IMPLIRET: Benchmarking the Implicit Fact Retrieval Challenge

Zeinab Sadat Taghavi*, **Ali Modarressi***, **Yunpu Ma, Hinrich Schütze**
Center for Information and Language Processing, LMU Munich
Munich Center for Machine Learning (MCML)
Correspondence: [zeinabtaghavi, amodaresi]@cis.lmu.de

Abstract

Retrieval systems are central to many NLP pipelines, but often rely on surface-level cues such as keyword overlap and lexical semantic similarity. To evaluate retrieval beyond these shallow signals, recent benchmarks introduce reasoning-heavy queries; however, they primarily shift the burden to query-side processing techniques – like prompting or multi-hop retrieval – that can help resolve complexity. In contrast, we present IMPLIRET, a benchmark that shifts the reasoning challenge to document-side processing: The queries are simple, but relevance depends on facts stated implicitly in documents through temporal (e.g., resolving "two days ago"), arithmetic, and world knowledge relationships. We evaluate a range of sparse and dense retrievers, all of which struggle in this setting: the best nDCG@10 is only 14.91%. We also test whether long-context models can overcome this limitation. But even with a short context of only thirty documents, including the positive document, GPT-o4-mini scores only 55.54%, showing that document-side reasoning remains a challenge. Our codes are available at github.com/ZeinabTaghavi/IMPLIRET.

1 Introduction

Retrieval systems play a pivotal role in many NLP applications, enabling models to utilize relevant information from large corpora such as document collections, web pages, or conversational histories (Lewis et al., 2020; Gao et al., 2023). Relevance in retrieval can be established through a range of connections, from explicit lexical or semantic similarity to more implicit, context-dependent associations. However, widely used retrieval systems are highly reliant on surface-level cues such as exact matches, repetition, or where a fact appears in the text (Ram et al., 2023; Coelho et al., 2024; Fayyaz et al., 2025). Additionally, many popular benchmarks (e.g., BEIR (Thakur et al., 2021)) do not surface these issues as their queries have lexical overlap with relevant documents (Shao et al., 2025). There are attempts to create reasoning-intensive datasets that push beyond lexical and surface-level matches. For instance, RAR-b (Xiao et al., 2024) reframes multiple-choice reasoning tasks into retrieval problems, BIRCO (Wang et al., 2024) collects multi-faceted questions across five domains, and BRIGHT (Su et al., 2025) uses full StackExchange problem descriptions as queries against the pages they cite. Since the reasoning burden lies on the query side, techniques like query expansion, chain-of-retrieval inference, or agentic retrieval can help models handle complex prompts and outperform standard retrievers (Wang et al., 2025; Song et al., 2025; Li et al., 2025).

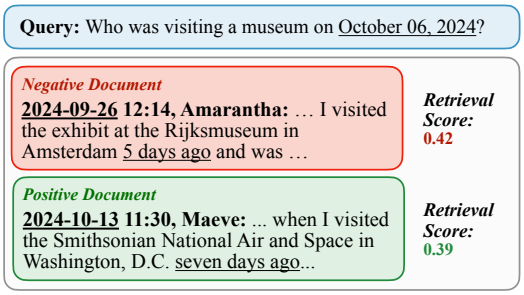

Figure 1: **An example from IMPLIRET:** a query and two sample documents, negative and positive. Retrieval of the relevant positive document requires surfacing implicit knowledge: that Maeve visited the Smithsonian on 2024-10-06.

In contrast, we present **IMPLIRET**, a benchmark that shifts reasoning to document-side processing: the queries are simple, but relevance depends on facts stated implicitly within the documents, spanning **arithmetic**, **temporal**, and **world knowledge** relationships that require inference to uncover. Figure 1 gives an example: the correct document requires resolving a reference to a date that is implicit, i.e., not stated directly. An effective retrieval

* Equal Contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 33168–33190
November 4-9, 2025

system must infer such implicit facts from the document content, ideally as part of the indexing process, in order to retrieve the correct result at query time. Yet current retrieval methods fail to capture the implicit signals needed for accurate retrieval. We evaluate sparse and dense approaches, including BM25 (Robertson and Zaragoza, 2009), ColBERT (Santhanam et al., 2022), and Dragon+ (Lin et al., 2023), and observe consistently poor performance: the best nDCG@10 is only 14.91% across our benchmark. To test whether long-context capabilities could mitigate the problem, we evaluate models in a setting where the positive document is included among several distractors. While GPT-o4-mini answers correctly when given only the positive document, its performance drops sharply even with just thirty documents in-context, achieving a ROUGE-1 recall of 55.54%. Our dataset IMPLIRET introduces a new setting that requires document-side reasoning for retrieval rather than query-side reasoning. IMPLIRET presents challenges for both retrieval and long-context processing, highlighting the need for models that can reason over implicit information embedded in large corpora.

2 IMPLIRET

In IMPLIRET, we construct examples whose relevance depends on information that is implicitly stated in the document, i.e., it can only be discovered through reasoning, not by surface-level overlap. IMPLIRET covers three reasoning categories: **World Knowledge**, **Arithmetic**, and **Temporal**.

We compile a **collection of implicit-tuple sets**. Within each set, a tuple links an implicit surface form that appears in a document to the explicit form that will appear in the query; see Fig. 1, e.g. (*"2024-10-13 ... seven days ago"*, *"October 06, 2024"*).

For every reasoning category, we create N such tuple sets. Each set T_i ($i = 1, \ldots, N$) contains M unique tuples ($|T_i| = M$). Tuples in the tuple sets are unique but not guaranteed to be unique throughout the collection of tuple sets. Hence, before document generation, we inject distinct auxiliary lexical entities (e.g. named entities, speaker names) into each tuple so that the documents generated from T_i remain distinguishable from those of T_j when $i \neq j$ (see Appendix A.4).

From each tuple in the tuple set, we generate a document, yielding a pool of documents $\mathcal{D}_{T_i}$ with $|\mathcal{D}_{T_i}| = M$. The document derived from $t_i \in T_i$ is the *only* positive for the query constructed from t_i, whereas all other documents in the global collection $\mathcal{D} = \bigcup_{i=1}^{N} \mathcal{D}_{T_i}$ – including those from tuples $t_i' \neq t_i$ in the same set and every document from any other set $T_j \neq T_i$ – are treated as negatives.

For each reasoning category, we generate two collections of tuple sets, one realized in the *uni-speaker* style and the other in the *multi-speaker* style, keeping their respective document pools separate to foster surface diversity. Thus, every query has exactly one positive document, while *every other document in the global collection* serves as a semantically irrelevant negative. In the remainder of this section, we detail the construction of the implicit-tuple sets and our procedure for generating documents and queries.

2.1 Generating Tuple Sets

Arithmetic. An arithmetic relation requires simple numerical reasoning. For instance, the query "Which bag costs $1,600?" can be answered by "The Prada bag costs $2,000, the Gucci bag is 20% cheaper," since $2,000 × 0.8 = $1600. Here, the model must identify the reference price, interpret the relative statement ("20% cheaper"), and perform the corresponding computation to infer the answer. Therefore, each tuple in the implicit tuple set takes the form $\big((p_1, r, e), p_2\big)$, where p_1 is the base price, r is the relative multiplier, $e \in \{\texttt{"Lower"}, \texttt{"Higher"}\}$ indicates the direction of the change, and p_2 is the queried price (e.g., $((2000, 0.2, \texttt{Lower}), 1600)$). We apply constraints to ensure that queried prices are unique, realistic, and well-distributed across the tuple set. Tuples are generated using a sampling algorithm that selects base prices and checks constraint satisfaction, backtracking as needed until M valid tuples are found (where M is the target number of documents indicated as "Docs" in Table 1). Full constraint details and sampling logic are provided in Appendix A.1.

World Knowledge. A world knowledge relation connects a textual mention to an external fact. For instance, the query "Who was in the UK?" can be answered by "Lenna was at Big Ben," based on the implicit fact that Big Ben is located in the UK. The model must identify the mentioned entity, retrieve the associated world fact, and use it to resolve the query. Each tuple is encoded as (*landmark*, *country*), e.g., ("Big Ben", "UK"). To build the tuple set, we collect landmark-country

Reasoning	Style	Docs	Tokens	
			Avg.	Total
Arithmetic	Uni-speaker	1500	553	830,098
	Multi-speaker	1500	142	213,750
World-Knowledge	Uni-speaker	1500	471	707,047
	Multi-speaker	1500	168	253,337
Temporal	Uni-speaker	1500	479	719,226
	Multi-speaker	1500	141	212,502

Table 1: **IMPLIRET statistics.** For each reasoning category and discourse style (uni-speaker vs. multi-speaker), we list the number of documents (*50 tuple sets* $\times$ *30 docs = 1500*), the average document length, and the total token count. Every document has exactly one associated query, so the document and query counts coincide.

pairs that are unambiguous, globally unique, free of lexical cues revealing the country, and refer to specific rather than generic locations. Candidates are sourced from Wikidata (Vrandečić and Krötzsch, 2014) and filtered using LLMs, embedding similarity, and web search verification. Full filtering criteria, prompts, and implementation details are provided in Appendix A.2. Here, we again generate a set of M tuples of each implicit tuple set.

Temporal. A temporal relation involves reasoning over relative dates; we gave an example in Figure 1. The model must identify the reference date (2024-10-13), interpret the relative time expression ("seven days ago"), and compute the resulting absolute date ("2024-10-06"). Each example is represented as a tuple $\big((d_B, R), D_L\big)$, where d_B is the base date explicitly mentioned in the document, R is a list of relative offsets (e.g. ["1 day after", "2 days after"]), and D_L is the list of resolved explicit dates (e.g., ["March 6th", "March 7th"]). We generate M such tuples under constraints that ensure date uniqueness, broad coverage across a fixed window, and realistic time offsets. Target date sequences are first sampled, then anchored to a base date to define relative expressions. The sampling algorithm verifies constraints and backtracks as needed until a valid set is found. Further details on constraints and sampling logic are provided in Appendix A.3.

2.2 Document-Query Pairs

We generate a document-query pair from every fact tuple, realizing it in one of two styles: *uni-speaker* (multi-turn chat) or *multi-speaker* (forum thread).

Uni-speaker (multi-turn chat). For each tuple, we create a short multi-turn dialogue. The same main conversant (e.g., "Alex") appears in every dialogue within a tuple set and never appears in any other tuple sets. To keep the interactions natural, the second conversant's name changes from one dialogue to the next. Depending on the reasoning category, the main conversant states which product they bought at a certain price (Arithmetic), mentions visiting a landmark (World Knowledge), or describes an activity that occurred on a specific date (Temporal). The query then targets the implicit fact contained in that statement: the product, person, or activity linked to the given price, country, or date.

Multi-speaker (forum thread, one post per user). Each tuple set receives a single **prompt** that serves as the thread's opening post. For that tuple set, we create a forum thread in which each post is authored by a different user, realizing one tuple, and all posts respond to the shared prompt. Thus, the thread mimics a discussion in which several users independently mention their purchase, visit, or scheduled activity, respectively. While the underlying actions mirror the uni-speaker setting, the query perspective shifts: instead of asking about an attribute of a known entity, it now asks which entity (product, person, or activity) satisfies a stated condition such as a price, location, or date.

Generation Pipeline. In both styles, i.e., in each conversation and post, every message includes a timestamp and speaker name (see Figure 1). In both styles, each example is produced via a three-step pipeline: (1) Entity binding: We assign entities (e.g., names, items, activities) to each tuple to create a plausible scenario and define the query target; (2) Document generation: We prompt an LLM to generate a chat or forum passage that embeds the entity and the implicit part of the tuple, without stating the explicit fact; (3) Verification: a second model attempts to extract the original tuple; we retain only examples where the intended fact is fully recoverable. This pipeline is supported by auxiliary lexical resources, including random names, brand-item pairs, and activity lists, as well as per-reasoning category prompt templates. We use GEMMA-3-27B-IT (Team, 2025) to synthesize the documents for each tuple.[1] Table 1 presents

[1]Details such as prompts and query templates are available in Appendix A.4.

IMPLIRET statistics[2].

Fluency and implicitness sanity check. We drew a stratified random sample of 72 instances (query–document pairs; 3 reasoning categories × 2 discourse styles × 12 per cell) and manually assessed each for (i) fluency, (ii) implicit support for the queried fact, and (iii) absence of explicit leakage (i.e., a verbatim statement of the fact). In this sample, all documents were fluent and supported the queries implicitly; under our rubric, we observed no cases of explicit leakage. Further details are provided in Appendix A.5.

3 Experiments

We employ IMPLIRET to probe whether state-of-the-art retrievers can perform *document-side reasoning*. Relevant documents are retrieved for each query among those documents that are in its corresponding (reasoning category and discourse style) group.

At test time, each query is compared to all its discourse style documents. Our evaluation covers a wide variety of retrieval methods: sparse lexical baseline BM25 (Robertson and Zaragoza, 2009; Lù, 2024); dense encoders CONTRIEVER, DRAGON+, and REASONIR (Izacard et al., 2021; Lin et al., 2023; Shao et al., 2025); late interaction model COLBERT V2 (Santhanam et al., 2022); and knowledge graph augmented retriever HIPPORAG 2 (Gutiérrez et al., 2025). Effectiveness is reported as nDCG@k in the main text; MRR@k appears in Appendix B.

4 Results

The nDCG@10 results across all reasoning categories are presented in Table 2. The highest average score, 14.91 (achieved by DRAGON+), shows the difficulty retrieval models face when reasoning over implicit facts in documents. More efficient baselines such as CONTRIEVER and BM25 perform substantially worse; notably, BM25 reaches just 12.24 due to its reliance on surface-level lexical overlap.

Performance varies across reasoning types: the Arithmetic category exhibits the largest performance spread (14.96 vs. 10.78), while it is narrowest for Temporal (12.83 vs. 10.98). Discourse style also plays a role: DRAGON+ scores 16.45% on multi-speaker examples compared to 13.37 on uni-speaker ones, suggesting that stylistic structure affects retrieval difficulty.[3]

[2]The tokens are counted using GPT-2 tokenizer (Radford et al., 2019).

Retriever	Reasoning			Average
	W. Know.	Arithmetic	Temporal	
Sparse				
BM25	14.69	11.06	10.98	12.24
Late-Interaction				
ColBERT v2	15.79	**14.96**	11.99	14.25
Dense Encoders				
Contriever	16.50	13.70	12.73	14.31
Dragon+	17.46	14.61	12.66	**14.91**
ReasonIR	**18.88**	10.78	11.25	13.64
Knowledge-Graph Augmented Indexer				
HippoRAG 2	16.62	14.13	**12.83**	14.53

Table 2: **Retrieval evaluation. nDCG@10** for our reasoning categories (world knowledge (W. Know.), arithmetic, and temporal), averaged over Uni-speaker and Multi-speaker documents) and "Average" of reasoning.

Experiment	k	Reasoning			Average
		W. Know.	Arithmetic	Temporal	
Llama 3.3 70B	1	**73.79**	**90.13**	**81.85**	**81.92**
	10	27.37	16.98	25.23	23.19
	30	17.43	4.42	10.29	10.71
GPT-4.1	1	**93.24**	**92.12**	**84.90**	**88.05**
	10	62.21	23.86	15.59	35.06
	30	53.91	9.28	6.93	22.90
GPT-o4-mini	1	**92.34**	**92.45**	**93.44**	**92.74**
	10	88.11	76.61	73.94	79.55
	30	75.44	76.31	14.86	55.54

Table 3: **RAG-style evaluation. ROUGE-1 (R-1) recall** for our reasoning categories (world knowledge (W. Know.), arithmetic and temporal, averaged over Uni-speaker and Multi-speaker documents) and "Average" across categories.

RAG Performance with an Oracle Retriever on Reason-Sensitive Documents. While retrieval quality clearly affects end-to-end performance, we ask whether an LLM with long-context capacity can still succeed *once* the relevant document is present. To test this, we use a retrieval-augmented generation (RAG) set-up with an **oracle retriever**, one that always includes the positive document in its top-k. The model sees the question together with k documents: one positive and $k-1$ hard negatives sampled from the same pool (among other $M-1$ samples), ensuring comparable style and topic. This configuration removes retrieval as a variable and isolates the LLM's document-side reasoning ability.

We evaluate three settings: k=1 (positive only),

[3]Full results per category and style in Appendix B.

k=10 (positive plus nine negatives), and k=30 (a full-pool setting where all documents from the pool are provided as context). The model receives the query along with the sequence of documents and must generate an answer. We evaluate three reader models: LLAMA 3.3 70B, GPT-4.1[4], and GPT-O4-MINI[5]. In Table 3, we report the average ROUGE-1 recall[6] scores to measure the overlap between the generated output and the positive answer (Lin, 2004). When given only the positive document (k=1), the models achieve average ROUGE-1 Recall of 81.92, 88.05, and 92.74. This suggests that the query itself is straightforward to answer once the relevant document is isolated. This also means that an LLM can solve the task if a high-performing retriever (which would retrieve the relevant document at rank 1) is available. However, as k increases (even with the positive included), performance declines, showing that LLMs struggle to focus on the correct evidence amid structurally similar negatives. This supports prior findings on long-context limitations and highlights the need for retrieving a small, focused set of documents rather than increasing context size (Kuratov et al., 2024; Modarressi et al., 2025).

Error Analysis RAG has two stages—retrieval and generation—so we analyze errors along two axes. **1. Retrieval side (Rank-1 vs. Positive).** For each query, we compare the retriever's top-1 passage with the annotated positive. We analyze the 60 queries where DRAGON+'s top-1 document differs from the positive (3 reasoning categories $\times$ 2 discourse styles $\times$ 10 queries), yielding 120 passages (top-1 and positive per query). We categorize misrank reasons into four groups: (i) *Word Overlap* (top-1 has extra query surface tokens), (ii) *Semantic Cue* (similar overlap but extra topical/theme terms), (iii) *Length* (overlap/semantics comparable; shorter passage chosen), and (iv) *Unknown* (indistinguishable under our heuristics). Table 4 shows that *Semantic Cue* is most frequent in arithmetic queries, and *Word Overlap* in temporal and world knowledge. **2. Generation side (Oracle-RAG, k=10 vs. all).** To isolate generation errors, we evaluate an oracle setting where the positive passage is guaranteed in context. We randomly select 60 queries ($3 \times 2 \times 10$) and evaluate two context sizes (k=10 and k=all, others selected randomly), yielding 120 cases. After reviewing outputs, we assign a single label to each incorrect answer: (i) *Malformation* (positive present, answer malformed), (ii) *No-answer/unrelated*, or (iii) *Distraction* (copied from a surface-similar distractor). Table 5 shows that *No-answer/unrelated* is most frequent, *Distraction* occurs mainly in temporal queries, and longer context reduces *Malformation* but raises *No-answer/unrelated*, suggesting that context length alone does not fix generation errors.

[4]Checkpoint: `gpt-4.1-2025-04-14`

[5]Checkpoint: `o4-mini-2025-04-16`

[6]R-1 Rec. = $\frac{|\text{Output Unigrams} \cap \text{Gold Answer Unigrams}|}{|\text{Gold Answer Unigrams}|}$

Reasoning	Word overlap	Semantic cue	Length	Unknown
Arithmetic	15%	55%	5%	25%
Temporal	55%	5%	35%	5%
W. Know.	50%	5%	30%	15%

Table 4: **Retrieval-side error types distribution for top-1 vs. positive.** For each reasoning category, we consider 20 query pairs (2 discourse styles $\times$ 10 queries); the percentages indicate the share of those 20 pairs for which the error type(column) was the primary reason the top-1 passage differed from the positive document (W. Know. = world knowledge).

Reasoning	k	Malformation	No-answer/Unrelated	Distraction
Arithmetic	10	40%	60%	0%
	all	20%	75%	5%
W. Know.	10	45%	55%	0%
	all	15%	85%	0%
Temporal	10	0%	35%	65%
	all	0%	40%	60%

Table 5: **Generation-side error type distribution under *oracle* RAG with two context sizes (k=10 vs. k=all)**, where the positive document is included in the context. Results are based on a randomly selected set of 60 queries (120 evaluated cases). Percentages are computed over incorrect answers within each (reasoning category, k) cell (W. Know. = World Knowledge).

5 Conclusion

We introduce IMPLIRET, a benchmark for evaluating retrieval models when relevance depends on document-side reasoning on implicit facts. Unlike prior datasets that emphasize complex queries, IMPLIRET shifts the reasoning burden to the documents. It covers three reasoning types – world knowledge, arithmetic, and temporal – and two discourse styles. Across sparse, dense, and KG-augmented retrievers, the best nDCG@10 is only 14.91. Even with GPT-o4-mini, given thirty documents including the positive, performance peaks at just 55.54%. These results highlight the difficulty of retrieving implicit facts and the need for models that can reason beyond surface cues.

Limitations

While our benchmark is carefully designed to evaluate implicit document-side reasoning in retrieval systems, it has the following limitations:

Synthetic Dataset. Documents and queries in IMPLIRET are synthesized using LLMs and structured templates. This allows control over the facts and how they are implicitly expressed, while avoiding conflicts. It also enables easy regeneration if data contamination or memorization is suspected. As with any synthetic benchmark, the data may differ slightly from naturally occurring text in discourse structure or topic diversity. All examples are in English and follow conversational formats (uni-speaker chats and multi-speaker forum posts). Although the use of LLMs helps ensure fluency, it introduces the risk of subtle hallucinations or unintended cues, which we address through automatic verification during dataset construction.

Reasoning Types & Level. In IMPLIRET, we only cover three simple categories of reasoning relations: arithmetic, temporal, and world knowledge, each with shallow composition. While the coverage of reasoning types is limited, the core finding remains: current retrievers struggle to locate relevant documents when reasoning is implicit, and LLMs fail to reliably attend to the correct evidence in long-context settings.

References

João Coelho, Bruno Martins, Joao Magalhaes, Jamie Callan, and Chenyan Xiong. 2024. Dwell in the beginning: How language models embed long documents for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 370–377, Bangkok, Thailand. Association for Computational Linguistics.

Mohsen Fayyaz, Ali Modarressi, Hinrich Schuetze, and Nanyun Peng. 2025. Collapse of dense retrievers: Short, early, and literal biases outranking factual evidence. *Preprint*, arXiv:2503.05037.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

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. *arXiv preprint arXiv:2502.14802*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Igorevich Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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.

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. *Preprint*, arXiv:2501.05366.

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.

Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6385–6400, Singapore. Association for Computational Linguistics.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *arXiv preprint arXiv:2407.03618*.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. 2025. Nolima: Long-context evaluation beyond literal matching. In *Forty-second International Conference on Machine Learning*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What are you token about? dense retrieval as distributions over the vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2481–2498, Toronto, Canada. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond.

Foundations and Trends® in Information Retrieval, 3(4):333–389.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. 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.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *Preprint*, arXiv:2504.20595.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations*.

Gemma Team. 2025. Gemma 3.

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)*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation. *Preprint*, arXiv:2501.14342.

Xiaoyue Wang, Jianyou Wang, Weili Cao, Kaicheng Wang, Ramamohan Paturi, and Leon Bergen. 2024. Birco: A benchmark of information retrieval tasks with complex objectives. *arXiv preprint arXiv:2402.14151*.

Chenghao Xiao, G Thomas Hudson, and Noura Al Moubayed. 2024. Rar-b: Reasoning as retrieval benchmark. *arXiv preprint arXiv:2404.06347*.

A Dataset Generation

In this appendix, we describe, for each of the three reasoning categories, *Arithmetic*, *World-Knowledge*, and *Temporal*, (i) how we construct the implicit-tuple sets and (ii) what arguments are required to synthesize their corresponding contexts. After covering tuple-set construction, we explain in Section A.4 how a language model is used to generate the final passages.

A.1 Arithmetic Reasoning

For each implicit tuple set in the collection, to generate M tuples for the Arithmetic category, we use Algorithm 1, which ensures that all tuples satisfy the these constraints: (i) all p_1 and p_2 values across the tuples are distinct, ensuring exactly one correct answer per query; (ii) all p_1 and p_2 values should be a multiple of 10, so that values resemble realistic prices; (iii) the p_2 values are evenly distributed across a predefined range of plausible prices, avoiding value clustering; and (iv) the resulting multiplier must have at most 2 decimals; hence, the required calculation is simple. The Algorithm1 returns a set of tuples of the form $((p_{1,i}, r_i, e_i), p_{2,i})$, where the two prices are mutually distinct and the multiplier r_i satisfies predefined numerical constraints. The construction guarantees uniform distribution across price ranges and ensures that each tuple encodes a plausible relative-price comparison suitable for reasoning-based retrieval.

A.2 World-Knowledge Reasoning

As described in Section 2, we gather landmark-country pairs under three constraints: (i) each landmark must refer to a globally unique location, avoiding names that could correspond to multiple places; (ii) the landmark name must not include lexical, semantic, or language-specific cues that reveal its country, avoiding surface-form shortcuts; and (iii) landmarks must refer to specific, recognizable sites rather than generic ones. To do so, we first assemble a seed list of unique landmark-country pairs via five steps: (i) issue a SPARQL query to Wikidata to retrieve every entity whose `instance-of` (P31) chain includes exactly one of the high-level place classes, museum (`Q33506`), university (`Q3918`), church building (`Q16970`), venue (`Q17350442`), or landmark (the superclass of `Q17350442`), and that is linked to exactly one sovereign state via the `country` (P17) property; (ii) for each hit, extract the English labels

Algorithm 1 Arithmetic Tuple Set

Require: Number of tuples M; *style* $\in$ {multi, uni}; Total number of attempts $limit$

if *style* $=$ multi **then**
  $(B_L, B_U) \leftarrow (50, 2050)$
else
  $(B_L, B_U) \leftarrow (50, 3050)$
end if
$\Delta \leftarrow \lfloor (B_U - B_L)/M \rfloor$
ImplicitTupleSet $\leftarrow \varnothing$
PriceSet $\leftarrow \varnothing$
repeat
  for $i \leftarrow 0$ **to** $M - 1$ **do**
    $p_{2,i} \leftarrow B_L + i\Delta$
    $r_i \leftarrow None$
    attempts$\leftarrow 0$
    repeat
      Sample $p_{1,i} \sim \text{Uniform}\big(\{x \in 10\mathbb{M} \mid B_L \leq x \leq B_U\}\big)$
      if $p_{2,i} > p_{1,i}$ **then**
        $r_i \leftarrow \dfrac{p_{2,i}}{p_{1,i}}$
      else
        $r_i \leftarrow \dfrac{p_{1,i}}{p_{2,i}}$
      end if
      attempts$\leftarrow$*attempts*$+1$
    until $(0 < r_i < 3$ **and** Round$(r_i,2) = r_i$ **and** $p_{1,i} \notin PriceSet$ and $p_{2,i} \notin PriceSet)$ **or** *attempts* exceeds *limit*
    if r_i **then**
      $e_i \leftarrow \begin{cases} \texttt{Lower}, & p_{2,i} < p_{1,i} \\ \texttt{Higher}, & \text{otherwise} \end{cases}$
      ImplicitTupleSet $\leftarrow$ ImplicitTupleSet $\cup\{((p_{1,i}, r_i, e_i), p_{2,i})\}$
    else
      $p_{2,i} \leftarrow p_{2,i} + 1$
      if $p_{2,i} = B_L + (i+1)\Delta$ **then**
        restart entire generation
      end if
    end if
  end for
until |ImplicitTupleSet| $= M$
return ImplicitTupleSet

of its enclosing administrative region (P131), city (P131 restricted to `Q515`), generic location (P276), and street (P669), yielding up to five concentric location strings; (iii) discard entities with missing, machine-generated, or multi-country labels, then drop any whose name tokens overlap these location strings; embed each remaining landmark-country pair with the 768-dimensional Contriever encoder and retain only those with cosine similarity below 0.25; (iv) pass the remainders to a 70B-parameter Llama-3.3 classifier that flags and removes generic names (e.g., "Downtown Club") or labels leaking their country, using exponential back-off retries until accepted; and (v) submit each accepted label to GPT-4o in web-search mode[7], prompting it to return the place's country in a dictionary format, and keep only those for which the model returns exactly one location. This process yields a balanced pool of 100 unique landmark-country pairs, ensuring we sample across different countries rather than multiple landmarks from the same country. Finally, for each implicit tuple set, we select a set of M distinct countries C and, for each $c \in C$, sample one landmark $l_c \sim \text{Uniform}(L_c)$ from that country's filtered list L_c. The resulting *implicit tuple sets* is as follows:

$$\textit{ImplicitTupleSets} \;=\; \{\,(l_c,\, c) \mid c \in C\,\}.$$

Generating N implicit tuple sets, we have our collection.

A.3 Temporal Reasoning

As described in Section 2, for each tuple set, we generate a implicit tuple sets of M tuples $\big((d_B, R), D_L\big)$ to have the following constraints: (i) all explicit dates in any D_L are unique within the tuple set, ensuring that each query maps to exactly one positive document; (ii) all dates used as base or resolved targets are evenly distributed across a fixed date window to avoid clustering; and (iii) all relative offsets in R must fall within a limited number of days from. Because context synthesis differs between multi-speaker and uni-speaker modes, we describe the two procedures separately in the following subsections.

A.3.1 Multi-Speaker: Tuple Construction

Here, we generate N implicit tuple lists, each containing M tuples. For generating them, we use Algorithm 2. The returned output contains all the information detailed before.

A.3.2 Uni-Speaker: Tuple Construction

We want to generate N implicit tuple sets, each containing M tuples. Each tuple set describes a main conversant's 28-day activity schedule. We categorize activities into three types:

1. **One-time:** executed exactly once in the 14-day period (we have 9 for then in each schedule).

2. **Repeating-Non-Sequential:** occurring on multiple, *non-consecutive* days (we have 3 for them in each schedule: 2 days, 3 days, and 2 days).

[7] Checkpoint: `gpt-4o-search-preview-2025-03-11`

Algorithm 2 Multi-Speaker Temporal Tuple Set

Require: $DateWindow$ from 2024-01-01 to 2024-12-31
DATESELECTION($n, DateWindow$): return n distance *date* in $DateWindow$ in which for $i \in [0, \ldots, n-2]$, days between $date_i$ and $date_{i+1}$ be between 2, up to 7 days, and the distance between $date_{n-1}$ and the end of $DateWindow$ be at least 14 days.

$\{\text{work\_date}_i\}_{i=0}^{19} \leftarrow$ DATESELECTION($20, DateWindow$)
for $i \leftarrow 0$ **to** 18 **do**
  $\text{message\_date}_i \leftarrow \text{work\_date}_{i+1}$
  $r_i \leftarrow (\text{message\_date}_i - \text{work\_date}_i).\text{days}$
  $TupleSet \leftarrow TupleSet \cup((\text{message\_date}_i, r_i), \text{work\_date}_i)$
end for
$\text{message\_date}_{19} \leftarrow \text{work\_date}_{19} +$ UNIFORMSAMPLE($(1, 7)$)
$r_{19} \leftarrow (\text{message\_date}_{19} - \text{work\_date}_{19}).\text{days}$
$TupleSet \leftarrow TupleSet \cup((\text{message\_date}_{19}, r_{19}), \text{work\_date}_{19})$
return $TupleSet$

3. **Repeating-Sequential:** performed on consecutive days (we have 3 for them in each schedule: 3 days, 3 days, and 4 days).

Each set covers two consecutive 14-day blocks, contains exactly M=30 activities, and is constructed in three phases: (i) scheduling activities without temporal overlap, (ii) selecting one *message time* per activity such that it differs from the scheduled lot, and (iii) packaging every activity into a tuple (day(s), start_hour, end_hour, message_time). Algorithm 3 details the procedure.

A.4 Synthesizing the Context

Now, for each reasoning category and document style, we have a list of implicit tuple sets containing all the information needed to have a consistent dataset. Consider having the auxiliary lexical content (personal names, daily-work verbs, brand names with corresponding items, and per-category forum topics and questions needed)[8], to each tuple of the implicit tuple list, we assign unique entities and then each tuple, contains all the information to generating the document in natural way. The exact item required for each reasoning category and style is mentioned in Table 6. Depending on the style, we proceed as follows:

A.4.1 Uni-Speaker (Chat-Style)

In the conversation-generation stage, we load the implicit tuple sets for each (category, style) pair. First, we use the

[8]Generated using Gemma-3-27B-it (Team, 2025)

Algorithm 3 Uni–Speaker Temporal Tuple Set

Require: period duration 14-days, day span 7:00–19:00
Auxiliary functions
PLACESCHEDULE(d, T, F, S): place a 2-4 hour scheduled slot, if T = 'Seq', for d consecutive days, if T = 'NonSeq', for d non-consecutive days, and if T = 'Once', for one day; mark blocks in F to remove the free times, add the slot into the schedule list S, and make a tuple from the list of occupied days D_L and start and end hours ((h_{start}, h_{end})) as $a = (D_{La}, h_{start}, h_{end})$ and returns it
SHIFTDATES(S,d_{off}): shift all the relative days to 14 days later if d_{off}=1.
SELECTQTIMES(S): Randomly selects a random day and hour (not exact time of start or end) as question time for each schedule to appear in the query.
DIFFTIMES(m, a): Returns a list of day differences between every scheduled day $d_{i,a}$ in the activity a and m ($(m - d_{i,a}).day$).

$S \leftarrow \varnothing$ ▷ Schedule list
$A \leftarrow \varnothing$ ▷ Activity list
for $period \in \{0, 1\}$ **do** ▷ two consecutive 14-day blocks
  $F \leftarrow \{d \mapsto$ INITFREE$(7, 19) \mid d = 1{:}14\}$ ▷ free-time map
  $d_{\text{off}} \leftarrow 14 \cdot period$ ▷ calendar shift
  for all len $\in \{3, 3, 4\}$ **do**
    $A \leftarrow A \cup$PLACESCHEDULE(len, *'Seq'*, F, S)
  end for
  for all len $\in \{2, 2, 3\}$ **do**
    $A \leftarrow A \cup$PLACESCHEDULE(len, *'NonSeq'*, F, S)
  end for
  for $i \leftarrow 1$ **to** 9 **do**
    $A \leftarrow A \cup$PLACESCHEDULE(len, *'Once'*, F, S)
  end for
  SHIFTDATES(S,d_{off})
end for
$\mathcal{Q} \leftarrow$ SELECTQTIMES(S)
TupleSet$\leftarrow \varnothing$
for all activity $a \in A$ **do**
  $m_a \leftarrow$ RANDOM($\mathcal{Q} \setminus \{\text{times}(a)\}$)
  $R_a \leftarrow$ DIFFTIMES(m_a, a)
  ▷ Message time
  $TupleSet \leftarrow TupleSet \cup((m_a, R_a), a)$
end for
return *TupleSet*

`STARTING_CONVERSATION_PROMPT` to generate $M = 30$ unique "starting phrases" for the tuple set (one for each tuple), ensuring that no two dialogues begin identically (Figure 2). Next, for each tuple, we prepend one of these starting phrases and feed the combination into the `CONVERSATION_GENERATION_PROMPT`; category-specific prompt templates and requirements are shown in Figure 5 for the Arithmetic, Figure 9 for the World-Knowledge, and Figure 13 for the Temporal reasoning. We then ask the model to produce exactly ten utterances per chat, verify the count, and regenerate any that do not meet this criterion. Finally, each completed conversation is submitted to a separate LLM

via the FEATURE_EXTRACTION_PROMPT, which must reconstruct the original tuple to confirm that the dialogue faithfully conveys the intended information (Figure 7 for the Arithmetic, Figure 11 for the World-Knowledge, and Figure 15 for the Temporal reasoning).

A.4.2 Multi-Speaker (Forum-Style)

In the forum-style generation stage, we similarly load the implicit tuple sets for each (category, style) pair. We first generate $M = 30$ unique starting phrases for the tuple set using the STARTING_CONVERSATION_PROMPT, so that each reply begins differently (Figure 4 for the Arithmetic, Figure 8 for the World-Knowledge, and Figure 12 for the Temporal reasoning). Then, for each tuple, we provide the forum topic, its base question, one starting phrases, and the tuple data to the CONVERSATION_GENERATION_PROMPT configured for forum responses; category-specific templates and requirements again appear in Figure 5 for the Arithmetic, Figure 8 for the World-Knowledge, and Figure 13 for the Temporal reasoning. We generate exactly five sentences per response. Finally, each forum reply is passed to a separate LLM via the FEATURE_EXTRACTION_PROMPT to extract the original tuple, ensuring the response accurately encodes the tuple's information (Figure 6 for the Arithmetic, Figure 10 for the World-Knowledge, and Figure 14 for the Temporal reasoning).

A.5 Human Evaluation Details

We performed a small, stratified sanity check to complement automatic validation. For each reasoning category $\times$ discourse style cell, we randomly selected two queries (12 instances per cell). For each selected query, candidate passages were ranked with the REASONIR retriever, and six passages were drawn by uniform sampling from two rank strata: top–20 (3) and bottom–20 (3). This yielded 72 passages in total. We assessed each passage for (i) *fluency*, (ii) *implicit support* of their corresponding queried fact (entailed but not stated verbatim), and (iii) *absence of explicit leakage*. In this sample, all passages were fluent and passed the implicitness and non-leakage checks under our rubric.

B Results

As explained in Section 4, our main retrieval metric is nDCG@10. For completeness, we also compute MRR@10, summarized in Table 7. The lower MRR@10 scores confirm that, across reasoning categories, the systems often fail to rank the positive documents first, underscoring the modest overall performance already suggested by nDCG. Granular results for the *Uni-Speaker* and *Multi-Speaker* settings are provided in Table 8. For the RAG-style evaluation, model outputs were generated using the prompt templates shown in Figures 16 and 17, and then evaluated using ROUGE-1 Recall against the reference answer.

Reasoning	Multi-speaker		Uni-speaker	
	Each Pool	Each Document	Each Pool	Each Document
Arithmetic	Topic, Forum-Base-Question	Conversant, Brand, Model Years (Two random numbers between 2013 up to 2024, the lower number is the price of the lower-priced item)	Main Conversant	Second Conversant, Shopping item, Low priced brand name, High priced brand
World-Knowledge	Topic, Forum-Base-Question	Conversant, Landmark	Main Conversant	Second Conversant, Landmark, Message Date
Temporal	Topic, Forum-Base-Question	Conversant, Item related to the Forum Base Question,	Main Conversant	Second Conversant, Daily work verb

Table 6: **Entity-assignment granularity for each reasoning category and document style.** Items listed under *Each Pool* are unique in that pool and are never reused across other pools. Items listed under *Each Document* are unique within their own set and never reused across other Documents in that pool. All entities are drawn from the auxiliary lexical resources described in Section A.4.

STARTING_CONVERSATION_PROMPT for Uni-Speaker (Chat Style) documents

Task
Generate {num_starting_points} distinct, natural-sounding first phrases suitable as the opening line of a response in an online forum discussion, for example, "I think", "In my point of view".

Requirements
- No numbering, bullets, or extra text before or after each sentence.
- Tone must be friendly, approachable, and universally applicable.
- They should be usable at the start of the response, not in the middle.
- Avoid any topic-specific references.
- Use general phrasing.
- Do not mention purchases or someone buying something.
- Do not include numerical references in the sentences.
- Do not use any locational information in the sentences.

Output Format
At least {num_starting_points} distinct phrases.
Separate each sentence with a blank line.

Figure 2: Prompt for generating a list of the first phrase in Uni-Speaker (Chat Style) documents. This prompt is used for all the reasoning categories of the Arithmetic, World-Knowledge, and Temporal.

Retriever	Reasoning			Average
	W. Know.	Arithmetic	Temporal	
Sparse Baseline				
BM25	9.55	7.42	6.83	7.93
Late-Interaction				
ColBERT v2	10.59	**9.63**	7.55	9.26
Dense Encoders				
Contriever	11.19	8.84	8.48	9.50
Dragon+	11.97	9.47	8.26	**9.90**
ReasonIR	**14.23**	7.13	7.78	9.71
Knowledge-Graph–Augmented Indexer				
HippoRAG 2	11.30	9.28	**8.57**	9.72

Table 7: **MRR@10** ranking metric scores for our reasoning category of World-Knowledge (W. Know.), Arithmetic, and Temporal, averaged over both Uni-speaker and Multi-speaker documents. The final "Average" column reports the mean MRR@10 across all reasoning categories.

Experiment	World-Knowledge				Arithmetic				Temporal				Average			
	Uni-speaker		Multi-speaker		Uni-speaker		Multi-speaker		Uni-speaker		Multi-speaker		Uni-speaker		Multi-Speaker	
	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10
Sparse Baseline																
BM25	9.23	14.18	9.86	15.20	9.23	14.05	5.61	8.07	6.33	10.20	7.33	11.76	8.26	12.81	7.60	11.68
Late-Interaction																
ColBERT v2	9.57	14.54	11.60	17.03	**9.45**	**14.62**	9.81	15.30	6.85	10.86	8.25	13.12	8.62	13.34	9.89	15.15
Dense Encoders																
Contriever	9.77	14.77	12.60	18.23	8.23	12.87	9.45	14.52	8.17	12.22	8.78	13.25	8.72	13.29	10.28	15.33
Dragon+	9.94	14.85	14.00	20.07	9.11	14.11	9.83	15.12	7.20	11.16	9.32	14.16	8.75	13.37	11.05	16.45
ReasonIR	7.75	10.54	**20.71**	**27.21**	3.87	5.68	**10.39**	**15.89**	2.95	4.17	**12.60**	**18.33**	4.86	6.80	**14.57**	**20.48**
Knowledge-Graph–Augmented Indexer																
HippoRAG 2	**9.95**	**14.94**	12.66	18.30	8.31	12.97	10.26	15.29	**8.18**	**12.24**	8.96	13.42	**8.81**	**13.38**	10.63	15.67

Table 8: **MRR@10** and **nDCG@10** for each reasoning category and discourse setting. The maximum value in every metric column is bold-faced. The final "Average" block shows per-setting means over the three categories.

STARTING_CONVERSATION_PROMPT for Multi-Speaker (Forum Style) documents

Task
Generate {num_starting_points} distinct, natural-sounding first phrases suitable as the opening line of a conversation between two friends, for example, "Hey! How's it going?", "Anything exciting happening?".

Requirements
- No numbering, bullets, or extra text before or after each sentence.
- Tone must be friendly, approachable, and universally applicable.
- They should be usable at the start of the conversation, not in the middle.
- Avoid any topic-specific references.
- Use general phrasing.
- Do not mention purchases or someone buying something.
- Do not include numerical references in the sentences.
- Do not use any locational information in the sentences.

Output Format
At least {num_starting_points} distinct phrases.
Separate each sentence with a blank line.

Figure 3: Prompt for generating a list of the first phrase in Multi-Speaker (Forum Style) documents. This prompt is used for all the reasoning categories of the Arithmetic, World-Knowledge, and Temporal.

CONVERSATION_GENERATION_PROMPT for Arithmetic Reasoning, Multi-Speaker (Forum Style) documents

```
**Task**
Generate a natural response to a forum question.

**Input**
- `topic`: A short topic of the forum discussion.
- `forum_question`: A base question posted in the forum.
- `user`: a dictionary with the following keys:
  - `name`: Name of the user.
  - `persona`: Persona of the user.
- `forum_post`: A list of three sentences:
  1. The price of an item from a certain brand's model in dollars (e.g., "Gaming Chairs from Secretlab model 2019: 1650 dollars").
  2. The price of another item from the same brand but a different model, described relative to the first item (e.g., "Secretlab, model 2019: 2.5 times more expensive than model 2016").
  3. A sentence stating which model of the brand was ultimately purchased (e.g., "model 2016 was purchased").
- `starting_phrase`: A starting phrase for the opening line of a response in an online forum discussion

**Requirements**
- Answer the `forum_question` by using the information in `forum_post`.
- You may incorporate details from `user["persona"]` about `user['name']` to make the response more natural.
- Explicitly mention the brand and model references, or the model that was purchased.
- Preserve the numeric references (prices, multipliers, etc.). You may write the numbers as words, but do not change their values (e.g., 3.5 → "three and a half").
- Use the `starting_phrase` as the opening line of the response.
- Write the relative price in a natural way (e.g., "The Secretlab 2019 model costs two and a half times as much as the 2016 model.").
- Only mention the information once in the response.

- Ensure all sentences are grammatically correct.
- Your generated answer must be coherent and make the answer sound like a real human reply in **five sentences**.

**Output Format**
Only **one line** of response, without any prefix or suffix.

INPUT: {context}
```

Figure 4: Prompt for generating the conversations for Multi-Speaker (Forum Style) documents in the Arithmetic reasoning.

CONVERSATION_GENERATION_PROMPT for Arithmetic Reasoning, Uni-Speaker (Chat Style) documents

```
**Task**
Generate a natural conversation between two people ("user_1" and "user_2") based on a shopping list.

**Input**
- `user_1`: A dictionary with the following keys:
  - `name`: Name of the first user.
  - `persona`: Persona of the first user.
- `user_2`: A dictionary with the following keys:
  - `name`: Name of the second user.
  - `persona`: Persona of the second user.
- `shopping_type`: Type of shopping.
- `item_to_buy`: The purchased item.
- `bought`: A list of three sentences:
  1. The price of the item in another brand.
  2. The price of the item in the brand bought, relative to the first.
  3. The brand bought.
  - `starting_phrase`: A starting phrase for the opening line of a conversation between two friends

**Requirements**
- In the conversation, `user_1['name']` must share a message describing their shopping experience: it was in the `shopping_type` category and they bought the `item_to_buy`.
- `user_2['name']` must engage naturally in the conversation but should not mention or comment on any shopping, timing, locational, or numerical information.
- You may use details from `user_1["persona"]` and `user_2["persona"]` to make the dialogue more natural.
- Mention the exact `shopping_type`, brands, and `item_to_buy` **once** in the conversation.
- Preserve all exact numbers and the original relative phrasing contained in the `bought` sentences.
- Explicitly state that `user_1` did **not** buy from the first brand.
- Explicitly state that `user_1` **did** buy from the second brand.
- **Place the complete shopping report in exactly one user_1 utterance of your choice** (it may be the 2nd, 3rd, 7th—any single line).
  - That utterance must contain the literal text of `shopping_type`, `item_to_buy`, **all brand names**, and **all numbers** from the three `bought` sentences.
  - After that line, neither speaker may repeat or partially restate those strings or figures; use indirect terms like "it", "the item", or "that second brand" instead.
  - No additional brands, items, or numerical prices may be introduced elsewhere.
- Use the `starting_phrase` as the opening line of the first utterance.
- All sentences must be grammatically correct.
- The conversation must consist of **exactly 10 utterances**, each on its own line.

**Output Format**
Only 10 lines of dialogue are separated by newlines. For each line, separate the user name (one of the values of `user_1['name']` or `user_2['name']`) and the utterance with a colon.

### EXAMPLE (structure only)
user_1['name']: <starting_phrase> ...
user_2['name']: ...
...


INPUT: {context}
```

Figure 5: Prompt for generating the conversations for Uni-Speaker (Chat Style) documents in the Arithmetic reasoning.

FEATURE_EXTRACTION_PROMPT for Arithmetic Reasoning, Multi-Speaker (Forum Style) documents

Task
Identify purchased items, their brands, and prices in a conversation transcript.

Input
You are given a forum response transcript with 4 sentences as INPUT.

Requirements
- Your task is to identify items purchased in the conversation, the brand of those items, and their prices. Note that the cost might be stated explicitly or described in relative terms. In such relative cases, you must calculate the final price numerically based on the information available.
- Identify any items that someone actually buys or mentions buying.
- Determine the brand associated with each purchased item (if specified).
- Extract or compute the price in dollars, performing calculations for relative pricing.
- If no purchases are found, return an empty list.
- Do not include any additional commentary.

Output Format
A list of dictionaries with keys:
- `item` (string): Name of the purchased item.
- `brand` (string): Brand name.
- `model` (integer): Model of the item.
- `price` (integer): Price in dollars.

INPUT: {context}

Figure 6: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Multi-Speaker (Forum Style) documents in Arithmetic reasoning.

FEATURE_EXTRACTION_PROMPT for Arithmetic Reasoning, Uni-Speaker (Chat Style) documents

Task
Identify purchased items, their brands, and prices in a conversation transcript.

Input
Input is a conversation transcript as a list of lines, each:
<user name>: <utterance>

Requirements
- Detect items someone buys or mentions buying.
- Determine the brand for each purchased item.
- Extract or compute the price in dollars, performing calculations for relative pricing.
- If no purchases are found, return an empty list.
- Do not include any additional commentary.

Output Format
A list of dictionaries with keys:
- `item` (string): Name of the purchased item.
- `brand` (string): Brand name.
- `price` (integer): Price in dollars.

INPUT: {context}

Figure 7: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Multi-Speaker (Forum Style) documents in Arithmetic reasoning.

CONVERSATION_GENERATION_PROMPT for World-Knowledge Reasoning, Multi-Speaker (Forum Style) documents

Task
Generate a natural response to a forum question.

Input
- `topic`: A short topic of the forum discussion.
- `forum_question`: A base question posted in the forum.
- `forum_post`: The location that the person wants to use for responding to the "forum_question".
- `user`: a dictionary with the following keys:
  - `name`: Name of the user.
  - `persona`: Persona of the user.
- `type_of_location`: The type of location that the person in the post is talking about in "forum_post".
- `starting_phrase`: A starting phrase for the opening line of a response in an online forum discussion

Requirements
- In the response, you should answer the "forum_question" by stating that you participated in the activity mentioned in "topic" at the "forum_post" location or at a place directly behind it.
- If the activity in `topic` would be inappropriate at the exact `forum_post` location, you may instead reference a suitable place immediately next to or behind it (e.g., "the dance studio just behind St. Mary's Church"), while still mentioning the `forum_post` location name exactly once.
- You can use the information in `user["persona"]` that is about the `user['name']` to make the response more natural.
- You should mention that the user was in the location mentioned in "forum_post" or behind it, and you can use the information in `type_of_location` to know the type of location mentioned in "forum_post", but make sure that you mention the location name exactly as it is in "forum_post".
- Only mention the `forum_post` location name once in the response.
- Do not alter the location in `forum_post`; use it exactly as it is without any changes.
- Do not mention any other location than the one in `forum_post`.
- Use the `starting_phrase` as the opening line of the response.
- Make sure that you generate grammatically correct sentences.
- Your generated answer must be coherent and must read naturally, as if a human is really answering the `forum_question`, in exactly 5 sentences.

Output Format
Only 1 line of response, without any prefix or suffix.

INPUT: {context}

Figure 8: Prompt for generating the conversations for Multi-Speaker (Forum Style) documents in the World-Knowledge reasoning.

CONVERSATION_GENERATION_PROMPT for World-Knowledge Reasoning, Uni-Speaker (Chat Style) documents

Task
Generate a natural conversation between two people ("user_1" and "user_2") based on a trip.

Input
- `user_1`: A dictionary with the following keys:
  - `name`: Name of the first user.
  - `persona`: Persona of the first user.
- `user_2`: A dictionary with the following keys:
  - `name`: Name of the second user.
  - `persona`: Persona of the second user.
- `trip_destination`: Destination of the trip.
- `type_of_location`: The type of location.
- `trip_purpose`: The purpose of the trip.
- `starting_phrase`: A starting phrase for the opening line of a conversation between two friends

Requirements
- In the conversation, user_1['name'] will share a trip-information message stating that they were at the `trip_destination` for the purpose specified in `trip_purpose`, while user_2['name'] must engage naturally but must not reveal or comment on any trip or locational information.
- Mention the `trip_destination` exactly once, spelled exactly as provided, and do not add or change any details. Do not mention its country or city.
- Mention the `trip_purpose` exactly once, spelled exactly as provided.
- If the activity in `trip_purpose` would be inappropriate at the exact `trip_destination` location, you may instead reference a suitable place immediately next to or behind it (e.g., "the dance studio just behind St. Mary's Church"), while still mentioning the `trip_destination` location name exactly once.
- Do not mention any other locational information in the conversation.
- user_2['name'] replies naturally without referencing trip or locational information.
- You may use the information in user_1["persona"] and user_2["persona"] to make the responses more natural.
- `type_of_location` describes the kind of place user_1 visited and can help make the conversation sound natural.
- Use the `starting_phrase` as the opening line of the first utterance.
- **Place the complete trip note in exactly one user_1 utterance of your choice** (it may be the 2nd, 3rd, 7th—any single line).
  - That utterance must contain the literal text of `trip_destination` and `trip_purpose`, each spelled exactly as provided.
  - After that line, either speaker may refer to the place or activity only indirectly (e.g., "there", "it", "that visit") and must never restate or partially repeat those exact strings.
  - No additional locational or purpose details may be introduced later.
- Make sure that you generate grammatically correct sentences.
- The conversation must consist of exactly 10 utterances.
- Each utterance is on its own line.

Output Format
Only 10 lines of dialogue are separated by newlines. For each line, separate the user name (one of the values of `user_1['name']` or `user_2['name']`) and the utterance with a colon.
EXAMPLE (structure only)
user_1['name']: <starting_phrase> ...
user_2['name']: ...
...

INPUT: {context}

Figure 9: Prompt for generating the conversations for Uni-Speaker (Chat Style) documents in the World-Knowledge reasoning.

FEATURE_EXTRACTION_PROMPT for World-Knowledge Reasoning, Multi-Speaker (Forum Style) documents

Task
Identify the location that the person was in.

Input
You are given a forum response transcript with 4 sentences as INPUT.

Requirements
- Your task is to identify the location that the person was in.
- Extract the exact name and do not change it.
- If no location is found, return an empty list.
- Do not include any additional commentary.

Output Format
A list of dictionaries with the key:
- `location` (string): Name of the location that the person was in.

INPUT: {context}

Figure 10: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Multi-Speaker (Forum Style) documents in the World-Knowledge reasoning.

FEATURE_EXTRACTION_PROMPT for World-Knowledge Reasoning, Uni-Speaker (Chat Style) documents

Task
Identify the destination of the trip and the purpose of the trip.

Input
Input is a conversation transcript as a list of lines, each:
<user name>: <utterance>

Requirements
- Detect the destination of the trip and the purpose of the trip.
- If no destination or purpose is found, return an empty list.
- Do not include any additional commentary.

Output Format
A list of dictionaries with keys:
- `destination` (string): Name of the destination of the trip.
- `purpose` (string): Name of the purpose of the trip.

INPUT: {context}

Figure 11: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Uni-Speaker (Chat Style) documents in the World-Knowledge reasoning.

CONVERSATION_GENERATION_PROMPT for Temporal Reasoning, Multi-Speaker (Forum Style) documents

Task
Generate a natural response to a forum question.

Input
- `topic`: A short topic of the forum discussion.
- `forum_question`: A base question posted in the forum.
- `forum_post`: The item related to the topic that the person wants to use for responding to the "forum_question".
- `user`: a dictionary with the following keys:
  - `name`: Name of the user.
  - `persona`: Persona of the user.
- `offset_days`: The relative date (e.g., "3 days ago") that the person in the post is talking about in "forum_post".
- `starting_phrase`: A starting phrase for the opening line of a response in an online forum discussion

Requirements
- In the response, answer the "forum_question" by stating that the user did the work on the date given in "offset_days", choosing a verb appropriate to the 'topic'.
- You can use the information in `user["persona"]` about `user['name']` to make the response more natural.
- Mention the `forum_post` item exactly once and do not mention any other item.
- Do not alter `offset_days`; use it exactly as written, though you may spell out its number component (e.g., "2 days ago" or "two days ago"). Do not convert it to a calendar date.
- The work must have occurred on a single day; avoid vague temporal expressions such as "until", "by the ...", "completed", or "finished".
- Do not mention any date other than the one in `offset_days`.
- Use the `starting_phrase` as the opening line of the response.
- Make sure that you generate grammatically correct sentences.
- Your generated answer must be coherent and sound natural, as if a real person is answering the `forum_question`, in exactly five sentences.

Output Format
Only 1 line of response, without any prefix or suffix.

INPUT: {context}

Figure 12: Prompt for generating the conversations for Multi-Speaker (Forum Style) documents in the Temporal reasoning.

CONVERSATION_GENERATION_PROMPT for Temporal Reasoning, Uni-Speaker (Chat Style) documents

Task
Generate a natural conversation between two people ("user_1" and "user_2") based on the given schedule.

Input
- `user_1`: A dictionary with the following keys:
  - `name`: Name of the first user.
  - `persona`: Persona of the first user.
- `user_2`: A dictionary with the following keys:
  - `name`: Name of the second user.
  - `persona`: Persona of the second user.
- `work`: The work task.
- `hours`: The hours that the work is to be performed.
- `offset_days`: A list describing when the work was done, relative to `message_time`. Each element is either a single relative day (e.g., '3 days ago', 'yesterday', 'today', 'in 2 days') or a span (e.g., 'Starting in 3 days for 4 consecutive days').
- `message_time`: The time that the conversation is being sent: [date of the message, day of the week, hour in 24h format]
- `starting_phrase`: A starting phrase for the opening line of a conversation between two friends

Requirements
- In the conversation, user_1['name'] will share a message describing their recent or upcoming work schedule and must mention the `work` and all `offset_days` in a single utterance.
- user_2['name'] must engage naturally in the conversation but should not mention or comment on any schedule, timing, or numerical details.
- You can use the information in user_1["persona"] that is about the user_1['name'], and user_2["persona"] that is about the user_2['name'] to make the response more natural.
- You should mention that the user_1['name'] did the 'work' on the specific day or days. Mention the day(s) of work using the same relative phrasing as in offset_days. You may express numbers as words (e.g., '2 days ago' or 'two days ago'), but do not rephrase or summarize the content of any span.
- All the work is being done in the same hour interval as specified in hours, you should not directly mention the end hour, but make sure that you accurately mention end hour relative to the start hour (e.g., "from 1 p.m. until 3 hours after that" or "from 9 in the morning for three hours"). Do not change the hours.
- Mention the `work` in the conversation exactly as it is (only change the tense if needed).
- **Place the full schedule in exactly one user_1 utterance of your choice** (it may be the 2nd, 3rd, 7th—any single line).
  - That utterance must:
    - include the literal text of `work` (tense may change),
    - repeat every phrase in `offset_days` verbatim, and
    - give the hour window exactly once, phrased relative to the start hour (e.g., "from 1 p.m. until three hours after that").
  - After that line, either speaker may refer to the activity only indirectly ("it", "those sessions", "the task") and must **never** restate the exact `work` string, schedule, or hours.
  - No new dates, spans, or numerical details may appear elsewhere.
- Do not change the "message_time" information. Ensure that the "hours" you use in the conversation for "work" are correct and accurate. For example, if the work is "updating a work log" and the "message_time" is ("2023-07-21", "Friday", 14), and the "hours" are (7, 10), You can use it like this: "2023-07-21", "Alaina", "I have to update a work log tomorrow from 7 in the morning for three hours."
- The message time is the time at which the conversation is being sent; Use the hour provided in message_time. For each utterance, randomly select a valid minute (00-59), ensuring that time either increases or remains the same across the 10 utterances.The final format of the message time should be like this: "YYYY-MM-DD HH:MM" (e.g., "2024-01-01 12:00").
- Use the `starting_phrase` as the opening line of the first utterance.
- Make sure that you generate grammatically correct sentences.
- The conversation must consist of exactly 10 utterances.
- Each utterance is on its own line.

Output Format
Only 10 lines of dialogue are separated by newlines. For each line, separate the final formatted message time and the user name (one of the values of `user_1['name']` or `user_2['name']`) with a comma, and separate the user name and the utterance with a colon.
EXAMPLE (structure only)
reformed_message_time, user_1['name']: <starting_phrase> ...
reformed_message_time, user_2['name']: ...
...

INPUT: {context}

Figure 13: Prompt for generating the conversations for Uni-Speaker (Chat Style) documents in the Temporal reasoning.

FEATURE_EXTRACTION_PROMPT for Temporal Reasoning, Multi-Speaker (Forum Style) documents

Task
Identify a work-related task described in the user's mention for the forum response and extract its temporal details. Specifically, you should:

1. Determine the work task (e.g., the action or project mentioned).
2. Identify any temporal expressions referring to when the work is to be performed. Convert relative time expressions (such as "tomorrow", "next week", etc.) into numerical offset_days (e.g., "1 day ago", "2 days ago", "3 days ago", etc.). Be very careful that the relevant dates are correct.

Input
You are given a forum response transcript with 4 sentences as INPUT.

Requirements
- Your task is to identify the work task and the `offset_days`.
- Mention the `offset_days` as a number with words (e.g., "1 day ago", "2 days ago", "3 days ago", etc.).
- Extract the exact work task and do not change it.
- If no work task or offset_days is found, return an empty list.
- Do not include any additional commentary.

Output Format
A list of dictionaries with the keys:
- `work` (string): The work task.
- `days` (string): The offset_days.

INPUT: {context}

Figure 14: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Multi-Speaker (Forum Style) documents in the Temporal reasoning.

FEATURE_EXTRACTION_PROMPT for Arithmetic Reasoning, Uni-Speaker (Chat Style) documents

Task
Identify a work-related task described in the conversation and extract its temporal details.

Input
Input is a conversation transcript as a list of lines, each:
<message time>, <user name>: <utterance>

Requirements
- Determine the work task (e.g., the action or project mentioned).
- Identify any temporal expressions referring to when the work is to be performed. Convert relative time expressions (such as "tomorrow", "next week", etc.) into absolute dates (YYYY-MM-DD) using the conversation date as a reference. Be very careful that the relevant dates be correct.
- Extract the time range mentioned for the task and express it as a tuple of two integers representing the start and end hours in 24-hour format.
- If no work task or offset_days is found, return an empty list.

Output Format
A list of dictionaries with keys:
- `work` (string): A string describing the identified task.
- `days` (list): A list of one or more dates (YYYY-MM-DD) on which the task occurs.
- `hours` (tuple): A tuple of two integers representing the start and end hours.

INPUT: {context}

Figure 15: Prompt for reconstructing the original tuple of implicit tuple set (extracting features) from generated conversations for Uni-Speaker (Chat Style) documents in the Temporal reasoning.

RAG_STYLE_PROMPT for Multi-Speaker (Forum Style) documents

```
**Task**
Answer the Question based on the context provided.

**Input**
- The INPUT contains a Topic, Forum Question and several Responses to the Forum Question.
- Each response is mentioned by a number in the following format:
Response {{number}}:
- Each response is separated by a new line.
- Each response contains the date, speaker and the message in the following format:
<date>, <speaker>: <message>

**Output**
return the final answer in a new line after "Answer:" without any prefix or suffix.

INPUT:
{context}

Answer the following question as precisely as possible, using the information provided in the responses. You may
rely on the response content, the time each response was sent, and who sent it.

Question: {question}

Answer:
```

Figure 16: Prompt for RAG-style experiment, while the input is forced to contain the positive document, in Multi-Speaker (Forum Style). This prompt is used for all the reasoning categories of the Arithmetic, World-Knowledge, and Temporal.

RAG_STYLE_PROMPT for Uni-Speaker (Chat Style) documents

Task
Answer the Question based on the context provided.

Input
- The INPUT contains several conversations between two users as Context and a Question.
- Each conversation is mentioned by a number in the following format:
Conversation {{number}}:
- Each conversation contains 10 utterances that are separated by lines.
- Each utterance contains the date, speaker and the message in the following format:
<date>, <speaker>: <message>

Output
return the final answer in a new line after "Answer:" without any prefix or suffix.

INPUT:
Context: {context}

Answer the following question as precisely as possible, using the information provided in the conversation. You may rely on the conversation content, the time each conversation was sent, and who sent it.

Question: {question}

Answer:

Figure 17: Prompt for RAG-style experiment, while the input is forced to contain the positive document, in Uni-Speaker (Chat Style). This prompt is used for all the reasoning categories of the Arithmetic, World-Knowledge, and Temporal.