Team CMU-LTI at TREC 2023 Tip-of-the-Tongue Track

Luís Borges[1,2], Jamie Callan[2], and Bruno Martins[1]

[1] Instituto Superior Técnico and INESC-ID, University of Lisbon, Portugal
[2] Carnegie Mellon University, USA

Abstract. This paper describes our submissions to the 2023 TREC Tip-of-the-Tongue (ToT) track. We opted for the common retrieval methodology of a Recall oriented first-stage retrieval, followed by the use of a more accurate re-ranker model. For first-stage retrieval, we considered a DPR retriever either aggregating the passages from the documents, or matching different parts of the queries against the abstract sections of the Wikipedia articles that describe the movies. Re-ranking was delegated to a Large Language Model (LLM) in a zero-shot setting, taking as input the movie titles from the first stage of retrieval. Results attest to the effectiveness of the proposed approach with the best run achieving an NDCG@1000 of 0.55.

1 Introduction

The challenges posed by Tip-of-the-Tongue (ToT) queries, and the shortage of previous work in this direction, motivated a TREC track on tip-of-the-tongue retrieval, with the release of ToT queries and a Wikipedia corpus. This paper describes our TREC submissions for this task, employing the typical retrieval methodology of leveraging a quick and Recall oriented first-stage retriever, and then using a more accurate, yet slower, model to re-rank the documents for each query, with the intent of maximizing retrieval accuracy.

For first-stage retrieval, we split the query into multiple sentences and search each sentence of interest with a retriever over the Wikipedia sections which are more likely to match the sentence. For example, a sentence stating the movie belonged to the horror category should be searched with a retriever over the Abstract sections of the Wikipedia corpus. We do this because not all sentences in a query should be important to find the relevant document, and the query could contain an excess of diverse or irrelevant information. For the purpose of the TREC submissions, we only consider a first-stage retriever over the abstract sections of Wikipedia. For re-ranking, we leverage the zero-shot capabilities of LLMs and feed GPT-4 [3] with large amounts of movie titles for simultaneous re-ranking, before exhausting any context size limits.

Our submissions involve a combination of two first-stage retrievers and two different ways of using the LLM. The first-stage retrievers are chosen as a simple DPR Max-P model [1], i.e., aggregating passage scores and choosing the maximum score to assess the entire document, and a DPR model which matches

query sentences with the Wikipedia abstracts. LLM re-ranking is done either with the top 100 movie titles from the first stage, or with the top 1000 titles over windows of size 100, leveraging a round-robin strategy to equally distribute the expected difficulty of the several windows.

2 Method

We first describe the two first-stage retrievers that were considered in the submitted runs, namely DPR Max-P and the abstract retriever, and then move along to the two use cases of GPT-4 for re-ranking, which are a top 100 re-ranker, and a top 1000 round-robin approach.

2.1 First-stage Retrieval

Our first attempt at a first-stage retriever employs a DPR retriever where a BERT-based model encodes the full query into a dense vector, another encoder encodes the document, and relevance is computed with a dot product between both representations. We consider DPR in a Max-P setting, given that the documents are often longer than 512 sub-words. In this setting, a relevance score is computed between the query and all the passages that compose the document. The score between the query and the document is the maximum score among the individual passage scores. A total of 1000 documents are retrieved for each query. This first retriever is hereby denoted as DPR Max-P.

Our second retriever tackles the long nature of Wikipedia documents, ToT queries, and the potentially irrelevant and diverse information within the queries. We focus query/document matching on specific parts of the texts that should be more useful for computing the relevance scores. We found that matching some of the query sentences solely with the abstract sections of Wikipedia lead to increased Recall over other Wikipedia document sections, or the full document. Formally, every query is split into multiple sentences, and those sentences are annotated with one or more codes that detail the kind of information that the sentence is providing (e.g., opinion, hedging, plot, scene, actor, etc.). These annotations were given by the organizers of the track. For every sentence in a query, if its codes are associated with the abstract section of Wikipedia (i.e., if the information from the sentence is typically contained in the abstract section of a document), then that sentence is encoded and searched over the Wikipedia Abstract collection. Once this process is complete, multiple ranked lists are generated, with one ranked list for every query sentence of interest. In order to compute the final first-stage ranking for downstream re-ranking, the score of a document is considered as the maximum score of that document among all the ranked lists. This retriever is hereby denoted as the abstract retriever. Again, the abstract retriever is implemented with a DPR Max-P model, and 1000 documents are retrieved for each sentence, and in the end, for each query.

2.2 Zero-shot Neural Re-ranking with LLMs

In order to perform re-ranking, an LLM is given the query and a numbered list of movie titles, and asked to return another ordered list of the same movie titles, sorted according to the likelihood that they refer to the given query. One of our two LLM use cases includes feeding the LLM with 100 movie titles, i.e. a relatively small amount of movies. This is faster, cheaper, and requires the LLM to process less information, compared to a re-ranking depth of 1000, which is the output length from the first stage of retrieval.

Our second LLM approach re-ranks the whole 1000 movies from first-stage retrieval in a round-robin fashion. Since feeding all of the 1000 movie titles to the LLM was impractical and potentially containing a lot of distracting information, we can instead move a sliding window over the 1000 movies, performing local re-ranking with the LLM over the document titles in the window, and sliding the window until all 1000 titles are processed. This allows re-ranking documents that are deeper in the initial ranking list, although the amount of positions that a relevant document can climb after re-ranking is limited to the window size. There is also an imbalance in the difficulty of the different windows, since the first windows are more likely to be difficult than most of the remaining windows. Ideally, for each window the LLM should process a set of movies with equivalent difficulty, in order to chose a set of movies as the highest ranked in that window. The selected movies could then be aggregated to form another set of movies, of higher Recall, which would again be re-ranked with the LLM, this time in one call only. This is the idea behind our round-robin approach. In order to re-rank 1000 movie titles, 10 groups of 100 titles are created. Titles are assigned to groups in an alternating fashion, i.e., the last digit of the original document ranking is the group ID. The LLM then re-ranks each group. Another ranking is generated by extracting the top 10 titles from the groups into a set of 100 movies. Finally, the aforementioned set of movies is again re-ranked by the LLM, generating a final ranking.

3 Datasets and Methodology

Unlike re-ranking, where the LLMs are applied in a zero-shot manner, our first-stage retrievers require an adequate amount of supervised data for model training. The amount of TREC ToT training queries was very small, limiting the quality of supervised retrievers trained solely on these queries. This motivated us to curate our own data, based on an existing Reddit ToT dataset [2]. The curation process is detailed in this section, together with the TREC queries and document collection. A subsection focusing on the methodology employed in this work concludes the section.

3.1 Curating the Training Dataset

The training dataset derives from a Reddit dataset for tip-of-the-tongue queries, extracted from the Reddit r/tipofmytongue subreddit. This dataset contains

about 1.3 million ToT queries, where around 47% are tagged as solved. Of those 1.3 million queries, 226,071 queries belong to the intended movie domain, and 52% of those 226k queries are annotated with a gold answer. Those 118k annotated movie queries were the ones considered for further processing.

The gold answers are given in natural language, with no direct mapping between natural language answers and Wikipedia titles. We therefore require a way of mapping an answer to a Wikipedia document from a Wikipedia corpus. The Wikipedia corpus is the English Wikipedia dump [3] on Huggingface, with identifier *20220301.en*, which consists of around 6.5 million documents. The mapping of Reddit answer to movie title was done by using the OpenAI API to prompt GPT-3.5 [4], asking for a movie title from the input answer. Once the LLM output a movie title, we used the difflib Python package to search for candidate titles among the Wikipedia movies, and associated the ToT query with the most similar candidate document title, given a minimum similarity threshold of 0.8. In the end, we end up with a collection 83,914 movie queries, each associated with the relevant Wikipedia document. This adds up to the 150 training queries released by TREC, leading up to a total training dataset size of 84,064 queries, together with the respective relevant documents.

3.2 The TREC Evaluation Dataset

The TREC organizers released 150 queries each for training, model development, and later, model testing. The organizers also released a smaller Wikipedia corpus, comprised of 231,848 documents. The 450 released queries are annotated at sentence level, with several types of codes. Examples include the *category* annotation, associated with a sentence that describes the movie category, the *plot* annotation, attached to sentences that describe the movie plot, or the *hedging* code, used when the user expresses uncertainty. The full set of codes can be found in the TREC website for the tip-of-the-tongue task [4].

3.3 Methodology

Data curation involves GPT-3.5, given its cheaper costs. Training DPR Max-P requires pairs of queries and movie titles from Wikipedia. The prompt to extract the movie titles from the Reddit queries is "The answer is {*answer*}. Give me the title of the movie referenced in the answer. Write the fewest words possible.". Training the Wikipedia section retrievers requires training data to be annotated with the codes from the previous section. For the TREC queries, those human annotations were released, so the remaining 84k queries needed to be annotated as well. To do this, GPT-3.5 is prompted to generate annotations for the Reddit training queries, based on the definition of the annotations from the TREC website, and considering as in-context examples three randomly sampled annotated queries from the 150 training queries given by the TREC organizers.

[3] https://huggingface.co/datasets/wikipedia/viewer/20220301.en
[4] https://trec-tot.github.io/guidelines

Table 1. The results of the four runs submitted to TREC, with the official metrics of NDCG and Recall, over the released 150 TREC testing topics.

Method	NDCG@1000	NDCG@10	Recall@5	Recall@1000
DPR Max-P	0.374	0.307	0.346	0.800
+ LLM Top 100	0.507	0.463	0.487	0.800
+ LLM Top 1000, round-robin	0.555	0.517	0.547	0.800
Abstract Retriever	0.076	0.026	0.033	0.360
+ LLM Top 100	0.153	0.123	0.133	0.360
+ LLM Top 1000, round-robin	0.258	0.248	0.273	0.360

Once the dataset is created, supervised training is done with DPR, instantiated with the *bert-large-uncased* checkpoint. Training is done for 10 epochs, with a learning rate of 2e-5, a batch size of 128, in-batch negatives, and a passage size of 512 subwords. The loss function is a contrastive loss, maximizing the score of the positive document against the in-batch negatives.

The baseline DPR models are trained with (query, relevant document) pairs, with the full queries and documents. For the Wikipedia abstract retriever, the training data consists of (sentence, relevant document) pairs, where the sentences are chosen to be the ones that are annotated with codes compatible with the abstract section of Wikipedia. Specifically, those codes are *temporal*, *category*, *genre audience*, *genre traditional tone*, *origin movie*, *origin language*, *production visual*, and *release date*.

Re-ranking is then called via the OpenAI API to GPT-4. In order to re-rank a window of 100 movie titles, the following prompt is passed to the API: "I am going to give you a question and a list of movies. Re-order the movies according to the likelihood that the question refers to the movie. Format the answer as a numbered list of 100 movies. Keep the same movie names. QUESTION: {*question*} MOVIE LIST: {*ordered movie list*}". Once the movie names are returned, a direct mapping is made between the returned title strings, and the previous titles from first-stage retrieval. In case of a mismatch, the difflib Python library is once again called to find the closest string.

4 TREC Submission Results

We submitted four runs to TREC, consisting of the combinations of two first-stage retrievers, and two LLM re-ranking strategies. The first-stage retrievers consisted of the abstract DPR retriever, which scored the highest Recall in our development experiments, and DPR Max-P retriever, with less Recall, but higher accuracy. Re-ranking with the LLM happened either over a cutoff of 100 or 1000 movies. Table 1 presents the results. The LLM was able to accurately perform re-ranking, regardless of depth. The abstract retriever resulted in a poor Recall over the test queries, despite out encouraging results over the development set. The abstracts of the relevant documents in the test set may have contained insufficient information to accurately match the query sentences. The query sentences may

Table 2. Comparison of our TREC scores with the median results from runs of other participants, over the 150 testing topics from TREC. W stands for win, L for loss, and T for tie. TMax stands for a tie with the max value, i.e., when our score equalled the maximum score from all participations. The two runs with DPR Max-P start with "dpr". LLM re-ranking over the top 100 ends with "-100", while the round robin over the top 1000 has its run ID ending with "-1k". The three reported metrics are the official metrics for the ToT track.

	NDCG@1000				NDCG@10				Recall@5			
Run	W	L	T	TMax	W	L	T	TMax	W	L	T	TMax
dpr-100	108	11	31	72	85	18	47	41	68	26	56	72
dpr-1k	109	11	30	80	99	13	38	77	77	22	51	82
abs-100	30	62	58	20	20	69	61	20	18	70	62	23
abs-1k	45	49	56	40	41	51	58	39	38	52	60	41

also have lacked enough context for accurate matching. Those issues were likely more present in the test queries, contrary to the development queries.

Table 2 compares the results of our four runs with the median, maximum, and minimum retrieval metrics from the runs of the other participating teams. From a total of 150 queries, for each run and retrieval metric, we count the number of TREC topics in which we scored higher, lower, or equalled both the median and the maximum score. The table highlights the solid performance of our runs with DPR Max-P, compared to the remaining runs. As previously mentioned, considering first-stage retrieval only with the abstracts showed to be subpar compared to the remaining teams.

5 Conclusions

This paper described our submission to the 2023 TREC ToT task. We considered a two stage architecture with an initial quick retrieval of candidate documents, and an accurate re-ranking of those same candidates. In the first stage of retrieval, we considered two choices, namely a simple DPR Max-P approach, and an abstract retriever which matched query sentences with the abstracts of the Wikipedia documents. We used GPT-4 for zero-shot re-ranking of the movie titles, in two ways. First, directly re-ranking the top 100 movie titles, and second, re-ranking 1000 movie titles through a sliding window and round-robin technique. Results on the TREC data showed DPR Max-P to be high in Recall both for development and test queries, but the abstract retriever only provided a high Recall on the development set. The LLM, for both use cases, was effective at re-ranking the movie titles in a zero-shot scenario. Our best submission, i.e. the conjunction of DPR Max-P with top 1000 GPT-4 re-ranking, consistently outperformed the median results for the individual TREC topics, and equalled the maximum NDCG in 80 out of those same 150 topics.

Acknowledgements This research was supported by the Portuguese Recovery and Resilience Plan through project C645008882-00000055, and through Fun-

dação para a Ciência e Tecnologia, with the Ph.D. scholarship SFRH/BD/150497/2019 under the CMU-PT Program, and with the INESC-ID multi-annual funding from the PIDDAC programme, corresponding to reference UIDB/50021/2020.

References

1. Reimers, Nils, and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." arXiv preprint arXiv:1908.10084 (2019).
2. Fröbe, Maik, Eric Oliver Schmidt, and Matthias Hagen. "A Large-Scale Dataset for Known-Item Question Performance Prediction." (2023).
3. OpenAI. "GPT-4 Technical Report" arXiv preprint arXiv:2303.08774 (2023).
4. Brown, Tom, et al. "Language Models are Few-Shot Learners." Advances in Neural Information Processing Systems 33 (2020): 1877-1901.