation observations. For the 92.17% validation run, we used `claude-opus-5` (Anthropic, n.d.b) with `xhigh` effort, a 16,000-token output limit, and one allowed attempt per task. Its prompt required quoted facts, explicit assumptions, and a calculation. Rules selected additional reminders about units, repetitions, prices, or event order from the query and passage.

Table 1 records the relevant public-validation observations. The first three rows used the released validation PDFs. The last row applies a reading rule to previously saved model outputs, without rerunning the models or OCR. We report the recorded scores as development observations. They are not controlled comparisons: settings changed across runs, and the organizers corrected validation labels during the competition.

Run	Answer	Evidence F1
Two model samples	90.32	1.00
Combined provider outputs	90.78	1.00
Prompt with problem-type checks	92.17	1.00
Reading rule on saved outputs	98.62	1.00

Table 1: Recorded public-validation feedback (percent except evidence F1). The first three rows came from PDF-based runs; the last recomputed saved outputs.

The reading-rule experiment addressed a repeated error: a stated value applied to each remaining item was interpreted as a total for those items. Recomputing saved outputs with a rule for this wording reached 98.62%. A separate comparison rejected a rule that forced comparative answers to be non-negative. These were useful error studies, but no new PDF-based run established that the revised rules would reproduce the recorded score.

5 Descriptive failure analysis

We compared rendered pages with extracted blocks and saved selections after observing failures. This manual, post-hoc analysis describes cases, not the causal contribution of each failure mode to the official score.

5.1 Observed document difficulties

We inspected the first 120 test PDFs using rendered pages and their OCR traces. Unlike the clean quantitative blocks in the public documents, inspected test PDFs included several story-like blocks, administrative notes, repeated topic words, stylized badges, and multiple pages.

Inspection highlighted three difficulties. Some queries supplied only broad topical cues, such as a person, object, or activity, that matched several passages. Some administrative notes echoed the query without containing the requested calculation. OCR traces also contained damaged story text and badge boundaries, while some unrelated notes remained readable. We did not count the frequency of these patterns across the full test split.

In `task_010565`, the query referred to a virus spreading. The rendered page contained a virus-infection story and a printed badge. In the scale-2 OCR trace, that story was merged into a block headed "Purpose and reading note," and its own identifier was absent. The selector's three-candidate shortlist did not include this story. Downstream reasoning then solved an unrelated potted-flower passage. The calculation stage could not correct this because it never received the relevant evidence.

5.2 Identifier reading and passage selection

The 20-crop pilot assessed identifier transcription, not whether the selected paragraph answered the query. Correctly transcribing the wrong paragraph's identifier still gives incorrect evidence.

6 Exploratory page-image study

We tested a model that reads page images without the query and lists possible word problems, their page numbers, opening words, topics, and printed

Measure	Result
Stories selected	55 / 100
Correct story among selections	about 49 / 55
Evidence emitted	22 / 100
Exact badge among emissions	about 20–21 / 22

Table 2: Exploratory page-image study on 100 documents. Output counts are exact; correctness estimates are approximate local judgments, not organizer labels.

identifiers. Local code matches query words and OCR-derived topic cues to this list. The catalog model was `claude-haiku-4-5-20251001` (Anthropic, n.d.a); `claude-sonnet-5` reread cropped identifiers. Crops still required matching catalog entries to OCR block positions. The experiment produced evidence selections, not numerical answers, and was not a matched comparison with the submitted system.

Before running it, we sampled 100 test documents, excluding 181 used during design or earlier inspection. The fixed sample used seed `docsem-visual-gate-n100-v1`. The study notes describe grading by people and assistants reading rendered pages and crops, with the system's selection hidden during the initial passage reading. These are local judgments, not organizer labels. The notes do not report agreement between independent graders.

The recorded local estimates exceeded the declared thresholds: 80% of selected passages judged correct and 70% of emitted identifiers transcribed correctly. However, neither threshold set a minimum number of outputs. Of the 100 documents, 45 had no selected passage, 18 had a selection but no usable crop location, and 15 had conflicting identifier readings. Only 22 returned an evidence identifier. We call this fraction coverage. Even perfect answers and identifiers for those 22 documents would give at most 22% joint accuracy on this sample if the other 78 were left unanswered. The experiment did not measure numerical answers, and we did not run it over the full test split.

More permissive matching on saved catalogs increased selections from 55 to 56, but admitted administrative notes in inspected cases. This used no new model calls and does not establish a limit for other selectors.

7 Lessons Learned

These observations motivate recommendations, not demonstrated remedies.

Check passage selection first. The virus case motivates inspecting the selected evidence before changing calculation instructions.

Test varied layouts. Compare readers or selectors on the same documents, changing one component at a time with the solver fixed. Report selection, identifier and answer accuracy, and coverage separately. Such controlled comparisons are needed to establish whether a change improves performance.

Separate identifier and passage checks. A correctly read identifier can still belong to the wrong passage.

Report coverage. The page-image study returned evidence for 22 of 100 documents; correctness among outputs alone missed this limitation.

8 Conclusion

EVICALC obtained 8.61% official test joint accuracy. Our descriptive analyses document a case of missing relevant evidence and a page-image prototype that returned evidence for only 22 of 100 documents. They motivate testing extraction and selection changes, but do not establish a single cause of the test result or demonstrate that the proposed changes improve performance.

Limitations

This report concerns one system and task. Models, prompts, readers, and metrics differed between public validation and test experiments. The score difference therefore cannot be attributed solely to document layout. The first 120 test documents were a convenience sample. Correctness judgments in the 100-document study are approximate, and raw annotation records would be needed to resolve the remaining uncertain cases. The rule applied to saved validation outputs was developed after inspecting errors. Its score is not a prospective test of a new PDF-based system. Hosted model calls can also vary between runs.

Ethical considerations

Predictions use released PDFs without private answers or source solution lookup. A plausible calculation from an unrelated passage remains a practical risk. Saved passages and expressions support inspection, not a guarantee of correctness.

References

Anthropic. n.d.a. Claude Haiku 4.5. Claude Platform model documentation. Accessed September 16, 2026.

Anthropic. n.d.b. Claude Opus 5. Claude Platform model documentation. Accessed September 16, 2026.

Anthropic. n.d.c. Claude Sonnet 5. Claude Platform model documentation. Accessed September 16, 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

OpenAI. n.d.a. GPT-4o. OpenAI API model documentation. Accessed September 16, 2026.

OpenAI. n.d.b. GPT-4o Mini. OpenAI API model documentation. Accessed September 16, 2026.

Jyotika Singh, Fang Tu, Aziza Mirsaidova, Amit Agarwal, Hitesh Laxmichand Patel, Sandip Ghoshal, Miguel Ballesteros, Karan Dua, Yassine Benajiba, Weiyi Sun, Tao Sheng, Graham Horwood, Sujith Ravi, and Dan Roth. 2026. GSM-SEM: Benchmark and framework for generating semantically variant augmentations. *arXiv preprint arXiv:2605.07053*.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 629–633.

Retrieve, Reason, Verify: A Modular Multi-Model System for DocSem

Pranavi Tadivalasa
Oracle
pranavi.tadivalasa@oracle.com

Abstract

DocSem requires systems to answer quantitative questions over document images while also returning the evidence supporting each prediction. We present *Retrieve, Reason, Verify*, a modular system that separates document transcription, evidence localization, numerical reasoning, deterministic execution, and selective multi-model verification. Under controlled evidence grounding on 217 validation instances, Qwen3.8-Max achieves 98.62% Joint Exact Accuracy. Combining Qwen3.8-Max and GPT-5.6 Sol as independent solvers, with Gemini 3.1 Pro invoked only when their numerical predictions disagree, improves validation performance to 99.08%. On the 1,730-instance hidden test set, which presents more challenging visual and grounding conditions, including degraded document quality, non-uniform evidence identifiers, and multiple plausible evidence passages, the complete end-to-end pipeline achieves 32.7% Joint Exact Accuracy. A manual audit of 50 hidden-test instances shows that the system-selected evidence differs from the manually identified evidence in 34 cases, whereas only two cases produce different numerical answers when the evidence agrees. These observations suggest that, within the audited sample, errors are concentrated more strongly in document grounding than in downstream numerical computation once the relevant evidence has been established.

1 Introduction

Quantitative question answering over document images requires more than extracting text and performing arithmetic. Before numerical reasoning can succeed, a system must identify the portion of the document containing the relevant quantitative scenario, recover that content accurately, preserve its associated evidence identifier, and determine the computation required by the query. Errors introduced at any of these stages can invalidate an otherwise correct numerical solution.

The DocSem shared task[1] brings these challenges together in a single evaluation setting. Systems receive a PDF document and a paraphrased query and must return both a numerical answer and the visible document block identifiers supporting that prediction. Queries may refer to the target scenario only indirectly, while documents can contain multiple numerical passages, repeated entities, visually degraded content, and administrative or contextual text with substantial lexical overlap. Consequently, lexical similarity alone is often insufficient for identifying the correct evidence. A successful system must recover the document structure, distinguish semantically relevant passages from plausible distractors, and preserve the correspondence between the selected evidence and the final numerical prediction.

We address this problem with *Retrieve, Reason, Verify*, a modular pipeline organized around a separation between document grounding and numerical computation. The grounding stage reconstructs the document into identifier-preserving text blocks and localizes the passage containing the complete quantitative scenario. The reasoning stage converts the grounded problem into an executable arithmetic expression, which is evaluated deterministically. Multiple reasoning models are then combined selectively, with an additional verifier invoked only when their numerical predictions disagree. This modular design enables numerical reasoning to be evaluated independently of document-grounding quality and facilitates stage-wise analysis of end-to-end errors.

Our experiments reveal a substantial difference between reasoning under fixed evidence and complete end-to-end document processing. When the relevant evidence is held constant, the strongest evaluated reasoning models solve nearly all validation instances correctly, and selective multi-

[1] https://docinsights-workshop.github.io/docinsights-2026/shared-task/

The Proceedings of Workshop on Document Intelligence and Understanding, pages 465–476
October 24, 2026 ©2026 Association for Computational Linguistics

model adjudication further reduces residual errors. End-to-end performance, however, is substantially lower when the system must first recover the relevant evidence from the document. A manual audit of hidden-test predictions further indicates that, within the inspected sample, discrepancies between system predictions and manually derived solutions occur substantially more often in evidence selection than in numerical reasoning when the same evidence is identified. This distinction motivates treating document grounding and numerical reasoning as separate sources of error rather than evaluating the system as a single opaque prediction process.

Our main contributions are:

1. We propose *Retrieve, Reason, Verify*, a modular architecture for quantitative document question answering that explicitly separates document grounding, numerical reasoning, deterministic execution, and selective verification.
2. We evaluate multiple reasoning models under controlled evidence grounding, enabling quantitative reasoning performance to be measured independently of document-localization errors.
3. We provide a manual analysis of the complete DocSem pipeline that distinguishes discrepancies in evidence recovery from discrepancies in downstream numerical reasoning and identifies grounding as a major source of error within the audited hidden-test sample.

2 Related Work

2.1 Document Question Answering

Document question answering requires models to jointly exploit textual content and the visual structure of document images. DocVQA established a benchmark for visual question answering over documents, emphasizing the importance of reasoning over document content beyond plain-text extraction (Mathew et al., 2021). Subsequent work has increasingly incorporated multimodal representations of text, layout, and visual appearance. LayoutLMv3 jointly models textual and visual information through unified multimodal pre-training (Huang et al., 2022), while Donut adopts an OCR-free formulation that maps document images directly to structured outputs (Kim et al., 2022).

These approaches primarily address the recovery and interpretation of document content. DocSem introduces an additional requirement of evidence attribution, whereby the predicted answer must be associated with the correct evidence identifier. Document processing must therefore preserve the correspondence between textual content and its original document-level identifier. Our system maintains this association during reconstruction by representing the document as identifier-preserving text blocks prior to evidence localization and numerical reasoning.

2.2 Numerical Reasoning and Program Execution

Numerical reasoning over documents has been studied in settings that combine evidence retrieval with multi-step computation. FinQA evaluates numerical question answering over financial reports, where systems must identify relevant facts and compose executable reasoning programs (Chen et al., 2021). TAT-QA extends this setting to hybrid evidence distributed across textual passages and tables (Zhu et al., 2021). These benchmarks demonstrate that numerical question answering depends jointly on accurate evidence selection and reliable execution of the required computation.

A related line of work separates language-based reasoning from numerical execution. PAL prompts language models to generate executable programs and delegates the resulting computation to a Python interpreter (Gao et al., 2023). Program of Thoughts similarly externalizes arithmetic operations into executable reasoning steps rather than relying on implicit calculation within the language model (Chen et al., 2023). Our system follows this general paradigm by deriving a constrained arithmetic expression from localized evidence and evaluating it deterministically. This separation enables errors arising from evidence selection, reasoning formulation, and numerical execution to be examined independently.

2.3 Reasoning Robustness and Verification

The reliability of numerical reasoning can be affected by irrelevant, distracting, or semantically similar context. Shi et al. demonstrate that irrelevant information can substantially degrade mathematical reasoning performance, particularly when distractors appear plausible with respect to the query (Shi et al., 2023). GSM-SEM further examines the sensitivity of mathematical reasoning systems to meaning-preserving semantic variation (Singh et al., 2026). These findings are relevant to document-level quantitative reasoning, where multiple passages may contain overlapping entities or

numerical information even though only one provides the complete evidence required to answer the query.

Prior work has also explored inference-time aggregation as a mechanism for improving reasoning reliability. Self-consistency samples multiple reasoning trajectories from the same model and selects the final answer based on agreement across the resulting solutions (Wang et al., 2023). In contrast, our system obtains independent predictions from distinct reasoning models and performs additional verification only when their numerical outputs disagree. This disagreement-triggered strategy concentrates additional reasoning effort on uncertain instances while avoiding an additional verification step when the primary solvers already agree.

3 System Description

Given a PDF document and a query, the system predicts a numerical answer together with the identifier of the supporting evidence. Figure 1 provides an overview of the architecture. The pipeline first reconstructs the document into identifier-preserving text blocks and then localizes the passage containing the relevant quantitative scenario. From the localized evidence, a reasoning model derives an executable arithmetic expression, which is evaluated deterministically. Independent solvers operate over the same grounded input, and disagreement cases are routed to a separate verifier.

Throughout the paper, we use *grounding* to refer collectively to document reconstruction and evidence localization. Transcription recovers textual content from the document image, reconstruction associates that content with document-level identifiers, localization selects the evidence required for the target problem, and reasoning derives the corresponding computation.

3.1 Core Pipeline

The core pipeline is designed to separate document grounding from numerical reasoning while preserving the correspondence between the selected evidence and the final prediction.

Document Reconstruction Each PDF page is rendered with PyMuPDF at 150–200 DPI and transcribed using PaddleOCR with PP-OCRv6 (PaddlePaddle, 2026). Because the validation documents are visually high quality, this rendering resolution is sufficient to preserve textual and numerical content without requiring higher-resolution processing. The OCR output is segmented into text blocks while retaining the evidence identifier associated with each passage. The resulting representation contains both the reconstructed document content and the set of valid identifiers required by the subsequent localization stage.

Evidence Localization Given the query, reconstructed text blocks, and valid evidence identifiers, Qwen2.5-7B-Instruct identifies the smallest evidence set containing the complete quantitative scenario. The model additionally extracts the mathematical question expressed within the selected passage. Separating passage selection from question extraction is useful when the user query refers to the target scenario indirectly or does not reproduce the wording of the underlying quantitative problem.

Quantitative Reasoning The reasoning model receives the localized evidence, extracted mathematical question, and original user query. It first identifies the target quantity and the numerical values, units, and relationships required for the computation and subsequently produces a Python-compatible arithmetic expression. The reasoning prompt explicitly accounts for unit conversion, percentage interpretation, comparison direction, omitted or duplicated quantities, and irrelevant numerical information. Constraining the solver to produce an executable expression separates semantic reasoning from the final arithmetic evaluation.

Deterministic Execution Following prior work on program-aided numerical reasoning (Gao et al., 2023; Chen et al., 2023), the generated expression is evaluated using a restricted Abstract Syntax Tree (AST)-based Python executor. The executor permits standard arithmetic operators, unary operations, and a predefined set of numerical functions while rejecting unsupported constructs. Malformed or unparsable expressions trigger up to two structured-output recovery attempts before the instance is treated as an execution failure. This design avoids relying on the language model to perform the final arithmetic operation internally.

Selective Multi-Model Verification Two reasoning models independently solve the same grounded problem. Their numerical outputs are normalized to a common representation and compared directly. Predictions with identical normalized values are accepted without additional verification. When the

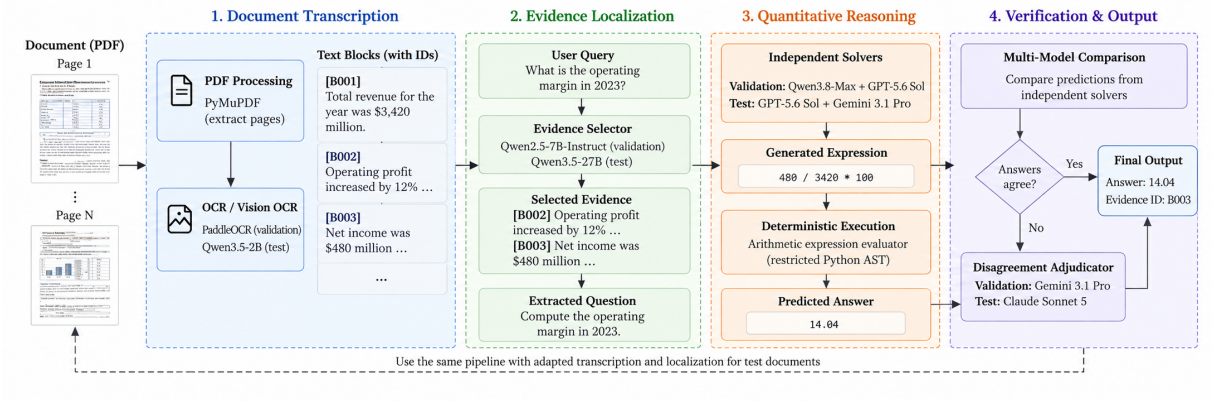


Figure 1: Overview of the proposed *Retrieve, Reason, Verify* pipeline. The document is reconstructed while preserving evidence identifiers, the relevant quantitative passage is localized, independent reasoning models derive executable computations, and disagreement cases are routed to a separate verifier.

predictions disagree, a separate verifier receives the localized evidence and both candidate derivations and independently reconstructs the required computation. Verification is therefore restricted to disagreement cases, concentrating additional reasoning capacity on instances for which the primary solvers do not reach the same result.

3.2 Hidden-Test Adaptations

The hidden-test setting presents additional challenges at the document-processing stage. Although reference answers and evidence annotations are unavailable, the input documents exhibit visual and formatting characteristics that differ from the validation data, including degraded text quality, compression artifacts, and non-uniform evidence identifiers. We therefore adapt the transcription and localization components using only observable properties of the unlabeled test documents. The interfaces among grounding, reasoning, deterministic execution, and verification remain unchanged.

Visual Transcription For hidden-test documents, we replace the OCR-based transcription used during validation with Qwen3.5-2B. Qualitative inspection of the unlabeled hidden-test documents indicated that conventional OCR pipelines, including PaddleOCR and Tesseract, were less reliable for low-contrast text and irregular identifier strings. We therefore use a vision-language model for transcription and instruct it to reproduce visible document content without normalizing capitalization, punctuation, numerical values, or identifier strings.

Format-Agnostic Block Reconstruction Unlike the validation data, hidden-test evidence identifiers do not follow a consistent prefix, length, or character pattern. The reconstruction stage therefore avoids imposing a predefined identifier format. Instead, the complete string preceding the first colon of each detected block is retained as the evidence identifier. Visually similar characters are preserved without normalization in order to maintain the document representation as faithfully as possible.

Evidence Localization The hidden-test localizer receives the reconstructed document and identifies the passage containing the target quantitative scenario. The predicted identifier is subsequently checked against the identifier set extracted during reconstruction. If the predicted identifier is invalid, localization is repeated while restricting the model to valid candidates, for at most two additional attempts. This constraint reduces identifier-formatting errors without modifying the underlying document content.

Reasoning and Adjudication Once evidence has been localized, two independent solvers derive the numerical solution from the same grounded input. Predictions with identical normalized numerical values are accepted directly. For disagreement cases, a separate adjudicator receives the localized evidence together with both candidate derivations and independently reconstructs the final computation. The hidden-test system therefore retains the same disagreement-driven verification strategy while adapting the transcription and localization components to the observed test-time document conditions.

4 Experimental Setup

4.1 Evaluation Protocol

Prompt templates, document-processing rules, and pipeline design choices are developed using the 908 labeled DocSem training instances. The 217 validation instances are held out from this development process and subsequently used for comparison of quantitative reasoning configurations. To decouple numerical reasoning from document-grounding errors, every validation solver receives the same localized evidence and extracted mathematical question. We refer to this setting as *controlled evidence grounding*.

The hidden test set contains 1,730 instances and is evaluated through the official shared-task framework. In contrast to controlled validation, hidden-test evaluation measures the complete end-to-end pipeline, including document transcription, reconstruction, evidence localization, numerical reasoning, and evidence prediction.

The two settings therefore evaluate complementary aspects of the system. Controlled validation measures reasoning performance conditional on fixed grounding, whereas hidden-test evaluation measures the effectiveness of the complete document-processing and reasoning pipeline.

4.2 Models and Configurations

We evaluate Grok 4.6 Fast, Gemini 3.1 Pro, Qwen3.7-Plus, Qwen3.8-Max, DeepSeek-R1-Distill-Qwen-32B, DeepSeek-R1, GPT-5.6 Sol, Claude Sonnet 5, and Kimi K2.5 Instant as quantitative reasoning models. Complete model references, stage assignments, and inference configurations are provided in Appendix A.

Grok 4.6 Fast is evaluated with both the initial and refined reasoning prompts, while Claude Sonnet 5 is evaluated under medium and higher reasoning-effort configurations. All remaining models use the refined structured reasoning prompt. The validation set is used to compare reasoning-model configurations after prompt and pipeline development on the training data.

The strongest validation configuration combines Qwen3.8-Max and GPT-5.6 Sol as independent solvers, with Gemini 3.1 Pro invoked only when their numerical predictions disagree. For the final hidden-test system, GPT-5.6 Sol and Gemini 3.1 Pro serve as the independent solvers, while Claude Sonnet 5 performs disagreement adjudication. Complete model and inference details are provided in Appendix A.

4.3 Prompting Strategy

Evidence localization and numerical reasoning are handled using separate prompt templates. The localization prompt instructs the model to identify the minimal evidence required to reconstruct the complete quantitative scenario rather than selecting passages solely on the basis of lexical similarity to the query. The localizer is explicitly restricted to evidence selection and does not perform the numerical computation.

The reasoning prompt requires the solver to identify the target quantity and the numerical relations necessary to answer the question before producing an executable arithmetic expression. The prompt additionally checks for unit consistency, percentage interpretation, comparison direction, duplicated quantities, and unsupported numerical values.

The adjudication prompt is designed to evaluate the underlying computation rather than select an answer according to model identity or majority preference. Given the grounded evidence and competing derivations, the adjudicator independently reconstructs the required calculation and may reject both candidate solutions when neither is supported by the evidence. Full prompt templates are provided in Appendix B.

4.4 Numerical Agreement and Execution

Before solver outputs are compared, numerical answers are converted to a common normalized representation using the same procedure throughout validation and hidden-test inference. Two predictions are considered to agree when their normalized numerical values are identical. Disagreement triggers the additional verification stage described in Section 3.

Expressions produced by the reasoning models are evaluated exclusively through the restricted AST-based executor. Invalid expressions are not evaluated directly and instead follow the structured-output recovery procedure, with at most two recovery attempts. This separation ensures that comparisons across reasoning models primarily reflect differences in problem interpretation and expression construction rather than differences in arithmetic execution.

4.5 Evaluation Metrics

We report the official DocSem metrics: *Answer Exact Accuracy*, *Evidence F1 (macro)*, and *Joint*

Model	Joint Exact Acc. (%)
Grok 4.6 (Fast, initial)	89.86
Qwen3.7-Plus	89.86
DeepSeek-R1-Distill-Qwen-32B	90.78
Kimi K2.5 (Instant)	96.31
Grok 4.6 (Fast, refined)	97.24
Gemini 3.1 Pro	97.24
Claude Sonnet 5 (medium)	97.24
Claude Sonnet 5 (higher)	97.70
GPT-5.6 Sol	98.16
DeepSeek-R1 (671B)	98.16
Qwen3.8-Max	**98.62**

Table 1: Performance of individual reasoning models on the 217 validation instances under controlled evidence grounding. Evidence is fixed and correct across configurations; consequently, Evidence F1 is 100% and Answer Exact Accuracy equals Joint Exact Accuracy.

Exact Accuracy.

Answer Exact Accuracy measures the proportion of instances for which the normalized predicted answer exactly matches the reference answer. For each task instance, evidence precision and recall are computed using set overlap between the predicted and reference evidence sets, and their harmonic mean gives the instance-level evidence F1. *Evidence F1 (macro)* is obtained by macro-averaging these scores across all instances, allowing partial credit when the predicted evidence set partially overlaps with the reference evidence.

Joint Exact Accuracy requires both the numerical answer and the complete evidence set to be correct for the same instance. We use *Joint Exact Accuracy* as the primary metric because it directly captures the joint requirement of correct numerical reasoning and exact evidence attribution.

5 Results

5.1 Validation Model Comparison

Table 1 reports the performance of individual reasoning models on all 217 validation instances under controlled evidence grounding. Because the same correct evidence is supplied to every solver in this setting, Evidence F1 is 100% across runs, and Answer Exact Accuracy therefore coincides with Joint Exact Accuracy. We consequently report Joint Exact Accuracy as the single summary measure for the validation reasoning comparison.

Qwen3.8-Max achieves the highest single-model Joint Exact Accuracy at 98.62%, corresponding to 214 of 217 validation instances. GPT-5.6 Sol and DeepSeek-R1 each achieve 98.16%, followed by Claude Sonnet 5 with higher reasoning effort at 97.70%. The relatively small differences among the strongest models indicate that several evaluated solvers achieve comparable performance once the relevant evidence has been fixed.

Configuration	Joint Exact Acc. (%)
Qwen3.8-Max	98.62
GPT-5.6 Sol	98.16
Qwen + GPT + Gemini adjudication	**99.08**

Table 2: Validation performance of disagreement-based multi-model adjudication under controlled evidence grounding.

5.2 Prompting and Reasoning Analysis

Prompt refinement has a substantial effect on Grok 4.6 Fast. The initial prompt achieves 89.86% Joint Exact Accuracy, whereas the refined structured prompt reaches 97.24%, an improvement of 7.38 percentage points.

The refined prompt introduces an explicit decomposition step before expression generation. The solver first identifies the target quantity, relevant values and units, dependencies among quantities, and potential distractors. It then verifies common sources of error, including percentage interpretation, comparison direction, unit conversion, and omitted or duplicated quantities, before producing the executable expression.

The resulting expression is evaluated separately by the restricted Python executor. This design assigns semantic interpretation and computation formulation to the reasoning model while keeping the final arithmetic evaluation deterministic. The improvement observed for Grok 4.6 Fast demonstrates that, in this setting, structured decomposition can substantially improve reasoning performance even when the underlying model and grounded evidence remain unchanged.

5.3 Multi-Model Adjudication

We next examine whether the residual errors of strong individual solvers are complementary. Qwen3.8-Max and GPT-5.6 Sol independently solve each validation instance using identical grounded evidence. Gemini 3.1 Pro is invoked only when their normalized numerical predictions disagree.

The adjudicated configuration achieves 99.08% Joint Exact Accuracy, corresponding to 215 of 217 validation instances and recovering one additional instance relative to the strongest individual solver.

This improvement indicates that the remaining errors of Qwen3.8-Max and GPT-5.6 Sol are not fully overlapping and that disagreement-based verification can resolve a subset of residual reasoning errors.

The improvement is modest in absolute terms, increasing Joint Exact Accuracy by 0.46 percentage points, and disagreement cases require an additional model invocation. Selective adjudication limits this additional inference cost to cases in which the primary solvers disagree, rather than applying verification uniformly to all instances.

5.4 Hidden-Test Performance

The final end-to-end system achieves 32.7% Joint Exact Accuracy on the 1,730-instance hidden test set. The hidden-test evaluation reports only Joint Exact Accuracy; separate Answer Exact Accuracy and Evidence F1 scores are not available for this split.

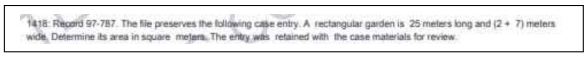
Unlike the controlled validation setting, this score incorporates errors from visual transcription, block reconstruction, evidence localization, exact identifier recovery, and numerical reasoning. The hidden-test result should therefore not be interpreted as a direct measure of reasoning degradation relative to validation. Instead, it reflects the combined performance of the complete document-processing and reasoning pipeline.

6 Manual Error Analysis

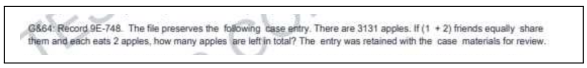
To investigate the sources of hidden-test error, we manually solve 50 instances from the 1,730-instance test set and compare the resulting evidence selections and numerical answers with the predictions produced by our system. For each instance, we inspect the corresponding document passages and determine the relevant quantitative problem, supporting evidence, and numerical solution directly from the document. Numerically equivalent answer representations are treated as agreement. Because this analysis covers only a subset of the hidden test set and relies on manual assessment rather than organizer-provided reference labels, the resulting counts are intended to characterize recurring failure modes rather than estimate the exact error distribution of the complete test set.

6.1 Grounding and Identifier Errors

Manual inspection reveals two recurring challenges in the document-grounding stage.



Visible identifier: `1418` → **Transcription:** `14I8`



Visible identifier: `G&64` → **Transcription:** `G864`

Figure 2: Representative identifier-transcription errors from the hidden test set. Although the surrounding passage remains largely readable, a local character-level transcription error changes the evidence identifier and can therefore affect exact evidence matching.

First, supporting passages are associated with opaque evidence identifiers rather than predictable labels such as `b01`. These identifiers may contain arbitrary combinations of letters, digits, capitalization, and punctuation, while exact evidence evaluation requires the predicted identifier to match the expected string.

Evidence recovery is therefore sensitive to localized transcription errors. Figure 2 illustrates two representative cases. In both examples, most of the surrounding passage is transcribed correctly, but a single character-level error changes the predicted evidence identifier. Because opaque identifiers provide little semantic redundancy, the surrounding passage often provides insufficient information to reconstruct an incorrectly recognized character.

Second, individual documents frequently contain multiple passages that are superficially relevant to the query. Administrative text may reuse query terms, while separate case descriptions may mention the same entities or contain similar numerical information. Evidence localization therefore requires identifying the passage that fully specifies the target quantitative scenario rather than relying primarily on lexical overlap.

This observation is consistent with prior findings that semantically plausible distractors can affect language-model reasoning (Shi et al., 2023). In the DocSem setting, however, an incorrect localization decision has an additional consequence: it changes the quantitative problem subsequently presented to the reasoning model.

6.2 Evidence and Answer Errors

Table 3 summarizes the relationship between the system predictions and manually derived solutions across the 50 inspected instances.

The system-selected evidence differs from the manually identified evidence in 34 of the 50 in-

System prediction vs. manual solution	Count
Same evidence, same numerical answer	14
Different evidence, same numerical answer	6
Same evidence, different numerical answer	2
Different evidence, different numerical answer	28
Total	50

Table 3: Comparison between system predictions and manually derived solutions for 50 hidden-test instances. Numerically equivalent answer representations are treated as agreement.

spected instances. The largest category consists of 28 cases in which both the evidence and numerical answer differ. Manual inspection reveals a recurring pattern in which the system localizes a different quantitative passage from the one identified during manual analysis and subsequently solves the problem contained in that passage. In these cases, the numerical disagreement appears to originate upstream from evidence localization rather than from arithmetic reasoning over the same evidence.

For example, in one inspected instance, the system selects evidence `E77` and produces an answer of 3650, whereas manual analysis identifies evidence `B74` and yields 346. The identifiers correspond to different quantitative passages in the same document, illustrating how an incorrect localization decision can determine the numerical problem subsequently presented to the solver.

A further six instances produce numerically equivalent answers despite differences between the system-selected and manually identified evidence. These cases include identifier-transcription differences as well as instances in which distinct candidate passages yield the same numerical result. Such examples demonstrate why answer accuracy alone does not fully characterize performance on a task that jointly evaluates numerical prediction and evidence attribution.

Only two of the 50 inspected instances produce different numerical answers when the system and manual analysis identify the same evidence. Within the audited sample, numerical prediction is therefore substantially more stable once grounding has been fixed. This observation is consistent with the high reasoning accuracy obtained under controlled evidence grounding, although the limited audit size and absence of organizer-provided hidden-test labels prevent conclusions about the exact error distribution of the complete test set.

Taken together, the audit suggests a stage-wise pattern of error propagation. Evidence localization determines which quantitative problem is presented to the solver, identifier recovery determines whether the supporting evidence is represented correctly for evaluation, and numerical reasoning operates only after these grounding decisions have been made. Within the inspected sample, errors in the upstream grounding stages are substantially more frequent than numerical errors when the relevant evidence has already been identified.

7 Discussion and Future Directions

The manual analysis suggests that further improvements may benefit from targeting the document-grounding stages of the pipeline. In particular, the audited examples expose a coupling between semantic evidence localization and exact identifier recovery: a system may identify the relevant passage while still failing evidence evaluation because the associated identifier is transcribed incorrectly. The following directions could reduce this dependency.

Region-Based Grounding Evidence localization need not depend on first recovering an opaque textual identifier. A system could instead preserve the spatial coordinates of detected text blocks and perform localization directly over document regions. The identifier associated with the selected region would then be recovered only when constructing the final prediction. Such a design would decouple semantic passage selection from identifier transcription and reduce the effect of character-level recognition errors on evidence localization.

Constrained Identifier Recovery The evidence identifiers appearing within a document constitute a finite candidate set. When identifier transcription is uncertain, a predicted string could therefore be reconciled against the observed candidates using character-level similarity, edit distance, or recognition confidence. This correction would operate only over identifiers already present in the document, preserving the requirement that the final output contain the exact canonical identifier expected by the evaluator.

Direct Visual Grounding An alternative is to reduce dependence on full-document transcription altogether. OCR-free document-understanding models such as Donut operate directly on visual document representations (Kim et al., 2022). A future DocSem system could first localize the relevant

page region using a vision-language model and subsequently transcribe only the selected evidence. This approach could limit error propagation from irrelevant or visually degraded regions of the document.

Grounding Verification The current verification stage operates after numerical reasoning, whereas the audit indicates that many observed errors arise earlier in the pipeline. A complementary grounding verifier could assess whether the selected passage contains all quantities, entities, and relationships required by the query before invoking the numerical solver. Disagreement between independently generated localization predictions could provide a natural trigger for this additional verification step.

Overall, these directions suggest shifting part of the verification effort from downstream numerical reasoning toward upstream document grounding. Such changes would preserve the modular structure of *Retrieve, Reason, Verify* while addressing the failure modes most frequently observed in the audited hidden-test examples.

8 Conclusion

We presented *Retrieve, Reason, Verify*, a modular system for document-grounded quantitative question answering that separates document grounding, numerical reasoning, deterministic execution, and selective multi-model verification. Under controlled evidence grounding, the strongest individual solver achieves 98.62% Joint Exact Accuracy, while disagreement-based adjudication improves performance to 99.08%.

The complete end-to-end system achieves 32.7% Joint Exact Accuracy on the 1,730-instance hidden test set. A manual audit of 50 test instances reveals substantially greater error in evidence selection than in numerical reasoning once the evidence is fixed: the system-selected evidence differs from the manually identified evidence in 34 cases, whereas only two cases produce different numerical answers when the evidence agrees.

These findings highlight an important distinction between reasoning accuracy under fixed grounding and reliability in end-to-end document processing. Within the audited sample, numerical reasoning is comparatively stable once the relevant evidence has been established, while transcription, passage localization, and exact identifier recovery introduce substantially more error. The results therefore motivate treating document grounding and numerical reasoning as separate components when designing and evaluating quantitative document question answering systems.

9 Limitations

Our analysis has several limitations. First, the hidden-test error analysis is based on 50 manually solved instances from a total of 1,730. The resulting counts are intended to identify recurring failure patterns and should not be interpreted as estimates of the exact error distribution over the complete hidden test set.

Second, organizer-provided reference answers and intermediate annotations are unavailable for the hidden test. The audit therefore relies on manual interpretation of the documents as a diagnostic reference. Although this enables qualitative comparison of system predictions with manually derived solutions, manual judgments may themselves be subject to interpretation and do not constitute official ground-truth annotations.

Third, the validation and hidden-test configurations use different transcription and localization models in response to observable differences in document quality and identifier formatting. The current experiments do not isolate the contribution of each adaptation independently. Controlled component-level ablations would be required to quantify the effects of visual transcription, passage localization, identifier recovery, and verification separately.

Fourth, the final pipeline relies on several proprietary reasoning models. Exact reproduction may therefore depend on continued access to the corresponding model versions, provider interfaces, and inference configurations. Although the deterministic execution procedure and overall modular architecture are model-independent, changes to proprietary systems may affect exact numerical reproducibility.

Finally, the observed failure patterns are specific to the DocSem setting. Datasets with cleaner document images, standardized evidence identifiers, fewer semantically similar passages, or different evidence-evaluation protocols may exhibit a different balance between grounding and numerical reasoning errors. Further evaluation on additional document question answering benchmarks would be necessary to determine how broadly these findings generalize.

References

Anthropic. 2026. Introducing Claude Sonnet 5.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Google. 2026. Gemini 3.1 pro: A smarter model for your most complex tasks.

Yupang Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091. Association for Computing Machinery.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. OCR-free document understanding transformer. In *Computer Vision – ECCV 2022*, pages 498–517. Springer.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

Moonshot AI. 2026. Kimi k2.5: Visual agentic intelligence.

OpenAI. 2026. GPT-5.6: Frontier intelligence that scales with your ambition.

PaddlePaddle. 2026. PP-OCRv6: Universal multilanguage text recognition.

Qwen Team. 2026a. Qwen3.5: Towards native multimodal agents.

Qwen Team. 2026b. Qwen3.7-plus.

Qwen Team. 2026c. Qwen3.8-max.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Jyotika Singh, Fang Tu, Aziza Mirsaidova, Amit Agarwal, Hitesh Laxmichand Patel, Sandip Ghoshal, Miguel Ballesteros, Karan Dua, Yassine Benajiba, Weiyi Sun, Tao Sheng, Graham Horwood, Sujith Ravi, and Dan Roth. 2026. Gsm-sem: Benchmark and framework for generating semantically variant augmentations.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

xAI. 2026. Grok 4.6.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Model and Prompt Details

A.1 Model Configurations

Table 4 summarizes the models used in the validation experiments and final hidden-test pipeline.

The strongest validation configuration uses Qwen3.8-Max and GPT-5.6 Sol with Gemini 3.1 Pro for disagreement adjudication. The final test system uses GPT-5.6 Sol and Gemini 3.1 Pro as independent solvers, with Claude Sonnet 5 as the adjudicator.

B Prompt Templates

This appendix provides the prompt templates used for the main model-based components of our pipeline. Dynamic inputs are represented using braces.

B.1 Evidence Localization Prompt

Evidence Localization Prompt

You are given a user query and a document divided into blocks with valid block identifiers.
Identify the smallest set of document blocks containing the complete quantitative problem relevant to the user query.
The user query may describe the desired information indirectly and may overlap lexically with unrelated passages. Treat it as a semantic retrieval cue rather than selecting evidence based only on shared words.
Do not solve the quantitative problem.
Select only the evidence required to reconstruct and solve the relevant problem. Every selected identifier must exactly match one of the provided valid identifiers.
Also extract the mathematical question stated in the selected evidence.
User query: {user_query}
Valid identifiers: {valid_ids}
Document blocks: {document_blocks}
Return the selected evidence identifiers and extracted mathematical question in the required structured format.

B.2 Quantitative Reasoning Prompt

Quantitative Reasoning Prompt

Solve the quantitative problem using only the provided evidence.
Identify the requested quantity, all relevant numerical values and units, and the mathematical relationships required to compute the answer.
Before returning the final result, verify that:

- all required quantities are included;
- irrelevant quantities are excluded;
- units are consistent or correctly converted;
- percentages and percentage-point differences are distinguished;
- the direction of comparisons is correct;
- repeated or grouped quantities are handled correctly; and
- every quantity used in the calculation is supported by the evidence.

Original user query: {user_query}
Mathematical question: {target_question}
Evidence: {evidence}
Return the result in the required structured format together with an explicit Python-compatible arithmetic expression.

B.3 Visual Transcription Prompt

Visual Transcription Prompt

Transcribe all visible document content faithfully.
Preserve the text exactly as visible, including capitalization, numerical values, punctuation, colons, line breaks, and block identifiers.
Do not summarize, paraphrase, explain, repair, or normalize the document text.
In particular, do not normalize visually similar characters in block identifiers. Reproduce each identifier exactly as it appears in the document.
Do not add Markdown formatting or commentary.
Return only the document transcription.

B.4 Evidence Identifier Correction Prompt

Evidence Identifier Correction Prompt

One or more selected evidence identifiers are invalid.
Select the supporting evidence again using only the valid identifiers provided below. Copy identifiers exactly as provided and do not modify, normalize, shorten, or reconstruct them.
The selected evidence must contain the complete quantitative problem relevant to the user query.
Valid identifiers: {valid_ids}
User query: {user_query}
Document blocks: {document_blocks}
Return valid evidence identifiers together with the extracted mathematical question in the required structured format.

B.5 Disagreement Adjudication Prompt

Disagreement Adjudication Prompt

Independently verify the quantitative problem using the provided evidence.
Do not select an answer based on the identity of a solver or its aggregate performance. Reconstruct the required computation directly from the evidence.
Check the requested quantity, relevant numerical values, units, calculation direction, and arithmetic in both candidate solutions.
If one candidate is correct, return that answer. If neither candidate is correct, independently derive the correct answer from the evidence.
Mathematical question: {target_question}
Evidence: {evidence}
Candidate solution 1: {candidate_1}
Candidate solution 2: {candidate_2}
Return the verified final answer in the required structured format.

Stage	Model / Configuration	Role
Validation OCR	PaddleOCR (PP-OCRv6) (PaddlePaddle, 2026)	Document transcription
Validation localization	Qwen2.5-7B-Instruct (Yang et al., 2024)	Evidence localization
Validation reasoning	Grok 4.6 (Fast) (xAI, 2026)	Quantitative reasoning
Validation reasoning	Gemini 3.1 Pro (Google, 2026)	Quantitative reasoning
Validation reasoning	Qwen3.7-Plus (Qwen Team, 2026b)	Quantitative reasoning
Validation reasoning	Qwen3.8-Max (Qwen Team, 2026c)	Quantitative reasoning
Validation reasoning	DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025)	Quantitative reasoning
Validation reasoning	DeepSeek-R1 (671B) (DeepSeek-AI, 2025)	Quantitative reasoning
Validation reasoning	GPT-5.6 Sol (OpenAI, 2026)	Quantitative reasoning
Validation reasoning	Claude Sonnet 5 (Anthropic, 2026)	Quantitative reasoning
Validation reasoning	Kimi K2.5 (Instant) (Moonshot AI, 2026)	Quantitative reasoning
Validation ensemble	Qwen3.8-Max + GPT-5.6 Sol	Independent solvers
Validation adjudication	Gemini 3.1 Pro	Disagreement adjudication
Test transcription	Qwen3.5-2B (Qwen Team, 2026a)	Visual transcription
Test localization	Qwen3.5-27B (Qwen Team, 2026a)	Evidence localization
Test reasoning	GPT-5.6 Sol + Gemini 3.1 Pro	Independent solvers
Test adjudication	Claude Sonnet 5	Disagreement adjudication

Table 4: Models used in the validation experiments and final hidden-test pipeline. Citations refer to the corresponding model release, model card, or official provider documentation.

B.6 Structured-Output Recovery Prompt

Structured-Output Recovery Prompt

Your previous response could not be parsed or contained an invalid final expression.
Re-evaluate the provided evidence and return a complete response in the required structured format.
Do not omit any required field. The arithmetic expression must be complete, numerical, and executable by the restricted Python evaluator.
Return only the corrected structured response.

A Cost-Efficient Vision-Text Cascade for Document QA: What Resolution Buys and What It Doesn't

Urjita Bedekar
Independent Researcher
urjita9@gmail.com

Abstract

This paper presents a two-stage vision-text cascade, submitted to the DocSem shared task at DocInsights Workshop, EMNLP 2026. A cheaper vision model (Claude Haiku 4.5) locates and transcribes the block that answers the question, and a stronger text-only model (Claude Sonnet 4.5) turns it into an arithmetic expression, which Python evaluates. Keeping the expensive model off the page images reduces cost by 14% against using one model for both, and the system reaches 88.02% answer accuracy on validation, up from 82.95%. On the test split the same system scores 7.28%, and this paper diagnoses the difference. The main constraint is block selection: test documents contain decoy blocks that repeat the question's words while asking something else. Prose inside the blocks is transcribed almost perfectly; identifiers, which carry no redundancy, run at 75–81%. I also find that the two splits are stored at different resolutions, 220 dpi against 71–79 dpi, and that enlarging a low-resolution page before sending it improves identifier transcription by 58 points. The gain stops at the provider's 1568 px cap, above which rendering higher delivers an identical image and accuracy does not meaningfully improve.

1 Introduction

Recent advances in Natural Language Processing for Information Retrieval have given rise to document-based question answering. This is an important case, as much of the data exists in documents. Document understanding and question answering over document images has shown remarkable progress (Mathew et al., 2021). The purpose of this system is to return an answer along with a citation to the passage it came from, so a reader can check it (Rashkin et al., 2023). This can be used in a wide variety of applications including but not limited to clinician notes, meeting notes, or product-specific information for an enterprise chatbot.

It is a common assumption that a system's behaviour on the test set is reliably predicted by its behaviour on the validation set. This did not hold here. The paper proposes a cost-efficient system which reached 88.02% accuracy on the validation set, up from 82.95% for a single model. The cost-efficient approach used here, an LLM cascade, is described by Chen et al. (2023). Evidence F1 reached 1.00 for 62 of the 69 teams on the validation leaderboard, while answer accuracy across those same systems ranged from 23% to 100%, so the attribution metric separates almost nobody. OCR (Smith, 2007) was also studied, but the watermark contaminated the extracted text and only 21% of identifier tokens were recovered correctly. This makes a VLM the more practical choice for a corpus whose PDFs contain no extractable text layer. To process multi-page image documents under strict resource constraints, the system uses a two-stage vision-text cascade. Stage 1 uses a lightweight, cheaper VLM to locate and transcribe blocks from the PDF. Stage 2 passes the resulting text tokens, and zero image tokens, to a stronger model which returns a formal arithmetic expression that is then evaluated by a Python interpreter. Gao et al. (2023) show that a stronger approach is one where the model reads the problem and generates the reasoning steps as a program.

Although the cascade reached 88.02% answer accuracy on validation, it scored 7.28% on the test set. Processing single pages rather than the full 8–9 page document recovered at least four points of joint accuracy. The larger cause is block selection: test documents contain decoy blocks that repeat the question's words while asking something else, and 18.9% of the blocks the system selected across the test split match that template. Confabulation by the VLM is present, with about one in four returned identifiers not appearing in the document, but it is too small on its own to account for the gap. This report analyses these failure modes in detail,

The Proceedings of Workshop on Document Intelligence and Understanding, pages 477–482
October 24, 2026 ©2026 Association for Computational Linguistics

along with negative results on standard methods: 3-sample self-consistency (Wang et al., 2023) gave no accuracy gain, and instruction-based reasoning rules stopped working when chain-of-thought (Wei et al., 2022) output is suppressed.

2 Task and Data

The dataset is the one released by Singh et al. (2026). Each instance gives you a PDF, a paraphrased question, and asks for two things back: an `answer` and an `evidence` field. The answer is a plain integer. The evidence is a list of the block identifiers the answer came from, as printed on the page. Answers are scored by exact match after normalising, evidence by macro-averaged F1 over the sets, and joint accuracy requires both to be right on the same instance. Joint accuracy is what the leaderboard ranks on.

There are 908 labelled training instances, 217 validation instances whose labels the organisers hold, and 1,730 unlabelled test instances. Every one of the 908 training answers is a plain positive integer, with no units, commas, decimals or minus signs.

The documents have no text layer. Each page is a single JPEG image. The text is drawn, not stored, so pulling it out with a PDF library returns nothing. That leaves two ways in: optical character recognition, or a vision-language model.

The two splits are built differently. Training and validation documents are titled *Regional Operations Brief*, run to two pages, number their blocks `b01:`, `b02:` and so on, and hold one solvable problem each. Test documents run 8 or 9 pages, and use different identifiers instead such as `R&25:`, `q-763U:`, `RAF-D4A:`, `74/D4:`. They also hold far more blocks.

The splits are also stored at different resolutions. Since every page is one JPEG, you can work out the resolution the corpus actually stores its text at: divide the image's pixel dimensions by the size it is placed at on the page. Across the first six documents of each split the answer is consistent, and the two ranges do not overlap (Table 4). An identifier set at about 8 pt is roughly 24 pixels tall at 220 dpi and roughly 8 pixels tall at 79 dpi before JPEG compression, and before the diagonal TESTING COPY watermark is laid over it.

3 System

This system is a two-stage cascade built for cost efficiency. Both stages are prompted, not trained, and both run through Anthropic's API using Claude Sonnet 4.5 and Claude Haiku 4.5. I tried several approaches before settling on this one. The final design uses the cheaper model to locate the right block and the more expensive model to do the reasoning, which took joint accuracy from 82.95% for a single-model version to 88.02%. The system underperformed on the test set, and I identify several failure modes for that below. The stages are described here.

3.1 Rendering

The PDFs have no extractable text layer, so there is no straightforward way to get the text out. I tried both OCR and a VLM. OCR was not effective here: the watermark corrupted Tesseract's character recognition into garbled strings such as `litenms iW` in place of "items in the", and `secnned relliow` in place of "second review". It recovered only 21% of identifiers correctly. Since VLMs handle layout structure as well as text, I went with a VLM. The provider downscales any image wider than 1568 pixels on the long edge, so I chose 135 dpi as a balance between the model's token budget and keeping small text legible.

3.2 Version 1: Single-Sonnet baseline

The baseline sends all rendered page images of a document to Sonnet 4.5 in a single API call. Sonnet both locates the block and computes the answer, returning the evidence identifier and the arithmetic together. This scored 82.95% answer and joint accuracy on the 217 validation tasks, at an average of $0.0137 per instance and $2.97 in total. Evidence F1 was 1.00.

3.3 Version 2: Cascade – Cheap Model Locates, Strong Model Computes

Haiku processes the image tokens to find the target block and transcribe its text, and Sonnet receives text only. Haiku initially returned the identifier with its trailing colon, which is not an exact match against the gold label, so I strip it with `clean_id()`. To keep arithmetic errors out of the model's hands, Python evaluates the returned expression locally. This gave the best validation accuracy, 88.02% at $0.0120 per instance. On the test set it scored 7.28%. Section 4 works through why: the largest

factor is block selection, with lower image resolution (71–79 dpi) and confabulation on high-entropy identifiers contributing.

3.4 Version 3: Per-Page Rebuild

Per-page isolation. To reduce confabulation across 8–9 page documents, v3 renders each page separately at 135 dpi and sends one API call per page.

Explicit block selection. The page prompt checks whether a page holds a block beginning `Record <id>. The file preserves the following case entry.`

Tuning the prompt. Every version was tested on the same held-out 20 instances. Tuning meant adding worked examples and rules about how to deliberate. The differences between versions (16/20 for the basic cascade, 18/20 with quantity rules and CoT) come from real changes rather than from fitting the prompt to those instances.

Batch infrastructure. Page images are built in parallel and the resulting requests are submitted through Anthropic's Batch API.

Refusal fallback. On some test instances the compute stage came back with `stop_reason = refusal` and no content. The request succeeds, so nothing throws an error, it just produces nothing. Since a missing answer counts as wrong, those instances would have been lost. I check `stop_reason` and whether any content came back, and send the same prompt to Haiku 4.5 instead, which answers them.

4 Analysis and Results

Evaluation follows the benchmark's own definitions: normalised exact match for numeric answers, exact set match for block evidence tokens, and joint accuracy, which requires both to be right on the same instance. System configurations were evaluated on a held-out development set (n=20, training instances 200–219, kept apart from the instances 1–5 used to develop the prompts) and on the official validation split (n=217).

4.1 Main System Performance & Pipeline Efficiency

Table 1 gives the official scores. On the validation split the cascade (V2) reached 88.02% answer accuracy with an evidence F1 of 1.00, improving on the single-model baseline (V1) by just over five points. On the held-out test split the same system, unchanged in every respect, obtained 7.28% answer accuracy. A revised system (V3), which issues one request per page rather than one per document, scored 11.21% joint accuracy.

Split	System	Answer	F1	Joint
Validation	V1: single model	82.95	1.00	82.95
Validation	V2: cascade	**88.02**	1.00	**88.02**
Validation	V2 + refinements	87.56	1.00	87.56
Test	V2: same system	**7.28**	10.17	—
Test	V3: per-page	—	—	**11.21**

Table 1: Official scores (%). System V2 is the only system evaluated on both splits.

Configuration	Answer	Evidence	$/inst.
Single model, vision + arith.	17/20	—	0.0140
Cascade	16/20	20/20	**0.0082**
+ CoT + quantity rules	**18/20**	20/20	0.0120
+ 3-sample self-consist.	18/20	20/20	0.0282

Table 2: Configurations on held-out training instances 200–219 (n=20).

The 88.02-to-7.28 comparison is the only controlled one available to me: identical code, identical prompts, identical models, two splits. It is an 81-point collapse, and the remainder of this paper is an account of it.

Table 2 reports the configurations compared during development, on training instances 200–219 held out from prompt construction. Four observations follow from it, and one goes against conventional thinking.

First, the cascade architecture is substantially cheaper. Moving the stronger model off the page images and onto approximately 300 tokens of extracted text reduces cost from $0.0140 to $0.0082 per instance, a 41% reduction, at 16/20 against the baseline's 17/20. At $n = 20$ and accuracy near 0.9 the standard error of a proportion is roughly ± 13 points.

Second, the configuration actually submitted costs $0.0120, a 14% reduction rather than 41%, and scores 18/20 on development data and 88.02% on validation, against the baseline's 17/20 and 82.95%. That configuration dominates the single-model baseline on both accuracy and cost, which is the claim made. The 41% figure belongs to a configuration I did not submit, and I separate the two because the distinction is easy to lose.

Third, the two prompt interventions do nothing in isolation and something together. Chain-of-thought alone leaves accuracy at 16/20. The

Configuration	$/inst.	1,730 inst.
Single model	0.0140	$64
Cascade	0.0082	$29
Cascade, Batch API	0.0041	**$15**

Table 3: Cost. Rates are $1/$5 per million input/output tokens for Haiku 4.5 and $3/$15 for Sonnet 4.5.

instruction to use every stated quantity, applied without visible reasoning, also leaves it at 16/20. Applied together, at identical prompt text, accuracy moves to 18/20.

Finally, self-consistency is a clear negative result in this task. Three independent samples per instance agreed on all 20 instances tested, so the majority vote is identical to a single sample while costing $0.0282, which is twice the single-model baseline for no measured change in accuracy.

The rates in Tables 2 and 3 are measured on the two-page development and validation documents. The single-model rate held at $0.0137 over the full 217-instance validation split, a run that cost $2.97, against $0.0140 on the twenty development instances. Test documents run to eight or nine pages, so the locate stage costs more there while the text-only compute stage does not change; the 1,730-instance totals are projections on that basis.

4.2 What Resolution Buys

The two splits are not stored the same way. Every page is a single JPEG, so the resolution the corpus holds its text at can be computed directly: divide the image's pixel size by the size it is placed at on the page. Across the first six documents of each split the answer is consistent and the ranges do not overlap (Table 4). Training pages are 220 dpi. Test pages are 71–79 dpi. An identifier set at about 8 pt is roughly 24 pixels tall in the first case and roughly 8 in the second, before JPEG compression and before the watermark is laid over it. This makes identifiers harder to read, but it does not explain the collapse. Table 5 shows what happens when I ask a model to transcribe every identifier on three pages of a test document and score the result against 48 identifiers I read by hand. At 135 dpi, Haiku recovers 36 of 48 and Sonnet recovers 39. At the source's own 79 dpi, Sonnet recovers 11. Identifiers are the hardest thing on the page because there is nothing to guess from. A blurry word in a sentence can be worked out from the words around it. `R09-A7F` cannot, because every character stands alone. When the model cannot read the string it returns something shaped like an identifier that is not in the document, because filling in a likely answer is what it does once the evidence runs out.

Split	Pages	Embedded JPEG	Native
Train (6 of 6)	2	1786×2526	**220 dpi**
Val (6 of 6)	2	1310×1853	**∼162 dpi**
Test (4 of 6)	8–9	572×809	**71 dpi**
Test (2 of 6)	8	643×910	**79 dpi**

Table 4: Resolution of the single JPEG comprising each page, over the first six documents of each split. An 8 pt identifier is roughly 24 px tall at 220 dpi and roughly 8 px at 79 dpi.

Model	Render	Exact	Not found
Haiku 4.5	135 dpi	36/48 (75%)	12
Sonnet 4.5	135 dpi	**39/48 (81%)**	9
Sonnet 4.5	79 dpi	**11/48 (23%)**	27

Table 5: Identifier transcription against 48 hand-verified identifiers on three pages of `task_010001`. The third row renders at the source's native resolution.

Enlarging the image matters far more than the model does. The two Sonnet rows are the same model on the same JPEG, 58 points apart, against six points for switching model at the same resolution. The likely reason is how the encoder splits an image into fixed patches: at 79 dpi an 8-pixel glyph sits inside one patch and is averaged into a single token, while the enlarged version spans two or three. Enlarging adds no detail, but it adds tokens, and for small text the tokens are what is short.

Resolution does not degrade performance smoothly. The drop from 220 to roughly 162 dpi between the development and validation splits cost nothing measurable, while the further drop into the test range coincided with an 81-point fall. Legibility appears to fail at a threshold rather than decline in proportion.

Accuracy also depends on what is being read. Table 6 measures three kinds of content on the same pages at four render resolutions, and the ordering holds at every one: word recall reaches 99.5% by

Render	Prose	Numbers	Identifiers
79 dpi (native)	80	59	19
100 dpi	98	87	62
135 dpi	**100**	96	**77**
200 dpi	100	**98**	71

Table 6: Recovery by content type and resolution (%), Haiku 4.5 on eight blocks from pages 1–3 of `task_010001`. Prose is unordered word recall; numbers and identifiers are exact match, so the columns are not directly comparable.

135 dpi, numbers 96%, and identifiers 77%, which is the highest they reach anywhere in the range tested. The difference is redundancy. A misread word is constrained by the words around it; a misread digit only by the arithmetic; an identifier such as `R24-82F` by nothing at all. Resolution is worth most where redundancy is least, and exact match over a high-entropy string admits no partial credit.

4.3 Classical OCR Baseline Evaluation and Rejection

Classical OCR (Tesseract) was evaluated and rejected. Against the 48 hand-verified identifiers it recovered 10 exactly (21%), which is disqualifying under exact-match evidence scoring; throughput was 13.67 s per page across eight threads, projecting to roughly 52 hours for the test split.

Its failures are, however, of a different kind. Consider `task_011327`, the instance where the vision pipeline returned an invented identifier, record number, and both quantities. OCR returned:

```
Grace counted (13+ 17)litenms
iW'the first review anct 13 in
the secnned relliow.
```

The prose is corrupt; the quantities are exact, and they are the ones the gold answer requires. `litenms iW'the` is self-evidently a failure to read. `R09-A7F` is indistinguishable from a genuine identifier without the document in hand. Two methods failed on the same page and only the weaker one signalled it.

5 Conclusion

Moving the strong model off the page images and onto extracted text made the system both cheaper and more accurate on validation: 88.02% against 82.95%, at $0.0120 against $0.0140 per instance. The same system, unchanged, scored 7.28% on the test split, and two things explain it. The first is selection: test documents contain decoy blocks that reproduce the question's words while asking something else. The second is resolution. The splits are rendered differently. Training pages hold their text at 220 dpi, validation at 162, and test at 71 to 79. The step from 220 to 162 cost nothing measurable. The step below it coincided with an 81-point fall, which suggests legibility fails at a threshold rather than declining in proportion. Resolution at inference time matters more than model choice. Rendering the same JPEG larger took identifier recovery from 23% to 81%, while moving from Haiku to Sonnet at the same resolution took it from 75% to 81%. Enlarging adds no detail, but it gives small glyphs more tokens to occupy. Since the resolution cap applies to whatever image is sent, sending page regions rather than whole pages might give each block more of that budget.

Limitations

Identifier transcription and block selection both contribute to the collapse, and this paper does not measure their relative weight. Because the content words overlap between the gold block and the decoys, the ranking ties, and the choice falls to whichever candidate happens to appear first in the results list. Scoring candidates on what a block asks rather than which words it contains separates the target from all three decoys in the document I enumerated; I could not evaluate this at scale, since test labels were not released. The block filter used on the test split was derived from the released test documents rather than from training data. Test scores were withheld after the first attempt, so the diagnosis in Section 4 proceeded without gold labels. The decomposition of the collapse into identifier reading and block selection is consistent with the observed score but is not measured against it.

Acknowledgments

I thank the organizers for putting this shared task together. The system, experiments, and measurements reported in this paper are my own. An AI assistant was used solely to improve the language of the original draft, which I reviewed and verified against my own experiment logs.

References

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2023. Measuring attribution in natural language generation models. *Computational Linguistics*, 49(4):777–840.

Jyotika Singh, Fang Tu, Aziza Mirsaidova, Amit Agarwal, Hitesh Laxmichand Patel, Sandip Ghoshal, Miguel Ballesteros, Karan Dua, Yassine Benajiba, Weiyi Sun, Tao Sheng, Graham Horwood, Sujith Ravi, and Dan Roth. 2026. Gsm-sem: Benchmark and framework for generating semantically variant augmentations. *arXiv preprint*.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*.

Xuezhi Wang et al. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

Author Index

Agu, Nkechinyere Nneka, 138
Ahuja, Naman, 196
Alon, Alvin Sarraga, 244
Argerich, Mauricio Fadel, 12
Arun, Govind, 155
Aydogan, Murat, 282

Baral, Rishabh, 196
Basile, Pierpaolo, 409
Baskin, Chaim, 357
Batista-Navarro, Riza, 244
Bedekar, Urjita, 477
Berker, Asya, 282
Bhat, Savita, 344
Biswas, Reetam, 217
Burlova, Albina, 287
Butakov, Nikolay Alekseevich, 400

Changalidis, Anton, 303
Chaudhary, Manav, 81
Cherepanov, Artem, 287
Chitale, Maitreya Prafulla, 81
Chizhov, Pavel, 303
Chun, Kent Chiu Jin, 232
Cordeiro, Jhonata Fernandes, 448

De Rosa, Daniele, 244
Deshpande, Aaditya, 344
Du, Quan, 324

Fahim, Tamjid Hasan, 440
Fajmut, Ales, 186
Friedl, Johannes, 244
Fürst, Jonathan, 12

Gabbasov, Timur, 400
Gangopadhyay, Tryambak, 138
Garimella, Aparna, 196
George, Ajo Babu, 155
Ghafouri, Hadis, 163
Giryes, Raja, 357
Godara, Divya, 460
Goldstein, Noon Pokaratsiri, 217
Grabovoy, Andrey, 174
Gupta, Harshit, 81
Gupta, Sachin, 460
Gupta, Vivek, 196

Han, Rinne, 138
Harary, Sivan, 357
Harish, Srinivas, 73
Healey, Jennifer, 428
Huo, Yifu, 324
Huong, Nguyen Thu, 186

Inuzuka, Marcelo Akira, 448

İncidelen, Mert, 282

Jin, Yichao, 232
Joseph, Sebastian Antony, 428

Kadusabe, Provia, 1
Karande, Shirish, 344
Kashkash, Yamen, 282
Kaushik, Abhishek, 1
Kazanaviciute, Vaiva, 303
Ke, Kenneth Zhu, 232
Kharaghani, Sepideh, 30, 46
Khodorchenko, Maria, 400
Kimhi, Moshe, 357
Kocbek, Primoz, 186
Krishna, Sidharth N, 155
Kumar, Vishnu Prasad Vijaya, 303
Kwan, Ching Yee Sonia, 232

Langlais, Pierre-Carl, 303
Lawless, Fiona, 1
Li, Chenyang, 100
Li, Junyi Jessy, 428
Li, Yiyang, 112
Liu, Zhuchenyang, 385

Ma, Tianyi, 112
MacDonald, Kurt, 46
Maksimov, Aleksandr, 400
Meštrović, Katarina, 186
Miao, Ruizhong, 100
Mikat, Aftab Hossain, 440
Murugesan, Keerthiram, 112
Musacchio, Elio, 409

Nenkova, Ani, 428
Neumann, Günter, 217
Neverov, Iaroslav, 303

Oguz, Cennet, 217

Pereira, Eduardo Fraga, 448
Posada, Alejandro, 30, 46
Pottanigari, Nikhil Reddy, 30, 46

Qian, Yiyue, 138
Queiroz, Marcos Paulo C M, 448

Raghunathan, Anand, 63
Rao, Susie Xi, 155
Ravi, Sujith, 100
Rivlin, Ehud, 357
Roth, Dan, 100

Salekin, Md Sirajus, 138
Sandalov, Aleksandr, 400
Schwartz, Eli, 357
Selvam, Surya, 63
Semeraro, Giovanni, 409
Shabtay, Nimrod, 357
Sheng, Tao, 100
Shetye, Ketaki Mangesh, 81
Shrivastava, Manish, 81
Siciliani, Lucia, 409
Siu, Alexa, 428
Song, Weiyang, 232
Spector, Artem, 357
Stasenko, Anastasia, 303
Steffen, Jörg, 217
Stiglic, Gregor, 186

Tadivalasa, Pranavi, 465
Tian, Fengning, 324

Uboldi, Asia Beatrice, 409
Upadhyay, Ritam, 196

Vadacchino, Saverio, 30, 46
Varma, Vasudeva, 81
Voznyuk, Anastasia, 174

Walser, Christoph Marjan Markus, 12
Wang, Chenglong, 324
Wang, Rongguang, 100
Wang, Yushuo, 232
Wang, Yuying, 100
Wang, Zehong, 112
Wang, Zichao, 428
Wang, Ziyi, 385
Wong, Chong Hwee, 232
Wong, Tiong Kiat, 232

Xiao, Tong, 324
Xiao, Yu, 385
Xing, Shunjie, 324
Xu, Haosong, 324

Yamshchikov, Ivan P., 303
Yang, Di, 324
Ye, Yanfang, 112
Yuxuan, Han, 232

Zampedroni, Mark Federico, 409
Zhang, Chunliang, 324
Zhang, Chuxu, 112
Zhang, Ruiyang, 273
Zhang, Shinan, 138
Zhang, Yao, 385
Zhang, Ying, 30, 46
Zhang, Zheyuan, 112
Zhao, Jingyuan, 232
Zhu, JingBo, 324
Zhu, Ziming, 324