

Benchmarking Aggregation Queries over Unstructured Text

Haojia Zhu¹, Jiahui Jin^{1,4}, Haoyu Li¹, Qinyuan Xu², Yuxi Liu¹,
Zihan Zhou¹, Hanchen Qiu¹, Jiaoyan Chen³, Jinghui Zhang¹

¹Southeast University, ²Imperial College London, ³The University of Manchester,

⁴Key Laboratory of Computer Network and Information Integration, MOE, China

{zhuhaojia, jjin, lihaoyu}@seu.edu.cn, qx225@ic.ac.uk,

{yuxiliu, zzhahn, qiuhanchen2002}@seu.edu.cn,

jiaoyan.chen@manchester.ac.uk, jhzhzhang@seu.edu.cn

Abstract

Aggregation query over free text is a long-standing yet underexplored problem. It requires exhaustive evidence collection, where systems must “find all” satisfying entities rather than merely “find one” plausible answer. Existing paradigms are not designed for this completeness requirement, leading to incomplete retrieval and systematic undercounting. This limitation stems from the lack of suitable benchmarks. In this work, we formalize corpus-bounded entity-level aggregation querying over text, where answers must be grounded in the given text corpus. We introduce AGGBENCH, an aggregation benchmark constructed from finance reports and research papers through a corpus-grounded pipeline with LLM-assisted human validation. AGGBENCH contains 498 aggregation queries, including both single- and multi-constraint queries with evidence-supported gold answers. Experiments on representative basic RAG, structured and schema-first and agentic methods show that existing systems struggle to preserve evidence coverage and accurately count satisfying entities, highlighting the need for completeness-oriented methods for aggregation over unstructured text. The data and code are available in <https://github.com/luck-seu/AGGBench>.

1 Introduction

Aggregation queries emerge in many real-world analytical tasks that require systematic inspection of large unstructured text collections, such as counting how many entities in a corpus satisfy natural-language constraints. They are common in domains including research (Tsafnat et al., 2014; Marshall and Wallace, 2019), finance (Xie et al., 2024), law (Niklaus et al., 2024), and investigative analysis (Spangher et al., 2025). Unlike standard QA, which can often be solved by finding one plausible answer, aggregation over text requires complete entity coverage. Figure 1 shows a financial example:

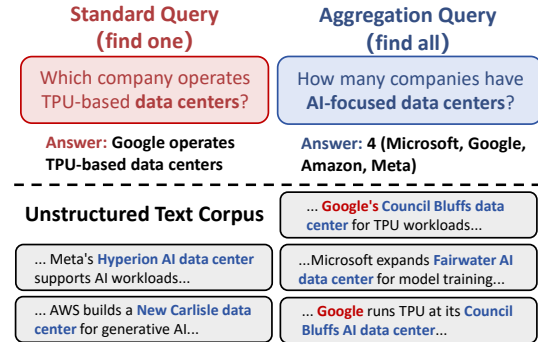


Figure 1: Examples of Aggregation Query and Standard Query over Unstructured Data.

How many companies have AI-focused data centers? Here, the target entities are company, and the phrase *have AI-focused data centers* specifies a constraint that must be verified from textual evidence. Answering the query therefore requires all qualifying data centers to be found, grounded, deduplicated, and counted. While such operations are straightforward over clean structured tables, most real-world data remains unstructured text (Harbert, 2021; Deng et al., 2025), making aggregation over text a practical but underexplored problem.

Some existing solutions are applicable to this problem, but they do not fully address this setting. (1) Structure and schema-first transformation. Since querying has been extensively studied over structured data, a natural solution is to first convert a text corpus into relational or graph databases and then apply Text-to-SQL or symbolic querying methods (Yu et al., 2018; Zhong et al., 2017). However, this requires costly preprocessing, and the resulting repositories are often incomplete or noisy (Katti et al., 2021; Siino et al., 2024). Recent semantic-operator systems relax purely structured operations with LLM-powered semantic processing (Liu et al., 2025; Patel et al., 2025; Shankar et al., 2025). However, aggregation in these systems still depends on prepared schemas or user-specified pipelines. (2) Retrieval-augmented reasoning. RAG frameworks (Lewis et al., 2020; Asai et al., 2024) and

agentic frameworks (Yao et al., 2023; Lee et al., 2024; Wu et al., 2024; Roucher et al., 2025; Li et al., 2025) enable LLMs to reason directly over the retrieved passages. But these methods were designed for normal QA, where the goal is to retrieve some highly relevant evidence, not to obtain complete coverage. Moreover, relevant mentions may exceed the fixed top- k retrieval window (Karpukhin et al., 2020), and retrieval omissions can directly lead to undercounting.

The lack of effective methods for aggregation querying is further rooted in a corresponding gap in evaluation. Existing QA benchmarks focus on factoid or open-domain retrieval tasks, where datasets such as HotpotQA (Yang et al., 2018) and Wiki-MultiHopQA (Ho et al., 2020) contain fewer than 5% of aggregation questions. Numerical reasoning datasets (e.g., DROP (Dua et al., 2019)) mainly test number extraction or comparison rather than complete entity-set identification. Recent benchmarks such as WideSearch (Wong et al., 2025) move closer to large-scale information collection, but target open-domain web search rather than corpus-bounded aggregation over a fixed text collection. Therefore, it is urgently required to construct benchmarks that cover aggregation querying scenarios as well as their evaluation.

Constructing such a benchmark is challenging because aggregation queries require complete entity sets with evidence-level grounding. Annotators must identify all satisfying entities within realistically sized text corpora and verify each one with supporting evidence, which makes exhaustive manual annotation highly expensive.

To fill this gap, we introduce AGGBENCH, a benchmark for corpus-bounded aggregation queries over unstructured text. We first formalize aggregation querying as an entity-level task in which a system must identify all entities of a target type that satisfy the query constraints and ground them in textual evidence. From finance and research corpora, we extract recurring entity types and candidate constraints and verbalize them into queries. Then we obtain gold entity sets and supporting evidence through LLM-assisted human annotation. We evaluate 15 representative methods, including basic rag, structured and schema-first and agentic methods. The results reveal that existing systems often fail to preserve evidence coverage and therefore lose evidence coverage and undercount satisfying entities, highlighting the need for completeness-oriented evaluation and methods.

Contributions:

- We introduce aggregation querying over unstructured text as a corpus-bounded entity aggregation task requiring complete, evidence-grounded identification of entities satisfying compositional natural-language constraints.
- We design a corpus-grounded benchmark construction pipeline and build **AGGBENCH**, a completeness-oriented benchmark with 498 aggregation queries, whose gold answers are annotated through LLM-assisted human annotation.
- We systematically evaluate representative methods on **AGGBENCH** across multiple LLM backbones and paradigms, showing that current systems often lose evidence coverage and undercount satisfying entities.

2 Related Work

We review benchmarks related to aggregation queries. Most QA benchmarks do not explicitly account for aggregation queries or completeness-oriented evaluation. Multi-hop QA datasets such as HotpotQA (Yang et al., 2018), ComplexWebQuestions (Talmor and Berant, 2018), MuSiQue (Trivedi et al., 2022), QASC (Khot et al., 2020), and 2Wiki-MultiHopQA (Ho et al., 2020) mainly require a small set of supporting passages, while aggregation questions are rare. Numerical reasoning datasets such as DROP (Dua et al., 2019) focus on arithmetic rather than exhaustive entity retrieval. Compared with existing QA benchmarks, which primarily evaluate the correctness of one or a few answers using metrics such as EM or F1, AGGBench evaluates corpus-level aggregation completeness through entity recall, chunk recall, and counting error. Although AGGBench contains 498 queries, each query requires exhaustive corpus-level entity discovery and evidence verification followed by multi-annotator validation, making its annotation cost fundamentally different from standard single-answer QA benchmarks.

Recent benchmarks have begun to study list-based or aggregation-style question answering (Ghosh et al., 2022; Katti et al., 2021; Lee et al., 2023; Zhang et al., 2025b; Dumitru et al., 2025). While they require multiple answer items or aggregate information across conversational logs and temporal records, they are often built in settings where metadata, structured relations, or subject-centric timelines can support the aggregation pro-

cess. Broad information-seeking benchmarks evaluate large-scale collection over Web or dynamic corpora (Yang et al., 2024; Wong et al., 2025; Lan et al., 2025), but focus on open-ended search rather than controlled, corpus-bounded aggregation with verifiable entity-level evidence. Our benchmark instead enables completeness-oriented evaluation and fine-grained diagnosis under large retrieval spaces.

Structured database benchmarks also cover aggregation through SQL or graph query languages (Zhong et al., 2017; Yu et al., 2018; Lee et al., 2021; Li et al., 2023; Lei et al., 2025), but assume structured schemas and records. Our benchmark instead studies completeness-oriented aggregation over free text.

3 Problem Statement

We target entity-level aggregation queries over text. Given a query q and a text corpus $\mathcal{C}$, the goal is to identify all entities satisfying the query constraints and perform an aggregation operation over the resulting entity set. Common aggregation operations include COUNT, SUM, and AVERAGE. In this work, we focus on COUNT aggregation, which asks for the number of satisfying entities.

Formally, an aggregation query is represented as

$$q = (T, \Phi, \Gamma), \quad (1)$$

where T denotes the target entity type, $\Phi = \{\phi_1, \phi_2, \dots, \phi_m\}$ denotes a set of constraints, and Γ denotes the aggregation operator. Throughout this work, we assume $\Gamma = \text{COUNT}$ and write the query as $q = (T, \Phi)$.

Entity type. T specifies the category of entities to be retrieved. Each query q is assumed to focus on a single entity type.

Constraint. Each query specifies a set of constraints Φ that constrain the target entity set. We treat each constraint ϕ as a boolean predicate over a single entity of type T , which evaluates to true if and only if there exists explicit textual evidence in the corpus that supports the entity satisfying the constraint. These constraints are composed using logical conjunction (AND) and disjunction (OR). More complex logical expressions can be equivalently represented through compositions of these operators (e.g., conjunctive or disjunctive normal form) (Huth and Ryan, 2004). Throughout this work, we assume a corpus-bounded setting, in which the corpus $\mathcal{C}$ serves as the sole source of

evidence for evaluating all constraints in q . The model is therefore prohibited from introducing external knowledge, in contrast to open-domain QA (Karpukhin et al., 2020; Wong et al., 2025).

Let $\mathcal{E}(\mathcal{C})$ denote the set of all entities mentioned in corpus $\mathcal{C}$. The aggregation result of query $q = (T, \Phi, \Gamma)$ over corpus $\mathcal{C}$ is defined as

$$\text{Ans}(q, \mathcal{C}) = \Gamma(\{e \in \mathcal{E}(\mathcal{C}) \mid \text{type}(e) = T \wedge \Phi(e) = \text{true}\}), \quad (2)$$

where $\Phi(e)$ denotes the evaluation of the composed constraint expression on entity e . In this work, we focus on COUNT aggregation, where $\Gamma(\cdot) = |\cdot|$. In Appendix A.1, we formalize existing methods and analyze why their design leads to incompleteness on aggregation queries.

4 Benchmark: AGGBench

Figure 2 gives an overview of the AGGBench construction pipeline. (1) We collect documents from domain-focused sources and preprocess them into a shared chunked text corpus. (2) We construct single-constraint queries by extracting and filtering corpus-derived entity types and constraints, then verbalizing each valid entity type–constraint pair into an aggregation query. (3) Human annotators read the corpus to identify all satisfying entities for each single-constraint query and record their supporting evidence spans. Overlapping annotations are used to measure reliability and resolve disagreements. (4) We construct multi-constraint queries by composing single-constraint queries with logical operators, and derive their gold answers from the corresponding answer sets. All prompts used for LLM and instructions for human annotation are provided in Appendix A.6 and A.8.

4.1 Source Corpus and Domain Selection

We collect a domain-focused document corpus $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$. Highly heterogeneous corpora are unsuitable for aggregation-query construction, because entity types and textual attributes are often scattered across unrelated topics, where many queries degenerate into single-instance “find-one” cases rather than aggregation problems. We therefore use domain-focused documents, where recurring entity types provide enough repeated instances for constructing and annotating corpus-bounded aggregation queries.

In AGGBench, $\mathcal{D}$ covers two domains: financial and research. For finance, we select 15 large-cap

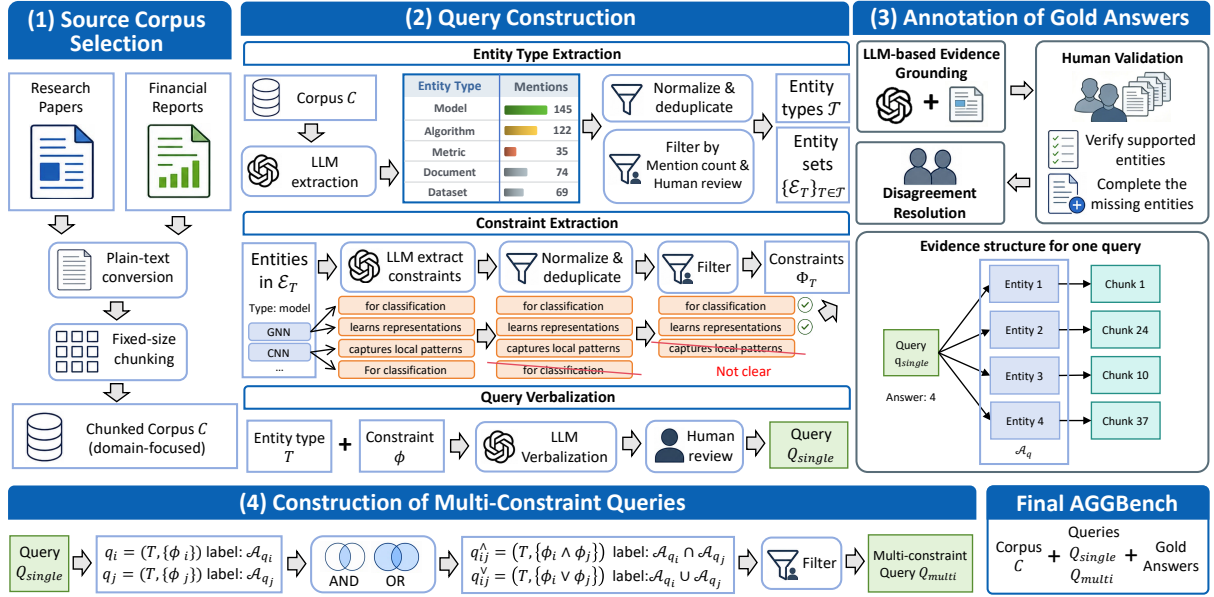


Figure 2: Overview of the AGGBench construction pipeline.

technology companies and retrieve their 2025 Form 10-K filings from SEC EDGAR (U.S. Securities and Exchange Commission, 2026), with an average chunk-text length of approximately 393,358 characters per document. For research, we manually collect 44 papers published between 2019 and 2025 in venues such as ACL and NeurIPS, with an average chunk-text length of approximately 101,812 characters per document.

Each document is converted into plain text, with non-textual content such as figures and tables removed. We split documents into chunks with a maximum length of 1000 characters and an overlap of 200 characters, producing the chunked corpus $C = \{c_j\}_{j=1}^M$, where each chunk c_j is linked to its source document d_i . All methods are evaluated on the same C to ensure fairness.

4.2 Partial Query Generation

The goal of query construction is to obtain single-constraint aggregation queries from the chunked corpus C . Each query is grounded in an entity type T , which defines the aggregation unit, and a constraint ϕ , which specifies the condition that target entities must satisfy. We denote the constructed query set as

$$\mathcal{Q}_{single} = \{(q, T, \phi)\}, \quad (3)$$

where q is the natural-language aggregation query. We construct $\mathcal{Q}_{single}$ through entity type extraction, constraint extraction, and query verbalization.

Entity type extraction. For each chunk c_j , we prompt an LLM to extract entity mentions and their

predicted entity types. Each mention is represented as $m = (n, T, d_i, c_j)$, where n is the entity name, T is the predicted type, and d_i, c_j are the source document and chunk identifiers. We normalize and deduplicate mentions by deterministic rules such as lowercasing and whitespace normalization, merging mentions with the same normalized name and type. This yields canonical entity sets $\mathcal{E}_T$ grouped by entity type, while preserving links to original mentions and chunks.

We then filter entity types that are too few or ill-defined for aggregation query construction. For each entity type T , we compute the number of entities $|\mathcal{E}_T|$ and its coverage over chunks and documents. We discard entity types with fewer than 5 canonical entities or coverage in fewer than 2 documents/chunks, since such types often lead to trivial “find-one” queries. Human annotators further remove entity types that are overly generic, ambiguous in boundary, or unsuitable as aggregation units. The remaining entity types form the retained type set $\mathcal{T}$.

Constraint extraction. For each retained type $T \in \mathcal{T}$, we collect the chunks linked to its canonical entities and prompt the LLM to extract attribute-value phrases that can serve as constraints. Each extracted constraint mention is represented as $r = (e, T, \phi, d_i, c_j)$, where e is the associated entity and ϕ is the candidate constraint. We normalize and deduplicate constraints under each entity type following the same protocol as query construction. We preserve their links to entities and evidence chunks, and obtain a candidate constraint set Φ_T .

We further filter constraints that are unsuitable for query construction. We discard constraints that appear fewer than 2 times, since they often produce trivial answer sets. Conversely, constraints that apply to almost all entities of the same type are removed because they do not define discriminative aggregation targets. Human annotators then remove constraints that are entity-specific or not directly supported by the source text. After filtering, each retained entity type T is paired with a validated constraint set Φ_T .

As shown in Figure 2, for the entity type $T = \text{model}$, entities such as GNN and CNN yield attribute-value phrases from their evidence chunks, such as “learns representations”. The normalization step merges equivalent values across entities, and the filtering step removes unclear values such as “captures local patterns”, retaining discriminative constraints for query construction.

Query verbalization. The validated entity type–constraint pairs are collected as

$$\mathcal{P} = \{(T, \phi) \mid T \in \mathcal{T}, \phi \in \Phi_T\}. \quad (4)$$

For each pair (T, ϕ) , we prompt the LLM to verbalize it into a natural-language aggregation query q , using in-context examples and requiring the generated query to preserve the original T and ϕ . For example, $T = \text{dataset}$ and $\phi = \text{“used for multi-hop question answering”}$ can be verbalized as “How many datasets are used for multi-hop question answering?” Human validators remove generated queries that are ambiguous, unnatural, or not directly verifiable from the corpus. This yields the single-constraint query set $\mathcal{Q}_{\text{single}} = \{q = (T, \{\phi\})\}$.

4.3 Gold Standard Annotation

After constructing the single-constraint query set $\mathcal{Q}_{\text{single}}$, we annotate gold answers on the corpus $\mathcal{C}$. For each query $q = (T, \phi)$, the gold answer is a deduplicated set of satisfying entities,

$$\mathcal{A}_q = \{e \in \mathcal{E}_T \mid \phi(e) = \text{true}\}, \quad (5)$$

where each entity must be supported by one document and chunk.

Two-stage annotation. We adopt a two-stage annotation pipeline. First, an LLM evaluates all query–chunk pairs and proposes candidate positives with entity names, document IDs, chunk IDs. These predictions are used only as pre-annotations. Second,

annotators are assigned a subset of documents. Human annotators read the assigned documents and perform two tasks for each query: (1) verify that each proposed entity is truly supported by the evidence, and (2) identify any entities of type T that satisfy ϕ but were missed by the LLM. Annotators are instructed to exhaustively read their assigned documents before making annotation decisions.

Quality control. A subset of documents is independently annotated by two annotators to measure consistency and identify ambiguous cases. For each disagreement, the annotator assigning the positive label presents the supporting text recorded during the independent annotation stage, and both annotators re-examine the evidence and its surrounding context. If the evidence does not lead to a shared interpretation, the case is flagged as unresolved. A query is removed when it is independently flagged as unresolved by at least two annotator pairs. In total, 15 queries are removed under this criterion. After removing these unresolved queries, positive-label consistency on the retained benchmark reaches 79%. All remaining disagreements are subsequently resolved through adjudication to establish the final gold labels. The annotation process involves 5 annotators and approximately 300 annotation hours.

We report mean pairwise Cohen’s κ (Cohen, 1960) and raw agreement, obtaining $\kappa = 0.496$ and 90.23% raw agreement. They were both calculated from the initial independent annotations, before any discussion, adjudication, or query removal. These statistics indicate high raw agreement but moderate chance-corrected agreement, reflecting the difficulty of resolving a small number of ambiguous entity–constraint relations. We also compare LLM pre-annotations with final human-validated labels, obtaining 91.85% accuracy and 0.603 F1, indicating that LLM pre-annotation improves efficiency but remains insufficient without human validation.

After validation, we normalize entity names following the same protocol as query construction, merge duplicate annotations, and conduct a final consistency check over entity names, evidence spans, and query constraints. The resulting benchmark contains 5110 validated entity annotations across 117 single-constraint queries, including 47 finance queries and 70 research queries.

4.4 Complete Query Generation

After obtaining validated single-constraint queries, we construct multi-constraint queries by compos-

ing queries with the same entity type. For two single-constraint queries $q_i = (T, \{\phi_i\})$ and $q_j = (T, \{\phi_j\})$, where $\phi_i \neq \phi_j$, we create composed queries using logical conjunction and disjunction:

$$q_{ij}^{\wedge} = (T, \{\phi_i \wedge \phi_j\}), \quad q_{ij}^{\vee} = (T, \{\phi_i \vee \phi_j\}). \quad (6)$$

The composed constraints are verbalized into natural-language queries using the same verbalization procedure. For example, when $T = \text{model}$, $\phi_i = \text{“used for classification”}$, and $\phi_j = \text{“captures local patterns”}$, the conjunction query asks: “How many models are used for classification and capture local patterns?” The disjunction query asks: “How many models are used for classification or capture local patterns?”

Their gold labels are derived directly from the validated single-constraint annotations. For conjunction, the gold entity set is the intersection $\mathcal{A}_{q_i} \cap \mathcal{A}_{q_j}$; for disjunction, it is the union $\mathcal{A}_{q_i} \cup \mathcal{A}_{q_j}$. The evidence for each resulting entity is inherited from the corresponding single-constraint annotations. We manually remove composed queries that are unnatural after verbalization, or have empty or uninformative answer sets.

4.5 Evaluation Protocol

We primarily evaluate aggregation accuracy. Given the predicted count $\hat{y}$ and gold count y , we report Absolute Count Error, $ACE = |\hat{y} - y|$, and Normalized Absolute Count Error, $NACE = |\hat{y} - y| / (y + \epsilon)$, where ϵ is a small constant for numerical stability. ACE measures absolute counting deviation, while NACE normalizes errors across queries with different answer sizes. To reduce the influence of outlier queries, we report the median ACE and NACE unless average.

We also report two recall-based diagnostics. Entity-level recall measures the proportion of gold entities recovered by the system: $\text{Recall}_{ent} = |\hat{\mathcal{E}} \cap \mathcal{E}| / |\mathcal{E}|$, where $\mathcal{E}$ is the gold entity set and $\hat{\mathcal{E}}$ is the output entity set. Chunk-level evidence recall measures the proportion of annotated gold evidence chunks covered by the system’s retrieved or selected chunks. Since an entity may be supported by multiple valid chunks, chunk-level recall is used only as a reference diagnostic signal.

5 Experiments

In this section, we conduct experiments on the proposed benchmark to analyze the behavior of different paradigms for aggregation over text.

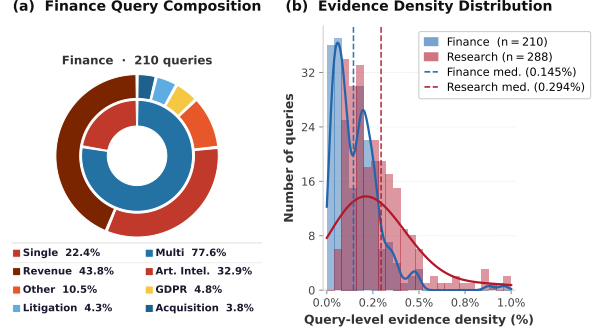


Figure 3: Benchmark characterization of AGGBench. (a) Chart shows the query split in the inner ring and entity composition in the outer ring. (b) Query-level evidence-density distribution across finance and research queries.

5.1 Experiment Setup

Reference Methods. We evaluate a diverse set of representative paradigms on our benchmark, as shown in Table 1 and Appendix A.2. We choose three categories of methods: Basic RAG, Structured and Schema-first Methods, and Agentic Framework. The selected methods include basic RAG and agent implementations, widely used research baselines, and representative open-source or commercial systems. The structured and schema-first category includes representative graph-based or GraphRAG-style retrieval approaches, including HippoRAG and LightRAG. More details are provided in the Appendix A.2.

LLM Models. The experiments were conducted on five representative LLMs—GPT-4o-mini, Gemini-3.5-Flash, Claude-Haiku-4.5, DeepSeek-V3.2, and Qwen3-32B. They cover both large and small models, open-source and proprietary settings, as well as reasoning and non-reasoning variants.

5.2 Benchmark Characterization

We characterize AGGBench from the perspective of aggregation difficulty.

Corpus and query statistics. Table 2 summarizes the corpus, query, and answer statistics of AGGBench across the finance and research domains. Despite aggregation labeling is particularly labor-intensive, AGGBench contains 498 queries, providing a non-trivial evaluation set for completeness-oriented aggregation. Figure 3(a) presents a nested pie chart for the finance domain, where the outer ring shows the distribution of entity types. Financial and research documents differ in style and constraint semantics: financial reports contain business-, AI-, and market-oriented descriptions, while research papers contain method-, dataset-, and task-oriented descriptions.

Table 1: Main performance comparison on the AGGBench dataset using GPT-4o-mini. For methods that do not explicitly output retrieved or selected evidence chunks, we omit chunk-level recall and mark it as “-”.

Category	Method	Finance				Research			
		NACE (med)	ACE (med)	Recall(entity)	Recall(chunk)	NACE (med)	ACE (med)	Recall(entity)	Recall(chunk)
Basic RAG Methods	Naive RAG	0.737	9.500	0.138	0.054	0.429	3.000	0.194	0.033
	FLARE (Jiang et al., 2023)	0.857	10.000	0.311	0.114	0.769	5.000	0.351	0.065
	RAT (Wang et al., 2024)	0.606	7.000	0.254	0.100	0.429	3.000	0.389	0.036
	DeepNote (Wang et al., 2025)	0.823	10.000	0.203	0.045	0.750	5.000	0.227	0.012
Structured and Schema-first Methods	HippoRAG (Gutiérrez et al., 2024)	0.750	9.500	0.144	0.057	0.500	3.000	0.220	0.042
	LightRAG (Guo et al., 2025)	0.667	7.000	0.170	0.058	0.364	2.000	0.223	0.029
	PageIndex (Zhang et al., 2025a)	0.800	9.500	0.143	-	0.750	5.000	0.080	-
	Palimpsest (Liu et al., 2025)	0.833	5.000	0.178	-	0.556	5.000	0.475	-
Agentic Frameworks	Naive Agent	0.586	6.000	0.214	0.074	0.500	3.000	0.306	0.032
	ReAct (Yao et al., 2023)	0.616	7.000	0.210	0.063	0.429	3.000	0.238	0.015
	ReadAgent (Lee et al., 2024)	0.500	7.000	0.398	-	0.667	4.000	0.273	-
	Multi Agent (Wu et al., 2024)	0.594	6.000	0.220	0.076	0.500	3.000	0.290	0.026
	Smolagents (Roucher et al., 2025)	0.768	8.000	0.186	0.069	0.538	3.000	0.260	0.049
	Search-o1 (Li et al., 2025)	0.667	8.000	0.162	0.054	0.875	7.000	0.317	0.044
	Claude Code (Anthropic, 2025)	0.579	7.000	0.180	-	0.444	3.000	0.237	-

Table 2: Statistics of AGGBench by domain. “Chars.” reports the total chunk-text length in millions of characters. “Chunks” reports the number of text chunks in each corpus. “Single Q.” and “Multi Q.” denote single-constraint queries and composed multi-constraint queries. “Avg. Ans.” is the average gold answer size across queries. “Evid. Dens.” reports the median query-level evidence density, computed as the fraction of gold evidence chunks among all chunks.

Domain	Chars. (M)	Chunks	Single Q.	Multi Q.	Avg. Ans.	Evid. Dens.
Research	4.48	4,755	70	218	7.03	0.294%
Finance	5.90	7,611	47	163	14.7	0.145%
Overall	10.38	12,366	117	381	10.26	0.210%

Regarding query structure, the 381 multi-constraint queries in AGGBench include 51 conjunctive (AND) queries and 330 disjunctive (OR) queries. The smaller number of AND queries mainly results from the construction protocol rather than sampling bias. Conjunctions naturally yield fewer qualifying entities because each entity must satisfy all constraints simultaneously. Consequently, many candidate AND queries contain at most one valid entity and are removed by our filtering procedure, resulting in fewer retained AND queries. These composed queries introduce set-level aggregation complexity, requiring systems to distinguish whether evidence should be intersected, unioned, and deduplicated. The inner ring of Figure 3(a) shows the split between single-constraint and multi-constraint queries in finance domain, which demonstrates that our benchmark covers diverse entity categories and includes queries with varying constraint complexity.

Answer and evidence distributions. One key challenge of aggregation over large text corpora is evidence sparsity: the evidence required to answer a query typically constitutes only a small fraction of the entire corpus. For each query q , we define evidence density as $\rho_q = |\mathcal{A}_q|/|C|$. As shown in

Table 2, the median query-level evidence density is only 0.210% overall and remains below 0.3% in both domains. This means that systems must locate a small number of relevant evidence chunks from a large candidate corpus, making the task highly dependent on effective retrieval and filtering. Figure 3(b) plots query-level evidence density, with the x-axis showing the query level evidence density and the y-axis showing the number of queries. Most queries have very low evidence density, making aggregation highly sensitive to retrieval omissions.

At the same time, AGGBench goes beyond single-answer QA. The average gold answer size is 14.7 in finance and 7.03 in research, requiring systems to find, verify, deduplicate, and count multiple satisfying entities rather than stop after retrieving one relevant passage.

5.3 Performance Landscape on AGGBench

We evaluate the performance and cost of representative paradigms and language models.

5.3.1 Performance of Existing Paradigms

Table 1 reports the main results. Overall, existing paradigms remain unreliable for aggregation queries over evidence-sparse corpora.

Basic RAG methods. Basic RAG methods provide direct evidence access, but their performance is unstable across domains. Iterative retrieval methods such as FLARE and RAT achieve higher entity recall than Naive RAG in several settings, showing that query refinement can expose more relevant evidence. However, higher recall does not consistently translate into lower count error. This reflects a mismatch between RAG and aggregation: RAG is designed to retrieve sufficient evidence for answer-

ing, but lacks an explicit objective to exhaustively collect all entities satisfying the query.

A natural question is whether the incompleteness of RAG can be addressed simply by retrieving more passages. We therefore vary the retrieval top- k from 5 to 50 for representative RAG methods. The results show that increasing k consistently exposes more relevant evidence, but does not lead to a corresponding improvement in aggregation accuracy. These results suggest that aggregation is not simply a retrieval-depth problem. Detailed results for different top- k values are provided in Appendix A.4.

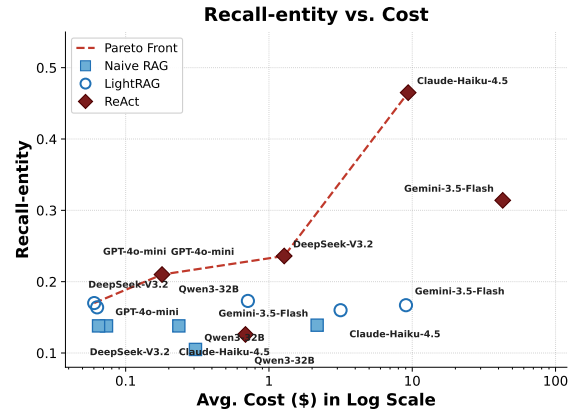
Structured and schema-first methods. Structured and schema-first methods do not consistently outperform basic RAG. LightRAG achieves strong count accuracy, especially on the research domain, but its entity recall remain limited. PageIndex and Palimpzest also show unstable behavior, with Palimpzest producing large errors on research. These results suggest that intermediate structures or semantic indexes can help organize evidence, but they do not guarantee completeness when constraints are not anticipated by the predefined schema or index.

Agentic frameworks. Agentic frameworks achieve competitive count accuracy in several settings, mainly due to their stronger planning and feedback-driven refinement. However, their entity recall is often lower than that of some RAG methods, suggesting that they are not better at exhaustive retrieval. Instead, agents can better interpret aggregation queries and adjust their behavior toward producing useful counts. Still, without mechanisms for preserving candidate coverage and validating completeness, they miss satisfying entities.

Overall, the main bottleneck is not merely reasoning over retrieved evidence, but maintaining sufficient evidence coverage before aggregation. More results and analysis are in Appendix A.4.

We further examine whether this conclusion holds across different logical compositions by separately evaluating AND and OR queries. Both subsets exhibit limited entity and evidence recall across methods, while OR queries generally incur larger absolute counting errors, particularly in the Finance domain, consistent with their larger answer sets. We also assess the robustness of the observed differences using paired Wilcoxon signed-rank tests on per-query results. The complete results are reported in Appendix A.7.

5.3.2 Comparison Across Base Models



approaches require costly corpus structuring; as shown in Appendix A.2.2, constructing such structured corpora incurs substantial computational/API cost and becomes brittle under domain shifts.

Agentic methods fail mainly due to unreliable aggregation over retrieved evidence. ReAct-style agents often issue multiple reformulated searches and accumulate many chunks in the context, but they lack a reliable mechanism to extract satisfying entities and deduplicate them across mentions. As a result, additional retrieval does not necessarily translate into complete or accurate counts. Other methods such as ReadAgent attempt a more exhaustive traversal of the corpus, but this substantially increases execution cost and still does not guarantee recovery of the complete entity set.

5.4.2 Common Error Patterns

Insufficient completeness awareness. Most errors stem from the lack of completeness-aware behavior. Existing frameworks are usually designed under a relevance-oriented assumption (Karpukhin et al., 2020), where finding several relevant chunks is treated as sufficient for answering. In contrast, aggregation queries require agents to follow stronger planning mechanisms and more stable execution patterns for systematically reading, verifying, and summarizing evidence.

Incorrect tool usage. About 20% of errors are related to incorrect tool usage. Models often repeatedly call the same tool with minor query variations, or fail to choose appropriate retrieval and filtering parameters. Future work could incorporate tool-awareness or tool-evolution learning to improve parameter selection and usage reliability.

Evidence hallucination. A smaller portion of errors comes from evidence hallucination. In these cases, the model mistakes numerical values or incidental statements in the source text for the final aggregation result, rather than deriving the answer from a verified set of satisfying entities.

6 Conclusion

In this work, we studied aggregation queries over unstructured text. We formalized this problem as corpus-bounded entity-level aggregation and introduced AGGBENCH, a completeness-oriented benchmark constructed from finance reports and research papers. Through a corpus-grounded query construction pipeline and LLM-assisted human validation, AGGBENCH provides evidence-supported gold answers. Our experiments show that existing

methods remain far from reliable on this task. Their failures mainly arise from incomplete evidence coverage, over-reliance on top-ranked passages, and insufficient mechanisms for tracking all satisfying entities across a large corpus. We hope AGGBENCH will support future work on completeness-aware retrieval and robust aggregation methods for unstructured corpora.

7 Limitations

Due to computational and API budget constraints, our experiments cover a representative set of LLM backbones and baseline paradigms, but do not exhaustively evaluate all available proprietary and open-source models. Since model capabilities evolve rapidly, future work can extend the comparison to a broader and continuously updated model pool. In addition, AGGBENCH currently focuses on research and financial documents. Although these domains provide realistic and evidence-sparse settings for entity-level aggregation over text, expanding the benchmark to more domains such as legal, biomedical, and enterprise documents would further test the generality of existing methods.

8 Ethical Impact

All source documents used in this study are publicly available financial records and research papers. We use them only for corpus-bounded benchmark construction and do not collect private user logs, behavioral traces, or data intended to profile individuals. Although some source documents may contain public names such as paper authors or company executives, our queries and annotations focus on entities such as methods, datasets, systems, products, organizations, and business items, rather than personal attributes. We also follow the licenses and terms of the original sources when constructing and releasing the benchmark.

Human annotation was used only to validate entity-level gold answers and supporting evidence. Annotators were informed of the research purpose and annotation procedure, and were compensated fairly for their work according to local institutional standards. We do not release annotator identities or any personal information about annotators. Since the annotation task involves publicly available documents and does not require collecting sensitive personal data from participants, it presents minimal privacy risk.

Acknowledgments

This work is supported by the National Science and Technology Major Project Grants No. 2026ZD0126000 and No. 2026ZD0126005, National Natural Science Foundation of China under Grants No.62572119, 62472094, and 62232004, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No.BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No.93K-9, and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization, Collaborative Innovation Center of Wireless Communications Technology, the Fundamental Research Funds for the Central Universities(2242026K30049). We also thank the Big Data Computing Center of Southeast University for providing the experiment environment and computing facility.

References

- Anthropic. 2025. Claude 3.7 sonnet and claude code. <https://www.anthropic.com/news/claude-3-7-sonnet>. Accessed: 2026-05-23.
- 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*.
- Jacob Cohen. 1960. **A coefficient of agreement for nominal scales**. *Educational and Psychological Measurement*, 20(1):37–46.
- Qiyang Deng, Jianhui Li, Chengliang Chai, Jinqi Liu, Junzhi She, Kaisen Jin, Zhaoze Sun, Yuhao Deng, Jia Yuan, Ye Yuan, Guoren Wang, and Lei Cao. 2025. **Unstructured data analysis using llms: A comprehensive benchmark**. *arXiv preprint arXiv:2510.27119*.
- 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378. Association for Computational Linguistics.
- Alexandru Dumitru, Venkatesh V, Adam Jatowt, and Avishek Anand. 2025. Evaluating list construction and temporal understanding capabilities of large language models. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, pages 369–379.
- Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. 2022. **Answering count queries with explanatory evidence**. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2415–2419. Association for Computing Machinery.
- Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. **LightRAG: Simple and fast retrieval-augmented generation**. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. **HippoRAG: Neurobiologically inspired long-term memory for large language models**. *Advances in Neural Information Processing Systems*, 37:59532–59569.
- Tam Harbert. 2021. **Tapping the power of unstructured data**. *MIT Sloan Ideas Made to Matter*.
- 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. International Committee on Computational Linguistics.
- Michael Huth and Mark Ryan. 2004. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2 edition. Cambridge University Press.
- Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. **Active retrieval augmented generation**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.
- 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. Association for Computational Linguistics.
- Anoop R. Katti, Kai Hui, Adrià de Gispert, and Hagen Fürstenau. 2021. **Question answering using web lists**. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*, pages 3132–3136. ACM.
- Tushar Khot, Peter Clark, Peter Jansen, and Ashish Sabharwal. 2020. **QASC: A dataset for question answering via sentence composition**. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.
- Tian Lan, Bin Zhu, Qianghui Jia, Junyang Ren, Haijun Li, Longyue Wang, Zhao Xu, Weihua Luo, and Kaifu Zhang. 2025. Deepwidesearch: Benchmarking depth

- and width in agentic information seeking. *arXiv preprint arXiv:2510.20168*.
- Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. 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 2261–2273.
- Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the 41st International Conference on Machine Learning*, pages 26396 – 26415. JMLR.org.
- Seongyun Lee, Hyunjae Kim, and Jaewoo Kang. 2023. LIQUID: A framework for list question answering dataset generation. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023)*, pages 13014–13024. AAAI Press.
- Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In *International Conference on Learning Representations*, volume 2025, pages 28691–28735.
- 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*, 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, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438.
- Chunwei Liu, Matthew Russo, Michael J. Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael J. Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the 15th Conference on Innovative Data Systems Research*, Amsterdam, The Netherlands.
- Iain J. Marshall and Byron C. Wallace. 2019. Toward systematic review automation: A practical guide to using Machine Learning tools in research synthesis. *Systematic Reviews*, 8(1):163.
- Kaisa Miettinen. 1998. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, MA.
- Joel Niklaus, Veton Matoshi, Matthias Stürmer, Ilias Chalkidis, and Daniel Ho. 2024. Multilegalpile: A 689gb multilingual legal corpus. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15077–15094. Association for Computational Linguistics.
- Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic operators and their optimization: Enabling llm-based data processing with accuracy guarantees in lotus. *Proceedings of the VLDB Endowment*, 18(11):4171–4184.
- Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. ‘smolagents’: a smol library to build great agentic systems. <https://github.com/huggingface/smolagents>.
- Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic query rewriting and evaluation for complex document processing. *Proceedings of the VLDB Endowment*, 18(9):3035–3048.
- Marco Siino, Ilenia Tinnirello, and Marco La Cascia. 2024. Is text preprocessing still worth the time? a comparative survey on the influence of popular preprocessing methods on transformers and traditional classifiers. *Information Systems*, 121:102342.
- Alexander Spangher, Tenghao Huang, Yiqin Huang, Lucas Spangher, Sewon Min, and Mark Dredze. 2025. A novel multi-document retrieval benchmark: Journalist source-selection in newswriting. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 180–204. Association for Computational Linguistics.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.
- Harsh Trivedi, Niranjana 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.

- Guy Tsafnat, Paul Glasziou, Miew Keen Choong, Adam Dunn, Filippo Galgani, and Enrico Coiera. 2014. [Systematic review automation technologies](#). *Systematic Reviews*, 3(1):74.
- U.S. Securities and Exchange Commission. 2026. SEC EDGAR Search Filings. <https://www.sec.gov/search-filings>. Accessed: 2026-05-20.
- Ruobing Wang, Qingfei Zhao, Yukun Yan, Daren Zha, Yuxuan Chen, Shi Yu, Zhenghao Liu, Yixuan Wang, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. [DeepNote: Note-centric deep retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19688–19715, Suzhou, China. Association for Computational Linguistics.
- Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. 2024. [RAT: retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation](#). ArXiv preprint arXiv:2403.05313.
- Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. 2025. [WideSearch: Benchmarking agentic broad info-seeking](#). *arXiv preprint arXiv:2508.07999*.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *Proceedings of the 1st Conference on Language Modeling (COLM 2024)*.
- Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyang Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, Zhiyang Deng, Yuechen Jiang, Zhiyuan Yao, Haoqiang Li, Yangyang Yu, Gang Hu, Jiajia Huang, Xiao-Yang Liu, Alejandro Lopez-Lira, Benyou Wang, Yanzhao Lai, Hao Wang, Min Peng, Sophia Ananiadou, and Jimin Huang. 2024. [Finben: A holistic financial benchmark for large language models](#). In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024 Datasets and Benchmarks Track. Curran Associates Inc.
- Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2024. [CRAG – comprehensive RAG benchmark](#). In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*. Datasets and Benchmarks Track.
- 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 (EMNLP)*, pages 2369–2380. Association for Computational Linguistics.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- 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 (EMNLP)*, pages 3911–3921. Association for Computational Linguistics.
- Mingtian Zhang, Yu Tang, and PageIndex Team. 2025a. Pageindex: Next-generation vectorless, reasoning-based RAG. *PageIndex Blog*. <https://pageindex.ai/blog/pageindex-intro>.
- Wentao Zhang, Woojeong Kim, and Yuntian Deng. 2025b. [From chat logs to collective insights: Aggregative question answering](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32811–32850. Association for Computational Linguistics.
- Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2SQL: Generating structured queries from natural language using reinforcement learning](#). ArXiv preprint arXiv:1709.00103.

A Appendix

A.1 Limitations of Existing Pipelines

Here we formalize existing methods and discuss their limitations. Following the main formulation, let $\mathcal{A}(q, \mathcal{C})$ denote the exact set of entities in corpus $\mathcal{C}$ that satisfy query q . For COUNT aggregation, the final answer is $\text{Ans}(q, \mathcal{C}) = |\mathcal{A}(q, \mathcal{C})|$. In practice, the corpus $\mathcal{C}$ is split into chunks $\{c_1, \dots, c_M\}$ and processed at the chunk level. Let $\mathcal{A}(q, c)$ denote the set of entities in chunk c that satisfy query q . The exact satisfying entity set is obtained by aggregating chunk-level results across the full corpus:

$$\mathcal{A}(q, \mathcal{C}) = \bigcup_{c \in \mathcal{C}} \mathcal{A}(q, c), \quad (7)$$

where duplicate entities across chunks are merged. Existing approaches can then be viewed as simplified approximations of Equation (7).

Directly computing the expression in Equation (7) is infeasible when $|\mathcal{C}|$ is large, so some existing systems adopt a rank-then-read strategy. A lightweight retriever assigns each chunk a relevance score and keeps only the top k chunks. These k chunks are fed to a large language model p_θ . When $k \ll \text{count}(q)$, many relevant entities never reach the model, and the resulting upper bound:

$$\widehat{\text{count}}_{\text{RAG}}(q) = |\text{AGG}_{c \in \mathcal{Z}_k(q, \mathcal{C})} \psi_q(c)| \quad (8)$$

can fall far below the true total. Otherwise, increasing k indiscriminately introduces large amounts of irrelevant text, inflates context length, and quickly exceeds the input limit of p_θ .

A second line of work replaces free-text reasoning with *schema-first* pipelines. Each chunk is pre-processed by an extractor T that maps text to structured tuples $T : \mathcal{C} \rightarrow \mathcal{S}$, yielding a relational or graph database $\mathcal{S}$. Then, the original query is rewritten into SQL or SPARQL form, denoted q^* , and executed as

$$\text{count}(q) = |\sigma_{q^*}(\mathcal{S})|, \quad (9)$$

where $\sigma_{q^*}(\cdot)$ returns all tuples satisfying the structured query. This approach suffers from two issues: building $\mathcal{S}$ demands expensive extraction and alignment, and the fixed schema restricts q^* to explicit attributes, making it hard to capture implicit or compositional conditions.

In short, rank-then-read RAG sacrifices recall for efficiency, while schema-first pipelines incur heavy upfront costs yet remain rigid. Neither class

can deliver complete aggregation at a practical inference cost; hence, a new framework is required to solve aggregation problems effectively.

A.2 Implementation details

A.2.1 Baseline Details

We provide detailed descriptions of the baseline methods introduced in Section 5.1, covering three categories and fifteen methods:

Basic RAG Methods. These kind of methods use top- k retrieval combined with chunk-based RAG prompting. We use LlamaIndex to build the Naive RAG system. For FLARE, RAT, and DeepNote, we design task-specific prompts for aggregation queries in both the Finance and Research domains, and preserve the original configurations of each method. All retrieval top- k are set to 5.

Structured and Schema-first Methods. These methods leverage graph-enhanced RAG or structure-aware retrieval instead of chunk-based retrieval. For HippoRAG and LightRAG, we construct the knowledge graphs with their default configurations respectively. For LightRAG, we use the hybrid mode and set the chunk top- k to 5. For PageIndex, we concatenate all PDF versions of filings in the Finance corpus into a single document collection named Financial Reports with a corresponding table of contents. Similarly, all paper PDFs in the Research corpus are merged into a collection named Proceedings. We then build page indexes on top of these collections for evaluation. For Palimpsest, we adopt the maxquality strategy.

Agentic Frameworks. These methods employ LLM-based agents to perform iterative reasoning and retrieval. Naive Agent is a basic agent framework that supports multiple calls to the retrieval tool. ReAct is implemented using LangChain and follows the classical thinking-action-observation loop. Multi-Agent enables a manager agent to create sub-agents and assign them specialized tasks. Smolagents is implemented based on its official library. All agentic frameworks mentioned above are equipped with a dense-retrieval tool implemented using ChromaDB, with the retrieval top- k set to 5 and the maximum number of iterations set to 10. For ReadAgent, we directly provide the raw corpus documents, and the framework performs document pagination and retrieval with its default settings. Search-o1 was originally designed for web search scenarios. We replace its web search

Table 3: Pricing details of the evaluated LLMs.

Model	Input (\$/M Tokens)	Output (\$/M Tokens)
GPT-4o-mini	0.1500	0.6000
Qwen3-32B	0.4000	1.6000
DeepSeek-V3.2	0.3160	0.4740
Gemini-3.5-Flash	1.5000	9.0000
Claude-Haiku-4.5	1.0000	5.0000

Table 4: Overall question-answering costs.

Category	Method	Cost(\$)	
		Finance	Research
Basic RAG Methods	Naive RAG	0.074	0.022
	FLARE	0.059	0.221
	RAT	0.353	0.735
	DeepNote	0.338	0.529
Structured and Schema-first Methods	HippoRAG	0.088	0.118
	LightRAG	0.059	0.103
	PageIndex	1.412	1.535
	Palimpsest	38.971	32.941
Agentic Frameworks	Naive Agent	0.088	0.224
	ReAct	0.179	0.034
	ReadAgent	6.397	7.868
	Multi Agent	0.471	0.912
	Smolagents	0.309	0.324
	Search-o1	0.162	0.309
	Claude Code	0.450	0.934

module with the same local dense-retrieval tool used above and design task-specific prompts for aggregation queries. We also evaluate Claude Code by directly issuing instructions in the terminal environment, and allow it to access the entire corpus and generate answers in the required output format.

A.2.2 Model Usage

In this paper, experiments are conducted on GPT-4o-mini, Gemini-3.5-Flash, Claude-Haiku-4.5, DeepSeek-V3.2 and Qwen3-32B. We access all LLM services through the DMXAPI model aggregation platform. The corresponding pricing details are provided in Table 3. The overall question answering costs for all experiments reported in Table 1 are summarized in Table 4. For structured methods, we additionally report the costs of constructing structured data representations, as summarized in Table 5.

A.3 Can Stronger Backbone LLM Solve Aggregation Problems?

To verify whether stronger base models can resolve the aggregation problem, we select one rep-

Table 5: Overall structure-construction costs.

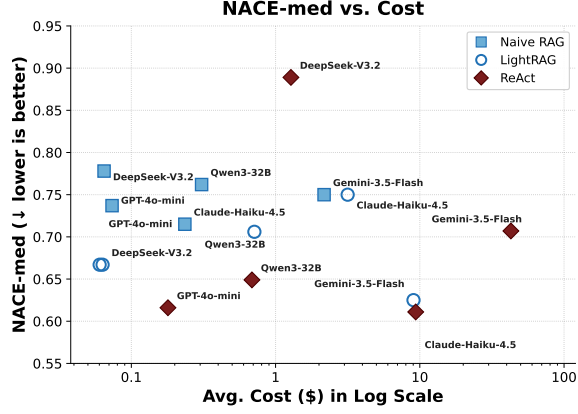
Method	Cost(\$)	
	Finance	Research
HippoRAG	1.471	1.834
LightRAG	7.353	3.904
PageIndex	8.824	3.162

resentative baseline from each of the three categories and evaluate their performance across different backbone models. Results are in Figure 5. Overall, stronger backbone models sometimes improve evidence coverage, particularly in terms of Recall-chunk and Recall-entity. For example, Gemini-3.5-Flash and Claude-Haiku-4.5 achieve substantially higher Recall metrics under the ReAct framework. This suggests that stronger instruction-following and reasoning capabilities can partially improve iterative retrieval and evidence collection. However, these gains are highly inconsistent across frameworks and are accompanied by dramatically increased inference cost. In several cases, stronger models yield only marginal improvements or even performance degradation despite significantly higher cost. That indicates that larger model size alone does not necessarily translate into better aggregation capability.

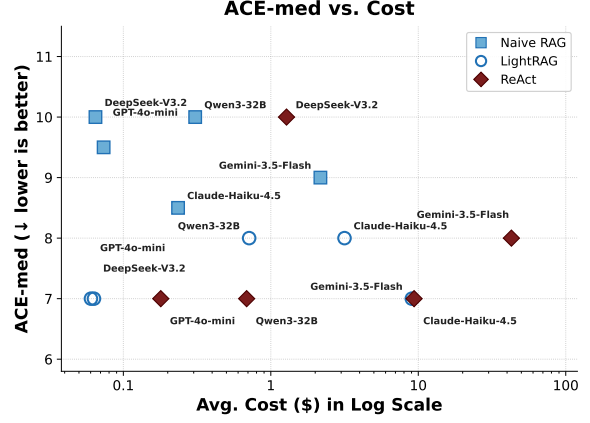
More importantly, even the strongest backbone models still exhibit relatively low Recall-chunk scores across all settings, suggesting that existing RAG and agentic pipelines remain fundamentally limited in exhaustive evidence collection. Although stronger models can partially mitigate the aggregation problem, they fail to resolve it within current baseline frameworks. These results indicate that simply scaling backbone LLMs is neither an effective nor an economical solution to the aggregation challenge posed by AGGBench.

A.4 Can Larger Top-k Retrieval Solve Aggregation Problems?

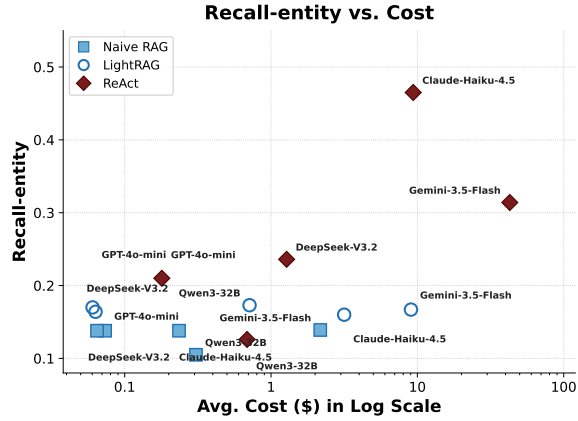
To verify whether enlarging the retrieval Top-k resolves the aggregation problem, we evaluated the performance of several RAG baselines under different Top-K. Results are summarized in Table 6. We observe that increasing Top-k raises chunk-level recall, sometimes by more than 0.09 compared to low-retrieval settings. However, this gain does not translate into better aggregation accuracy. On AGGBench Research domain, naive RAG exhibits higher recall as top-k grows, yet both NACE(mean) and NACE(med) remain nearly unchanged or even



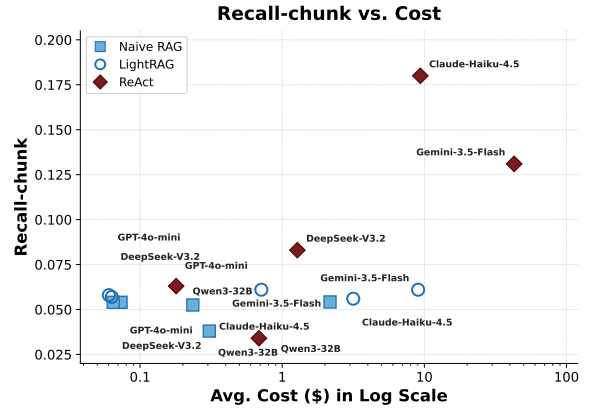
(a) NACE-med vs. Cost



(b) ACE-med vs. Cost



(c) Recall-entity vs. Cost



(d) Recall-chunk vs. Cost

Figure 5: Performance versus average per-query cost (USD, log scale) across all four AGGBench metrics. Each point represents one backbone model under one retrieval configuration. Dashed lines indicate the Pareto front.

worse. LightRAG shows a similar trend: although recall on AGGBench increases from 0.03 to over 0.12, its NACE values fluctuate with larger top-k, reflecting unstable accuracy. For DeepNote, enlarging top-k yields smoother recall improvements, but aggregation accuracy also oscillates and eventually worsens at higher retrieval sizes, revealing the model’s sensitivity to noisy evidence.

Overall, results show that existing RAG methods cannot solve aggregation problems by enlarging Top-k. Even when evidence is retrieved, the inclusion of noise often undermines accuracy.

A.5 Dataset Details

Finance corpus. The finance corpus is constructed from company annual reports filed through the U.S. Securities and Exchange Commission (SEC) EDGAR system. We collect Form 10-K filings whose report dates fall in 2025, and select companies in descending order of market capitalization. Form 10-K filings are used because they provide long-form, company-level descriptions of busi-

Table 6: Effect of retrieval Top-k on baselines evaluated on AGGBench Research domain.

Method	Top-k	NACE(Mean)	NACE(Med)	Recall
Naive RAG	5	0.446	0.429	0.033
	10	0.419	0.333	0.050
	20	0.711	0.333	0.077
	50	1.406	0.417	0.121
LightRAG	5	0.450	0.364	0.029
	10	0.418	0.375	0.044
	20	0.434	0.333	0.076
	50	0.463	0.375	0.123
Deepnote	5	0.621	0.667	0.015
	10	0.785	0.667	0.037
	20	0.963	0.615	0.049
	50	0.920	0.556	0.094

ness operations, risk factors, financial conditions, products, markets, and strategic directions, making them suitable for evaluating aggregation queries over realistic financial text. Table 17 lists the SEC EDGAR filings used in our finance corpus.

Research corpus. The research corpus is constructed from full-text research papers collected from official publication or preprint sources, such

as conference proceedings and open-access paper repositories. We focus on papers in areas related to large language models, retrieval, agents, graph learning, and data-centric AI, where recurring entities such as methods, datasets, models, tasks, and evaluation settings naturally support entity-level aggregation queries. For each paper, we convert the original document into plain text and apply the same chunking and annotation pipeline as used for the finance corpus. Table 18 lists the research papers used in our research corpus.

A.6 Human Instructions

A.6.1 Instructions for Query Construction

Table 7: Instructions for query construction validation.

Instructions for Query Construction

Task Instruction.

You will be asked to validate intermediate outputs used for constructing aggregation queries. These outputs include candidate entity types, candidate constraints, and generated natural-language queries. Your goal is to remove candidates that are unsuitable for constructing corpus-bounded aggregation queries. You should make decisions only based on the provided corpus snippets and should not use external knowledge.

Entity Type Validation.

For each candidate entity type, you will be given its type name, sampled entity mentions, normalized canonical entities, and representative evidence chunks. Keep an entity type only if it defines a clear and countable aggregation unit, such as a dataset, model, company, product, method, system, data center, or business segment. The boundary of the type should be clear enough that you can decide whether a mention belongs to this type. Remove entity types that are overly generic, ambiguous, too broad, or not suitable as counting units, such as “thing”, “item”, “factor”, “area”, or “information”. Also remove types that are merely textual roles or vague descriptions and cannot be consistently grounded in the corpus. A valid entity type should have enough repeated instances in the corpus to support aggregation queries.

Constraint Validation.

For each candidate constraint, you will be given the entity type, the constraint phrase, associated entities, and supporting chunk. Keep a constraint only if it directly specifies a property, function, domain, condition, or usage of the target entity type. The constraint must be explicitly supported by the provided chunk, rather than inferred from external knowledge. Remove constraints that are vague, overly generic, unsupported, ambiguous, too rare, too broad, or not useful for distinguishing entities. Examples of overly generic constraints include “important”, “various”, “large”, and “different”, unless the corpus gives them a precise meaning. Also remove constraints that are entity-specific, apply to almost all entities of the same type, or cannot be naturally interpreted as a query condition.

Query Validation.

For each generated query, you will be given the original entity type T , the constraint ϕ , and the generated natural-language query q . Keep a query only if it preserves the original entity type and constraint without adding new conditions. The query should ask for an aggregation result, usually in the form of “How many ...”. It should be natural, understandable, corpus-bounded, and answerable using the provided corpus.

Remove queries that are ambiguous about what should be counted, require external knowledge or subjective judgment, change the meaning of the original type–constraint pair, or introduce extra assumptions.

A.6.2 Instructions for Gold Answers

Table 8: Instructions for gold answer annotation.

Instructions for Gold Answers

Task Instruction.

You will be given a set of aggregation queries and a set of assigned documents split into text chunks. For each query, you will see its target entity type T , constraint ϕ , natural-language question q , and LLM-proposed candidate positives. Your task is to construct a corpus-bounded gold answer set.

For each query, first check the LLM-proposed candidates. For every proposed entity, decide whether it is a concrete entity of the required type T , whether the provided chunk explicitly supports the constraint ϕ , and whether the entity should be counted as a positive answer. Reject candidates whose evidence is missing, vague, only weakly related to the constraint, or dependent on external knowledge. Also reject generic category names, descriptions, or entity types that are not concrete entities. After verifying the proposed candidates, read the assigned documents and identify any additional entities that satisfy the same query but were missed by the LLM. For each valid entity, record a canonical entity name, the source document ID, and the chunk ID copied from the corpus. If the same entity appears multiple times or under slightly different names, merge these mentions into one canonical entity entry and keep the clearest supporting evidence.

Only count entities that satisfy the query within the given corpus. If an entity is mentioned but the constraint is not explicitly supported, do not count it. If the evidence is ambiguous, mark the case for discussion rather than making an unsupported decision.

During disagreement resolution, annotators should revisit the query definition, the source text, and decide whether the entity is concrete, whether the constraint is explicitly supported, and whether the annotation is consistent with other entities for the same query. For each disagreement arising from independent annotation, the annotator assigning the positive label first presents the supporting text recorded during the independent annotation stage and explains why it satisfies the query constraint. Both annotators then re-examine the supporting text and its surrounding context. If they still cannot reach a shared interpretation, the case is marked as unresolved rather than forcing either a positive or negative label. A query is removed from the benchmark only when it is independently flagged as unresolved by at least two annotator pairs.

A.6.3 Instructions for Multi-Constraint Queries

Table 9: Instructions for multi-constraint query validation.

Instructions for Multi-Constraint Queries

Task Instruction.

You will be given multi-constraint aggregation queries generated by composing two validated single-constraint queries with either conjunction or disjunction. For each composed query, you will see the entity type T , the two original constraints ϕ_i and ϕ_j , the logical operator, the generated natural-language query, and its derived answer set.

Your task is to decide whether the composed query should be kept in the benchmark. Keep the query only if the generated natural-language form correctly preserves the intended logical meaning. For conjunction, the query should require entities to satisfy both constraints. For disjunction, the query should require entities to satisfy at least one of the two constraints. The query should be natural, understandable, and unambiguous about what should be counted.

Remove the query if the verbalization changes the meaning of either constraint, introduces additional assumptions, makes the logical relation unclear, or produces an unnatural question. Also remove queries whose answer set is empty, trivial, or uninformative for evaluation. Do not use external knowledge when making this decision; judge only whether the composed query and its derived answer set are valid under the existing annotated corpus.

A.7 Additional Results

A.7.1 Performance by Logical Operator

To examine whether the observed difficulty of aggregation is specific to a particular logical operator, we separately evaluate the 381 multi-constraint queries according to their composition operator. The benchmark contains 51 conjunctive (AND) queries, including 44 Finance and 7 Research queries, and 330 disjunctive (OR) queries, including 119 Finance and 211 Research queries. Tables 10 and 11 report the complete results for all evaluated methods.

Across both operators, current methods continue

to exhibit limited entity and evidence recall, indicating that the completeness bottleneck is not specific to either conjunction or disjunction. OR queries generally incur substantially larger absolute count errors than AND queries, particularly in the Finance domain. This is consistent with OR queries typically producing larger answer sets, for which missed entities accumulate into larger counting errors. Importantly, no method consistently dominates across both operators and domains, further suggesting that existing RAG, structured retrieval, and agentic pipelines lack a reliable mechanism for preserving corpus-level completeness before aggregation.

A.7.2 Statistical Significance

We select one representative method from each paradigm: Naive RAG from Basic RAG Methods, LightRAG from Structured and Schema-first Methods, and ReAct from Agentic Frameworks. We conduct paired Wilcoxon signed-rank tests on per-query NACE, ACE, entity-level recall, and chunk-level recall. Table 12 reports the resulting p -values.

Overall, these results show that the evaluated paradigms differ substantially in evidence-recovery behavior, while differences in final counting accuracy are more dependent on the particular method and domain.

A.8 Prompt Templates

This section lists the prompt templates used in the LLM-assisted construction of AGG-Bench. Placeholders such as {chunk}, {entity}, {entity_type}, and {query} are replaced with the corresponding corpus content or query information during benchmark construction.

Table 10: Performance on multi-constraint conjunctive (AND) queries. NACE and ACE report median normalized and absolute count error, respectively. Ent. R and Chunk R denote entity-level and chunk-level recall.

Method	Finance				Research			
	NACE ↓	ACE ↓	Ent. R ↑	Chunk R ↑	NACE ↓	ACE ↓	Ent. R ↑	Chunk R ↑
NaiveRAG	0.410	2.000	0.197	0.087	0.250	1.000	0.120	0.024
FLARE	0.667	3.000	0.333	0.155	0.667	2.000	0.143	0.029
RAT	0.417	2.000	0.269	0.143	0.750	3.000	0.286	0.066
DeepNote	0.625	4.000	0.290	0.069	0.500	2.000	0.167	0.000
HippoRAG	0.657	2.000	0.192	0.054	0.500	2.000	0.311	0.006
LightRAG	0.624	3.000	0.234	0.092	0.727	3.000	0.381	0.000
PageIndex	0.667	3.000	0.212	–	0.667	2.000	0.119	–
Palimpzest	1.000	4.000	0.077	–	0.750	12.000	0.588	–
NaiveAgent	0.500	2.000	0.296	0.111	0.750	3.000	0.262	0.048
ReAct	0.646	4.000	0.226	0.058	0.333	1.000	0.083	0.008
ReadAgent	0.688	4.000	0.519	–	0.500	2.000	0.119	–
MultiAgent	0.572	3.000	0.298	0.111	0.500	2.000	0.155	0.048
Smolagents	0.750	4.000	0.292	0.114	0.667	2.000	0.143	0.000
Search-ol	0.667	4.000	0.171	0.061	2.208	9.000	0.344	0.024
Claude Code	0.950	5.000	0.251	–	0.500	3.000	0.343	–

Table 11: Performance on multi-constraint disjunctive (OR) queries. NACE and ACE report median normalized and absolute count error, respectively. Ent. R and Chunk R denote entity-level and chunk-level recall.

Method	Finance				Research			
	NACE ↓	ACE ↓	Ent. R ↑	Chunk R ↑	NACE ↓	ACE ↓	Ent. R ↑	Chunk R ↑
NaiveRAG	0.793	14.000	0.116	0.044	0.455	3.000	0.176	0.037
FLARE	0.909	16.000	0.315	0.090	0.800	6.000	0.343	0.065
RAT	0.628	11.500	0.263	0.092	0.429	3.000	0.371	0.031
DeepNote	0.750	14.000	0.163	0.033	0.750	5.000	0.209	0.012
HippoRAG	0.750	13.000	0.121	0.050	0.500	4.000	0.021	0.032
LightRAG	0.684	13.000	0.234	0.047	0.333	2.000	0.211	0.00570
PageIndex	0.806	15.000	0.131	–	0.769	6.000	0.079	–
Palimpzest	1.000	6.000	0.190	–	0.415	4.000	0.502	–
NaiveAgent	0.579	11.000	0.188	0.066	0.455	3.000	0.284	0.026
ReAct	0.619	12.000	0.198	0.049	0.455	4.000	0.223	0.001
ReadAgent	0.400	7.000	0.387	–	0.667	5.000	0.284	–
MultiAgent	0.586	11.000	0.190	0.066	0.429	3.000	0.279	0.025
Smolagents	0.768	15.000	0.154	0.055	0.538	4.000	0.253	0.052
Search-ol	0.667	12.000	0.158	0.048	0.889	7.000	0.301	0.044
Claude Code	0.533	9.000	0.181	–	0.429	3.000	0.240	–

Table 12: Paired Wilcoxon signed-rank tests between representative methods from the three evaluated paradigms. Non-significant entries report exact p -values, while statistically significant entries are denoted by * $p < 0.05$, ** $p < 0.01$, and *** $p < 0.001$.

Dataset	Method A	Method B	NACE	ACE	Chunk R.	Entity R.
Research	Naive RAG	LightRAG	0.1940	*	0.4009	**
Research	Naive RAG	ReAct	0.1047	0.0555	***	***
Research	LightRAG	ReAct	0.4944	0.1211	**	0.6522
Finance	Naive RAG	LightRAG	0.0783	0.0957	***	***
Finance	Naive RAG	ReAct	***	***	0.4269	***
Finance	LightRAG	ReAct	0.2281	0.2953	***	***

Table 13: Prompt templates for entity type extraction in financial reports.

Entity Type Extraction

You are an expert in financial information extraction from SEC 10-K filings.

Task:

Extract key entities from the given text and classify them into predefined categories.

Example Entity Types:

- Organization:
company, institution, or external entity.
- Product:
named product, platform, system, or service offered by a company.
- Technology:
technical concept, method, or capability.
- Market:
industry sector or application domain.
- Strategy:
clearly stated long-term business direction, initiative, or strategic focus.
- Risk:
potential factor that may negatively impact the company.
- Regulation:
law, policy, or government restriction affecting the company.
- Event:
specific business action or occurrence.
- Metric:
quantitative financial or operational indicator.

Guidelines:

- Extract only explicitly mentioned entities.
- Use concise and canonical names (avoid long phrases).
- Do not infer or hallucinate entities not present in the text.
- Avoid duplicates.
- If an entity could belong to multiple types, choose the most specific and appropriate one.

Output Format:

Return a JSON array ONLY, with no additional text:

```
```json
[
 { "type": "...", "name": "..." }
]
```
```

Note:

1. Extract only entities explicitly discussed in the text, and do not speculate
 2. Use Standard English terms for entity names
 3. If there are no extractable entities, return an empty list []
- Output only the JSON array, no text.

Text:

{chunk_text}

Table 14: Prompt templates for constraint extraction.

Attribute-value Constraint Extraction

You are an expert in entity-centric information extraction.

Task:

Extract constraints (restrictive or descriptive phrases) that constrain or specify the meaning of the entity "{entity_name}" from the text snippet. Constraints refer to attribute values that restrict or describe the entity.

Definition of "constraint":

A constraint is a short phrase that restricts or adds specific meaning to the entity. It typically captures:

Type: "generative AI" → "generative"

Domain / application: "AI for healthcare" → "healthcare"

Scale / scope: "large-scale AI systems" → "large-scale"

Condition / context: "AI in cloud environments" → "cloud environments"

Purpose / function: "AI for fraud detection" → "fraud detection"

In this task, constraints refer to attribute values, not attribute names.

For example, in "AI for healthcare applications", the attribute can be "domain" or "application", but the constraint to extract is "healthcare".

Strict Rules:

1. Extract ONLY phrases that directly constrain or specify the entity.
2. The constraint must be semantically tied to the entity, not general sentence context.
3. Keep constraints SHORT, preferably 1 to 4 words.
4. Prefer meaningful phrases over single vague words.
5. DO NOT extract:
 - Full clauses or long explanations
 - Generic words, such as "important", "various", or "many"
 - Attribute names, such as "type", "domain", "application", "context", "purpose", or "function"
6. Extract at most 3 constraints.
7. If no valid constraint exists, return an empty list.

Examples:

"generative AI models" → ["generative"]

"AI for healthcare applications" → ["healthcare"]

"large-scale AI systems" → ["large-scale"]

"AI used in cloud environments" → ["cloud environments"]

"AI for fraud detection" → ["fraud detection"]

"AI is transforming industries" → []

Text snippet:

"{context}"

Output format (STRICT JSON, no extra text):

```
{{
  "constraints": ["string", "string"]
}}
```

Table 15: Prompt templates for query verbalization.

Query Verbalization

Generate a question based on the given entity and constraint.

Generate a question in JSON format:

entity_level: A question about how this entity is described by/associated with the given constraint. This question asks about the specific entity's relationship with the constraint.

Output format (STRICT JSON):

```
{{
  "entity_level": "question about {entity} and {constraint}"
}}
```

Examples:

Input:

entity: competition

constraint: online

Output:

How many online competitions are in the corpus?

Input:

entity: data center

constraint: accelerated computing

Output:

How many accelerated computing data centers are in the corpus?

Now generate the question:

Input:

entity: {entity}

constraint: {constraint}

Output:

""

""Based on the following entity, constraint, and source context, generate a question:

Entity: {entity}

Constraint: {constraint}

Source Context:

{source}

Generate a question in JSON format:

entity_level: A question about how this entity is described by/associated with the given constraint. This question asks about the specific entity's relationship with the constraint.

Output format (STRICT JSON):

```
{{
  "entity_level": "question about {entity} and {constraint}"
}}
```

Requirements:

- Questions must be answerable from the source context
- Use natural, clear English
- Do not include any text outside the JSON object

Table 16: Prompt templates for LLM-based query–chunk pre-annotation.

LLM-based Query–chunk Pre-annotation

System message:

You are a precise evidence extractor for financial document analysis. You will be given one text chunk from a financial filing and a structured list of questions. Each question has a query_id, entity_type, attribute, and question text.

Use all four fields together as matching guidance, not only the question text.

Your task is to decide, for each question, whether the chunk explicitly supports it. If it does, extract one concrete named entity for that question.

Strict rules:

1. Use only the given chunk. Do not use external knowledge.
2. Do not answer the final count directly.
3. If the chunk does not explicitly support a question, omit that question from matches.
4. The entity_name must be a concrete entity explicitly mentioned in the chunk. It should be a proper name, a specific named product, a named transaction, or a named system.
5. Never use generic category phrases as entity_name. Forbidden examples include: cloud service providers, companies, customers, business, service, product, platform, market, revenue, competition, litigation, acquisition, and applications. If the chunk only discusses a category but does not name a concrete entity satisfying the question, return no match for that question.
6. Only match a question if the chunk explicitly supports a concrete named entity. Do not infer a match from the company name alone.
7. Do not use a table of contents, SEC cover page, filing metadata, address, phone number, trading symbol, or generic risk–factor text as evidence, unless the question directly asks about those fields.
8. Return only valid JSON. Do not use markdown. Do not use code fences. Do not include any explanation outside the JSON.

User message:

CHUNK ID: {chunk_id}

SOURCE FILE: {source_file}

CHUNK INDEX: {chunk_index}

CHUNK TEXT:

{chunk_content}

QUESTIONS:

{questions}

For each question that is explicitly supported by this chunk with a concrete named entity, add one entry to "matches".

If no question is supported, return:

```
{
  "chunk_id": "{chunk_id}",
  "source_file": "{source_file}",
  "chunk_index": {chunk_index},
  "matches": []
}
```

If there are matches, return only this JSON:

```
{
  "chunk_id": "{chunk_id}",
  "source_file": "{source_file}",
  "chunk_index": {chunk_index},
  "matches": [
    {
      "query_id": "q_001",
      "question": "the original question text",
      "entity_name": "Google Cloud",
      "entity_type": "public cloud service",
      "reason": "short reason grounded only in this chunk"
    }
  ]
}
```

Table 17: Examples of SEC EDGAR Form 10-K filings used in the finance corpus.

| # | Company | Ticker | Exchange | Report Date | Filing Date | Source |
|----|---------------------------|--------|----------|-------------|-------------|----------------------------|
| 1 | NVIDIA | NVDA | Nasdaq | 2025-01-26 | 2025-02-26 | SEC filing |
| 2 | Apple | AAPL | Nasdaq | 2025-09-27 | 2025-10-31 | SEC filing |
| 3 | Alphabet (Google) | GOOG | Nasdaq | 2025-12-31 | 2026-02-05 | SEC filing |
| 4 | Microsoft | MSFT | Nasdaq | 2025-06-30 | 2025-07-30 | SEC filing |
| 5 | Amazon | AMZN | Nasdaq | 2025-12-31 | 2026-02-06 | SEC filing |
| 6 | Meta Platforms (Facebook) | META | Nasdaq | 2025-12-31 | 2026-01-29 | SEC filing |
| 7 | Tesla | TSLA | Nasdaq | 2025-12-31 | 2026-01-29 | SEC filing |
| 8 | Micron Technology | MU | Nasdaq | 2025-08-28 | 2025-10-03 | SEC filing |
| 9 | Netflix | NFLX | Nasdaq | 2025-12-31 | 2026-01-23 | SEC filing |
| 10 | Oracle | ORCL | Nasdaq | 2025-05-31 | 2025-06-18 | SEC filing |
| 11 | AMD | AMD | Nasdaq | 2025-12-27 | 2026-02-04 | SEC filing |
| 12 | Palantir | PLTR | Nasdaq | 2025-12-31 | 2026-02-17 | SEC filing |
| 13 | Cisco | CSCO | Nasdaq | 2025-07-26 | 2025-09-03 | SEC filing |
| 14 | IBM | IBM | Nasdaq | 2025-12-31 | 2026-02-24 | SEC filing |
| 15 | Salesforce | CRM | Nasdaq | 2025-01-31 | 2025-03-05 | SEC filing |

Table 18: Research papers used in the research corpus.

| # | Title | Year | Venue | Source |
|----|---|------|---|-----------------------|
| 1 | <i>Advanced RAG Models with Graph Structures: Optimizing Complex Knowledge Reasoning and Text Generation</i> | 2024 | ISCEIC | Paper |
| 2 | <i>Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval</i> | 2021 | ICLR | Paper |
| 3 | <i>Answering Complex Questions by Joining Multi-Document Evidence with Quasi Knowledge Graphs</i> | 2019 | SIGIR | Paper |
| 4 | <i>AVIS: Autonomous Visual Information Seeking with Large Language Model Agent</i> | 2023 | NeurIPS | Paper |
| 5 | <i>Boosting Language Models Reasoning with Chain-of-Knowledge Prompting</i> | 2024 | ACL | Paper |
| 6 | <i>CancerKG.ORG: A Web-scale, Interactive, Verifiable Knowledge Graph-LLM Hybrid for Assisting with Optimal Cancer Treatment and Care</i> | 2024 | CIKM | Paper |
| 7 | <i>Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources</i> | 2024 | ICLR | Paper |
| 8 | <i>CodexGraph: Bridging Large Language Models and Code Repositories via Code Graph Databases</i> | 2025 | NAACL | Paper |
| 9 | <i>Complex Logical Reasoning over Knowledge Graphs using Large Language Models</i> | 2023 | arXiv preprint | Paper |
| 10 | <i>Empowering Large Language Models to Set up a Knowledge Retrieval Indexer via Self-Learning</i> | 2024 | arXiv preprint | Paper |
| 11 | <i>Enhancing Textbook Question Answering with Knowledge Graph-Augmented Large Language Models</i> | 2024 | ACML | Paper |
| 12 | <i>Exploring Large Language Model for Graph Data Understanding in Online Job Recommendations</i> | 2024 | AAAI | Paper |
| 13 | <i>From Local to Global: A Graph RAG Approach to Query-Focused Summarization</i> | 2024 | arXiv preprint | Paper |
| 14 | <i>Generative Subgraph Retrieval for Knowledge Graph-Grounded Dialog Generation</i> | 2024 | EMNLP | Paper |
| 15 | <i>Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs</i> | 2024 | ACL | Paper |
| 16 | <i>GraphCoder: Enhancing Repository-Level Code Completion via Code Context Graph-Based Retrieval and Large Language Models</i> | 2024 | arXiv preprint | Paper |
| 17 | <i>GraphGPT: Graph Instruction Tuning for Large Language Models</i> | 2024 | SIGIR | Paper |
| 18 | <i>Graph Neural Network Enhanced Retrieval for Question Answering of LLMs</i> | 2024 | arXiv preprint | Paper |
| 19 | <i>Graph of Records: Boosting Retrieval Augmented Generation for Long-context Summarization with Graphs</i> | 2025 | ACL | Paper |
| 20 | <i>GraphReader: Building Graph-based Agent to Enhance Long-Context Abilities of Large Language Models</i> | 2024 | EMNLP | Paper |
| 21 | <i>G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering</i> | 2024 | NeurIPS | Paper |
| 22 | <i>Knowledge Graph-Augmented Language Models for Knowledge-Grounded Dialogue Generation</i> | 2023 | arXiv preprint | Paper |
| 23 | <i>Knowledge Graph Enhanced Retrieval-Augmented Generation for Failure Mode and Effects Analysis</i> | 2025 | Journal of Industrial Information Integration | Paper |
| 24 | <i>Knowledge Graph Prompting for Multi-Document Question Answering</i> | 2024 | AAAI | Paper |
| 25 | <i>Knowledge Guided Text Retrieval and Reading for Open Domain Question Answering</i> | 2019 | arXiv preprint | Paper |
| 26 | <i>KnowledgeGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases</i> | 2023 | arXiv preprint | Paper |
| 27 | <i>Language is All a Graph Needs</i> | 2024 | EACL | Paper |
| 28 | <i>Large Language Models based Graph Convolution for Text-Attributed Networks</i> | 2024 | OpenReview submission | Paper |
| 29 | <i>LightRAG: Simple and Fast Retrieval-Augmented Generation</i> | 2025 | EMNLP | Paper |
| 30 | <i>LLaGA: Large Language and Graph Assistant</i> | 2024 | ICML | Paper |
| 31 | <i>Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation</i> | 2024 | arXiv preprint | Paper |
| 32 | <i>MEG: Medical Knowledge-Augmented Large Language Models for Question Answering</i> | 2024 | arXiv preprint | Paper |
| 33 | <i>Mitigating Large Language Model Hallucinations via Autonomous Knowledge Graph-based Retrofitting</i> | 2024 | AAAI | Paper |
| 34 | <i>MuseGraph: Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining</i> | 2024 | arXiv preprint | Paper |
| 35 | <i>ProLLM: Protein Chain-of-Thoughts Enhanced LLM for Protein-Protein Interaction Prediction</i> | 2024 | bioRxiv | Paper |
| 36 | <i>Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning</i> | 2024 | ICLR | Paper |
| 37 | <i>RHO: Reducing Hallucination in Open-domain Dialogues with Knowledge Grounding</i> | 2023 | ACL | Paper |
| 38 | <i>RiTeK: A Dataset for Large Language Models Complex Reasoning over Textual Knowledge Graphs</i> | 2024 | arXiv preprint | Paper |
| 39 | <i>RuleRAG: Rule-Guided Retrieval-Augmented Generation with Language Models for Question Answering</i> | 2024 | arXiv preprint | Paper |
| 40 | <i>SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning</i> | 2023 | CoRL | Paper |
| 41 | <i>Simple is Effective: The Roles of Graphs and Large Language Models in Knowledge-Graph-Based Retrieval-Augmented Generation</i> | 2025 | ICLR | Paper |
| 42 | <i>StructRAG: Boosting Knowledge Intensive Reasoning of LLMs via Inference-Time Hybrid Information Structurization</i> | 2025 | ICLR | Paper |
| 43 | <i>Structure Pretraining and Prompt Tuning for Knowledge Graph Transfer</i> | 2023 | WWW | Paper |
| 44 | <i>Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph</i> | 2024 | ICLR | Paper |