Modular Sparse and Dense Retrieval for Evidence-Constrained Biomedical Question Answering

Ganesh Chandrasekar[1], **Benjamin Lofo Follo**[1], **Aleksandr Vinokhodov**[1], **Sabine Bergler**[1]
[1]Computational Linguistics at Concordia (CLAC Lab), Concordia University, Montreal, Canada

Abstract

We describe a submission to TREC BioGen Task B, which generates short answers grounded in PubMed and cited with PMIDs. Our approach is modular: a sparse pipeline (BM25 with cross-encoder reranking) and a dense pipeline (MedCPT bi-encoder with cross-encoder) are both coupled with evidence-constrained generation that preserves PMIDs end to end. Each pipeline runs under two retrieval budgets (narrow and wide) using a shared query set formed by the original question plus three reformulations, yielding five runs in total including a baseline. We outline design choices for reformulation, reranking, evidence selection, and citation control, and we report official results for answer quality and citation metrics.

1 Introduction

TREC BioGen Task B (Gupta et al., 2025)[1] asks systems to generate a concise multi-sentence answer grounded in PubMed and cited with explicit PMIDs.

Each input instance is a *topic* that includes an *id*, a *question*, a short *narrative*, and a topic *label*. The system must return a single paragraph with bracketed PMIDs as citations, under the constraints of at most 250 words and at most three citations per sentence. Table 1 summarizes the Task B input/output format and the generation constraints used in our development setup.

We build two retrieval-augmented pipelines with strict citation control: a sparse pipeline using Okapi BM25 with a cross-encoder reranker (Robertson and Zaragoza, 2009), and a dense pipeline using MedCPT bi-encoders with a MedCPT cross-encoder reranker (Jin et al., 2023). We use Qwen2.5 for query reformulation and evidence-constrained generation (Yang et al., 2024; Hui et al., 2024).

[1]`https://trec-biogen.github.io/docs`

Input	id, question, narrative, topic label
Output	One paragraph with bracketed PMIDs
Dev set	30 topics (IDs 181–210)
Constraints	$\leq$ 250 words, $\leq$ 3 citations per sentence
Citation control	PMIDs preserved end to end

Table 1: Task B specification used in development.

Across all runs, we preserve PMIDs from retrieval through selection to generation, and we constrain generation so that the model can only cite PMIDs that appear in the provided evidence block. We report official run-level results using the track's automatic evaluation with BioACE (Gupta et al., 2026).

2 Background

Our sparse pipeline builds on Okapi BM25 (Robertson and Zaragoza, 2009) with cross-encoder reranking to improve first-stage ranking quality (Nogueira and Cho, 2019; Li et al., 2023).

Our dense pipeline follows dual-encoder retrieval, which learns separate encoders for queries and documents and retrieves by embedding similarity (Karpukhin et al., 2020; Xiong et al., 2020). In the biomedical setting, we use MedCPT query, article, and cross-encoders trained contrastively on PubMed-scale data (Jin et al., 2023), enabling semantic retrieval over titles and abstracts beyond lexical overlap.

For generation, we use vLLM (Kwon et al., 2023) with **Qwen2.5-Coder-Instruct** (32B, GPTQ-Int4), relying on the Qwen technical reports for model and chat formatting details (Yang et al., 2024; Hui et al., 2024). To broaden lexical and semantic coverage in retrieval, we generate multiple reformulations per topic and pool retrieved candidates across subqueries by PMID. For each PMID, we keep the best reranker score observed and record which subqueries retrieved it (for analysis). For evidence selection in the *sparse* pipeline, we combine TF-IDF relevance with Maximal Marginal Relevance (MMR) to keep the prompt focused and reduce redundancy (Carbonell and Goldstein, 1998).

3 Environment and Dependencies

3.1 Hardware

Runs were executed on a single NVIDIA A100 GPU; light preprocessing ran on CPU.

3.2 Software

We use a Conda environment with Python 3.10. Retrieval uses Pyserini over Lucene with Java 21 (Lin et al., 2021; Yang et al., 2017). Generation uses vLLM (Kwon et al., 2023) with a Qwen family model `Qwen/Qwen2.5-Coder-32B-Instruct-GPTQ-Int4` (Yang et al., 2024; Hui et al., 2024), which we refer to as the Qwen-Instruct model in the rest of this paper. Models are loaded with Hugging Face Transformers. Evidence selection in the *sparse* pipeline uses scikit-learn for TF-IDF and MMR (Carbonell and Goldstein, 1998). Sparse retrieval relies on BM25 (Robertson and Zaragoza, 2009). Dense retrieval and reranking use MedCPT bi- and cross-encoders (Jin et al., 2023). Table 2 summarizes the software stack and the role of each component used in our system.

3.3 Contribution to task evaluation

We provide a read-only web viewer to support manual inspection of our submitted runs.[2] For each topic, the viewer displays the generated answer, sentence-level PMID citations, and the retrieved evidence for the cited PMIDs, including the PubMed title and abstract text. The viewer also supports run-to-run comparison across our five submissions to facilitate evidence tracing and citation verification.

4 System Overview

We adapted the official BioGen 2025 starter kit[3] and built a modular retrieval-augmented generation pipeline with strict citation control. For each topic, we form a fixed query set consisting of the original question plus three reformulated questions (Section 5). We refer to each element of this set as a *subquery*. We retrieve evidence independently for each subquery and then pool candidates across subqueries by PMID to produce a single candidate set for the topic.

[2] `https://trec-biogen-taskb-viewer.vercel.app`
[3] `https://github.com/trec-biogen/starter-kit-2025`

We evaluate two retrieval budgets. In **narrow** mode, we retrieve the top 25 BM25 hits per subquery and keep the top 10 after reranking. In **wide** mode, we retrieve the top 200 BM25 hits per subquery and keep the top 30 after reranking. These settings mainly change the size and diversity of the pooled candidate set.

All runs generate the final answer with Qwen under the same output constraints, and we enforce at most 250 words and at most three citations per sentence. We preserve PMIDs end to end: every PMID that appears in brackets in the final answer must be present in the evidence block provided to the generator. Internally, the generator cites *evidence indices*, which are mapped back to bracketed *PMIDs* in post-processing, making citation parsing deterministic at evaluation time. The full prompt templates are provided in Appendix A.

Submitted Runs: We submit five runs that differ by retrieval type (sparse versus dense) and retrieval budget (narrow versus wide):

- **Run A (baseline, narrow):** BM25 on the original question only, cross-encoder reranking, then Qwen answer generation with bracketed PMIDs.
- **Run B (sparse, narrow):** BM25 on all subqueries, cross-encoder reranking, pool by PMID, evidence selection, then Qwen answer generation.
- **Run C (dense, narrow):** BM25 on all subqueries, cross-encoder reranking, pool by PMID, MedCPT bi-encoder rescoring and MedCPT cross-encoder reranking, then Qwen answer generation.
- **Run D (dense, wide):** Same as Run C, but with the wide retrieval budget.
- **Run E (sparse, wide):** Same as Run B, but with the wide retrieval budget.

Baseline run (Run A): The baseline retrieves candidates using BM25 from the original question only. After deduplication by PMID, we rerank candidates using a cross-encoder conditioned on the original question and provide the top evidence to Qwen to generate a single paragraph. Post-processing enforces the length and citation caps and removes any PMID not present in the evidence.

Component	Library	Notes
Retrieval (BM25)	Pyserini + Java 21	Lucene backend
Sparse cross-encoder rerank	HF Transformers	`cross-encoder/ms-marco-MiniLM-L-6-v2`
Dense retrieval	HF Transformers	MedCPT `ncbi/MedCPT-Query-Encoder` and `ncbi/MedCPT-Article-Encoder`
Dense cross-encoder rerank	HF Transformers	`ncbi/MedCPT-Cross-Encoder`
Generation runtime	vLLM	`Qwen/Qwen2.5-Coder-32B-Instruct-GPTQ-Int4`
Tokenization	HF Transformers	Qwen tokenizer and chat template
Selection	scikit-learn	TF-IDF and MMR

Table 2: Software stack and roles.

Sparse pipeline (Runs B and E): For the sparse pipeline, BM25 retrieval and cross-encoder reranking are performed per subquery. We then pool candidates across subqueries by PMID, keeping the best reranker score per PMID and recording which subqueries retrieved it. Each pooled document is summarized with Qwen and split into sentence-level bullets, where each bullet includes its PMID. We select a compact set of bullets using TF-IDF relevance followed by MMR for diversity, and we pass only these selected bullets to Qwen for final answer generation.

Dense pipeline (Runs C and D): The dense pipeline starts from the same BM25 retrieval and PMID pooling stage. We then rescore pooled candidates using MedCPT dual encoders: a query encoder for the topic-side text and an article encoder for the candidate title and abstract. We rank candidates by embedding similarity and apply the MedCPT cross-encoder to rerank the shortlist. The top evidence is provided to Qwen for final answer generation under the same citation and length constraints.

5 Reformulation

We use query reformulation to increase retrieval coverage by retrieving over the original question plus three paraphrastic variants. The motivation comes from our MedHopQA pipeline for multi-hop questions, where we issued multiple subqueries to expose missing hops. In BioGen, we do not perform hop decomposition; instead, we use paraphrastic variants to improve lexical and semantic coverage for the same information need.

Importantly, the aggregation differs across the two settings. In BioGen, we retrieve and rerank per subquery, then pool candidates by PMID using the maximum reranker score observed for that PMID. In MedHopQA, subqueries are used to support multi-hop reasoning and later answer construction, rather than PMID-based pooling. Nonetheless, both settings use multiple subqueries per topic to broaden evidence coverage (Saisudha et al., 2025).

During the shared-task submission, reformulations were generated using only the original question as input (topic and narrative were not used). In post-task analysis, we expect that including the full topic context (topic and narrative) would yield variants that better reflect the intended information need and reduce ambiguity, especially for under-specified questions. We treat this change as future work.

We generate exactly three reformulations using Qwen served via vLLM (Kwon et al., 2023; Yang et al., 2024; Hui et al., 2024). The prompt requests: (i) a more formal, clinically phrased version, (ii) a direct, patient-focused inquiry, and (iii) a version that focuses on one specific aspect of the question, such as diagnosis, treatment, or prognosis. Together with the original question, this yields four subqueries per topic. Run A uses only the original question, while Runs B–E use the full subquery set (Section 4).

Output format and rule-based extraction: To keep the query set stable, we instruct the model to output exactly three complete questions numbered `1.`, `2.`, and `3.`, with no additional text. We then apply a fixed rule-based parser: we keep only lines that begin with `1.`, `2.`, or `3.`, strip the numbering, trim whitespace, remove exact duplicates while preserving order, and keep at most three distinct reformulations. This prevents commentary from entering retrieval and makes the reformulation stage reproducible for a given model output.

Both narrow and wide modes use the same subquery set; the only difference between these modes is the retrieval budget applied downstream.

6 Baseline (Run A)

6.1 BM25 retrieval (original only)

Run A uses the original question only and does not apply reformulation. We retrieve the top 25 documents using BM25 from the indexed PubMed title and abstract text. Each hit is stored as a **(pmid,**

text) pair, where *text* concatenates the title and abstract. We then deduplicate candidates by PMID to obtain a unique candidate list for the topic. Next, we rerank the candidate list with a cross-encoder conditioned on the original question. After reranking, we keep the top 10 PMIDs as the evidence set for generation.

We format the generation prompt using the topic, narrative, and the evidence block. Evidence is formatted as indexed evidence lines of the form `[i] (pmid) text`, and the model is instructed to cite by evidence index using square brackets (e.g., `[1]` or `[1,3]` or `[2-4]`). The generator produces a single paragraph under the task constraints.

6.2 Post-processing and constraints

After generation, we apply deterministic post-processing to enforce the task limits and to ensure that all citations are traceable to the provided evidence. First, we normalize whitespace in the generated text. We then segment the paragraph into sentences using punctuation-based splitting. For each sentence, we extract citation markers in square brackets and interpret them as evidence indices (including comma-separated lists and index ranges such as `[2-4]`). We map these evidence indices to PMIDs using the index-to-PMID lookup constructed when building the evidence block for the prompt. Any citation that does not map to a provided evidence item is discarded.

We deduplicate PMIDs within each sentence and cap the number of citations per sentence to three by keeping the first occurrences. We then replace evidence-index citations with the corresponding bracketed PMIDs and drop any sentence that has no remaining valid PMID citations. Finally, we enforce the 250-word answer limit by removing trailing sentences until the total word count is within the cap.

This enforcement can reduce Answer Recall or Answer Completeness in cases where the model generates additional relevant sentences but fails to attach valid evidence-index citations, or when the final sentences contain useful content that is removed to satisfy the hard word cap. However, it improves citation validity by ensuring that every remaining sentence is explicitly supported by retrieved evidence.

7 Sparse Pipeline (Runs B and E: BM25)

7.1 Reformulation and BM25 retrieval

Runs B and E use a fixed query set consisting of the original question plus three reformulations (Section 5). We retrieve evidence independently for each query branch. For each branch, BM25 returns the top k hits, where k=25 in the narrow setting (Run B) and k=200 in the wide setting (Run E). Each branch is then reranked immediately using a cross-encoder that scores query–document pairs, and we keep the top m reranked hits per branch (m=10 for Run B and m=30 for Run E).

After reranking and truncation within each branch, we pool results across branches by PMID. Pooling performs a union over PMIDs: if a PMID appears in multiple branches, we keep a single copy and retain (i) the highest cross-encoder score observed for that PMID and (ii) the set of query branches that retrieved it. The pooled candidate list is then sorted by this best cross-encoder score. We do not apply an additional reranking model after pooling, so the final pooled order is determined solely by the maximum per-branch cross-encoder score for each PMID.

7.2 LLM summaries for pooled candidates

For each pooled candidate document, we generate a short summary using the same Qwen model used elsewhere in the pipeline. The summarization prompt conditions on the original question and asks for a concise, neutral 1–2 sentence summary, prioritizing study outcomes and quantitative details when present. We store each result as a $(pmid, summary)$ pair to preserve the PMID association for downstream evidence construction.

7.3 Evidence bullets

To construct a compact evidence block, we split each summary into sentence-level bullets and attach the PMID to each bullet. To control prompt length, we cap the number of bullets per document and truncate long bullets to a fixed word limit (28 words in our runs). This truncation is a practical budget choice to keep the evidence block within the generator context window. It can remove some detail when the summary sentence is long, so it is a potential source of lost evidence signal, especially for recall-oriented content. A natural alternative is to generate bullet-style summaries directly; we leave this as future work.

7.4 Evidence selection with TF-IDF and MMR

We rank and select bullets in two stages. First, we compute TF-IDF similarity between each bullet and the query set (original plus reformulations) and use this as a relevance signal. Second, we apply Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to choose a diverse subset of bullets, balancing relevance against redundancy with already-selected bullets. In our implementation, TF-IDF provides the relevance scores, and MMR performs greedy selection on top of those scores (with λ=0.75). This ordering keeps the selected evidence focused on the question while reducing near-duplicate statements.

Reversing the order (diversity-first, relevance-second) could increase topical breadth, but it risks selecting off-focus bullets early and can reduce precision. A more stable compromise is to apply MMR on a TF-IDF shortlist (top-N bullets) rather than on the full set, which we treat as a reasonable extension if we need to push coverage further.

7.5 Answer generation

The selected bullets are formatted into an evidence block where each bullet is associated with an evidence index and its PMID. We then prompt Qwen to produce a single cohesive paragraph where each sentence ends with bracketed citations that refer only to evidence indices (mapped back to PMIDs). The same post-processing and constraint enforcement described in Section 6.2 is applied, including the hard 250-word cap and the three-citations-per-sentence cap.

8 Dense Pipeline (Runs C and D: MedCPT)

8.1 Reformulation and BM25 retrieval

Runs C and D use the same fixed query set as the sparse pipeline: the original question plus three reformulations (Section 5). We retrieve evidence independently for each query branch. For each branch, BM25 returns the top k hits, where k=25 in the narrow setting (Run C) and k=200 in the wide setting (Run D). Each branch is then reranked immediately using the same cross-encoder reranker, and we keep the top m reranked hits per branch (m=10 for Run C and m=30 for Run D). This initial cross-encoder reranking is used only for per-branch pruning before pooling; final dense ranking is performed by MedCPT rescoring and the MedCPT cross-encoder (Section 8.2).

After reranking and truncation within each branch, we pool results across branches by PMID using the same procedure as in Section 7.1. Pooling performs a union over PMIDs and retains the highest cross-encoder score observed per PMID, along with the set of query branches that retrieved it. This pooled candidate list is the input to the MedCPT rescoring stage.

8.2 MedCPT rescoring and reranking

From the pooled candidate list, we apply MedCPT dual-encoder rescoring using `ncbi/MedCPT-Query-Encoder` for the query side and `ncbi/MedCPT-Article-Encoder` for the document side (Jin et al., 2023). We build query text from the topic context (topic and narrative) and the question variants (original plus reformulations). We encode each variant with the MedCPT query encoder and aggregate embeddings by averaging to form a single query representation. Each pooled candidate is encoded from its title and abstract using the MedCPT article encoder. Candidates are ranked by cosine similarity, and we keep up to the top 80 by bi-encoder score.

We then apply the MedCPT cross-encoder `ncbi/MedCPT-Cross-Encoder` to rerank this bi-encoder shortlist and keep the top 24 candidates by cross-encoder score. Finally, we retain the top 16 candidates as the evidence set passed to the generator.

8.3 Answer generation

Evidence is formatted as indexed lines of the form `[i] (pmid) snippet`, so the model cites indices rather than PMIDs directly. Qwen is prompted to produce a single cohesive paragraph where each sentence ends with bracketed citations using only evidence indices, which are mapped back to PMIDs deterministically. The same post-processing and constraint enforcement described in Section 6.2 is applied, including the 250-word hard cap and the three-citations-per-sentence cap.

9 Official Results

9.1 Evaluation setup and metrics

Official scoring follows the BioGen Task B evaluation setup (Gupta et al., 2025) and uses BioACE for automatic evaluation (Gupta et al., 2026). We report run-level metrics for: (i) **answer quality**

Run	Answer Precision	Answer Recall	Answer Completeness	Answer Correctness
A: Baseline	88.31	32.34	**77.63**	66.90
B: Sparse (narrow)	89.04	35.15	67.70	64.19
C: Dense (narrow)	92.28	36.41	69.53	65.74
D: Dense (wide)	**93.83**	**36.73**	72.56	**67.33**
E: Sparse (wide)	90.99	35.80	66.77	64.56

Table 3: Official run-level answer quality metrics for our submitted runs.

Run	Citation Coverage	Citation Support Rate	Citation Contradiction Rate
A: Baseline	74.87	**97.38**	0.44
B: Sparse (narrow)	95.45	91.12	**0.39**
C: Dense (narrow)	94.90	90.37	0.46
D: Dense (wide)	92.21	87.46	1.39
E: Sparse (wide)	**95.83**	87.32	0.72

Table 4: Official citation metrics for our submitted runs.

(Answer Precision, Answer Recall, Answer Completeness, Answer Correctness) and (ii) **citation quality** (Citation Coverage, Citation Support Rate, Citation Contradiction Rate). All values are percentages. Higher is better for all metrics except Citation Contradiction Rate, where lower is better.

9.2 Results

Tables 3 and 4 report official results across five runs varying retrieval type and retrieval budget.

Dense versus sparse

On answer quality (Table 3), the dense MedCPT runs (C, D) outperform the sparse runs (B, E) on precision-focused metrics. Run D achieves the best Answer Precision (93.83) and Answer Correctness (67.33) among our runs, while Run C is next-best on these two metrics (92.28 precision, 65.74 correctness). The sparse runs improve recall relative to the baseline, but they remain slightly below the dense runs on recall and notably below them on precision (e.g., Run E recall 35.80 vs. Run D recall 36.73; precision 90.99 vs. 93.83).

On citation metrics (Table 4), the sparse runs achieve the highest Citation Coverage (Run E: 95.83; Run B: 95.45) and the lowest Citation Contradiction Rate among our runs (Run B: 0.39). The dense runs remain competitive on coverage in the narrow setting (Run C: 94.90), but contradiction increases in the wide dense setting (Run D: 1.39). Overall, within our submissions, dense reranking improves answer precision/correctness, while sparse evidence selection yields stronger citation coverage and lower contradiction.

Narrow versus wide

Increasing the retrieval budget produces small but consistent gains in Answer Recall within each pipeline: sparse recall increases from 35.15 (Run B) to 35.80 (Run E), and dense recall increases from 36.41 (Run C) to 36.73 (Run D). In the dense pipeline, the wide setting also improves Answer Completeness (69.53 in Run C to 72.56 in Run D) while keeping high precision.

For citations, the narrow setting yields higher support and lower contradiction. In the dense pipeline, moving from narrow to wide reduces Citation Coverage (94.90 to 92.21) and Citation Support Rate (90.37 to 87.46), while increasing Citation Contradiction Rate (0.46 to 1.39). In the sparse pipeline, Citation Support Rate also decreases from 91.12 (Run B) to 87.32 (Run E), even though coverage remains high (95.45 to 95.83). These results suggest that widening the evidence pool slightly improves recall, but makes it harder to maintain consistently supported citations.

Baseline

The baseline (Run A) performs strongly on two metrics within our submissions: it has the highest Answer Completeness (77.63) and the highest Citation Support Rate (97.38). At the same time, it has the lowest Answer Recall (32.34) and the lowest Citation Coverage (74.87). This is consistent with the baseline using only the original question (no reformulations) and a smaller evidence pool, which can maintain highly supported citations but limits coverage of the information need.

10 Conclusion

We presented a modular system for TREC BioGen Task B with a baseline run, sparse BM25+cross-encoder runs, and dense MedCPT runs. All runs share a fixed query set (original plus three reformulations) and enforce strict evidence control such that every bracketed PMID is traceable end to end. Narrow versus wide retrieval budgets let us trade off recall and prompt focus while keeping downstream selection and generation consistent.

Official results show that the wide MedCPT run achieves the strongest Answer Precision and Answer Correctness among our runs, and that wide settings increase Answer Recall compared to narrow in both sparse and dense pipelines. On citation metrics, Runs B to E achieve high Citation Coverage with low Citation Contradiction Rate under the same citation constraints.

Future work will (i) incorporate topic and narrative into reformulation to better preserve the full information need, and (ii) systematically explore alternative retrieval techniques and design variants at each stage of the pipeline (retrieval, reranking, pooling, evidence selection, and generation) to characterize step-wise trade-offs under the same citation constraints.

References

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Deepak Gupta, Davis Bartels, and Dina Demner-Fushman. 2026. Bioace: An automated framework for biomedical answer and citation evaluations. *https://arxiv.org/abs/2602.04982*.

Deepak Gupta, Dina Demner-Fushman, William Hersh, Steven Bedrick, and Kirk Roberts. 2025. Overview of TREC 2025 Biomedical Generative Retrieval (BioGen) Track. In *The Thirty-Fourth Text REtrieval Conference Proceedings (TREC 2025)*, NIST Special Publication. National Institute of Standards and Technology (NIST).

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Qiao Jin, Won Kim, Qingyu Chen, Donald C Comeau, Lana Yeganova, W John Wilbur, and Zhiyong Lu. 2023. Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 39(11):btad651.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2023. Parade: Passage representation aggregation for document reranking. *ACM Trans. Inf. Syst.*, 42(2).

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2356–2362, New York, NY, USA. Association for Computing Machinery.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Harikrishnan Gurushankar Saisudha, Ganesh Chandrasekar, and Sabine Bergler. 2025. Agentic and non-agentic multi-hop systems for medical question answering. Zenodo. Conference proceeding, published Aug 14, 2025, v1.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *CoRR*, abs/2007.00808.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1253–1256, New York, NY, USA. Association for Computing Machinery.

A Prompts

A.1 Reformulation prompt

Listing 1: Reformulation prompt

```
Instruction: Using the Original Question,
    rewrite the question in three different
    ways, each a complete, grammatically
    correct question. Preserve meaning; vary
    phrasing.
- One version may use more formal clinical
    language.
- One version may be a direct, patient-focused
     inquiry.
- One version may focus on a specific aspect (
    diagnosis, treatment, or prognosis).

Original Question: {item.question}

Return only the three reformulated questions
    in this format:
1. ...
2. ...
3. ...
Do not include any other text. The response
    must start with '1.'.
```

A.2 Summarization prompt

Listing 2: Summarization prompt

```
Summarize the biomedical EVIDENCE below in
    1--2 sentences (≤80 words).
Keep concrete findings and numbers (%, OR/HR/
    RR, CI, p-values) exactly.
Avoid speculation; do NOT add new facts.
{maybe_topic}{maybe_narrative}
(PMID {pmid}) EVIDENCE:
{evidence}

Summary:
```

A.3 Generation prompt

Listing 3: Generation prompt

```
Instruction: Using the topic, narrative, and
    EVIDENCE below, write a SINGLE cohesive
    paragraph of {sentences_min}-{
    sentences_max} sentences (target {
    answer_words_max} words; hard cap {
    answer_words_max}).
Every sentence MUST end with square-bracket
    citations using ONLY evidence indices, e.
    g., [1] or [1,3] or [2-4]. Use AT MOST {
    citations_per_sentence_cap} citations per
     sentence. Prefer specific quantitative
    claims (%, CI, OR/HR/RR, p-values). Do
    not add section headings or bullets.

Topic: {topic}
Narrative: {narrative}
Original Question: {original_question}
Reformulated Questions: {reformulated_joined}

EVIDENCE (compact claims; cite by index):
{evidence_lines}
Answer:
```