Yale NLP at TREC 2024: Tip-of-the-Tongue Track

Rohan Phanse
Yale University
rohan.phanse@yale.edu

Gabrielle Kaili-May Liu
Yale University
kaili.liu@yale.edu

Arman Cohan
Yale University
arman.cohan@yale.edu

Abstract

This paper describes our submissions to the TREC 2024 Tip-of-the-Tongue (ToT) track. We use a two-stage pipeline consisting of DPR-based retrieval followed by reranking with GPT-4o mini to answer ToT queries across three domains: movies, celebrities, and landmarks. Two of our runs performed retrieval using a "general" DPR model trained to handle queries from all domains. For our third run, we developed an approach to route queries to multiple "expert" DPR models each trained on a single domain. To build training sets for our DPR models, we collected existing ToT queries and generated over 100k synthetic queries using few-shot prompting with LLMs. After retrieval, results were reranked either listwise or using a combined pointwise and listwise approach. Our results demonstrate the efficacy of our three submitted approaches, which achieved NDCG@1000 scores ranging from 0.51 to 0.60.

1 Introduction

The Tip-of-the-Tongue (ToT) task refers to the challenge of retrieving the identifier of an item from generally verbose and vague recollections of that item [1]. The TREC ToT track was launched in 2023 to motivate the development of information retrieval (IR) solutions for this task [1].

In this paper, we describe our submissions to the TREC 2024 ToT track. We propose a two-stage pipeline consisting of retrieval using Dense Passage Retriever (DPR) models [2] followed by reranking with GPT-4o mini to answer ToT queries across three domains: movies, celebrities, and landmarks. To build training datasets for our DPR models, we implemented the pipeline proposed by Borges et al. [3] in TREC ToT 2023 to create a dataset of movie queries and collect few-shot examples of celebrity and landmark queries. To address the scarcity of existing celebrity and landmark queries on the Internet, we developed a few-shot prompting approach to generate over 100k synthetic queries using GPT-4o mini.

After preparing training sets for each domain, we trained a single "general" DPR model to handle queries from all domains and used it in our first two runs. In addition, we developed an approach to route queries to multiple single-domain "expert" DPR models for our third run. We used GPT-4o mini to rerank the results retrieved by our DPR models. We developed an initial pointwise reranking stage that we used along with Borges et al.'s [3] listwise round-robin approach in our first run. We only performed listwise reranking in our other two runs to measure the specific contribution of our proposed pointwise reranking stage to overall performance.

2 Methods

We begin by describing the process of preparing and generating our training sets in Section 2.1. We then discuss our "routing" retrieval approach in Section 2.2 and our reranking pipeline in Section 2.3.

2.1 Query Generation

To build the training datasets for our DPR models, we paired the pipeline proposed by Borges et al. [3] for collecting real-world ToT queries with our few-shot prompting approach for generating synthetic queries using LLMs.

Motivated by Borges et al.'s [3] success with collecting 118k ToT movie queries from Fröbe et al.'s TOMT-KIS dataset of 1.3M+ Reddit posts [4], we employed various filtering techniques to extract relevant posts about celebrities and landmarks from this dataset. We ultimately obtained only $3,051$ posts regarding celebrities and 117 posts regarding landmarks that satisfied our criteria of having an answer and not containing links, which we specified to avoid abrupt queries that simply relied on linked content.

To address this data scarcity, we opted to generate synthetic training datasets for the celebrity and landmark domains with the goal of constructing diverse, realistic queries varied in writing style, difficulty, and query length. We generate multiple types of queries

across three difficulty settings ("easy", "medium", and "difficult") and three length settings ("short", "medium", and "long") for a total of nine categories, each containing queries generated with the same difficulty and length settings.

Our first task was to collect real queries from TOMT-KIS to form our training dataset for the movie domain and our pools of few-shot examples for the celebrity and landmark domains. We implemented Borges et al.'s [3] pipeline for extracting answers from Reddit posts with GPT-3.5 Turbo and matching them to the titles of Wikipedia articles in the corpus using the Python library `difflib`. We adhered closely to their proposed pipeline aside from two modifications. First, we gave GPT-3.5-Turbo the option to respond with `"NO NAME"` in the title extraction stage to filter out posts without valid answers. Second, we confirmed whether the words in the extracted title were contained in the Reddit post's chosen answer to eliminate cases of hallucinations from GPT-3.5 Turbo. Similarly, we checked that words in the matched Wikipedia title were contained in the extracted title during the title matching stage.

With this pipeline, we obtained a training dataset of 71,449 query-answer pairs about movies from an initial collection of 127,283 posts in TOMT-KIS that contained either "movie" or "film" in their titles. From an initial group of 3,051 celebrity-related posts, which we identified by searching for tags such as `[Celebrity]` or `[Actor]` in the post title, we obtained 1,387 celebrity queries with answers. Similarly, we used our pipeline on 117 landmark-related posts we identified in TOMT-KIS by searching for tags such as `[Landmark]` or `[Place]`. We also manually searched over the r/TipOfMyTongue subreddit[1] for recent landmark-related posts (c. 2023–2024) to feed into our pipeline or process by hand. In total, we collected 57 landmark queries with answers and 82 without answers. Discounting queries reserved for our test sets, we obtained a total of 1,187 celebrity queries and 114 landmark queries to use as few-shot examples.

Following collection of few-shot examples, we obtained the names of various celebrities and landmarks to serve as the answers for our synthetically generated queries. We scraped names from 96 Wikipedia articles containing lists of celebrities or notable people. We additionally scraped names of highly popular individuals whose articles appeared in the top 5,000 Wikipedia articles for each month from December 2015 to July 2024. To identify if an article was about a person, we checked whether the phrase "born" and a month or an open parenthesis followed by a month appeared in the first few sentences of the article body. In total, we collected 36,491 unique celebrity names using these two approaches. We also collected 18,449 unique landmark names from 296 Wikipedia articles containing lists of landmarks and places using the same webscraping techniques.

We decided to effectively double the size of our training datasets by randomly assigning a third of the answers to one of the nine total categories, a third to two categories, and a third to three categories. Repeated answers were sent to different categories to ensure generation of different types of queries for each occurrence. For our synthetic celebrity dataset, we utilized 8,000 answers per category to achieve a total of 72,000 answers. For our landmark dataset, we utilized 4,000 answers per category to achieve a total of 36,000 answers.

```
Your task is to generate a question for the answer
of "{answer}" from the perspective of someone who is
trying to remember the name of this {domain} they
have forgotten.  You must use the exact same writing
style as the questions below.  Your question cannot
contain the answer.  {difficulty_requirement} Your
question must have a similar length to Question #3,
which has {L} words.  You will be penalized if your
question is shorter than {max(0, L - 10)} words or
longer than {L + 10} words.

Answer #1:  {few_shot_answer_1}
Question #1:  {few_shot_query_1}
...
Answer #4:  {answer}
Question #4:
```

Figure 1: Few-shot prompt for generating a query. Few-shot examples #2 and #3 omitted for brevity.

Finally, we generated synthetic queries to pair with these answers using GPT-4o mini. In addition to 3-shot prompting, we utilized custom prompts for each difficulty and query length setting as shown in Figures 1 and 2. To control for the difficulty of queries, we substituted the `{difficulty_requirement}` specification in the prompt template (Figure 1) with one of the three options shown in Figure 2. We observed that the difficulty of ToT queries often corresponds to their level of vagueness. Thus, we prompted GPT-4o mini to generate specific and helpful queries for the "easy" setting, vague and unhelpful queries for the "difficult" setting, and a mix of both for the "medium" difficulty setting. We validate that our approach results in measurable differences in difficulty between categories in Section 3.1.

While we explicitly encouraged GPT-4o mini to generate false memories for the "difficult" setting as

[1] https://www.reddit.com/r/tipofmytongue

```
Easy: The person in your question remembers many
details about the {domain} they have forgotten, so
your question must include multiple specific facts
and helpful details. You will be penalized if the
question is vague or difficult to answer.

Medium: The person in your question remembers
only a few details about the {domain} they have
forgotten, so your question must include one or two
specific and helpful facts but also some vague and
unhelpful details.

Difficult: The person in your question remembers
almost nothing about the {domain} they have
forgotten, so your question must only include
unhelpful vague details and false memories. You
will be penalized if the question is too specific or
easy to answer.
```

Figure 2: Difficulty requirement prompts by category.

seen in Figure 2, we also did not fact check any of the generated queries. Thus, we utilized both intentional and unintentional hallucinations from LLMs to better simulate the ToT task.

To encourage GPT-4o mini to emulate the writing style observed in real celebrity or landmark queries, we performed 3-shot prompting. Few-shot examples were randomly drawn from one of three pools created by sorting all few-shot examples by query length and dividing the list into thirds. We sampled from the lower third of the length distribution for "short" queries, the middle third for the "medium" length queries, and the upper third for "long" queries. We prompted GPT-4o mini to generate its query to have the same number of words as the third few-shot example query. We found the most success in generating queries that met a length target when the given examples had the same or similar lengths to the target.

Queries	**Movies**	**Celebrities**	**Landmarks**
Total	74,449	75,561	36,320
Real	71,449	1,187 (x3)	32 (x10)
Synthetic	—	72,000	36,000
Provided	300 (x10)	—	—

Table 1: Breakdown of the different types of queries present in the training datasets for each domain. Upsampling by a factor of 3 is denoted by (x3).

We provide a breakdown of our single-domain training datasets in Table 1. We used the 300 queries in the `train` and `dev1` sets provided to us by the TREC ToT organizers in our movie dataset. We upsampled these queries by a factor of 10 to give them greater weight during training. We upsampled the real queries in the celebrity and landmark domains to encourage our DPR models to improve upon both real and synthetic queries during training.

We also combined these individual datasets into our full training dataset with 74,449 movie queries, 75,561 celebrity queries, and 72,640 landmark queries for a total of 222,650 queries. We obtained the 72,640 landmark queries by upsampling our landmark dataset by a factor of 2. This allowed us to achieve approximately equal representation for each domain. We used our full dataset to train a single "general" DPR model to handle all domains and provide more training details in Section 3.2.

2.2 Routing

In addition to training a single general DPR model on all domains, we developed an approach to route a query to one of multiple "expert" DPR models. Each expert model was trained on a single domain's training set for over three times as many epochs compared to the general model. In this section, we refer to the general model as DPR-All and the expert models for the movie, celebrity, and landmark domains as DPR-M, DPR-C, and DPR-L, respectively. We investigate both approaches to determine their relative efficacy for the ToT task across domains.

Our preliminary experiments demonstrated that DPR-M and DPR-L outperformed DPR-All by a small margin on our internal test sets. On the other hand, DPR-All outperformed DPR-C by a noticeable margin. These results may be due to the slight overrepresentation of the celebrity domain in our multi-domain training dataset, which contains about 1k more celebrity queries than movie queries and roughly 3k more than landmark queries. Therefore, we concluded that training a model on a single domain offers increased performance for underrepresented domains in DPR-All's training dataset. However, the general model is comparable and may offer greater robustness due to the increased diversity of its training queries.

To ensure our routing approach directs queries to the best possible model, we route movie and landmark queries to their respective expert models while sending celebrity and "uncertain" queries to the general model. We used GPT-4o mini as our router and iterated on our prompting strategy until we achieved near-100% accuracy in our experiments with only a small amount of "uncertain" classifications. Our final router prompt is included in Figure 3.

As shown in Figure 3, we used the labels of `MOVIE`, `PERSON`, and `PLACE` to refer to the movie, celebrity, and landmark domains respectively, and remove ambiguity inherent in the definitions of celebrity and

```
You are given a question where someone is trying
to remember the name of either a person, movie, or
place they have forgotten.

Respond with PLACE if the person asking the question
is describing a place.  Respond with PERSON if they
are describing a person.  Respond with MOVIE if they
are describing a movie.  If an actor from a movie is
being described, respond with PERSON.

If this task was difficult due to ambiguity or
multiple categories being present in the question,
respond with your best guess along with the phrase
UNCERTAIN.
```

Figure 3: Domain router prompt.

landmark. As celebrity queries about actors were overwhelmingly classified as `MOVIE` by GPT-4o mini, we added a sentence to handle this case. We also found that requiring the model to respond with both its best guess and `UNCERTAIN` for ambiguous cases instead of only `UNCERTAIN` stopped it from overusing this tag.

2.3 Reranking

For reranking, we develop an initial pointwise reranking stage to pair with Borges et al.'s [3] round-robin approach. To rerank the results retrieved by our DPR models, we adopt Borges et al.'s successful round-robin strategy of using batches during listwise reranking. We used GPT-4o mini as our reranker to balance cost and efficacy of the process. While Borges et al. [3] ranked 100 results at once with GPT-4, we limited the reranking process to a maximum of 20 results at a time to reduce costs. For larger inputs, we observed that GPT-4o mini's performance would quickly degrade and become prone to meaningless repetition of variations of the same title or name.

More generally, let N be the total number of results to be reranked in a round-robin manner and B be the number of batches. To evenly distribute the N results among the B batches, the ith result is sent to batch $i \mod B$ at position $\lfloor \frac{i}{B} \rfloor$. Then, listwise reranking is performed on each batch of $\frac{N}{B}$ results and the top $\frac{N}{B^2}$ results are collected from each batch to create a final batch. Finally, the final batch is reranked to determine the top $\frac{N}{B}$ spots in the final ranking. For the remaining results, the ith result in batch j is sent to position $B \cdot i + j$ in the final ranking.

For our runs that only used listwise reranking, we employed a smaller-scale version of Borges et al.'s approach that only reranked the top 100 results of the 1000 results retrieved by our DPR models. We used 5 batches of 20 results (i.e., $N = 100$ and $B = 5$).

To expand the scope of our reranking to all 1000 results retrieved by DPR, we developed a pointwise reranking stage to select the 100 most relevant results according to scores generated by GPT-4o mini. These results were then passed on to our top 100 round-robin reranker to produce the final ranking.

We first distributed the 1000 results retrieved by our DPR models into 50 batches of 20 results (i.e., $N = 1000$ and $B = 50$). For each batch, we prompted GPT-4o mini to generate relevance scores on a scale of 1 to 10 for 20 names as shown in Figure 4.

```
You are given a question and a list of 20 names of
either all movies, celebrities, or landmarks.  For
each name in the list, respond with the line number
and the name followed by a score from 1 to 10 on how
likely this name is the answer to the question.
```

Figure 4: Pointwise reranking prompt.

In our experiments, we found the most success when using an open-ended prompt that did not give explicit scoring criteria beyond setting a range. We observed that GPT-4o mini tended to give the majority of results low scores from 1 to 4 and would give high scores of 7+ to a small group of results that it deemed the best. We found this behavior to be helpful as it prevented the upper end of the score range from becoming saturated.

After generating scores with GPT-4o mini for all 1000 results, we sorted the results in a pointwise fashion in descending order to form the ranking S. Ties were broken by choosing the result that appeared earlier in the original DPR ranking. We passed the top 100 results in S to our round-robin reranker and assigned its output to the top 100 slots of the final ranking. We filled in the remaining slots with the results in S that appeared after the top 100.

3 Implementation

3.1 Data Statistics

We provide example queries from our synthetic training datasets in Table 2. We can observe characteristics of real ToT queries in these synthetic queries such as the use of an informal writing style, recollections of memories, and expressions of uncertainty. Note that the two "easy" queries in Table 2 contain specific and helpful clues while the "medium" and "difficult" queries are much more vague (i.e., the intended outcome of our query generation prompts in Figure 2).

We evaluate each category of our synthetic datasets with the BM25 retriever and provide plots of the

	Example Synthetic Queries By Category
Short Length + Easy	Famous athlete known for javelin throwing, has a unique record with a notable technique, competed in the 1980 Olympics. Who is he?
Short Length + Diff.	some kind of historical building I vaguely remember from a trip with my family ages ago. I think it was in Europe, maybe in a small town? It had a lot of beautiful gardens around it, and I feel like there were some old paintings or something inside. I keep confusing it with other castles, but this one definitely stood out. Any ideas? Thanks!
Med. Length + Med. Diff.	Modern sculptor known for his large-scale, geometric metal pieces. I think his work often incorporates light in interesting ways, and he might have been influenced by contemporary architecture or mid-century design. He created some installations for public spaces, but I'm really struggling to recall his name. He doesn't come to mind easily, so any hints would help!
Long Length + Easy	A luxury accommodation near a famous golf course overlooking stunning coastal views I visited this beautiful place a few years back, and I've been trying to recall its name ever since. It's located near Pebble Beach on California's Monterey Peninsula, surrounded by picturesque landscapes, featuring lush green fairways and majestic cypress trees. The building itself had a classic, elegant design with a cozy fireplace in the lobby. I remember enjoying a superb meal at their highly-rated restaurant that focused on local seafood and fine wines. There was also an inviting outdoor patio where we could soak in the ocean breeze. If anyone can help jog my memory about the name of this amazing lodge, that would be fantastic!

Table 2: Example synthetic queries by category from the celebrity and landmark training sets.

BM25 Recall@1000 scores per category in Figure 5. We use this metric as a proxy for the average difficulty of a category's synthetic queries because it tests the most basic ability of queries to simply be retrieved at all. Under this framework, we validate that our query generation approach succeeded in creating noticeable differences in difficulty between categories with different difficulty settings.

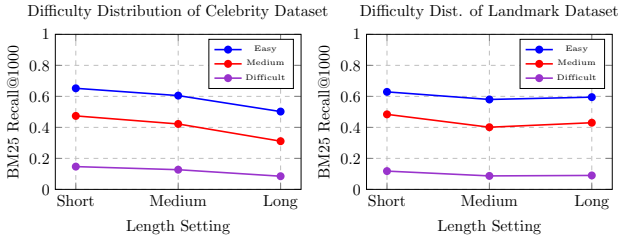

Figure 5: Plots of the difficulty distribution across categories for the celebrity and landmark datasets. Each one of a dataset's nine categories is represented a point, and points corresponding to the same difficulty setting are connected in a line.

For categories with the same difficulty setting in the celebrity dataset, we observed an increase in difficulty as query length increased (Figure 5), which is consistent with Arguello et al.'s [1] findings in TREC ToT 2023. For the landmark dataset, we observed an exception to this trend, possibly due to biases affecting difficulty that arose from the extremely small size of the landmark few-shot example pools.

In addition, we include the average query word length for each category of our synthetic datasets in Table 3. We designed our query generation process to randomly sample each synthetic query's target length from few-shot pools of actual ToT queries. Thus, the average query lengths in Table 3 are consistent with the natural length distribution of real ToT queries in the celebrity and landmark domains.

	Celebrity Dataset			Landmark Dataset		
	Short	Medium	Long	Short	Medium	Long
Easy	40.95	77.33	135.70	53.92	90.89	134.28
Medium	40.08	75.74	135.44	52.29	90.05	134.21
Difficult	41.05	79.18	142.31	54.55	93.35	139.20

Table 3: Average number of words per query for each category in the celebrity and landmark datasets.

3.2 Experimental Setup

We used the Tevatron framework created by Gao et al. [5] to train our DPR models, which we initialized from the `bert-base-uncased` checkpoint [6]. We utilized a learning rate of 2e-5, a batch size of 32, a maximum query size of 128 tokens, and a maximum passage size of 512 tokens. We used the sole answer for each query as its positive document and provided two negative documents per query during training.

We used the *baseline_bm25* code provided by the TREC ToT organizers to run BM25 on our training queries [1]. We then chose the two highest ranked non-answers according to BM25 as each query's negative documents. We also employed the organizers' *baseline_gpt4_db* code to match our reranked titles to document IDs in the corpus [1].

DPR models handling a single domain were trained for 10 epochs. Our DPR model handling all three domains together was trained for 3 epochs to maintain approximately equal overall training time.

4 Results

We submitted three runs to TREC and share evaluation results on the official TREC 2024 ToT test set in Table 4. We used our "routing" retrieval approach from Section 2.2 in the `dpr-router-lst-rerank` run and our "general" model approach described at the end of Section 2.1 in our other two submitted runs. We used our combined pointwise and listwise reranking approach in the `dpr-pnt-lst-rerank` run and only performed listwise reranking in our other two runs as described in Section 2.3.

To compare our two retrieval approaches, we examine `dpr-lst-rerank` and `dpr-router-lst-rerank`, which differ in only the retrieval method used. As both runs displayed similarly high Recall@1000 performance, the two retrieval approaches share similar recall performance with reranking only slightly

Table 4: Results of the three runs submitted to TREC.

Run	NDCG@10	NDCG@1000	MRR@1000	Recall@1000
dpr-pnt-lst-rerank	0.5708	0.6049	0.5482	0.8417
dpr-lst-rerank	0.5010	0.5424	0.4860	0.8400
dpr-router-lst-rerank	0.4664	0.5098	0.4515	0.8333

changing the Recall@1000 score if at all. The "general" model approach resulted in higher NDCG performance for `dpr-lst-rerank` compared to the "routing" approach for `dpr-router-lst-rerank`, suggesting the advantage of the former retrieval approach over the latter.

We compare our two reranking approaches by inspecting `dpr-pnt-lst-rerank` and `dpr-lst-rerank`, which differ only by the addition of the initial pointwise reranking stage we propose. We observe a significant jump in NDCG@1000 performance from 0.54 in `dpr-lst-rerank` to 0.60 in `dpr-pnt-lst-rerank`, demonstrating the effectiveness of pointwise reranking for the ToT task.

5 Conclusions

This paper described our submissions to the TREC 2024 ToT track. We proposed a two-stage pipeline of DPR-based retrieval followed by reranking using GPT-4o mini, swapping in two different retrieval strategies and two reranking approaches. We developed a few-shot prompting approach using GPT-4o mini to reliably generate synthetic training queries of multiple difficulties and lengths. We observed that training a "general" DPR model to handle queries from all domains outperformed routing queries to single-domain "expert" DPR models. We found that our combined pointwise and listwise reranking approach improved performance when compared to our runs that only utilized listwise reranking. Possible future extensions of these retrieval and reranking strategies include using them upon a greater number of domains or with a different set of models.

Acknowledgments

We gratefully acknowledge the support and feedback provided by members of the Yale NLP Lab throughout this project. We are thankful to the Yale College Dean's Office and the YES Scholars program for funding this research and making it possible.

References

[1] Jaime Arguello, Samarth Bhargav, Fernando Diaz, Evangelos Kanoulas, and Bhaskar Mitra. Overview of the TREC 2023 Tip-of-the-Tongue Track. In *The Thirty-Second Text REtrieval Conference (TREC 2023)*. NIST, 2024. `https://trec.nist.gov/pubs/trec32/papers/Overview_tot.pdf`.

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 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. ACL, 2020. `https://doi.org/10.18653/v1/2020.emnlp-main.550`.

[3] Luís Borges, Jamie Callan, and Bruno Martins. Team CMU-LTI at TREC 2023 Tip-of-the-Tongue Track. In *The Thirty-Second Text REtrieval Conference (TREC 2023)*. NIST, 2024. `https://trec.nist.gov/pubs/trec32/papers/CMU-LTI.T.pdf`.

[4] Maik Fröbe, Eric Oliver Schmidt, and Matthias Hagen. A Large-Scale Dataset for Known-Item Question Performance Prediction. In *QPP++@ ECIR*, 2023. `https://ceur-ws.org/Vol-3366/paper-03.pdf`.

[5] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Tevatron: An Efficient and Flexible Toolkit for Dense Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3120–3124, 2023. `https://doi.org/10.1145/3539618.3591805`.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. ACL, 2019. `https://doi.org/10.18653/v1/N19-1423`.