Retriever-Aware Query Rewriting for Tip-Of-The-Tongue Search via Preference Optimization

Vinicius L. C. Faria
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
0009-0001-2565-7768

Rodrygo L. T. Santos
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
0000-0003-2921-8444

***Abstract*—Tip-of-the-Tongue (ToT) known-item retrieval is a challenging task that relies heavily on long, specific, and inaccurate user descriptions resulting from memory failure. In this work, we address an efficient, production-friendly first-retrieval pipeline designed to address the unique challenges of bridging the gap between user intent and target documents. Because ToT queries are often verbose and specific due to memory failure, retrieving the first candidate document set from a large indexed collection requires robust semantic translation. We focus our training on the query side through Supervised Fine-Tuning (SFT) distillation, asymmetric query encoder tuning with aggressively denoised hard negatives, and Simple Preference Optimization (SimPO). Our empirical evaluations demonstrate that this multi-stage tuning process yields substantial improvements across all key retrieval metrics. Furthermore, the strong independent performance of the Harrier-OSS model validates the use of decoder-only architectures for dense ToT retrieval, though the inclusion of simpler models like BM25 proved invaluable for enriching the rewrite training data, outperforming the query-rewrite focused runs published on TREC 2025.**

***Index Terms*—information retrieval, tip-of-the-tongue, first retrieval, query rewriting**

I. Introduction

With recent advancements in natural language processing, current search engines and retrieval systems are well suited for known-item retrieval, where the searcher provides specific keywords or identifiers for the desired item. Modern retrieval pipelines aim to bridge the lexical gap between documents and queries by comparing both items semantically using neural models, which provide a deeper comparison between relevant documents and the user's original intent.

The described retrieval systems, however, also struggle to resolve queries in which the user cannot provide reliable identifiers or precise names and instead relies on long, specific, and verbose descriptions, commonly referred to as Tip-of-the-Tongue (ToT) queries. Such queries are typically a result of human memory failure and frequently occur with movies, celebrities, and landmarks with which the user has had contact in the past but cannot recall major or precise details. Even with modern pipelines such as RAG (Retrieval Augmented Generation), usual search engines provide unsatisfactory results, and the problem is usually manually solved by other users in online forums. Figure 1 showcases a typical ToT query.

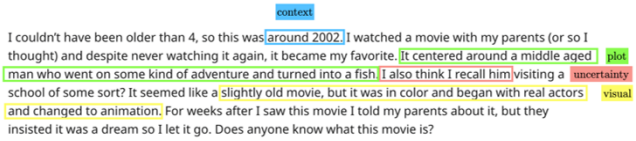

Fig. 1: Example of a Tip-of-the-Tongue query

In this work, we aim to address an efficient and production friendly solution for ToT known-item retrieval queries, specifically addressing the first-retrieval step, which involves obtaining the first candidate document set from a large indexed collection. Due to the niche of the problem and the heavy offline cost of indexing large corpora, we wish to analyze performance by explicitly freezing all document structures, such as the index itself and the document encoders, while tuning a small LLM rewriter and query-related structures. Since asymmetric tuning and query rewriting with a single retriever can be difficult, we also aim to leverage the effect of using multiple language architectures — lexical, encoder-only, and decoder-only — to formulate a more robust signal for all training steps. Due to the nature of TREC tracks, this work aims for a mostly quantitative approach, with decisions guided according to the established metrics from each experiment.

II. Related Work

Tip-of-the-Tongue known-item retrieval was featured as a research topic by the TREC (Text REtrieval Conference) in the 2023, 2024, and 2025 editions, moving to NTCIR (NII Testbeds and Community for Information Access Research) for the 2026 edition. The latest published 2025 edition received several solutions [1], being significantly harder than the 2024 edition [2] while featuring a much larger corpus and open domain questions.

The highest performing solution of the TREC 2025 edition [3] utilized a combination of a dense and an LLM retriever, multiple query rewrites, and rank fusion. Specifically for

the movie domain, it involved finetuning a large 8 billion parameter retriever and building a separate, smaller index, which was highly beneficial since such queries constituted about half of the test domain, featuring non-synthetic queries. Other high performing solutions featured plain query rewriting with a lexical retriever [4] and multiple dense and lexical retrievers [5]. In this work, we also aim to employ query rewriting and multiple retrievers; however, we specifically generate a separate, retriever-aware rewrite for each retriever, adapting to its underlying language architecture while also tuning the LLM model. Additionally, we differ from other fine-tuning solutions by explicitly maintaining all index structures frozen, allowing for a more efficient and production-friendly solution due to the large corpus.

One solution from TREC 2025 also directly focused on query rewriting [6] by establishing a ranking pipeline with a single dense retriever and re-ranker, and tuning an LLM rewriter via Direct Preference Optimization (DPO) [7]. The preference pairs are obtained directly from the dense retriever and re-ranker scores. We employ a similar pipeline but utilize multiple first-retrievers and retriever-aware rewrites, constructing multiple preferences per query regarding each retriever. Furthermore, we diverge from such work by utilizing rank-aware metrics for determining preference pairs and applying a supervised finetuning pre-training step.

Outside the TREC ecosystem, recent work has shown that query rewriting utilizing Large Language Models can be highly beneficial for web search queries [8] [9]. The recently proposed Think-then-Rewrite framework [10] has established state-of-the-art performance for domain-specific rewrites, where the LLM generates a chain of thought and the respective rewrite, being tuned by GRPO [11] with rank-aware metrics. We utilize a similar approach to this framework; however, we opt for supervised finetuning and preference optimization for stability instead of traditional RL techniques, which we recognize is not the best approach for reasoning tasks.

III. Methodology

The experimental methodology was designed with explicit consideration of the specific characteristics of Tip-of-the-Tongue (ToT) queries, which are assumed to be associated with a single, uniquely relevant document, corresponding to the vaguely recalled item.

A. Corpus Pre-Processing & Baseline Retrieval

Modern retrieval pipelines increasingly employ hybrid search paradigms, which combine dense and lexical retrieval mechanisms and have been shown to yield superior performance across standard evaluation metrics [12]. An ensemble of lexic and neural retrievers were employed in both training and evaluation pipelines.

Initially, we employ a pre-processing step for the corpus due to model constraints and data augmentation. We initially remove the tail of each document, which often contains irrelevant sections such as "See Also" and "External Links". For neural retrievers, which benefit from semantic meaning, documents were chunked into 256 token passages with a 30 token overlap, which we empirically deemed a good balance between length and semantic expressiveness. Additionally, we augment such passages by appending the document title and current subtitle to the beginning of each text to ground the current scope of the passage, especially since the passage breaks are completely arbitrary. An augmented passage is presented in figure 2.

Example Passage

Anarchism - Section: Etymology, terminology, and definition. more so. Many scholars reject anarcho-capitalism as a misunderstanding of anarchist principles. While opposition....

Fig. 2: Example augmented passage for neural retrievers

To aggregate passages back into documents, we experimented with two pooling methods: max-pooling and top-3 sum pooling, which sums the total score of the first 3 passages of the document.

For inference, we also add an additional retrieval process, $Fused$, which is obtained by applying Reciprocal Rank Fusion (RRF) for all retriever established rankings. We initially perform two baseline retrievals, which involve no model tuning. First, we utilize the raw, user-defined queries. Secondly, we utilize a small LLM to introduce zero-shot rewrites: one for semantic models and another for lexical models.

B. Supervised Fine-Tuning (SFT) via Teacher Distillation

We aim to replicate a similar procedure to the Think-then-Rewrite framework, but with preference tuning instead of traditional RL techniques. As such, we employ a two-step pipeline consisting of supervised finetuning (SFT) followed by preference tuning. The initial SFT helps to establish an initial grounding of the task to the model and is widely utilized [13]. In our case, such a step distills the chain-of-thought-behavior also utilized by the selected framework and teaches well structured rewrites for each retriever.

We constructed a prompt portraying the rewrite task for ToT queries, including a specific section on the target retriever to which the query will be applied that explicitly names the retriever and describes what an ideal rewrite consists of — for instance, lexical retrievers typically require heavy use of keywords. We then generate one "golden rewrite" per query/retriever pair utilizing a high-capacity teacher LLM $\mathcal{M}_T$. Following [14], we add a post-processing procedure in which we discard queries in which the LLM deduced the target entity with its internal parameters, as well as responses with incorrect formatting.

Utilizing the golden rewrites, we employ distillation via Supervised Fine-Tuning to train a smaller, more efficient stu-

dent model $\mathcal{M}_S$ that inherits the chain-of-thought, formatting, and styling constraints of the teacher but does not distinguish between a well-performing rewrite and a poorly performing one. A single model is utilized for generating all retriever-specific rewrites; the shared task is controlled by the retriever name and description given in the prompt. We believe that the use of multiple retrievers and rewrites for the same query is a strong form of data augmentation and can lead to a more robust training pipeline.

Figure 3 showcases the full retriever-aware pipeline for retrieval utilized by this step, which is refined through further tuning of query encoders and preference tuning of the rewriter.

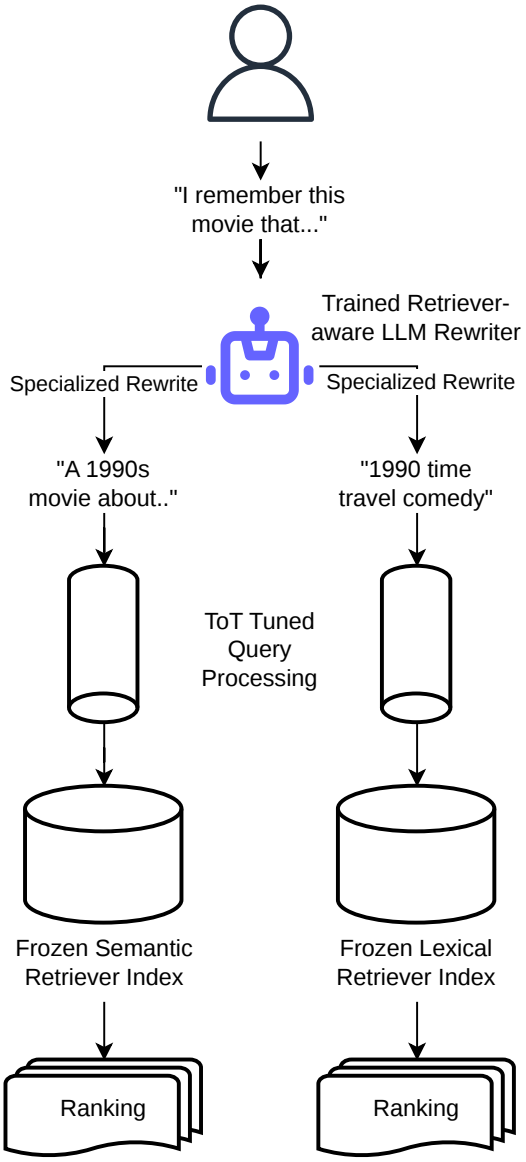

Fig. 3: Showcase of the retriever-aware pipeline

C. Asymmetric Query Encoder Tuning

Aiming to further improve the query encoder quality for the ToT task, we utilize rewrites obtained from the previously distilled model to mine a small hard-negative dataset for the asymmetric fine-tuning of such encoders. Due to the nature of the task, we employ a highly careful mining process: ToT queries are very descriptive and typically have only one relevant document. However, several other documents can share similar semantic meanings (e.g., Star Wars vs. Star Trek). Treating such closely related documents as negatives can be disruptive for the tuning process, especially in the asymmetric case, which risks embedding skew. We position this step between SFT and preference tuning because it provides a strong baseline rewrite model $\mathcal{M}_S$, while keeping it untuned for ranking quality so that the main contribution comes from our final rewriter rather than from overfitting to the trained retrievers.

To address this, we utilize a RocketQA-like approach [15] to denoise the hard negatives, which involves applying a re-ranker to the initial result for each query/retriever. The starting rankings are extracted individually from each retriever pipeline. We utilize the dense retriever rewritten query, as the re-ranker typically benefits heavily from semantic meaning. Due to the length of the documents, we rank the chunked passages of the golden document (utilizing the same re-ranker model) according to the dense rewritten query to find the most relevant sections of the document, used as the target "golden passage" for training.

With the original and re-ranked results for each query and retriever, we aggressively prevent the inclusion of unlabeled false negatives by discarding the Top 10 re-ranked passages and passages extracted from the reference golden document. From the remaining candidates, we construct a stratified hard-negative dataset with a varying number of negatives per retriever. The distribution is designed to expose the encoder to varying levels of difficulty:

- **Retriever-Biased Distractors (50%):** Passages ranked high by the initial retriever (top 30) but low by the re-ranker (below position 100). These force the model to unlearn retriever-specific biases and lexical traps.
- **Boundary Negatives (25%):** Passages ranked moderately (top 30–60) by the CE, providing challenging semantic distractors.
- **Easy Negatives (25%):** Background passages ranked below 120 by both the retriever and the re-ranker to maintain global vector space uniformity.

The result of this process is a dataset containing instances with a retriever-aware query, a corresponding "golden passage," and the hard-mined negatives, which is then utilized to train the query encoders using contrastive learning. For each retriever, 50% of the training hard negatives were drawn from its own rankings, and the other half was evenly sampled from the remaining retrievers, thereby diversifying the training data while maintaining the trained retriever as the main focus.

D. Simple Preference Optimization (SimPO)

To encourage better rewrites for each retriever, we employ preference tuning via Simple Preference Optimization (SimPO) [16], which was preferred over Direct Preference

Optimization (DPO) [7] due to its lower memory footprint and higher efficiency, as it does not require a frozen reference model. We increased the temperature of the original LLM $\mathcal{M}_S$ and produced 5 different rewrites for each query/retriever pair, and then executed the pipeline utilizing the tuned encoders for the initial rankings.

Preference tuning methods require a dataset of chosen and rejected pairs (y_w, y_l). To construct the dataset, we evaluated the downstream retrieval effectiveness of each rewrite using a composite reward function, defining a "retriever-as-a-judge" approach. For a given candidate rewrite, we calculate a reward $\mathcal{R}$ that balances the hard ranking position of the golden document with the retriever's scoring confidence:

$$\mathcal{R} = 0.1 \cdot \text{CMI} + \text{S-MRR}$$

The smoothed Mean Reciprocal Rank (S-MRR) applies a logarithmic decay based on the 0-indexed rank of the golden document, defined as $\text{S-MRR} = \frac{1}{\log_2(\text{rank}+2)}$. If the golden document is not retrieved within the candidate pool, S-MRR defaults to 0.

The Contrastive Mutual Information (CMI) signal was utilized similarly to the Think-then-Rewrite framework [10]. We begin by normalizing the original retrieval scores within the candidate pool into z-scores, then apply a local softmax function parameterized by a temperature τ to yield a probability distribution over the candidates. Then, the CMI is computed as the log-probability of the golden document minus the log-probability of the mean negative documents: $\text{CMI} = \ln(p_{gold}) - \ln(\bar{p}_{neg})$. A static penalty of -10.0 is applied if the target is entirely missing.

Finally, for each query, the rewrite maximizing $\mathcal{R}$ is selected as the chosen response y_w, and the rewrite minimizing $\mathcal{R}$ is selected as the rejected response y_l. To maintain a strong gradient signal during SimPO training, we enforce a strict structural margin, discarding any queries where $\mathcal{R}_{y_w} \leq \mathcal{R}_{y_l}$ to avoid training on ties.

IV. Experimental Setup

Experiments were conducted on a computing node equipped with a single NVIDIA A100 (80GB) GPU. We utilize the TREC 2025 corpus as a reference, containing 6 million full english Wikipedia documents. We utilize the dev1, dev2, and train datasets from the 2025 edition, composed of 428 real-user train queries in the movie domain, and the NTCIR 2026 english train dataset, which consists of 4,000 synthetically generated queries in the open domain, totaling 4,428 queries. The test dataset of the 2025 TREC track comprises 622 queries, representing a balanced mixture of real and synthetically generated open-domain information needs. We extracted the top 3000 ranked passages for retrieval rounds that required the passage-to-document pooling.

We split the training dataset into two distinct parts. The first subset, containing 2,500 queries, was used for supervised fine-tuning of the query rewriter and for tuning the query encoder. The remaining queries were used for preference tuning.

To establish our frozen document indexes and query encoders, we hand-picked 3 distinct state-of-the-art retriever models based on performance, size, and language architectures:

- BM25 [17] - classic purely lexical model
- SPLADE [18] (*naver/splade-v3*) - learned sparse retriever built from a traditional encoder transformer architecture
- Harrier-OSS (*microsoft/harrier-oss-v1-0.6b*) - text embedding model built from an LLM, featuring a decoder-only architecture. As of the time of writing, the model is very new and has shown very good performance with a relatively small model size [19].

For the raw query baseline, we utilized standard PyTerrier [20] parameters for all models. We utilize *meta-llama/Llama-3.1-8B-Instruct* [21] with FP16 quantization for the zero-shot query rewriter baseline due to its wide use in literature.

For the Supervised Fine-Tuning (SFT) phase, we utilized *deepseek-ai/DeepSeek-R1-Distill-Llama-70B* [22] with 4-bit quantization as our high-capacity teacher model ($\mathcal{M}_T$) to generate the target golden rewrites, while the student model ($\mathcal{M}_S$) was initialized from *meta-llama/Meta-Llama-3.1-8B-Instruct*. We fine-tuned the model using Low-Rank Adaptation (LoRA) with a rank of $r = 32$, a scaling factor of $\alpha = 64$ applied to all attention and feed-forward linear layers. To optimize instruction-following capabilities, the loss was computed exclusively on the generated completion tokens. We utilize FP16 quantization for inference.

During the asymmetric query encoder tuning, we employed *nvidia/llama-nemotron-rerank-1b-v2* as the cross-encoder reranker for the denoising process. The query encoders for both SPLADE and Harrier-OSS were trained individually using an InfoNCE loss to separate the golden passages from our stratified hard negatives. Both tuning processes are highly specialized, with SPLADE requiring FLOPs loss and Harrier-OSS requiring calibration of the EOS token. Training parameters for both neural models were obtained by a grid-search, validated from the dev dataset of NTCIR 2026 track, with 500 synthetic queries. For BM25, we use a simple grid-search.

Finally, to construct the candidate pool for Simple Preference Optimization (SimPO), we generated multiple distinct rewrites per query/retriever pair by elevating the student model's sampling temperature to 0.7. For our composite reward calculation, we utilized $\tau = 10.0$ for SPLADE due to the high score magnitude obtained due to integer quantization and $\tau = 1.0$ for both BM25 and Harrier-OSS. The SimPO alignment was executed on the merged SFT model by injecting a new LoRA adapter, utilizing the identical rank, scaling, and targeted modules as in the previous phase. FP16 was also used for inference.

V. Results

The first round of retrieval is primarily focused on recall to allow for a more detailed re-ranking step. Since each query has one relevant document, $Recall@K$, $Success@K$ and $Hit@K$ all refer to the same value. Max pooling was typically the strongest performer on recall metrics, and top-3 sum pooling

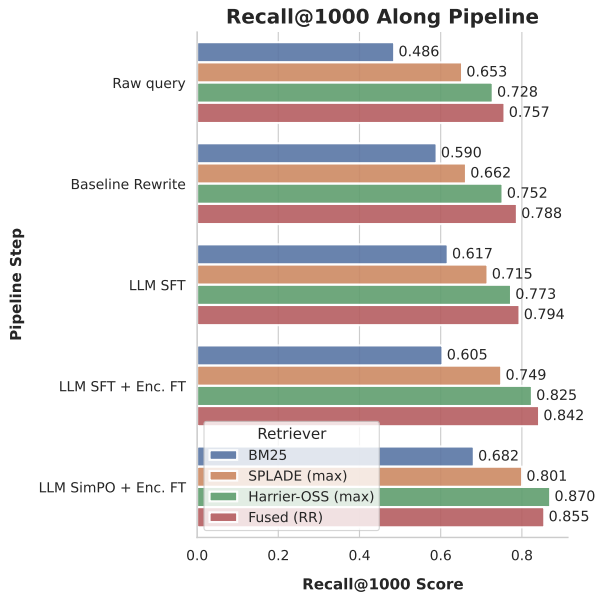

Fig. 4: Recall@1000 along the training pipeline. We omit the top-3 pooling method for conciseness due to weaker performance compared to max pooling.

on precision metrics, so they will be highlighted individually in visualizations. Results were extracted utilizing PyTerrier [20] and *ir_measures* [23].

Figure 4 demonstrates retrieval performance along the training pipeline. Primarily, the initial baseline rewrite, as utilized by most TREC 2025 teams, demonstrates a swift performance jump, especially for BM25, due to spelling errors and similar issues. However, the biggest performance jumps were obtained from query encoder fine-tuning and, most importantly, the LLM preference tuning step, which includes an almost 0.045 positive delta for all retrievers. BM25 itself surpasses SPLADE's results on raw queries while being a much cheaper model.

Table I shows the full results of every pipeline step according to official TREC 2025 metrics, which are directly comparable to the published results in [1]. Although hyperparameters and training decisions were selected with an emphasis on recall, it is noteworthy that remaining metrics also increased in the final step of the pipeline. The preference tuning step showed statistically significant increases for all metrics, and the query encoder also showed significant improvements in all cases except for BM25, where the lack of significance is likely attributable to the simplicity of the grid search. Retriever fusion also achieves very strong results, but Harrier-OSS consistently surpasses it over the training steps, mainly thanks to its greater capacity for generalization, especially when compared with the much smaller and simpler SPLADE and BM25 models. However, we believe the use of weaker retrievers are still beneficial due to the enrichment of query rewriter training and hard negative examples. The final $Recall@1000$ from the complete pipeline is very strong, being second only to SRCB [3] runs from the previous edition.

Figure 5 addresses whether the retriever-as-a-judge confounds rewrite quality with verbosity. Length hacking is a well-documented failure of preference-optimization methods such as DPO and SimPO on free-form generation, where the policy learns to produce longer outputs simply because the reward happens to correlate with length rather than content. To test this, we bin the chosen rewrites of each retriever into five quintiles by word count and plot the mean chosen reward with 95% confidence intervals. Across all three retrievers, the slope is either flat or negative ($r \in [-0.15, -0.09]$), meaning that longer rewrites do not obtain a higher reward. The absence of a positive length-reward correlation indicates that the preference signal is unlikely to induce a verbosity drift during alignment, and that improvements observed after preference tuning can be attributed to the rewriter extracting and inducing better features of the original query.

VI. Conclusion and Future Work

In this work, we presented an efficient, production-friendly first-retrieval pipeline designed to address the unique challenges of Tip-of-the-Tongue (ToT) known-item retrieval. Because ToT queries are often verbose, specific, and inaccurate due to memory failure, bridging the gap between user intent and target documents requires robust semantic translation. We addressed this by leveraging multiple language architectures—lexical (BM25), encoder-only (SPLADE), and decoder-only (Harrier-OSS)—while explicitly freezing the document indexes to avoid the prohibitive costs of re-indexing large corpora. By focusing our training on the query side through Supervised Fine-Tuning (SFT) distillation, asymmetric query encoder tuning with aggressively denoised hard negatives, and Simple Preference Optimization (SimPO), we generated highly effective, retriever-aware query rewrites.

Our empirical evaluations demonstrate that this multi-stage tuning process yields substantial improvements across all key retrieval metrics. The complete pipeline outperforms several runs from the previous edition, including ranking quality metrics such as nDCG@1000, even without employing a reranking stage. It achieves the highest position among all runs except those from SRCB, while also being significantly more suitable for production thanks to its use of frozen indexes and significantly smaller embedding models. Furthermore, the strong independent performance of the Harrier-OSS model validates the use of decoder-only architectures for dense ToT retrieval, though the inclusion of simpler models like BM25 proved invaluable for enriching the rewrite training data, especially since this solution also outperforms the query-rewrite focused runs published on TREC 2025.

Since TREC is highly quantitative and given the scope of this study, we did not perform validation procedures such as ablation studies; these are left for future work. Possible future directions include conducting further experiments on strategies for constructing the retriever ensemble and how they affect the rewriter's training; applying retriever masking to design a robust, unified rewriting framework that operates

TABLE I: Pipeline results on the TREC 2025 test dataset. † indicates statistical significance from the previous pipeline step with $p < 0.05$, extracted from paired t-test. Results in bold highlight the best performing retrieval method along the pipeline step and metric.

Metric	Pipeline step	BM25	SPLADE (max)	SPLADE (top3)	Harrier-OSS (max)	Harrier-OSS (top3)	Fused (RR)
nDCG@10	Raw query	0.122	0.165	0.160	0.198	0.167	**0.221**
	Baseline Rewrite	0.170†	0.163	0.139	0.232†	0.194†	**0.238**†
	LLM SFT	0.161	0.183	0.156	0.237	0.195	**0.244**
	LLM SFT + Enc. FT	0.132	0.219†	0.183†	**0.276**†	0.230†	0.242
	LLM SimPO + Enc. FT	0.216†	0.282†	0.224†	**0.318**†	0.268†	0.286†
RR	Raw query	0.111	0.149	0.146	0.176	0.146	**0.201**
	Baseline Rewrite	0.156†	0.151	0.126	0.209†	0.170†	**0.211**
	LLM SFT	0.144	0.169	0.144†	**0.216**	0.177	0.215
	LLM SFT + Enc. FT	0.118	0.199†	0.166†	**0.245**†	0.208†	0.220
	LLM SimPO + Enc. FT	0.199†	0.257†	0.203†	**0.284**†	0.242†	0.258†
Recall@1000	Raw query	0.486	0.653	0.659	0.728	0.741	**0.757**
	Baseline Rewrite	0.590†	0.662	0.675	0.752	0.749	**0.788**†
	LLM SFT	0.617	0.715†	0.727†	0.773	0.770	**0.794**
	LLM SFT + Enc. FT	0.605	0.749†	0.749†	0.825†	0.834†	**0.842**†
	LLM SimPO + Enc. FT	0.682†	0.801†	0.797†	**0.870**†	0.868†	0.855
nDCG@1000	Raw query	0.170	0.230	0.225	0.269	0.238	**0.294**
	Baseline Rewrite	0.226†	0.232	0.210	0.302†	0.262†	**0.310**†
	LLM SFT	0.221	0.256†	0.232†	0.309	0.270	**0.316**
	LLM SFT + Enc. FT	0.196	0.289†	0.256†	**0.345**†	0.305†	0.325
	LLM SimPO + Enc. FT	0.279†	0.350†	0.297†	**0.386**†	0.339†	0.363†

consistently across all retrievers; and deploying the trained rewriter on a novel retriever, handling the retrieval step in a zero-shot fashion. We also plan to build a standard re-ranker for submitting runs to NTCIR's ToT English 2026 edition.

VII. Use of AI

Generative AI has been utilized in this work for polishing this report (rephrasing, grammar, etc) and auxiliary coding tasks.

References

[1] J. Arguello, F. Diaz, M. Fröebe, T. E. Kim, and B. Mitra, "Overview of the trec 2025 tip-of-the-tongue track," in *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, ser. NIST SP xxxx, Gaithersburg, Maryland, 2025.

[2] J. Arguello, S. Bhargav, F. Diaz, T. E. Kim, Y. He, E. Kanoulas, and B. Mitra, "Overview of the trec 2024 tip-of-the-tongue track," in *Text REtrieval Conference (TREC)*, NIST. TREC, March 2025. [Online]. Available: https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2024-tip-of-the-tongue-track/

[3] H. Li, Y. Zhang, J. Zhou, Y. Zhang, S. Jiang, and B. Dong, "Srcb at trec 2025: Million llms and tip-of-the-tongue tracks," in *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, ser. NIST SP xxxx, Gaithersburg, Maryland, 2025.

[4] M. Fröbe, J. H. Merker, E. O. Schmidt, M. Potthast, and M. Hagen, "Webis at trec 2025: Tip-of-the-tongue track and autojudge," in *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, ser. NIST SP xxxx, Gaithersburg, Maryland, 2025.

[5] W. Zhou, R. Mehta, and A. Miyaguchi, "Ds@gt at trec tot 2025: Bridging vague recollection with fusion retrieval and learned reranking," in *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, ser. NIST SP xxxx, Gaithersburg, Maryland, 2025.

[6] A. P. Nader and R. L. T. Santos, "Ufmg at trec 2025: Retriever-aligned query rewriting for tip-of-the-tongue retrieval," in *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, ser. NIST SP xxxx, Gaithersburg, Maryland, 2025.

[7] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," 2024. [Online]. Available: https://arxiv.org/abs/2305.18290

[8] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting for retrieval-augmented large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.14283

[9] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," 2022. [Online]. Available: https://arxiv.org/abs/2212.10496

[10] A. Li, Y. Shi, Y. Si, Y. Wu, M. Cai, X. Tan, Y. Wang, C. Sun, X. Liu, and K. Kuang, "Think then rewrite: Reasoning enhanced query rewriting for domain specific retrieval," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, pp. 15 045–15 053, 03 2026.

[11] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.03300

[12] D. Sultania, Z. Lu, T. Naik, F. Dernoncourt, D. S. Yoon, S. Sharma, T. Bui, A. Gupta, T. Vatsa, S. Suresha, I. Verma, V. Belavadi, C. Chen, and M. Friedrich, "Domain-specific question answering with hybrid search," 2024. [Online]. Available: https://arxiv.org/abs/2412.03736

[13] W. Xiao, Z. Wang, L. Gan, S. Zhao, Z. Li, R. Lei, W. He, L. A. Tuan, L. Chen, H. Jiang, Z. Zhao, and F. Wu, "A comprehensive survey of direct preference optimization: Datasets, theories, variants, and applications," 2026. [Online]. Available: https://arxiv.org/abs/2410.15595

[14] Y. He, T. E. Kim, F. Diaz, J. Arguello, and B. Mitra, "Tip of the tongue query elicitation for simulated evaluation," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 3398–3407. [Online]. Available: https://doi.org/10.1145/3726302.3730335

[15] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, and H. Wang, "Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering," 2021. [Online]. Available: https://arxiv.org/abs/2010.08191

[16] Y. Meng, M. Xia, and D. Chen, "Simpo: Simple preference optimization with a reference-free reward," 2024. [Online]. Available: https://arxiv.org/abs/2405.14734

[17] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, p. 333–389, Apr. 2009. [Online]. Available: https://doi.org/10.1561/1500000019

[18] C. Lassance, H. Déjean, T. Formal, and S. Clinchant, "Splade-v3: New baselines for splade," 2024.

[19] H. Cho, R. Kang, and Y. Kim, "Skillret: A large-scale benchmark for skill retrieval in llm agents," 2026. [Online]. Available: https://arxiv.org/abs/2605.05726

[20] C. Macdonald and N. Tonellotto, "Declarative experimentation in

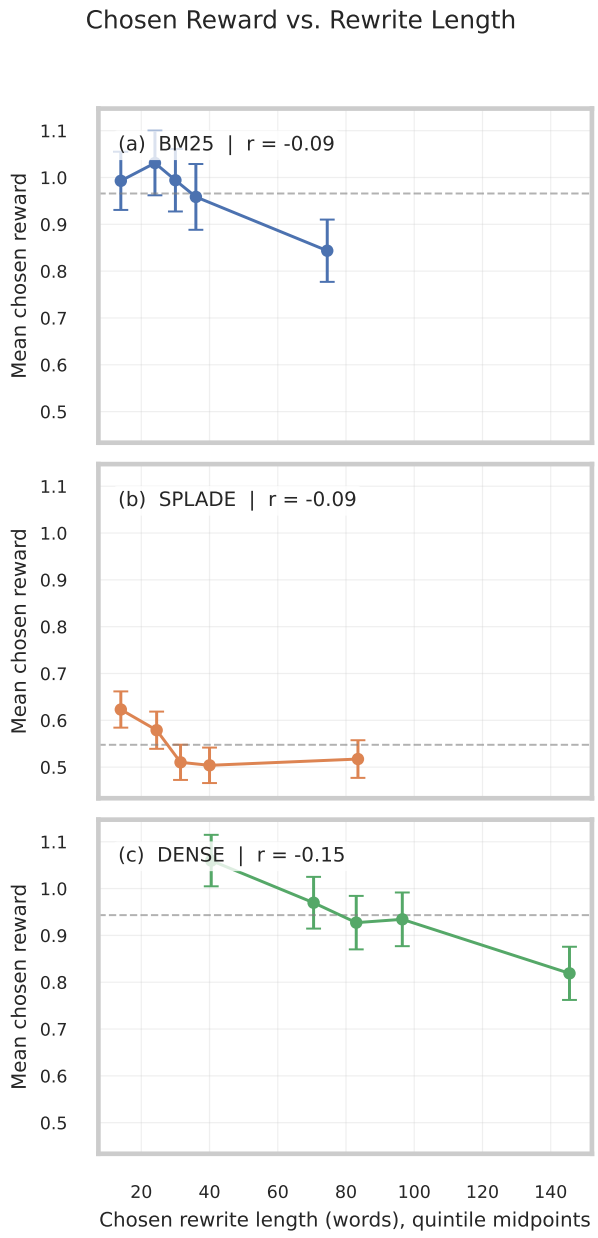

Fig. 5: Chosen reward as a function of rewrite length, per retriever. Queries are binned into five quintiles of chosen-rewrite word count; points show the mean chosen reward with 95% confidence intervals, against a dashed global-mean baseline. The negative slopes ($r \in [-0.15, -0.09]$) indicate the retriever-judge rewards precision over verbosity, precluding length-based reward hacking during preference optimization.

information retrieval using pyterrier," in *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, ser. ICTIR '20. ACM, Sep. 2020, p. 161–168. [Online]. Available: http://dx.doi.org/10.1145/3409256.3409829

[21] A. G. et al, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[22] D.-A. et al., "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[23] S. MacAvaney, C. Macdonald, and I. Ounis, "Streamlining evaluation with ir-measures," in *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 13186. Springer, 2022, pp. 305–310. [Online]. Available: https://doi.org/10.1007/978-3-030-99739-7_38

APPENDIX

A. *Prompts*

The utilized system message is constant across all stages (teacher golden generation, student SFT, and SimPO):

System Prompt

You are an expert Information Retrieval Query Rewriter.

The user prompt utilized for the teacher generation and the tuned rewriter is shown at figure 6 instantiated per query and per retriever, with the placeholder `{retriever_type}` replaced by one of `BM25`, `SPLADE`, or `DENSE`:

For the baseline rewrites, however, we employ a separate, untuned *Llama-3.1-8B-Instruct* rewriter with a distinct prompt, shown in Figure 7. This is necessary due to formatting constraints and the fact that the model does not inherently support the Chain-of-Thought reasoning required by the think-then-query template used in the tuned pipeline. Such behavior was only consistently achieved after the SFT phase.

User Prompt for the Tuned Rewriter

Your task is to translate an ambiguous "Tip-of-the-Tongue" user query into an optimal search string for a retriever_type retrieval system.

User Query: {tot_query}

Instructions for {retriever_type}:
{get_retriever_prompt(retriever_type)}

CRITICAL CONSTRAINTS:
1. DO NOT try to guess the exact name of the target entity (e.g., the specific movie name or book title). If you guess wrong, it will ruin the retrieval.
2. Instead, focus on extracting the core concepts, themes, actions, and constraints (like dates or genres) from the user's query.
3. Clean up any conversational noise ("I'm trying to remember...", "Does anyone know...").
4. Add 2-3 synonyms for the most distinctive concepts only. Avoid over-expanding generic terms like "movie" or "book".

In your <think> tags, reason about which parts of the user's query are the most important anchors, and what safe expansions would help retriever_type.
After you are done thinking, wrap your final rewritten query in <query></query> tags.

Fig. 6: Prompt for the tuned rewriter

User Prompt for the Baseline

You are an expert retrieval assistant. Analyze the user's Tip-of-the-Tongue description.
1. Extract a clear list of atomic search cues (keywords, fragments, entities).
2. Synthesize those cues into a single, highly descriptive but concise while retaining nuance, grammatically fluent sentence that a dense retrieval model can embed effectively.

You must respond ONLY with a valid JSON object. Do not include markdown formatting, preambles, or explanations.

Fig. 7: Prompt for baseline rewriter