Enhancing Retrieval Systems with Inference-Time Logical Reasoning

Felix Faltings[1]*, **Wei Wei**[2], **Yujia Bao**[2]
[1]MIT, [2]Center for Advanced AI, Accenture
faltings@mit.edu, {wei.h.wei, yujia.bao}@accenture.com

Abstract

Traditional retrieval methods rely on transforming user queries into vector representations and retrieving documents based on cosine similarity within an embedding space. While efficient and scalable, this approach often fails to handle complex queries involving logical constructs such as negations, conjunctions, and disjunctions. In this paper, we propose a novel inference-time logical reasoning framework that explicitly incorporates logical reasoning into the retrieval process. Our method extracts logical reasoning structures from natural language queries and then composes the individual cosine similarity scores to formulate the final document scores. This approach enables the retrieval process to handle complex logical reasoning without compromising computational efficiency. Our results on both synthetic and real-world benchmarks demonstrate that the proposed method consistently outperforms traditional retrieval methods across different models and datasets, significantly improving retrieval performance for complex queries.

1 Introduction

Retrieval systems are integral to many applications, including search engines, question-answering systems, and recommendation platforms (Baeza-Yates et al., 1999; Lewis et al., 2020; Gao et al., 2023). Modern systems operate by transforming user queries into vector representations and retrieving documents based on cosine similarity within an embedding space (Reimers, 2019; Wang et al., 2023; Zhao et al., 2024; Lee et al., 2024). This approach is highly efficient and scalable, as cosine similarity computations are fast and can handle large-scale data. However, the reliance on cosine similarity and static embeddings often limits the system's ability to understand and process complex queries that involve logical constructs such as negations.

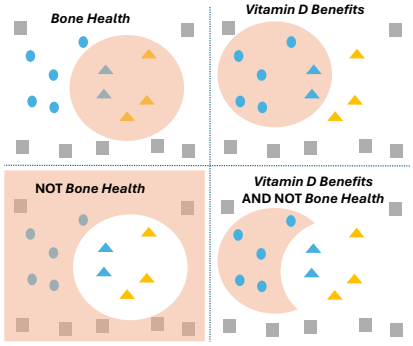


Figure 1: Given a query "What are the benefits of vitamin D, focusing on benefits other than bone health?", we first convert the query into the logical expression "Vitamin D Benefits AND NOT Bone Health". We then calculate the cosine similarity scores for each term (top row) and combine these scores to generate the final results.

Large Language Models (LLMs) have demonstrated remarkable capabilities in inference-time reasoning (Wei et al., 2022; Yao et al., 2024). Recently, (Jiang et al., 2023; Gu et al., 2022; Sun et al., 2023; Luo et al., 2023) have successfully applied LLM's reasoning capability to improve the retrieval performance for knowledge-based question answering, however, the application of inference-time reasoning for general retrieval systems remains relatively unexplored.

We posit that integrating logical reasoning at inference time is equally crucial for retrieval tasks, especially when dealing with complex queries that cannot be accurately represented using simple similarity measures (Meghini et al., 1993; Ounis and Paşca, 1998). Consider a query like "What are the benefits of vitamin D, focusing on benefits other than bone health?" A traditional retrieval system, relying solely on cosine similarity, may struggle to exclude documents related to bone health due to the embedding's inability to represent the negation effectively. The system tends to retrieve documents that are globally similar to the query, failing to account for specific logical instructions, such as

*Work done during an internship at Accenture.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 449–463

exclusions or combinations of concepts.

To address this limitation, we propose a novel inference-time reasoning framework for retrieval systems that explicitly incorporates logical reasoning into the retrieval process. Our approach involves three key steps. First, **Logical Query Transformation**: We utilize an LLM to parse and rewrite the natural language query into a logical form, such as "A AND B OR NOT C," where A, B, and C represent different semantic components of the query. Second, **Term Embedding and Similarity Computation**: Each term identified in the logical form (A, B, C, etc.) is individually encoded into the embedding space. We compute the cosine similarity between each term's embedding and the embeddings of documents in the corpus. Third, **Score Composition Based on Logical Relations**: We combine the similarity scores of each term in accordance with their logical relations. Our approach adds minimal overhead since the embedding can be performed in parallel.

We validate our approach through comprehensive experiments. We first create synthetic datasets with queries of varying logical complexity to test the limitations of existing retrieval algorithms. Our findings indicate that as the number of logical terms increases, the performance of traditional retrieval methods degrades significantly, while our method better maintains performance, demonstrating robustness against query complexity. We also evaluated our algorithm on three real-world datasets: NFCorpus (Boteva et al., 2016), SciFact (Wadden et al., 2020) and ArguAna (Thakur et al., 2021). Specifically, we augmented these three datasets with natural language queries that target compositional reasoning. We tested our method using four commonly used embedding models. The results show that our approach consistently outperforms baseline methods across all models and datasets, confirming its effectiveness in practical scenarios.

2 Method

Given a natural language query, *"What are the beneifts of vitamin D, focusing on benefits other than bone health?"*, we first transform it into a logical expression using a pre-trained large language model (Dubey et al., 2024), *"Vitamin D Benefits" AND NOT "Bone Health"*, where the terms in quotes can be any string of text. Given a document, these queries can be seen as logical expressions, which we evaluate in a fuzzy way (Novák et al., 2012), using the scores from a dense retrieval model to assign truth values to each clause. The fuzzy evaluation of the expression then gives a composite retrieval score for the given document. In the following sections, we present the concrete details of our method, starting with the syntax of the logical queries, followed by the retrieval semantics.

2.1 Query Syntax

Queries are made up of terms–which can be any string of text–combined using operators. We allow three operators, AND, OR, and NOT. Formally, the syntax of the language is described by the following simple grammar,

$$
\begin{aligned}
T &\rightarrow U \text{ OR } U \mid U \\
U &\rightarrow V \text{ AND } V \mid V \\
V &\rightarrow \text{ NOT } W \mid W \\
W &\rightarrow \textit{string} \mid (T)
\end{aligned}
$$

where the use of distinguished non-terminals, T, U, V, and W enforces an operator priority, NOT $\succ$ AND $\succ$ OR , which is itself overridden by parentheses.

2.2 Query Semantics

For each term t_j in a query, and each document D_i in a corpus, we compute a score s_{ji} using the dense retrieval model. Usually, this is the cosine similarity between the embedding vectors of the term and document. The semantics of the query then tell us how to combine the scores s_{ji} into a single score s_i which we can use for retrieval.

Consider a query of the form,

$$(t_1 \text{ OR } t_2) \text{ AND } \text{ NOT } t_3.$$

Then, for document D_i, if s_{1i}, s_{2i}, and s_{3i} are the scores obtained from the dense retrieval model, we take the composed retrieval score to be,

$$s_i = \text{OP}_{AND}(\text{OP}_{OR}(s_{1i}, s_{2i}), \text{OP}_{NOT}(s_{3i})),$$

where OP_{AND}, OP_{OR} and OP_{NOT} are functions that define how scores should be combined depending on the query operator. We detail our choice of operators in the next section.

In general, each query can be parsed using the grammar described above, resulting in a parse tree, which is directly translated into a tree of operations acting on the scores s_{1i}, s_{2i}, and s_{3i} as shown in Fig. 2. A more complicated example is given in the appendix. This tells us how to compute the final retrieval score s_i. More formally, this could be written as an attribute grammar (Knuth, 2005).

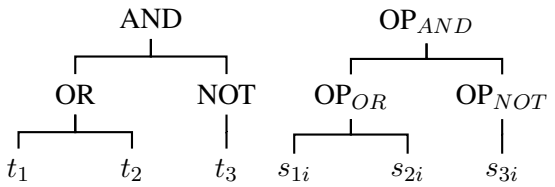


Figure 2: Example parse tree (left) and corresponding graph of operations (right).

2.3 Score Operations

The choice of the operators OP_{AND}, OP_{OR}, OP_{NOT} should reflect the logical semantics of the query. For example, for the conjunction t_1 AND t_2, with scores s_{1i} and s_{2i}, the composite score should be low if *any* of the two scores is low. In this work we consider three approaches,

1. First, we take the perspective of fuzzy logic, where we view the retrieval scores as fuzzy set memberships. The fuzzy membership to a union of sets, corresponding to the logical OR operation, is then taken to be the maximum of the individual fuzzy memberships, and the membership to an intersection, corresponding to the AND operation, is given by the minimum (Fox and Sharat, 1986).

2. Second, we also choose operators based on probability theory. The probability of two independent events occurring simultaneously is $P(A \cap B) = P(A)P(B)$, the probability of the union of two events is upper bounded by $P(A \cup B) \leq P(A) + P(B)$ and the probability of the complement of an event is $P(\neg A) = 1 - P(A)$. Motivated by this we use the arithmetic operations $\times$, $+$ and $1 - \cdot$ for AND, OR and NOT respectively.

3. Finally, we also use heuristically motivated operators such as addition for AND, so that a document that scores highly for individual queries will score highly for their conjunction; and $1/\cdot$ for NOT, so that a document that scores highly for a query will score low for its negation.

In summary, the full set of operator choices is,

$$\text{OP}_{AND}(x, y) = x * y \,|\, x + y \,|\, \min(x, y)$$
$$\text{OP}_{OR}(x, y) = x + y \,|\, \max(x, y)$$
$$\text{OP}_{NOT}(x, y) = 1 - x \,|\, 1/x$$

We evaluate all combinations of these operators in our experiments. Our default choice is $\text{OP}_{AND}(x, y) = x * y$, $\text{OP}_{OR}(x, y) = x + y$, and $\text{OP}_{NOT}(x, y) = (1 - x)$, which we find to work best.

3 Results

We start by validating the performance of the logical retrieval system itself on synthetic data. Next, we assess the system's utility for retrieval on real data. In all our experiments, we evaluate using four base embedding models: Nvidia's `NV-Embed-V1` (Lee et al., 2024), Mistral's `nv-embedqa-mistral-7b-v2` and OpenAI's `text-embedding-v3-large` and `text-embedding-v3-small`.

3.1 Synthetic Data

Three Term Queries We first evaluate performance on all possible queries formed of three terms using synthetic data. This gives 32 "templates", such as,

$$t_1 \text{ AND } t_2 \text{ OR } \text{ NOT } t_3$$

The three placeholders t_1, t_2, and t_3 are filled in by terms. For each possible template, we generate 100 queries by filling in the placeholders with random topics from a set generated by Llama3-70b. For each query we then generate documents that match and don't match the query with Llama3-70b. For example, for the query,

"dog" AND "cat" AND "mouse",

we generate one document that matches, which is related to all three terms, and three documents that don't match, which are related to all but one of the terms. See Appendix H for more details.

The results are presented in Table 1. We report the standard nDCG@10 in all our results. We show the results when passing the query directly to the retrieval model (base) vs. composing the retrieval model scores for each term (logical). For reference, the performance when using random scores is around 0.7. We see that logical retrieval outperforms the baseline, with the most gains coming from queries with negations. We did not see large differences between embedding models. See Table 7 in the appendix for a breakdown.

Scaling Number of Queries We look at performance as the number of terms in the queries scales, focusing this time solely on queries consisting of OR or AND operators. For example,

"dog" AND "mouse" AND . . . AND "cat"

	Number of negations			
	0	1	2	3
Base	0.95	0.77	0.65	0.52
ITLR	**0.99**	**0.97**	**0.96**	**1.00**

Table 1: nDCG@10 Results on synthetic data. Dense and logical retrieval systems were evaluated on synthetically generated test cases for all 32 possible logical queries with three terms. We show results broken down by the number of negations in the queries.

OP_{AND}	OP_{OR}	OP_{NOT} $1-x$	OP_{NOT} $1/x$
$\min(x,y)$	$\max(x,y)$	0.86	0.86
	$x+y$	0.92	0.87
$x*y$	$\max(x,y)$	0.90	0.89
	$x+y$	**0.97**	0.90
$x+y$	$\max(x,y)$	0.91	0.81
	$x+y$	**0.97**	0.82

Table 2: nDCG@10 results for logical retrieval on the same data from Table 1 for different choices of operators, using `NV-Embed-V1`. Each entry represents a choice for each of the three operators.

We generate data in the same way as before. Our results are presented in Fig. 3. We see that the gains from logical retrieval increase as the number of terms increases. This is more pronounced for the AND queries than the OR queries, likely since each AND query has a single positive match whereas each OR query has many matches.

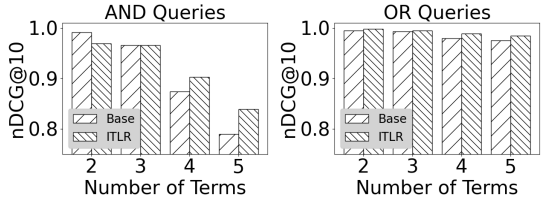


Figure 3: Performance as the number of terms scales. Baseline dense retrieval and logical retrieval were evaluated on queries connected by AND and OR clauses, with increasing number of clauses.

3.2 Operator Combinations

We tested all combinations of operators OP_{AND}, OP_{OR} and OP_{NOT} proposed in section 2.3 on the three term query data using the `NV-Embed-V1` embedding model. We present the results in Table 2. We see that our default choice works best, although the alternative using $\text{OP}_{AND}(x,y) = x+y$ works equally as well. Note that the common choice made in fuzzy logic of $\text{OP}_{AND}(x,y) = \min(x,y)$ and $\text{OP}_{OR}(x,y) = \max(x,y)$ performs quite poorly.

3.3 Real Data

Our previous experiments consider very small corpora constructed specifically for each query. We now turn to real datasets. In order to ensure the queries have sufficient compositionality, we generate queries using Llama3-70b. As in our synthetic experiments, we create 3-term logical queries from templates, filled in with topics extracted from the dataset. We create 30 queries per template, resulting in 960 total queries. We ask Llama3-70b to turn these queries into natural language questions and throw away the original queries. We also use Llama3-70b to label the relevance of each document in the corpus to each of the questions. In the appendix we give the prompts that were used and show through examples that the generated queries are realistic.

We compare three methods. The **Baseline** feeds the question to the dense retrieval model. The **BRIGHT** baseline first asks Llama3-70b to reason about the question and feeds the reasoning trace to the retrieval model. This is the method used in (Su et al., 2024). Finally, our **ITLR** method asks Llama3-70b to generate a logical query from the question, which is fed to our logical retrieval system. See Sec. 3.4 for human evaluation results

	NFcorpus	**SciFact**	**ArguAna**	**Legal**
`NV-Embed-V1:`				
Baseline	0.56	0.51	0.51	0.46
BRIGHT	0.67	0.59	0.58	0.52
ITLR	**0.74**	**0.64**	0.64	0.59
`text-embedding-v3-large:`				
Base	0.63	0.59	0.63	0.56
BRIGHT	0.70	0.63	0.66	0.60
ITLR	0.73	**0.64**	**0.67**	**0.66**
`text-embedding-v3-small:`				
Base	0.56	0.50	0.55	0.50
BRIGHT	0.68	0.59	0.65	0.57
ITLR	0.67	0.54	0.63	0.61
`nv-embedqa-mistral-7b-v2:`				
Base	0.54	0.50	0.40	0.41
BRIGHT	0.48	0.39	0.29	0.27
ITLR	0.67	0.61	0.59	0.42

Table 3: nDCG@10 Results on real data. For each dataset taken from BEIR (Thakur et al., 2021), compositional questions were generated using Llama3-70b. We show results for three embedding models and three methods. Legal short for LegalBenchCorporateLobbying.

	Number of negations			
	0	**1**	**2**	**3**
Base	**0.81**	0.60	0.51	0.36
Reasoning	**0.81**	0.68	0.64	0.56
Logical	0.76	**0.71**	**0.76**	**0.73**

Table 4: Breakdown of Table 3 for `NV-Embed-V1` on NFCorpus by number of negations.

Base Dataset	Corpus Size	Gen. Queries
NFCorpus	3,633	960
ArguAna	8,674	960
SciFact	5,183	960
Legal	340	960

Table 5: **Dataset statistics** Legal short for LegalBench-CorporateLobbying.

	NFCorpus	SciFact	ArguAna
Accuracy	87.29	86.25	85.1

Table 6: Accuracy of question transformation into logical queries.

showing that Llama3-70b is able to successfully formulate logical queries.

We report nDCG@10 results in Table 3, for four datasets derived respectively from the NFCorpus, SciFact, ArguAna and LegalBenchCorporateLobbying tasks, accessed through MTEB (Muennighoff et al., 2022). Table 5 gives the statistics of the datasets used in our experiments.

We see that ITLR achieves the best performance overall, beating all baselines in a majority of cases. This performance gap becomes larger with more negations, as seen in Table 4. The lack in improvement without negations may be due to the use of three term queries. Since simpler queries are likely better represented in the training distribution of the retrieval models, they are easy enough to process. In our synthetic experiments (Figure 3) we show that, as the number of terms increases, ITLR outperforms baselines including on cases without negations.

3.4 Accuracy of Query Transformation

Because our system is reliant on transforming user queries from natural language into our logical syntax, we investigated the accuracy of Llama3-70b on this query transformation task. For all the questions in three of our real datasets–NFCropus, SciFact and ArguaAna–we asked human annotators to assess whether or not the logical query generated by the LLM accurately reflected the original question. As can be seen in Table 6, the generated logical queries tend to capture the original question quite well, with accuracies of 85% to 87%. In appendix B we also examined the queries labelled as inaccurate and found that in the cases we considered the errors tended to be minor.

4 Conclusion

In this paper, we propose an inference-time logical reasoning framework that addresses the limitations of traditional retrieval methods in managing complex queries with logical constructs. The framework is highly efficient, enabling concurrent computation of retrieval scores for each term. By integrating logical reasoning directly into the retrieval process, our framework consistently outperforms traditional methods on both synthetic and real-world benchmarks, demonstrating particular strength in handling queries with a higher frequency of negations and AND operations.

5 Limitations

The logical retrieval system we presented in this paper presents some limitations, as it still underperforms on queries without any negations. We identify some concrete problems that could be addressed in future work. First, in most scenarios, the queries to retrieval systems, such as questions from users, are not given as logical formulas. It is also unreasonable to expect users to write logical formulas on their own. Hence, the system is reliant on reformulating queries into logical queries. While we used a simple prompt to achieve this, it is possible that better performance could be obtained by finetuning a reformulation model. Second, we observed in preliminary experiments that the performance of the system can be improved by calibrating the scores of the underlying retrieval model. For example, when processing AND queries, some terms may receive generally higher retrieval scores than others, biasing retrieval towards documents that match those terms but not the others. We did not find any simple methods to calibrate the scores, but this could be accomplished when training the retrieval model, or by training a calibration model on a large dataset.

Ethical Considerations We do not foresee any immediate risks of our work as we are not releasing any major new artifacts, such as pre-trained models, which could be used in adverse ways. Retrieval systems are limiting points in applications such as retrieval augmented generation (RAG) since downstream answers are generated based on the documents provided by the retrieval system. Poor retrieval systems may skew the information retrieved from corpora, and thus improving the faithfulness of retrieval systems may be broadly beneficial.

References

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edward A Fox and S Sharat. 1986. A comparison of two methods for soft boolean interpretation in information retrieval. Technical report, TR-86-1. Virginia Tech. Department of Computer Science.

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

Yu Gu, Xiang Deng, and Yu Su. 2022. Don't generate, discriminate: A proposal for grounding language models to real-world environments. *arXiv preprint arXiv:2212.09736*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Donald E Knuth. 2005. The genesis of attribute grammars. In *Attribute Grammars and their Applications: International Conference WAGA Paris, France, September 19–21, 1990 Proceedings*, pages 1–12. Springer.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, et al. 2023. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint arXiv:2310.08975*.

Carlo Meghini, Fabrizio Sebastiani, Umberto Straccia, and Costantino Thanos. 1993. A model of information retrieval based on a terminological logic. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–307.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Vilém Novák, Irina Perfilieva, and Jiri Mockor. 2012. *Mathematical principles of fuzzy logic*, volume 517. Springer Science & Business Media.

Iadh Ounis and Marius Pașca. 1998. Relief: Combining expressiveness and rapidity into a single system. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pages 266–274.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, et al. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *arXiv preprint arXiv:2407.12883*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

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)*.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

Embedding Model	Base	ITLR
NV-Embed-V1	0.74	**0.97**
text-embedding-v3-large	0.71	**0.97**
text-embedding-v3-small	0.71	**0.97**

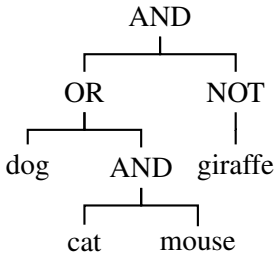
Table 7: nDCG@10 results on synthetic data, broken down by embedding model.

A Additional Parsing Example

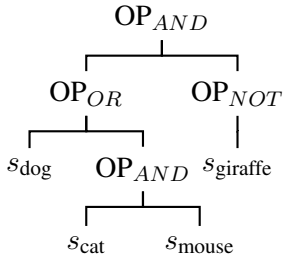
Consider the query,

(“dog" OR “cat" AND “mouse") AND NOT “giraffe"

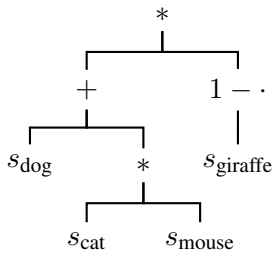
The corresponding parse tree will be,



And the computational graph for combining the scores is,



With our actual choice of operators this looks like,



Written out as a formula this gives the final retrieval score as

$$s = (s_{\text{dog}} + s_{\text{cat}} * s_{\text{mouse}}) * (1 - s_{\text{giraffe}}).$$

B Additional Results

Breakdown by Embedding Model Table 7 breaks down the synthetic data results by embedding model. We see that there is little difference between the embedding models we considered.

Accuracy of Query Transformation We investigated how accurate Llama3-70b was at transforming natural language questions into logical queries. Below we give an example of inaccurate queries from the NFCorpus, SciFact and ArguAna datasets respectively.

- **Example 1** Natural language question:

What are some studies on the health benefits of anthocyanins that are not specifically focused on cancer prevention or targeted towards men?

LLM generated query:

"Health benefits of anthocyanins" AND (NOT "cancer prevention" OR NOT "targeted towards men")

Here, the original question is ambiguous about which studies should be excluded.

- **Example 2** Natural language question:

  What are some recent research topics in the fields of neurology and oncology?

  LLM generated query:

  "Recent research in neurology" OR "Current research topics in neurology" AND "Recent advancements in oncology" OR "Current studies in oncology"

  Here the AND should have been an OR .

- **Example 3** Natural language question:

  What are some news articles that are not about agricultural development in Africa or Brexit, or are about a topic unrelated to peacekeeping?

  LLM generated query:

  "News articles" AND NOT ("Agricultural development in Africa" OR "Brexit" OR "Peacekeeping")

  Here, "Peacekeeping" should be outside the NOT clause.

C Data Examples

Below we provide examples of queries from our NFCorpus dataset.

- **Example 1**

  What are the risk factors for pancreatic cancer, excluding those related to MRSA in swine farms, or what are the benefits of cruciferous compounds in cancer prevention?

- **Example 2**

  What are the health benefits and risks of a vegetarian diet that does not include dairy products, and are there any natural alternatives to dairy that can provide similar nutritional value?

- **Example 3**

  What are some ways to prevent cancer through diet, excluding the effects of xenohormesis mechanisms, and specifically considering the potential benefits of cherries or other foods rich in phenolic compounds?

- **Example 4**

  What are the dietary factors that can help prevent cancer, excluding those related to polycyclic aromatic hydrocarbons?

- **Example 5**

  What are the effects of bioactive compounds on colon or prostate cancer, excluding studies on their mechanisms of action?

Below, we give the corresponding logical queries generated by the LLM,

- **Example 1**

  ("Risk factors for pancreatic cancer" AND NOT "MRSA in swine farms") OR "Benefits of cruciferous compounds in cancer prevention"

- **Example 2**

  "Health benefits of a vegetarian diet without dairy products" AND "Risks of a vegetarian diet without dairy products" AND ("Natural alternatives to dairy products" OR "Plant-based alternatives to dairy products") AND ("Nutritional value of dairy products" OR "Nutritional benefits of dairy alternatives")

- **Example 3**

  "Dietary prevention of cancer" AND NOT "xenohormesis" AND ("cherries" OR "foods rich in phenolic compounds")

- **Example 4**

  "Dietary factors that help prevent cancer" AND NOT "Polycyclic aromatic hydrocarbons"

- **Example 5**

  "Effects of bioactive compounds on colon cancer" OR "Effects of bioactive compounds on prostate cancer" AND NOT "Mechanisms of action of bioactive compounds"

D Reformulation Prompts

Here we give the prompts used for query reformulation in our reasoning methods from Sec. 3.3.

BRIGHT Reasoning prompt

```
Here is a user query:

\{question\}

(1) Identify the essential question in the query.
(2) Think step by step to reason about what should be included in the relevant
↪  documents.
(3) Draft an answer.
```

Logical Formula Prompt

```
I have a document retrieval system that processes logical queries.
These queries can be of the form,
"term1" AND "term2" OR "term3" AND NOT "term4"

The meaning of the operators AND, OR and NOT should be obvious:
- AND means the retrieved document should be related to both terms
- OR means the retrieved document can be related to either term
- NOT means the retrieved document should not be related to the given term

Given a natural language question from a user, I want to use the retrieval system to
↪  gather documents that contain information relevant to the user's question.
I need you to create suitable logical query to the retrieval system.
Remember that each of the individual terms can be a keyword, a phrase, a sentence,
↪  or even
```

	Number of negations			
	0	1	2	3
Base	0.95	0.77	0.65	0.52
ITLR	0.99	0.97	0.96	**1.00**
Calibrated ITLR	**1.00**	**0.98**	**0.97**	**1.00**

Table 8: Expanded version of Table 1 including calibration.

```
a whole document. So don't limit yourselves to keywords.
For example, the following question,
"What is the impact of eating fresh oranges on pancreatic cancer risk,
and its relationship to stage II diabetes"
Could be answered with the query,
"Impact of fresh orange consumption on pancreatic cancer risk" AND
"What is the relationship of eating fresh oranges to stage II diabetes?"
A query that only used keywords, such as
"oranges" AND "consumption" AND "pancreatic cancer" AND "stage II diabetes"
would lose much of the meaning of the original question! It's not clear if
↪  consumption relates
to oranges, so a document that talks about consuming figs, and peeling oranges
would match this query!

Here is the user's question,
\{question\}

Can you come up with a suitable logical query to the retrieval system? Only include
↪  the query
in your answer.
```

E Calibration

We also experimented with calibrating the scores from the retrieval models before combining them in ITLR. On the synthetic data, for a given embedding model and for each term, we generated 20 positive and 20 negative documents using Llama3-70b. This gave us a dataset of documents $x_1, \ldots, x_N$, with embedded cosine similarities $s_1, \ldots, s_N$ and labels $y_1, \ldots, y_N \in \{0, 1\}$. We then fit a simple calibration model,

$$\hat{y}_i = \sigma((s_i - \tau) * \lambda),$$

using gradient descent. This calibration offered some improvements on our synthetic dataset, as can be seen in Table 8, which is an expanded version of Table 1 with calibration.

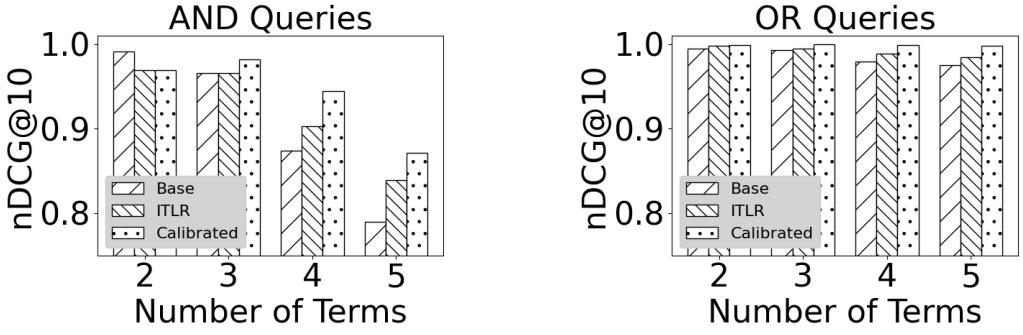


Figure 4: Expanded version of Fig. 3 including calibration.

Model	Number of Model Parameters	Access	License
`Llama3-70b`	$70 * 10^9$	API	cc-by-nc-4.0
`NV-Embed-V1`	$7.85 * 10^9$	API	llama3
`text-embedding-3-large`	Undisclosed	API	commercial
`text-embedding-3-small`	Undisclosed	API	commercial

Table 9: Dataset statistics

F Additional Data Details

The datasets were accessed through MTEB (Muennighoff et al., 2022) under an Apache 2.0 license. These datasets all contain English text.

G Model Details

Table 9 gives details on the embedding models and LLMs used in our experiments, including parameter counts and how they were accessed.

H Synthetic Data Generation

Consider a query,

"mouse" OR "dog" AND NOT "cat".

Any document that matches this query can be categorized by the individual terms of the query that it does or does not match. For example, for this query, there are two categories,

1. Matches mouse but not cat
2. Matches dog but not cat

Any of these categories is a list of terms that the document matches, and a list of terms that it doesn't match. This gives an easy way to generate matching documents using Llama3-70b by providing terms that should or shouldn't be matched.

There are many documents that don't match the query, but we want to evaluate against challenging negatives. We can create these by taking one of the categories above and swapping a condition. For example, a positive document will match "mouse" but not "cat". A hard negative could match "mouse" *and* "cat". In general, given a list of terms that the document should or shouldn't match, we simply move one term to the opposite category. So a hard negative will match all the terms it should to match the query, except one. Or it will avoid all the terms it should, except one.

Thus, to generate documents for our synthetic data, we first enumerate all the combinations that positive documents should or shouldn't match. We then create all the hard negatives by swapping one of the terms. For each set of terms to match or not match, we create one document. For three term queries, this will result in at most three positive documents. We also keep up two three negative documents. Hence for each query, we generate up to six documents.

I Synthetic Queries

We found that the queries in original datasets used in our experiments are overly simple. For example, in the NFCorpus dataset the queries are the titles of web articles such as "Philippines". To better evaluate the retrieval performance under complex user questions, we created a set of queries based on topics found in the corpus and then used an LLM to evaluate how relevant each passage was to each query. The passages are completely unchanged. This gives us natural and challenging queries with a better labelling of relevant passages, while retaining the complexity of a real world document corpus.

To generate the queries, we extract topics from random documents in the corpus. We then create logical queries using the extracted topics. Finally, we transform these into natural language questions. The Llama3-70b prompts we used are given below.

Topic Extraction Prompt

```
You will be given a document. You need to extract all the salient topics from it.
The topics should range from general to specific. Here is the document:
{format_doc(doc)}

Please give the salient topics as a list with one topic per line.
Don't include anything else in your answer.
Sort the topics from most general to most specific.
```

Query to Question Prompt

```
I have a document retrieval system that processes logical queries.
These queries can be of the form,
"term1" AND "term2" OR "term3" AND NOT "term4"

The meaning of the operators AND, OR and NOT should be obvious:
- AND means the retrieved document should be related to both terms
- OR means the retrieved document can be related to either term
- NOT means the retrieved document should not be related to the given term

I want to evaluate the performance of a human user to use this retrieval system.
Given a natural language question, the user needs to come up with a logical query
↪  that will best
retrieve relevant documents.
In order to make a dataset for evaluation, I want to operate in reverse. I have
↪  collected
many logical queries, and I would like to come up with a corresponding natural
↪  language question.
Then I can give the question to a user and see how well the recover they original
↪  query.

So, given the following logical query, can you come up with such a natural language
↪  question?
Here's the query,
{query}

What question would you come up with? Only include the question in your answer.
```

J Human Annotations

Human annotators tasked with evaluating the LLM generated queries were paid a fair wage of $25 an hour. They were given the following instructions

```
Instruction for Human Annotator: Logical Expression Validation
Task Overview
You will be given a natural language question and a corresponding logical
↪  expression generated by an LLM (Large Language Model). Your task is to
↪  determine whether the logical expression accurately represents the intended
↪  meaning of the question.
A correct logical expression should:
- Capture the key intent of the question.
- Properly reflect any exclusions, inclusions, or constraints mentioned.
- Maintain the logical relationships between elements.
```

```
Evaluation Criteria
1. Accuracy - Does the logical expression correctly interpret the intent of the
↪  question?
2. Completeness - Are all relevant aspects of the question included in the logical
↪  expression?
3. Exclusions - If the question explicitly excludes something, does the logical
↪  expression handle this correctly?
4. Logical Structure - Are the AND, OR, and NOT operators used correctly to reflect
↪  the relationships in the question?
If the logical expression is correct, mark it as valid. If incorrect, mark it as
↪  invalid and provide an explanation of the error.
Examples
Positive Examples (Correct Expressions)
Example 1:
- Natural Language Question: ``How does vitamin D benefit your health? I already
↪  know about bone health, so I want to know other benefits.''
- Parsed Logical Expression: health benefits of vitamin D AND NOT bone health
- Explanation: The logical expression correctly retrieves information about vitamin
↪  D's health benefits while excluding bone health, as specified in the question.
Example 2:
- Natural Language Question: ``What are some movies directed by Christopher Nolan,
↪  excluding superhero films?"
- Parsed Logical Expression: movies directed by Christopher Nolan AND NOT superhero
↪  films
- Explanation: The logical expression correctly filters out superhero films while
↪  still retrieving Nolan's other movies.
Example 3:
- Natural Language Question: ``Which laptops have at least 16GB RAM and either an
↪  Intel i7 or AMD Ryzen 7 processor?"
- Parsed Logical Expression: laptops AND 16GB RAM AND (Intel i7 OR AMD Ryzen 7)
- Explanation: The expression correctly captures the requirement of 16GB RAM and
↪  allows either processor type, as intended.
Negative Examples (Incorrect Expressions)
Example 4:
- Natural Language Question: ``How does vitamin D benefit your health? I already
↪  know about bone health, so I want to know other benefits."
- Parsed Logical Expression: health benefits of vitamin D OR NOT bone health
- Error: The use of OR NOT instead of AND NOT changes the meaning. The expression
↪  may return results that are completely unrelated to vitamin D.
Example 5:
- Natural Language Question: ``What are some movies directed by Christopher Nolan,
↪  excluding superhero films?"
- Parsed Logical Expression: movies directed by Christopher Nolan OR NOT superhero
↪  films
- Error: The OR NOT operator incorrectly allows movies that aren't superhero films
↪  but might not be directed by Nolan, which is not what the question asks.
Example 6:
- Natural Language Question: ``Which laptops have at least 16GB RAM and either an
↪  Intel i7 or AMD Ryzen 7 processor?"
- Parsed Logical Expression: laptops AND (16GB RAM OR Intel i7 OR AMD Ryzen 7)
```

- Error: The use of OR within the parentheses makes it possible for laptops with only Intel i7 or AMD Ryzen 7 (but less than 16GB RAM) to be included, which is incorrect.

Final Notes

- Pay close attention to negations (NOT). Misplacing them can completely alter the meaning.
- Ensure correct grouping with parentheses. Ambiguities in logic can lead to unintended results.
- Rephrase the natural language question in a structured way before checking the logical expression.

Your accuracy in annotation ensures that the model correctly understands and processes logical constraints in natural language.