Tip-of-the-Tongue Retrieval leveraging Large Language Models

Aprameya Bharadwaj **Chantal D Gama Rose** **Dheeraj Pai**

João Coelho **Vinay Nair**

Abstract

In this project, we propose a pipeline for Tip-of-the-Tongue (ToT) Retrieval, focusing on ToT queries referencing movies. This task is characterized by long, verbose, and complex queries often containing uncertainty. Motivated by the challenges posed to current retrieval systems by such queries, we leverage LLMs as query decomposers, capitalizing on their ability to effectively break down intricate queries into smaller sequences that are easier to deal with. Furthermore, we study the usage of LLMs as zero-shot re-rankers, since they have shown to achieve state-of-the-art performance on other passage re-ranking tasks. We apply our techniques to the TREC-ToT [1] dataset, successfully improving the provided GPT-4 baseline by 65%. Code is publicly available here.

1 Introduction

Current Information Retrieval (IR) systems are tailored for situations where users can describe information needs precisely. The TREC-ToT track focuses on the retrieval of movies where the user poses a complex query, containing e.g. memories, uncertainty, comparisons, and exclusion criteria. Figure 1 shows an example of such a query, highlighting the challenging hedging sentences, and social cues that do not contribute to the context. Traditional IR systems often show limited effectiveness in this scenario [2], since most available benchmarks do not represent it, and resolving ToT queries may need more information than what is available on a common query-document pair [27]. Given the widespread popularity of movies and the consequent extensive amount of data available about them on the internet, the training of LLMs like GPT-4 can naturally involve a significant exposure to movie-related information, ultimately enhancing their ability to comprehend and resolve movie-based ToT queries.

[1] https://trec-tot.github.io/

> Movie from the early 2000s I believe about three people living in an apartment but never running into each other. One woman and two men are in the apartment. The woman is the realtor or owner of the apartment and at least one of the guys is a squatter/homeless. It is a Korean or Chinese film I think. Art house flick I think it won a few awards from film festivals like Cannes. Help if you can!

Figure 1: Tip-of-the-tongue query. Hedging sentences highlighted in yellow, and a social courtesy highlighted in green. Our model correctly identifies "Vive L'Amour" as the movie the user sought.

Hence, in this work, we aim to bridge the gap between traditional IR systems and the demands of ToT retrieval, by exploring possible approaches to integrate LLMs into the retrieval pipeline. First, in Section 3.1, we introduce our retrieval methods, including insights and solutions for the limited training data available for the TREC competition, and on the usage of LLMs as query decomposers, motivated by the fact that shorter and less-hedging queries should be easier for first stage retrievers to deal with [40, 28]. Considering that LLM-based re-ranking achieves state-of-the-art results in zero-shot passage re-ranking tasks [50], in Section 3.2 we detail our LLM-based re-ranking pipeline, which consists in top-100 listwise re-ranking followed top-10 pointwise re-ranking.

Our results (Section 5) show that our pipeline is able to surpass a GPT-4 baseline by 65%, showing the efficacy of the proposed methods. Finally, in Section 6, we provide a qualitative analysis of the results, concluding that the performance of the LLM as a re-ranker can be linked to popularity metrics, as such metrics can influence the amount of training data regarding a movie the LLM was exposed to.

2 Related Work

This section covers related work, initially presenting Large Language Models and Information Retrieval. The intersection of these two fields is motivated, and the Tip-of-the-Tongue Retrieval task introduced.

2.1 Large Language Models

Recently, there has been a notable shift in machine learning for natural language processing. This shift emphasizes using large pre-trained language models in various tasks, leading to significant progress across multiple fields, with models like LLaMa [51], PALM 2 [1], and GPT 4 [38] leading the way. Some fine-tuning approaches have demonstrated impressive zero-shot generalization to different tasks, e.g. RLHF [10], and instruct fine-tuning [52]. However, this still requires training models with a very high number of parameters. Multiple approaches have been proposed to overcome this issue, for example by introducing a new (small) set of parameters and freezing the remaining ones [14, 22, 25], by selecting only a small percentage of trainable parameters [55, 13], or by re-parameterizing the model [15, 9, 29]. Prompt-based methods [31], like few-shot prompting [7], are also a useful alternative, effectively exploiting LLMs generation capabilities without requiring further model fine-tuning.

2.2 Information Retrieval

Within the field of IR, neural methods currently achieve state-of-the-art results, particularly on tasks that include ranking short text passages according to relevance towards user questions [54]. Most approaches rely on Transformer-based neural language models, either following a bi-encoder architecture [45, 43] when performing top-N retrieval, or a cross-encoder architecture [35, 39] for top-N re-ranking.

Bi-encoders encode queries and passages independently [45, 43]. This allows the offline indexing of individual passage representations through methods that support the fast execution of maximum inner product searches such as FAISS [20]. Conversely, cross-encoders generate a representation for the concatenation of a query and a passage, directly modeling the interactions between these two components [35, 39]. The representation can then be used to predict a score for the passage to the query.

2.3 Intersection between IR and LLMs

Given the advancement on LLMs, research efforts have emerged to explore the synergies between retrieval and generation methods. For instance, retrieval-augmented generation [19, 23, 6, 44] leverages a retrieval model to get relevant information to contextualize a prompt, enhancing responses to factual questions. Another example is query generation, where an LLM is used to generate queries to be used as training data [5, 18] or for document expansion [37, 36, 12]. In this work, we follow studies that use LLMs as re-rankers through prompting [49, 41]. This can be done in a listwise fashion [33], pair-wise fashion [42], or pointwise fashion [21].

2.4 Tip-of-the-Tongue Retrieval

Tip-of-the-Tongue Retrieval is characterized by complex queries, often containing uncertainty and inaccuracies, comprising a user description of an item they have previously encountered but can't recall the identifier (Figure 1). Following the intuition that smaller and simpler queries should be easier for current retriever models to deal with, query decomposition techniques have been proposed for this task [28], where the initial query is broken into sub-queries focusing on different data types (dates, possible title mentions, directors, writers, etc ...), and fed to multiple specialized retrievers. In the TREC-Clinical Trials track [46], which deals with a different retrieval task with large complex queries, one approach named NQS [40] also followed decomposition, where a seq2seq model is used to generate queries given a large clinical trial description. To foster growth in this area, two datasets were created by scraping the /r/tipofmytongue subreddit [11, 4]. Experiments on these datasets show that standard retrievers struggle with ToT queries, motivating further research.

3 Proposed Methods

We address the ToT retrieval task through the implementation of a retrieval and subsequent re-ranking pipeline. Specifically, we start by using a system that, when provided with a query, can efficiently retrieve the subset of the most relevant movies from an extensive corpus, focusing on achieving a high Recall to ensure comprehensive coverage. Then, the retrieved top-N movies are re-ranked using a computationally more expensive method, which should generate superior relevancy estimates.

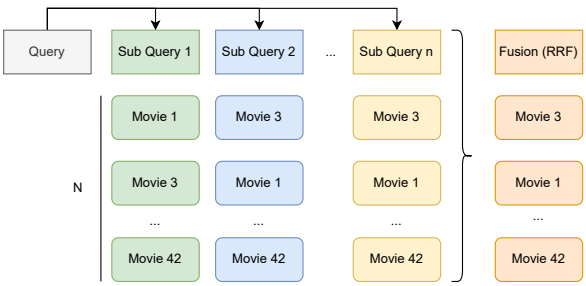

Figure 2: Query decomposition and fusion process. LLM decomposes query into n sub-queries. Top-N retrieval is performed individually, and the results are fused through RRF (Eq. 4).

3.1 First Stage Retrieval

The authors of the TREC track provide two retrieval baselines. A sparse one leveraging BM25 [47], and a dense one leveraging a `distil-bert` [48] model. Given a query q and a document corpus D, we can use the above methods to efficiently score every document $d \in D$, hence performing top-N retrieval on the whole corpus. BM25 scoring can be written as follows:

$$\mathrm{s}_{\mathrm{BM25}}(q,d) = \sum_{t \in q} \frac{\mathrm{tf}(t,d) \times \mathrm{idf}(t) \times (k_1+1)}{\mathrm{tf}(t,d) + k_1 \left(1 - b + \frac{b|d|}{\mathrm{avgdl}}\right)} . \tag{1}$$

In the previous equation, t is a term of the query, b and k_1 are hyperparameters, avgld is the average length of the documents in the collection, $\mathrm{tf}(t,d)$ is the frequency of term t within passage d, and $\mathrm{idf}(t)$ is the inverse document frequency for term t. Assuming we have a model f which maps an input sequence to an embedding space, dense retrieval scoring can be computed as follows:

$$\mathrm{s}_{\mathrm{dense}}(q,d) = \mathrm{sim}(f(q), f(d)) . \tag{2}$$

In the previous equation, sim is a similarity function between latent representations. In this work, we consider the cosine similarity. Next, we introduce training objectives and the techniques we employed to extend the dense baseline, namely by considering a self-supervised warm-up step of the dense retrieval model, more supervised data, and by leveraging query decomposition.

Table 1: Details of the training data used in this work. Lengths given in number of GPT-2 tokens.

	TREC-ToT	Reddit-ToT	Movie-ICT
Supervised	Yes	Yes	No
Queries	150	10777	161156
Movies	231852	14863	161156
Avg Query Length	181	144	116
Avg Movie Length	653	454	482

3.1.1 Training Data and Objectives

The training data provided for the TREC competition, hereby referred to as TREC-ToT, totals 150 queries, each labeled with a movie. The model is trained with a standard contrastive objective, aiming to minimize the following cross-entropy loss:

$$L = -\frac{1}{n} \sum_i \log \frac{e^{\mathrm{s}_{dense}(q_i,d_i)}}{e^{\mathrm{s}_{dense}(q_i,d_i)} + \sum_j e^{\mathrm{s}_{dense}(q_i,d_{ij}^-)}} , \tag{3}$$

where (q_i, d_i) are positive pairs, and $\{d_{ij}^-\}$ is the set of negative movies for query q_i, provided within the same training batch. The model provided by the authors uses BM25 to sample negatives in the first epoch, and self-hard negatives in a second epoch. Moreover, in-batch negatives are also used, i.e., a positive example associated with a query q_i within a batch is used as a negative example for all other q_j in the same batch.

While useful, the training data is limited. As such, we extend the training by using the movie corpus provided for the task in a self-supervised warm-up step. The training objective follows Equation 3, and training examples were sampled following an Inverse Cloze Task (ICT) [8] adapted for this domain, where given a movie textual description, a sentence is sampled and used as a query. The remainder of the sentence is labeled as a positive example for the query. We do not sample negatives for this task, instead relying on in-batch negatives alone. We refer to this data source as Movie-ICT.

Furthermore, we leverage the Reddit-ToT dataset [4] which provides supervised data scraped from the /r/tipofmytongue subreddit. We consider only the subset regarding movies, although there are also samples regarding books. The queries are similar to the TREC-ToT data but are not labeled with sentence-level annotations. Table 1 summarizes the datasets we used and their dimensions.

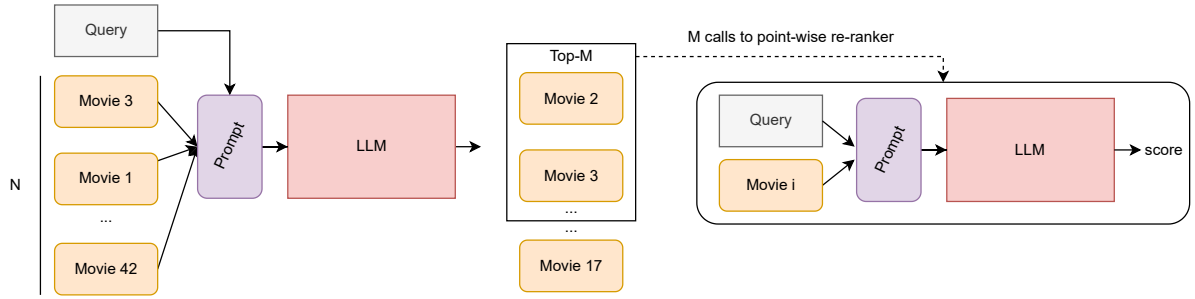

Figure 3: Re-ranking pipeline. LLM re-orders the top-N movie list. Point-wise re-ranker refines the top-M results

.

3.1.2 Query Decomposition

The original query in this task is much larger than queries traditionally posed to search systems. Retrieving the top-N movies directly from this query may be sub-optimal due to both its size and its complexity. To overcome this issue, we leverage query decomposition: each query q is decomposed into n sub-queries, (q_1,...,q_n). Then, each sub-query is used independently to retrieve the top-N movies. The n rankings are combined through Reciprocal Rank Fusion:

$$\text{score}(m) = \sum_{i=1}^{n} \frac{1}{k + \text{rank}_\text{i}(m)} \ . \qquad (4)$$

In the previous equation, k is a hyperparameter tuned to 60, m is a movie, and $\text{rank}_i(m)$ is its position in the ranked list generated by q_i.

We use sentence-by-sentence decomposition as provided by annotations in the dataset, and extend it by leveraging LLMs to generate the decompositions. This process is illustrated in Figure 2. To perform LLM decomposition, we start by cleaning social cues in the user input, e.g. "Help if you can!". Then, in order to remove other irrelevant snippets that do not contribute to the retrieval process, we prompt the LLM to decompose the queries into smaller independent sentences that only retain information that can help in movie title retrieval (A.1.1). Furthermore, the LLM was asked to retain important information from previous sub-queries in the later sub-queries, so as to ensure that each sub-query has as much useful detail as possible.

Decomposition is applied when doing inference for first-stage retrieval. For training queries, we'll only apply it to the 150 queries in the TREC-ToT dataset. Applying it to Movie-ICT wouldn't be sound because the unsupervised queries are sentences from movie descriptions, and applying it to Reddit-ToT would be too expensive (A.3).

3.2 Re-ranking

Supervised methods like cross-encoders excel at re-ranking tasks, but they require amounts of training data unavailable in this domain. Hence, we consider LLM-based zero-shot re-ranking, as it has achieved state-of-the-art results in other tasks [50].

We propose a two-step pipeline for LLM-based re-ranking, consisting of top-N listwise re-ranking (where the top-N was obtained from the retriever), followed by top-M (M < N) pointwise re-ranking. The idea is to have a broader, less detailed model reorder the results from the dense-retriever, and further improve the ranking by applying a more focused, detailed approach in a smaller sample of top movies. This pipeline is depicted in Figure 3.

For top-N listwise re-ranking, the model is given the query and a list of N titles to re-rank. Using only the title instead of full movie descriptions results on smaller prompts, which are cheaper and faster to execute. This, however, relies on the hypothesis that the LLM has seen enough movie-related data during training to be able to infer relevancy towards the query from the title alone. The prompt used for listwise re-ranking (A.1.4) considers integer identifiers for each movie in the top-N, and asks the model to re-order the integers.

Regarding top-M pointwise re-ranking, the model is used to attribute a score to a single (query, movie) pair. This allows for a deeper interaction between the query and the movie, where we are able to consider more information than just the title since we are only dealing with a single sample. This approach capitalizes on the better interaction to refine the model's understanding and provide better ranking judgments, potentially improving the overall results of the re-ranking process. The prompt used for pointwise re-ranking (A.1.5) asks the model to return a relevancy score for the (query, movie) pair being analyzed.

4 Experimental Setup and Evaluation

We use the data previously described and summarized in Table 1. Regarding evaluation data, we perform it on the development split of the TREC-ToT dataset, since the test set labels are undisclosed until the results of the TREC track are communicated. This split contains 150 evaluation queries.

Our dense retrievers are based on the `distil-bert` model, more specifically the `distilbert-base-uncased` checkpoint available at HuggingFace Transformers [53]. To train the dense retriever we leverage the SentenceTransformers framework [45]. Pyserini [26] is used for sparse indexes, and FAISS [20] for the dense ones. For GPT-4, we leverage the OpenAI API and the `gpt-4-1106-preview` model.

In order to stay within our budget, we perform top-100 listwise re-ranking followed by top-10 pointwise re-ranking. These values were chosen by inspecting the retriever recall cuts to ensure proper coverage, and considering the cost to run each re-ranking step.

Results are reported in terms of the official metrics of the task: Recall, and Normalized Discounted Cumulative Gain (NDCG) [17]. During first stage retrieval, our focus is on achieving a high Recall, since the performance on the re-ranking stage is inherently constrained by the Recall of the initial retriever. For re-ranking, the emphasis shifts to the main metric of the task, which is the NDCG@1000. We also compute the Precision@1 to assess the model's ability to present the most relevant movie as the top-ranked result, reflecting its immediate utility to users seeking highly pertinent answers.

5 Empirical Results

In this section, we discuss our set of experiments. Baselines provided by the organizers are replicated, and we detail the results of our proposed methods, which lead to a 65% improvement in terms of NDCG over the GPT-4 baseline.

5.1 GPT-4 Baselines

We start by examining the zero-shot performance of GPT-4 in this task. The TREC track organizers provide a GPT-4 baseline, where the entire query was fed to GPT-4 and it was prompted to return up to 20 movie titles, ordered by relevance to the query (A.1.2). We further consider a baseline where GPT-4 is prompted to return a single movie title (A.1.3).

Results are shown in the first group of Table 2, showing that GPT-4 is able to resolve 15% of the queries with a single guess. This value goes up to 18% when prompted to return more than just one movie. We attribute this change in precision mostly to the non-determinism of the model. These results highlight the difficulty of the task, and motivate the usage of a different strategy, as GPT-4 alone has subpar performance.

5.2 First Stage Retrieval

Our first experiments aimed to replicate the TREC track sparse (BM25) and dense (distil-bert) baselines. We were able to do so, replicating the provided NDCG@1000 value, and also computing other metrics of interest to us, as reported in the second group of Table 2. Specifically, we'll focus on the Recall@100 for the retrieval step, since the objective is to provide comprehensive coverage before the re-ranking step. In this case, we can see that the dense model achieves a stronger result, improving the BM25 baseline by 62%. However, both these methods are weaker than the GPT-4 baselines in terms of P@1 and NDCG. This indicates that training effective retrievers for this task may be hard due to the nature of the queries and the lack of training data, motivating a method that combines the strength of retrievers and LLMs.

5.2.1 Query Decomposition

In the third group of Table 2 we present the results for our experiments with query decomposition on top of a sparse retriever. These results show that query decomposition is in fact beneficial in terms of recall. Using sentence-level decomposition on top of BM25 yielded an 18% improvement in terms of Recall@100, while the best LLM query decomposition method yielded a 50% improvement.

For the LLM decomposition, we experiment with manually removing different sentences from the query that are annotated in the dataset as "social" (as in social niceties), "hedging" (sentences containing uncertainty), or "opinion" (heavily opinionated sentences). Although intuitively it makes sense that such snippets would not contribute to the retrieval process, we find that if we remove "hedging" and "opinion", the results worsen, most likely due to loss of information given the high incidence of such sentences in the data. On the other hand, removing "social" niceties helps Recall, since this information does not provide any useful context, and it also makes our prompts smaller.

Table 2: Results on the development splits of the TREC-ToT dataset (150 queries), including baselines, query decomposition for sparse and dense retrieval, variations in dense retrieval training data, and the final re-ranking pipeline on top of the best retrieval results.

	P@1	R@10	R@100	R@1000	NDCG@10	NDCG@100	**NDCG@1000**
GPT-4 zero-shot 1 movie	0.153	0.153	0.153	0.153	0.153	0.153	0.153
GPT-4 zero-shot 20 movies	0.180	0.276	0.320	0.320	0.233	0.240	0.240
BM25	0.080	0.093	0.180	0.407	0.086	0.104	0.131
distil-bert (TREC-ToT)	0.040	0.147	0.360	0.660	0.085	0.127	0.163
BM25 + sentence decomposition	0.046	0.100	0.213	0.473	0.060	0.082	0.114
BM25 + LLM decomposition	0.026	0.133	0.273	0.553	0.082	0.111	0.144
distil-bert + LLM decomposition	0.053	0.133	0.340	0.626	0.087	0.131	0.165
distil-bert (Movie-ICT + TREC-ToT)	0.027	0.153	0.353	0.706	0.086	0.127	0.170
distil-bert (Reddit-ToT + TREC-ToT)	0.073	0.233	0.433	0.696	0.145	0.185	0.217
distil-bert (Movie-ICT + Reddit-ToT + TREC-ToT)	0.046	0.213	**0.500**	**0.713**	0.126	0.184	0.210
GPT-4 top-100 listwise re-rank	0.266	0.406	0.500	0.500	0.340	0.359	0.359
GPT-4 top-100 listwise re-rank (shuffled top-100)	0.280	**0.420**	0.500	0.500	0.355	0.370	0.370
GPT-4 top-10 pointwise (re)re-rank (title only)	0.307	0.420	0.500	0.500	0.369	0.384	0.384
GPT-4 top-10 pointwise (re)re-rank (title + description)	**0.327**	0.420	0.500	0.500	**0.381**	**0.396**	**0.396**

We used the query decomposition methods on top of the distil-bert model, however, we notice that while there are gains in terms of NDCG and P@1, the Recall gets slightly worse. This is problematic because this metric directly bottlenecks the performance of the re-rankers.

5.2.2 Training Data

Regarding the effect of training data, results can be seen in the fifth group of Table 2. Applying either the supervised Reddit-ToT data or the unsupervised Movie-ICT before fine-tuning on the TREC-ToT dataset improves the Recall. Combining both delivers the stronger result, achieving our strongest performance of 50% Recall@100, a 38% improvement over the distil-bert baseline, which considers the model training only on the TREC-ToT data. It's also worth noting that the NDCG achieved by these retrievers is around 21%, which is better than the GPT-4 baseline with a single movie. The scarcity of large-scale training data has been acknowledged by other authors as one of the challenges for ToT Retrieval [4, 28]. Our results strengthen this fact, motivating efforts for both supervised and unsupervised data collection.

Finally, we note we did not perform query decomposition on queries from Reddit-ToT or Movie-ICT, since that would be too expensive and out-of-budget (A.3). However, given the results of its application in sparse retrieval, and also the improvements in NDCG for dense retrieval when considering the TREC-ToT queries alone, it may be a possible path to follow to extend this work.

5.3 Re-ranking

As previously stated, we start by conducting top-100 listwise re-ranking on top of the best first stage retriever in terms of Recall. Results on the sixth group of Table 2 show that this is highly effective, improving the retrieval NDCG@1000 by 76%, and the P@1 by 600%. This confirms the hypothesis that LLMs such as GPT-4 can internally contextualize themselves about movies only given their titles, even when being asked to resolve complex information needs such as ToT queries.

It is also worth noting that randomly shuffling the top-100 input movie titles did not have a negative influence on the result. In fact, it achieved a higher performance. While other authors argue and show that LLMs fail to properly model long sequences, losing context in the middle [30], we hypothesize that this result was due to (i) our prompts not being large enough for context to be lost (user query + 100 titles occupy on average 1.5k tokens; the GPT-4 version that we used supports up to 128k), and (ii) the non-determinism already observed and reported when zero-shot prompting GPT-4 with ToT queries.

Regarding pointwise top-10 re-ranking, we apply it on top of the best listwise result. The last group of Table 2 shows that pointwise re-ranking considering only the title already has a positive impact, improving the NDCG obtained in the listwise step by 7%. Further conditioning the prompt on the movie description leads us to our best result of 39.6% NDCG, a 65% improvement over the GPT-4 baseline provided by the TREC track organizers.

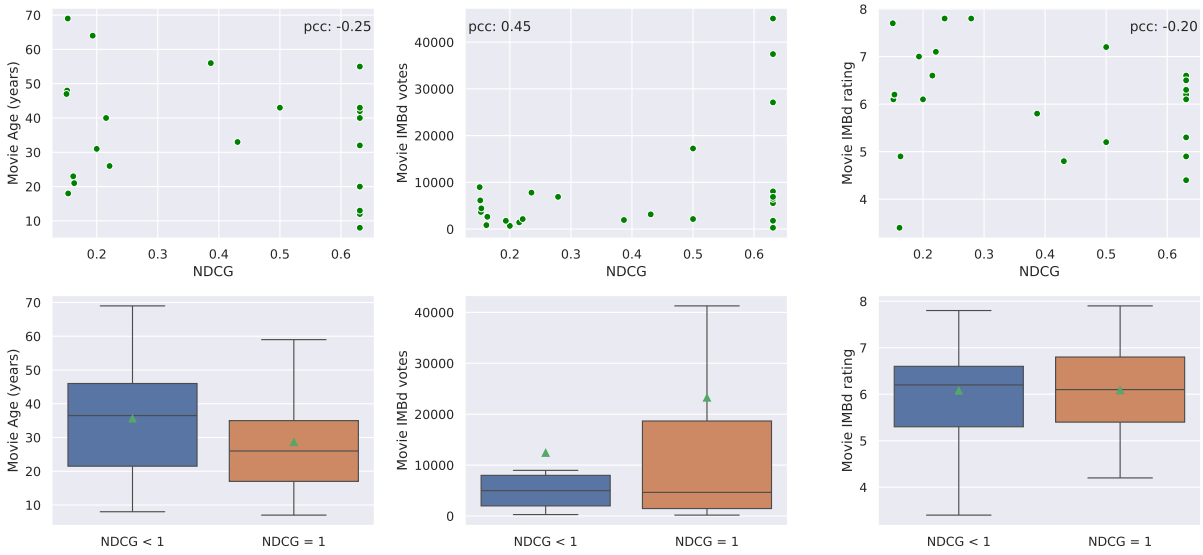

Figure 4: Top: Scatter plots depicting data points where GPT failed to resolve the query (NDCG < 1), illustrating the relationship between NDCG scores and movie-related metrics: age, IMBb votes, and IMDb rating. Each data point represents the movie labeled as relevant to the query, showing the distribution of NDCG scores in relation to these metrics. Bottom: boxplots comparing the distributions of the previous metrics for queries where GPT fails (NDCG < 1), and queries where GPT is successful (NDCG = 1).

This may indicate that while the LLM is able to contextualize itself with just the title, some movies can benefit from extra information, probably due to low popularity which can lead to less training data. To follow up on this intuition, Section 6 provides a qualitative analysis of the results, aiming to understand the impact of movie-related metrics (age, rating, etc...) on the performance of the LLM re-ranker.

As for the hallucinations of GPT-4 when performing re-ranking given our prompts (A.1.4, A.1.5), we first note that it adhered to the stipulated prompt output format for 98% of the calls (for both listwise and pointwise), hence making its usage possible due to an output that can be parsed deterministically. For listwise re-ranking, we associated integer identifiers with the title, asking the model to re-rank the identifiers. This was done to avoid hallucinations when writing title names, which could lead to mismatches between the titles in the output, and those in our corpus. Finally, we note that for the 2% of the cases that the model did not adhere to the stipulated output, it was due to the presence of hyperlinks in the text of the query, for which the model returned an answer informing that it can't open links, hence can't answer. Simply re-prompting without change solved this problem.

6 Qualitative Analysis

We investigated queries where the GPT-4 re-ranker was not able to fully resolve to the correct movie (i.e., where query NDCG < 1), aiming to discern underlying patterns that might contribute to diminished performance from an LLM. We hypothesize that the magnitude of online discussions about a movie directly impacts the availability of training data associated with it, subsequently influencing GPT-4's effectiveness in handling complex ToT queries linked to it.

Figure 4 shows NDCG distributions alongside correlations with specific metrics that act as proxies for a movie's online discussion volume: the movie's age, the number of votes on IMDb[2], and the IMDb rating. These metrics were chosen due to their potential influence on the extent of online discourse surrounding a movie. Older movies may have less contemporary discussions, while the number of IMDb votes reflects the level of engagement and interest within IMDb's community. Moreover, the IMDb rating offers an insight into the perceived quality of the movie.

Regarding the movie age, we first note that all movies linked to the queries analyzed in our study predate 2022, hence aligning with the training data available to the GPT-4 model used in this work

[2]https://www.imdb.com/

(`gpt-4-1106-preview`), which incorporates information up to April 2023. Analysis of the data reveals a weak negative correlation, implying that there might be a slight tendency for the NDCG to decrease with older movies, but the relationship is not very strong. Additionally, on average, we observed that the movies associated with queries that the model failed to resolve tend to be older compared to those it correctly identified. The number of IMDb votes follows a similar pattern, with a moderate positive correlation which indicates that NDCG may increase with the number of votes. Again, looking at the distributions, the movies associated with queries that were successfully resolved tend to have a higher number of votes, reinforcing the association between higher engagement, and the model's successful resolution of queries.

As for the IMDb rating, we observed very similar distributions for queries where the model succeeded versus those where it failed, indicating a minimal impact of IMDb rating on the model's accuracy in resolving queries. While IMDb ratings offer insights into public perception, it appears that the model's ability to accurately resolve queries isn't strongly influenced by this specific metric. This fact can further strengthen the claim that the model relies on the volume of discussions or engagement around a movie, as indicated by the number of IMDb votes, rather than the overall sentiment conveyed through ratings.

It's important to highlight that our analysis is based on a relatively small sample of movies. Consequently, the insights drawn from this data might not attain statistical significance. However, despite the limited sample, this study offers valuable initial perspectives on the potential relationships between movie attributes and GPT-4's performance as a re-ranker for ToT Retrieval. Further exploration with larger and more diverse datasets could yield more definitive conclusions.

7 Conclusion and Future Work

This paper presents an effective ToT Retrieval pipeline that combines the strengths of both a dense retriever and a GPT-4 re-ranker. Notably, employing either of the components in isolation resulted in subpar results, underscoring the benefits of their combined utilization.

The efficacy of our proposed methods can be summarized in the following key observations: (i) query decomposition improved first stage retrieval; (ii) collecting more training data for ToT Retrieval further boosts the retrieval results; (iii) GPT-4 displayed the ability to re-rank movies based solely on their titles; and (iv) pointwise re-ranking improved the listwise results, where conditioning the model on both title and description yielded stronger results than relying only on the title.

While the listwise re-ranking approach is restricted to domains where LLMs can contextualize a whole item based solely on a global identifier (for instance, famous movies, books, or songs), the remaining components of the pipeline should be useful for other retrieval scenarios with complex queries.

As for future work, we start by highlighting that diversifying reliance on GPT models mitigates the risk of a single point of failure. Using and fine-tuning open-source models specifically for this task could aid reproducibility and extension of this work [56, 32].

Moreover, given the constraint to our research budget, re-ranking was performed at limited depths (e.g., 100 and 10), hence consequently restricted by recall limitations at those levels. Scaling these experiments, for example by conducting full top-1000 pointwise re-ranking, could yield direct improvements. Still related to scaling, we could use a stronger base model when training the dense retriever, for instance one based on the SentenceT5 [34], which uses both the encoder and the decoder to generate the representations.

Finally, the token length of the dataset often exceeds the 512-token limit of retrieval models for both queries and documents. While not addressed in this study, employing long-sequence models during first stage retrieval could further enhance results [3, 24, 16].

8 Ethical Statement

With our project solely utilizing TREC data, our ethical focus centers on adherence to TREC's data usage guidelines. While the data is anonymized, we remain vigilant to prevent any ethical breaches, especially concerning the privacy of the users who posed the ToT queries that were used in this work.

We recognize an ethical concern regarding research reproducibility arising from the inherent non-deterministic nature of outputs generated by OpenAI's models, posing challenges in replicating results consistently.

References

[1] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *ArXiv*, abs/2305.10403, 2023.

[2] J. Arguello, A. Ferguson, E. Fine, B. Mitra, H. Zamani, and F. Diaz. Tip of the tongue known-item retrieval: A case study in movie identification. In *International Conference on Human Information Interaction and Retrieval*, pages 5–14, 2021.

[3] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The Long-Document Transformer. *ArXiv*, abs/2004.05150, 2020.

[4] S. Bhargav, G. Sidiropoulos, and E. Kanoulas. 'It's on the tip of my tongue' A new Dataset for Known-Item Retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 48–56, 2022.

[5] L. H. Bonifacio, H. Abonizio, M. Fadaee, and R. F. Nogueira. InPars: Data Augmentation for Information Retrieval using Large Language Models. *ArXiv*, abs/2202.05144, 2022.

[6] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving Language Models by Retrieving from Trillions of Tokens. In *International Conference on Machine Learning*, 2022.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Conference on Neural Information Processing Systems*, 2020.

[8] W. Chang, F. X. Yu, Y. Chang, Y. Yang, and S. Kumar. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*, 2020.

[9] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *ArXiv*, abs/2305.14314, 2023.

[10] P. Fernandes, A. Madaan, E. Liu, A. Farinhas, P. H. Martins, A. Bertsch, J. G. C. de Souza, S. Zhou, T. Wu, G. Neubig, and A. F. T. Martins. Bridging the gap: A survey on integrating (human) feedback for natural language generation, 2023.

[11] M. Fröbe, E. O. Schmidt, and M. Hagen. A large-scale dataset for known-item question performance prediction. 2023.

[12] M. Gospodinov, S. MacAvaney, and C. Macdonald. Doc2query-: When less is more. In *Advances in Information Retrieval - European Conference on Information*, 2023.

[13] D. Guo, A. M. Rush, and Y. Kim. Parameter-efficient transfer learning with diff pruning. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[14] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[16] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, 2021.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[18] V. Jeronymo, L. H. Bonifacio, H. Abonizio, M. Fadaee, R. de Alencar Lotufo, J. Zavrel, and R. F. Nogueira. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *ArXiv*, abs/2301.01820, 2023.

[19] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active Retrieval Augmented Generation. *ArXiv*, abs/2305.06983, 2023.

[20] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *ArXiv*, abs/1702.08734, 2017.

[21] M. Khalifa, L. Logeswaran, M. Lee, H. Lee, and L. Wang. Few-shot reranking for multi-hop qa via language model prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15882–15897, 2023.

[22] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[23] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela.

Retrieval-Augmented Generation for Knowledge-Intensive NLP Task. In *Annual Conference on Neural Information Processing Systems*, 2020.

[24] C. Li, A. Yates, S. MacAvaney, B. He, and Y. Sun. PARADE: Passage Representation Aggregation for Document Reranking. *ArXiv*, abs/2008.09093, 2020.

[25] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[26] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *International Conference on Research and Development in Information Retrieval*, 2021.

[27] K. Lin, K. Lo, J. E. Gonzalez, and D. Klein. Decomposing complex queries for tip-of-the-tongue retrieval. *ArXiv*, abs/2305.15053, 2023.

[28] K. Lin, K. Lo, J. E. Gonzalez, and D. Klein. Decomposing complex queries for tip-of-the-tongue retrieval, 2023.

[29] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022.

[30] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *ArXiv*, abs/2307.03172, 2023.

[31] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9), 2023.

[32] X. Ma, L. Wang, N. Yang, F. Wei, and J. Lin. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. *ArXiv*, abs/2310.08319, 2023.

[33] X. Ma, X. Zhang, R. Pradeep, and J. Lin. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*, 2023.

[34] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics*, 2022.

[35] R. Nogueira and K. Cho. Passage Re-ranking with BERT. *ArXiv*, abs/1901.04085, 2019.

[36] R. Nogueira and J. Lin. From doc2query to doctttttquery. *Technical Report*, 6, 2019.

[37] R. F. Nogueira, W. Yang, J. Lin, and K. Cho. Document Expansion by Query Prediction. *ArXiv*, abs/1904.08375, 2019.

[38] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[39] R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *International Conference on Knowledge Discovery & Data Mining*, 2019.

[40] R. Pradeep, Y. Li, Y. Wang, and J. Lin. Neural Query Synthesis and Domain-Specific Ranking Templates for Multi-Stage Clinical Trial Matching. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2325–2330, 2022.

[41] R. Pradeep, S. Sharifymoghaddam, and J. Lin. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. *ArXiv*, abs/2309.15088, 2023.

[42] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, and M. Bendersky. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. *ArXiv*, 2023.

[43] 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. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, 2021.

[44] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham. In-Context Retrieval-Augmented Language Models. *ArXiv*, abs/2302.00083, 2023.

[45] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Empirical Methods in Natural Language Processing*, 2019.

[46] K. Roberts, D. Demner-Fushman, E. M. Voorhees, S. Bedrick, and W. R. Hersh. Overview of the trec 2021 clinical trials track. In *Proceedings of the Thirtieth Text REtrieval Conference*, 2021.

[47] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 2009.

[48] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[49] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, and Z. Ren. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *ArXiv*, abs/2304.09542, 2023.

[50] R. Tang, X. Zhang, X. Ma, J. Lin, and F. Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *ArXiv*, abs/2310.07712, 2023.

[51] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[52] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. *ArXiv*, abs/2212.10560, 2022.

[53] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[54] A. Yates, R. Nogueira, and J. Lin. Pretrained transformers for text ranking: BERT and beyond. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[55] E. B. Zaken, Y. Goldberg, and S. Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Annual Meeting of the Association for Computational Linguistics*, 2022.

[56] X. Zhang, S. Hofstätter, P. Lewis, R. Tang, and J. Lin. Rank-without-GPT: Building GPT-Independent Listwise Rerankers on Open-Source Large Language Models. *ArXiv*, abs/2312.02969, 2023.

A Appendix

A.1 Prompts

Here we show the prompt formats we used through our experiments. Re-ranking prompts were adapted from RankLLaMA [32]. The GPT-4 Zero shot prompt for 20 movies is from the authors of the TREC-ToT [3] track.

A.1.1 Query Decomposition Prompt

Prompt Input: query q
You are an utility that decomposes complex user-formed descriptions of movies into smaller independent sentences that aid in movie name retrieval. Incorporate as much historical information in each sentence and lengthen them by appending synonyms and words at the end that may resemble the movie title and description and help in retrieval.
Original Query: q
Decomposed Query:

A.1.2 GPT-4 Zero-Shot 20 Movies

Prompt Input: query q
You are an expert in movies. You are helping someone recollect a movie name that is on the tip of their tongue. You respond to each message with a list of 20 guesses for the name of the movie being described. Important: you only mention the names of the movies, one per line, sorted by how likely they are the correct movie with the most likely correct movie first and the least likely movie last. Given below is the movie description:
q

A.1.3 GPT-4 Zero-Shot 1 Movie

Prompt Input: query q
You are an expert in movies. You are helping someone recollect a movie name that is on the tip of their tongue. You respond to each message with a single guess for the name of the movie being described.**important**: you only mention the names of the movie and nothing else. Given below is the movie description:
q

A.1.4 Listwise Re-ranking Prompt

Prompt Input: query q, movie titles t_i
I will provide you with 100 movies, each indicated by a numerical identifier between [], e.g., [1], [2]. Rank the movies based on their relevance to the user query: q.
- [0] t_0
- [1] t_1
...
- [99] t_{99}
User query: q.
Rank the 100 movies above based on their relevance to the query. Use your best knowledge about the movies given their titles. All the movies should be included and listed using the identifiers, in descending order of relevance. The output format should be [] > [], e.g., [4] > [2]. Only respond with the ranking results, do not say any word or explain.

A.1.5 Pointwise Re-ranking Prompt

Prompt Input: query q, movie title t
I will provide you with an user query, a movie title and a small description. Score for the movie from 1 to 10 in the format 'x,xxx', with respect to relevance to the user query. A score of 1 indicates that the movie is not relevant to the query, and a score of 10 indicates that you are fully confident that it is a match.
User Query: q
A possible movie is: Title: t[title]. Description: t[description].
Use your best knowledge about the movie, use its title and the description to contextualize yourself. The output format should be the score alone, following the format 'x,xxx' i.e., '1,212' or '7,999'. Only respond with the score, in the correct format, 'x,xxx', do not say any word or explain.
Score:

A.2 Training Details

Hyperparameters for distil-bert training. We keep the best model among each validation step per epoch.

- Epochs: 20
- Learning rate: 6e-5
- Weight decay: 0.01
- Batch size: 10
- Hard negatives per query: 5
- Others: default from distil-bert and/or SentenceTransformers.

A.3 OpenAI API Costs

The costs we incurred with our methods (on average) for 150 queries:

- Zero-shot 1 movie: 1$
- Decomposition: 3$
- Listwise top-100 re-ranking: 9$.
- Pointwise top-10 re-ranking: 3$ title only; 6$ title and description.

[3] https://trec-tot.github.io/

A.4 Negative Results

We tried some approaches that did not lead to promising results. Here are their descriptions:

- **Pointwise re-ranking through perplexity**: Some OpenAI models return token log probabilities, which allows for the computation of the perplexity of a sentence. We hypothesized that the perplexity of a (query, relevant movie) pair would be lower than a (query, irrelevant movie) pair, and that this scoring method would be more natural than having the model output a score. However, the strongest model that has the log probability feature available (`text-davinci-003`) did not manage to improve on GPT-4 listwise re-ranking.
- **Cheaper re-rank models**: Before moving to GPT-4, we tried GPT-3.5 as our listwise re-ranker. However, it would return around 60% of malformed ranked lists, i.e., not following the format we stipulated in the prompt (A.1.4). Hallucinations included multiple repeated movies in the list, lists much shorter than the original 100 movies, identifiers outside the 0-99 range, and poorly formatted lists. This made the result impossible to parse (automatically).

B Team Member Contributions

We all acknowledge the following contributions:

- Aprameya Bharadwaj: Implemented the Zero shot ranking and also the pointwise re-ranking code for both prompt based and perplexity based approach.
- Chantal D Gama Rose: Carried out prompt engineering to perform query decomposition of the input. Ran experiments with BM25 for first stage retrieval.
- Dheeraj Pai: Implementation and experimentation of dense retrieval with query decomposition.
- João Coelho: BM25+RRF implementation. Dense retrieval warm-up task and external data sources. Initial listwise re-ranking implementation. Experimentation and result analysis.
- Vinay Nair: Listwise re-ranking implementation, experimentation, and prompt engineering.