OldJoe at AVeriTeC: In-context learning for fact-checking

Farah Ftouhi*[†], Russel Dsouza*, Lance Gamboa, Jinlong Liu, Asim Abbas,
Yue Feng, Mubashir Ali, Mark Lee, Venelin Kovatchev
School of Computer Science, University of Birmingham

Abstract

In this paper, we present the system proposed by our team *OldJoe*, for the 8th edition of the AVeriTeC shared task, as part of the FEVER workshop. The objective of this task is to verify the factuality of real-world claims. Our approach integrates open source large language models, SQL, and in-context learning. We begin with embedding the knowledge store using a pretrained embedding language model then storing the outputs in a SQL database. Subsequently, we prompt an LLM to craft relevant questions based on the input claim, which are then used to guide the retrieval process. We further prompt the LLM to generate answers to the questions and predict the veracity of the original claim. Our system scored 0.49 on the HU-METEOR AVeriTeC score on the dev set and 0.15 on the Ev2R recall on the test set. Due to the time constraint we were unable to conduct additional experiments or further hyperparameter tuning. As a result, we adopted this pipeline configuration centered on the `Qwen3-14B-AWQ` model as our final submission strategy. The full pipeline is available on GitHub.[1]

1 Introduction

In an era where information spreads rapidly across digital platforms, manual fact-checking struggles to keep pace with the vast volume of content generated daily. This growing challenge has sparked increasing interest in the development of automated fact-checking systems with a focus on efficient, reproducible and open-source methodologies. In this context, the AVeriTeC Shared Task[2] was introduced to evaluate systems capable of assessing the factuality of claims using a structured knowledge base. We created a system that combines large language models, SQL databases, and in-context learning. Our pipeline has the following components: (1) a postgreSQL database, which stores the embeddings and chunks for evidence documents from the knowledge store; (2) Question Generation, where we generate questions and queries based on the given claim; (3) Retrieval and Re-ranking, which uses `pgvector` and `postgres` to retrieve evidence for each query; (4) Answer Generation, where we generate answers for each question using the retrieved evidence chunks; and (5) Veracity Check, where we assign the final veracity label based on the generated question-answer pairs. The entire pipeline is illustrated in Figure 1.

*Main contributors
[†]Corresponding author: fxf482@student.bham.ac.uk
[1]https://github.com/farahft/OldJoe
[2]https://fever.ai/task.html

2 Related Work

One of the earliest studies to frame fact-checking as a computational task was introduced by (Vlachos and Riedel, 2014), who aimed to replicate the traditionally manual process of claim verification using NLP techniques. This foundational work paved the way for other efforts, most notably the introduction of the FEVER dataset(Thorne et al., 2018), a large-scale benchmark designed to advance research in claim verification against textual sources. Over the years, the increasing spread of misinformation(Das et al., 2023) has further elevated fact-checking as a critical area of research and led to multiple studies.

More recently, (DeHaven and Scott, 2023) proposed BEVERS, a simple yet highly effective pipeline that achieves state-of-the-art results on both the FEVER and SciFact (Wadden et al., 2020) datasets. This growing interest has also motivated the organization to launch the FEVER(Schlichtkrull et al., 2024) workshop, which evaluate systems using the real-world claim dataset AVeriTeC(Schlichtkrull et al., 2023). Participants in these workshops have employed a wide range of approaches — from systems relying on APIs(Rothermel et al., 2024) to those based on fine-tuned open-source models (Sevgili et al., 2024), reflecting the diversity and rapid evolution of methods in this domain.

3 Methodology

In this section we present our system pipeline as shown in Figure 1. We start by preparing our evidence database from the knowledge store given by the organisers. Next, we build our question-query generator to generate questions and queries that guide retrieval for a given claim. Then, we build the answer generator to answer the questions and queries based on the retrieved evidence. Finally, these answers are used to determine the veracity of a given claim.

3.1 Evidence Embeddings

Before creating the embeddings, we first semantically split each evidence document into chunks with a maximum length of 2048 characters using `semantic-text-splitter` [3] that offers methods for splitting text into smaller chunks, aiming to reach a target chunk size while prioritizing splits at semantically meaningful boundaries. We then explored several approaches to generating the embeddings and storing them in a database. One approach we attempted involved using `Alibaba-NLP/gte-large-en-v1.5` (Zhang et al., 2024) and FAISS to store the texts, their embeddings, and the corresponding metadata as pickled files (Douze et al., 2024). We also looked into using a ChromaDB vector database to store both the embeddings and metadata in a singular vector. Ultimately, our final system generates sentence embeddings with `jina-ai/jina-embeddings-v3`[4] with a maximum model length of 2048 and relies on the `postgres` extension, `pgvector`[5], for storing both the embeddings and the content of the evidence chunks together. Evidence for each claim is stored in a separate table, enabling efficient and accurate retrieval of relevant evidence.

3.2 Question and Query generation

The next step in our inference pipeline is to generate questions along with search queries. For this task, we use `Qwen/Qwen1.5-14B-Chat-AWQ` [6] (QwenLM Team, 2025), a recent reasoning model that is quantized to fit within 24GB of VRAM set up to 8192 as a max length. For each claim, we prompt this model to first analyse and reason about the claim before generating four questions that, when answered, would provide easy insight into the veracity of the claim.

To support veracity prediction and improve both interpretability and retrieval quality, we first prompt the model to generate questions from the original claims. Using the generated questions we prompt the model to generate refined search queries designed to better capture the information need and guide evidence retrieval. This two-step process enables the system to retrieve more relevant evidence chunks, which are then used to form Q-A pairs that inform the final veracity prediction

The prompts for question and query generation are provided in Appendix A and B respectively.

Question Generation Example

Claim: Trump Administration claimed songwriter Billie Eilish Is Destroying Our Country In Leaked Documents
Question: Are there any official documents from the Trump Administration that explicitly state Billie Eilish is destroying the country?

3.3 Evidence Retrieval and Re-ranking

We use the questions and queries generated in Section 3.2 to retrieve evidence. For each question, corresponding search queries along with the question itself are used to retrieve and simultaneously rerank candidate evidence chunks. Parallel to the various embedding and data warehousing approaches we explored, we also compared the effectiveness of four approaches in retrieving evidence from our knowledge base:

1. the FAISS retrieval method, which uses cosine similarity to quantify the distance between query embeddings and evidence embeddings

2. the BM25 (Robertson et al., 2009) algorithm, which retrieves the most relevant evidence using keyword search without relying on embeddings

3. a hybrid score combining BM25 scores with FAISS-based cosine similarity scores between query and evidence embeddings

4. reciprocal rank fusion (RRF) scoring (Cormack et al., 2009), which collates the BM25-

[3] https://pypi.org/project/semantic-text-splitter/
[4] https://huggingface.co/jinaai/jina-embeddings-v3
[5] https://github.com/pgvector/pgvector
[6] https://huggingface.co/Qwen/Qwen1.5-14B-Chat-AWQ

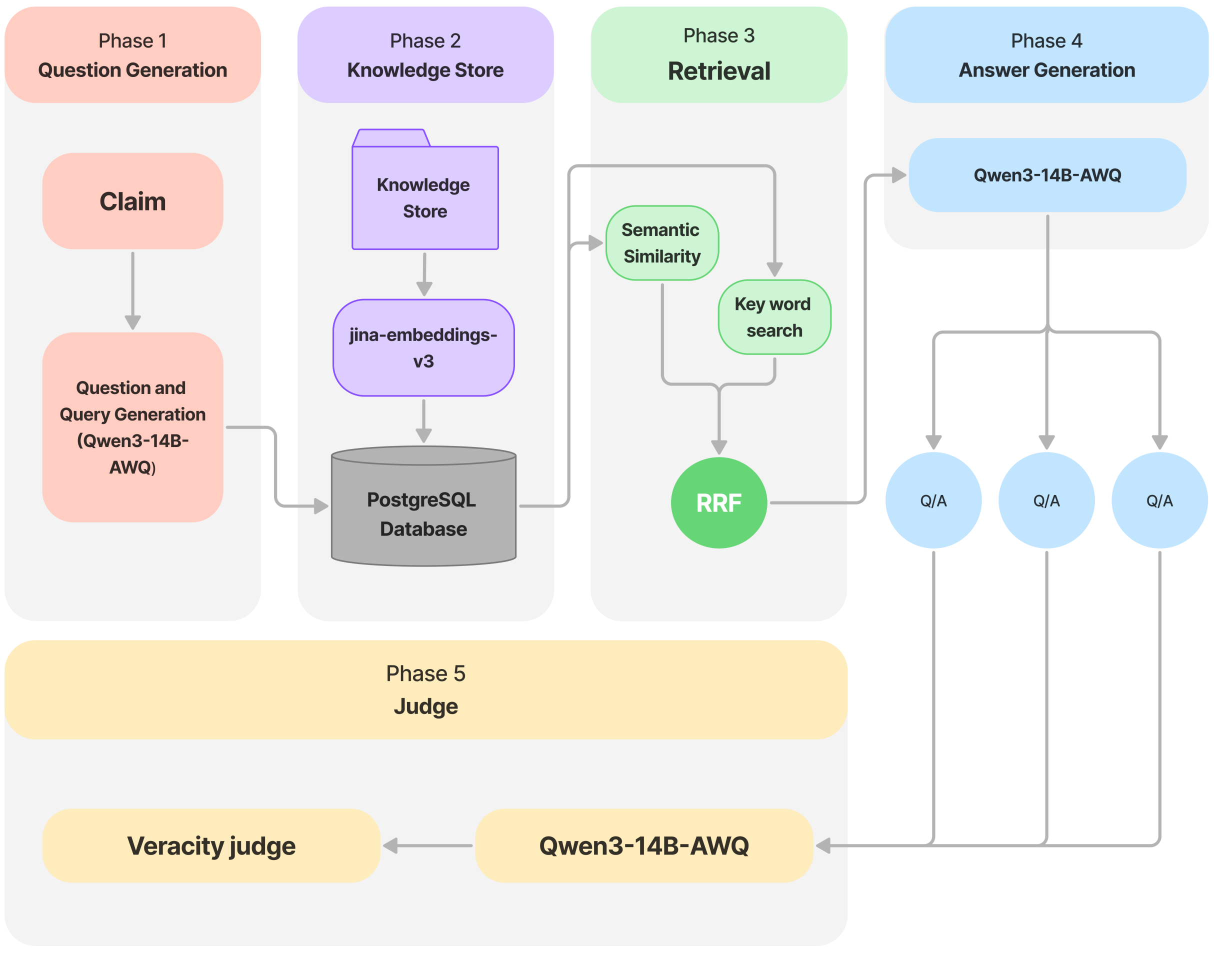

Figure 1: System Pipeline.

based and pgvector-based rankings of evidences and embeddings stored in our postgreSQL database

We used the Prometheus evaluation metric (Pombal et al., 2025) a multilingual LLM-as-a-judge framework that supports both reference-based and reference-free evaluation enabling direct assessment and pairwise comparison of long-form outputs, to assess the best retrieval approach. The results in Table 1 show that retrieving evidence from postgreSQL through RRF scoring provided the best results.

Consequently, the entire retrieval-reranking process is performed in a single SQL query for efficiency and speed. Combining BM25-based keyword search and pgvector-based semantic search, the SQL query retrieves the eight most question-relevant evidence chunks identified by each search method. The BM25 and pgvector ranks and scores for each retrieved chunk are then collated using RRF with a penalty factor of 4 to ensure balanced contributions between the BM25 and pgvector search strategies. Based on this RRF scoring mechanism, the system returns the 10 highest-ranked evidence chunks for each question.

Approach	Prometheus Evaluation Metric
FAISS	1.20
BM25	2.90
FAISS & BM25	1.80
postgreSQL RRF	**3.80**

Table 1: Comparison of retrieval models using the Prometheus evaluation metric.

The SQL query for retrieval and re-ranking is provided in Appendix E.

3.4 Answer Generation

The penultimate step in our inference pipeline is answer generation. The process is similar to Section 3.2 and uses the same reasoning model: `Qwen/Qwen3-14B-AWQ` (QwenLM Team, 2025). The model is prompted to generate the answers based on the question and corresponding retrieved evidence. The prompt used can be found in Appendix C.

Answer Generation Example

Question: Are there any official documents from the Trump Administration that explicitly state Billie Eilish is destroying the country?
Answer: No. As reported by The Washington Post and corrected by the New York Post, the leaked documents from the Trump administration did not state that Billie Eilish was destroying the country. Instead, they recorded Eilish's own statement criticizing Trump, who she said was destroying our country and everything we care about. The Washington Post initially misattributed this quote to Trump officials but later issued a correction clarifying that the statement was Eilish's own, not an official assessment by the administration.

3.5 Veracity Judgment

The final stage of the pipeline is veracity judgment, where each claim is classified into 4 different labels: *Supported*, *Refuted*, *Conflicting/Cherry-Picking* or *Not Enough Evidence*. Similar to Sections 3.2 and 3.4, we prompt the model to predict the claim's veracity given the claim and the question-answer pairs generated in Section 3.4. The prompt used can be found in Appendix D.

4 Experiments

This section presents results of the experiments we conducted to determine which reasoning model provided the most optimal performance for our fact-checking pipeline.

4.1 Hardware

All the experiments were conducted on a single machine equipped with two AMD EPYC9334 32-core CPUs, 1 TB of RAM, and two 1TB NVMe SSD and 4TB NVMe SSD. The system also included 8 NVIDIA L40s GPUs each with 48GB of memory. The complete pipeline including the database, language models, and all other components were packed into a Docker image totaling 230 GB in size. This containerized setup can be run on systems with at least 24 GB of GPU memory.

4.2 Experimental Results

The processes to generate the embeddings and insert them into our postgreSQL knowledge store database collectively took approximately 4 hours. Claim labeling on the dev set required an average of 45 seconds per claim, totaling around 6.3 hours for the entire dataset.

We employed three models in performing the question generation, question-and-answer generation, and veracity prediction tasks. These models were all compatible with the 24GB GPU RAM hardware setup described in Section 4.1. Table 2 shows each model's scores for the three tasks mentioned above, as evaluated using the official 2024 Shared Task metrics. Among the models, `qwen3-14b-awq` returned the highest scores when paired with in-context learning and applied into our inference pipeline. The model achieved an accuracy of 0.494 and an AVeriTeC score of 0.42 on the dev set. Owing to limited time, we could not explore alternative configurations or pipelines , we proceeded with the mentioned pipeline which showed promising results during Q and Q+A stages.

4.3 Final Submission Results

Our system was evaluated on the Ev2R framework proposed by (Akhtar et al., 2024), which introduces reference-based, proxy-reference and reference-less scores for evidence evaluation in automated fact-checking. Our system achieved a mean runtime of 84.57 seconds per claim, a Q+A Ev2R recall of 0.387 a Q-only Ev2R score of 0.182 and an overall AVeriTeC score of 0.151.

5 Conclusion

This paper describes Team *OldJoe*'s submission to the AVeriTeC Shared Task the FEVER workshop. We explored various strategies for embedding, storing, and retrieving evidence chunks for better veracity prediction. We evaluated multiple language models and investigated the effectiveness of applying them and in-context learning for automated fact-checking. While our system demonstrates a promising performance on the dev set when evaluated by the HU-METEOR metric, further improvements are necessary to enhance its generalisation and achieve better results on the test set.

Limitations

This system has been designed for the FEVER shared task and is structured to meet the requirements and limitations of the task. The performance of the model outside of the parameters of the task

Model	Q score	Q/A score	Accuracy	AVeriTeC Score
`llama-3.1-8B`	0.411	0.27	0.388	0.38
`qwen3-8B-fp8`	0.410	0.28	0.41	0.414
`qwen3-14b-awq`	0.411	0.27	**0.494**	**0.492**

Table 2: Comparison of models performance using HU-meteor Accuracy and AVeriTeC scores.

might differ significantly. Due to time and computational constraints, we were not able to fully finetune our system on the data. It is likely that the system performance can improve substantially with additional finetuning and access to more powerful hardware.

Acknowledgments

The computations described in this research were performed using the Baskerville Tier 2 HPC service (https://www.baskerville.ac.uk/). Baskerville was funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

References

Mubashara Akhtar, Michael Schlichtkrull, and Andreas Vlachos. 2024. Ev2r: Evaluating evidence retrieval in automated fact-checking. *arXiv preprint arXiv:2411.05375*.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Anubrata Das, Houjiang Liu, Venelin Kovatchev, and Matthew Lease. 2023. The state of human-centered nlp technology for fact-checking. *Information processing & management*, 60(2):103219.

Mitchell DeHaven and Stephen Scott. 2023. Bevers: A general, simple, and performant framework for automatic fact verification. *arXiv preprint arXiv:2303.16974*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

José Pombal, Dongkeun Yoon, Patrick Fernandes, Ian Wu, Seungone Kim, Ricardo Rei, Graham Neubig, and André FT Martins. 2025. M-prometheus: A suite of open multilingual llm judges. *arXiv preprint arXiv:2504.04953*.

QwenLM Team. 2025. Qwen3 technical report. `https://github.com/QwenLM/Qwen3/blob/main/Qwen3_Technical_Report.pdf`. Accessed: 2025-05-14.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Mark Rothermel, Tobias Braun, Marcus Rohrbach, and Anna Rohrbach. 2024. InFact: A strong baseline for automated fact-checking. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 108–112, Miami, Florida, USA. Association for Computational Linguistics.

Michael Schlichtkrull, Yulong Chen, Chenxi Whitehouse, Zhenyun Deng, Mubashara Akhtar, Rami Aly, Zhijiang Guo, Christos Christodoulopoulos, Oana Cocarascu, Arpit Mittal, James Thorne, and Andreas Vlachos. 2024. The automated verification of textual claims (AVeriTeC) shared task. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*.

Michael Sejr Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. 2023. Averitec: A dataset for real-world claim verification with evidence from the web. In *Thirty-thh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Özge Sevgili, Irina Nikishina, Seid Muhie Yimam, Martin Semmann, and Chris Biemann. 2024. UHH at AVeriTeC: RAG for fact-checking with real-world claims. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 55–63, Miami, Florida, USA. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 workshop on language technologies and computational social science*, pages 18–22.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and

Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. *arXiv preprint arXiv:2004.14974*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*.

Appendix

A Question Generation Prompt Template

```
You are an advanced fact-checking AI, tasked with generating highly targeted,
investigative questions to verify claims.
Each question should probe a unique and essential aspect of verification.
You are NOT fact-checking the claim. You're job is to generate questions to
enable better fact checking.

### Instructions
Generate {{ n_questions }} **distinct** and **non-redundant** questions.
Each question should target a **different** key dimension of verification
Each question must be **necessary**: answering it should bring us **measurably
closer to a veracity judgment**.
Each question must be atomic and include all the **necessary** and **sufficient
** information while being **concise**.

### Additional Notes
Before finalizing each question, consider:
1. What specific aspect of the claim does this question interrogate?
2. Would answering this question significantly impact veracity assessment?
3. Is this question fundamentally different from the others?
Make sure that the question *is a question*
{{ response_format }}


### Task:
Claim: "{{ claim }}"

Generated Questions:
```

B Query Generation Prompt Template

```
You are a retrieval-optimization AI that transforms fact-checking questions
into **search
queries** for evidence gathering.

Your job is to generate **{{ n_queries }} diverse, high-recall queries** that
can retrieve
**useful evidence** to help answer the following investigative question.

### Goals
- Cover **different phrasings**, **semantic angles**, and **terminological
variations**.
- Balance between **specificity** and **generalization** to maximize evidence
retrieval.
- Optimize for both **keyword** and **semantic search systems**.

### Techniques
Use the following techniques to generate diverse queries:
- Strip to core facts, entities, and concepts
- Use synonyms, rephrasing and related concepts.
- Break down the question into subcomponents.
- Vary terminology (formal/informal, technical/common).
- Include relevant entities or contexts.
- Reformulate to target potential evidence phrases (e.g., "according to", "
experts say",
etc.)
{{ response_format }}

### Task
Question: "{{ question }}"

Generated Search Queries:
```

C Answer Generation Prompt Template

```
You are an advanced fact-checking AI, tasked with answering questions based on
provided evidence.
You have been given a question and a set of evidence chunks retrieved from a
database to answer that question.
Your goal is to synthesize a well-reasoned answer supported by the evidence.
You must ground your answer only in the evidence provided and avoid speculation
 or unsupported claims.
Name your sources and be journalistic in your approach and response.
### Instructions
- Read and analyze all the provided evidence chunks.
- Identify relevant information that directly supports or refutes the question.
- Think step-by-step and reason about the evidence logically and cautiously.
- Acknowledge uncertainty or lack of coverage if the evidence is incomplete or
contradictory.
- Align your final answer with the type of question. If it is a **yes/no
question**, your answer must begin with either Yes or No and remain strictly
within that framing.
- Avoid speculation, assumptions, or invented content.
- Distill a short name for the source of each from the provided URL.
- When citing evidence, refer to the **source name** (e.g., "as reported by The
 Guardian" or "as reported in the New York Times") instead of just the chunk
number or URL.
{{response_format}}
### Evidence Chunks:
{% for chunk in chunks %}
[CHUNK [{{ loop.index }}] START]
URL:{{ chunk.source_url }}
CONTENT: {{ chunk.content }}
[CHUNK [{{ loop.index }}] END]
{% endfor %}

### Task
QUESTION: "{{ question }}"

ANSWER:
```

D Veracity label Generation Prompt Template

```
You are an advanced fact-checking AI tasked with determining the veracity of
claims based on evidence.
### Inputs
CLAIM: "{{ claim }}"
EVIDENCE:
{% for qa in qa_pairs %}
QUESTION: {{ qa.question }}
ANSWER: {{ qa.answer }}
{% endfor %}

### Task
Based on the evidence above, provide step by step reasoning followed by a final
 verdict label for the claim.
### Labels
- "Supported": The evidence fully supports the claim
- "Refuted": The evidence contradicts the claim
- "Conflicting": Different pieces of evidence support and contradict the claim
- "Not Enough Evidence": Insufficient evidence to make a determination
{{ response_format }}
### Instructions
Ensure your verdict is:
- Strictly based on the provided evidence
- Considers all available information
- Acknowledges any uncertainties or gaps

ANSWER:)
```

E SQL Query

```
SQL("""
        WITH
        input_queries AS (
            SELECT
                qid,
                qtext,
                qembedding
            FROM
                UNNEST(%(qtexts)s::text[], %(qembeds)s::vector[]) WITH
                ORDINALITY
                AS t(qtext, qembedding, qid)
        ),
        semantic_search AS (
            SELECT
                t.id,
                iq.qid,
                1.0 / (%(srpenalty)s + RANK() OVER (PARTITION BY iq.qid ORDER
                BY t.embedding <=> iq.qembedding)) AS score
            FROM {tname} t, input_queries iq
            ORDER BY t.embedding <=> iq.qembedding
            LIMIT %(slimit)s
        ),
        keyword_search AS (
            SELECT
                t.id,
                iq.qid,
                1.0 / (%(krpenalty)s + RANK() OVER (PARTITION BY iq.qid ORDER
                BY ts_rank_cd(to_tsvector('english', content), plainto_tsquery
                ('english', iq.qtext)) DESC)) AS score
            FROM {tname} t, input_queries iq
            WHERE to_tsvector('english', content) @@ plainto_tsquery('english',
             iq.qtext)
            ORDER BY ts_rank_cd(to_tsvector('english', content),
            plainto_tsquery('english', iq.qtext)) DESC
            LIMIT %(klimit)s
        ),
        combined AS (
            SELECT id, SUM(score) AS total_score
            FROM (
                SELECT * FROM semantic_search
                UNION ALL
                SELECT * FROM keyword_search
            ) s
            GROUP BY id
        )
        SELECT
            t.doc_id, t.source_url, t.chunk_index, t.content, c.total_score
        FROM combined c
        JOIN {tname} t ON t.id = c.id
        ORDER BY c.total_score DESC
        LIMIT %(topk)s
        """)
```