Tip-of-the-tongue (ToT) Retrieval Leveraging Large Language Models

Aprameya Bharadwaj, Chantal D Gama Rose, Dheeraj Pai, João Coelho, Vinay Nair

Introduction

- ToT retrieval is challenging for current IR systems due to **long, verbose, and complex queries,** often containing **uncertainty.**
- We work on the **TREC-ToT movie** dataset, containing **150 train queries**, **150 evaluation queries,** and a **230000-movie corpus.**
- Motivated by the **popularity of movies**, and recognizing the (potential) extensive exposure of LLMs to this domain during training, we combine a dense retriever and an LLM re-ranker to return an ordered list of 100 candidates for each query:

"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 squator/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!"

1. Vive L' Amour
2. Take Care of My Cat
3. Sorum

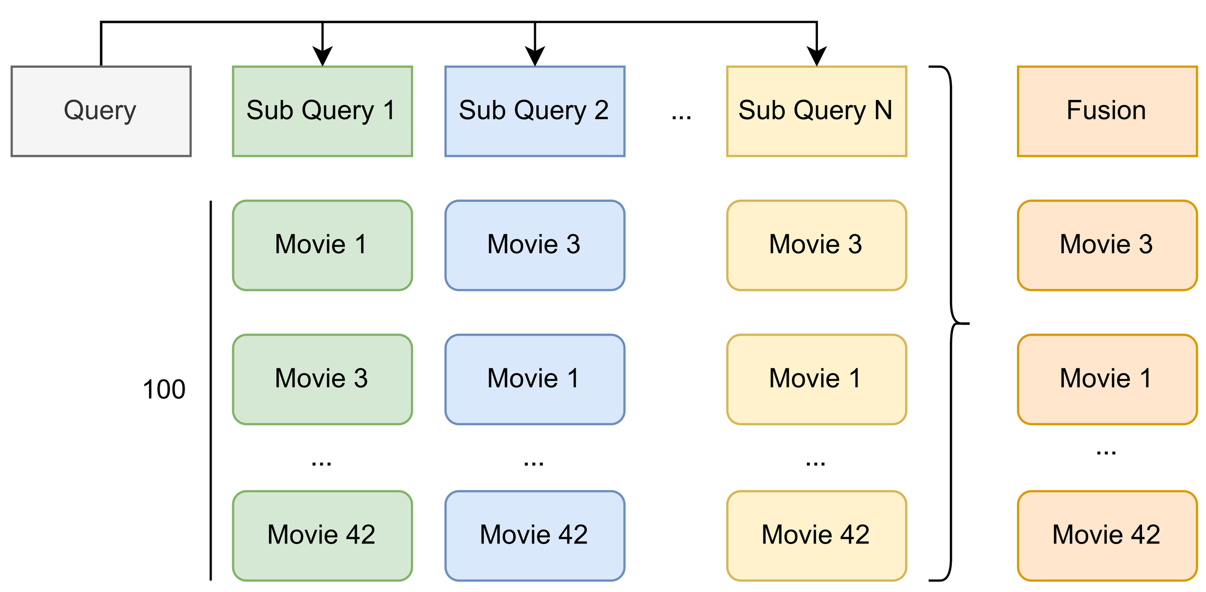
...

100. The Eighth House

Methods

Query Decomposition

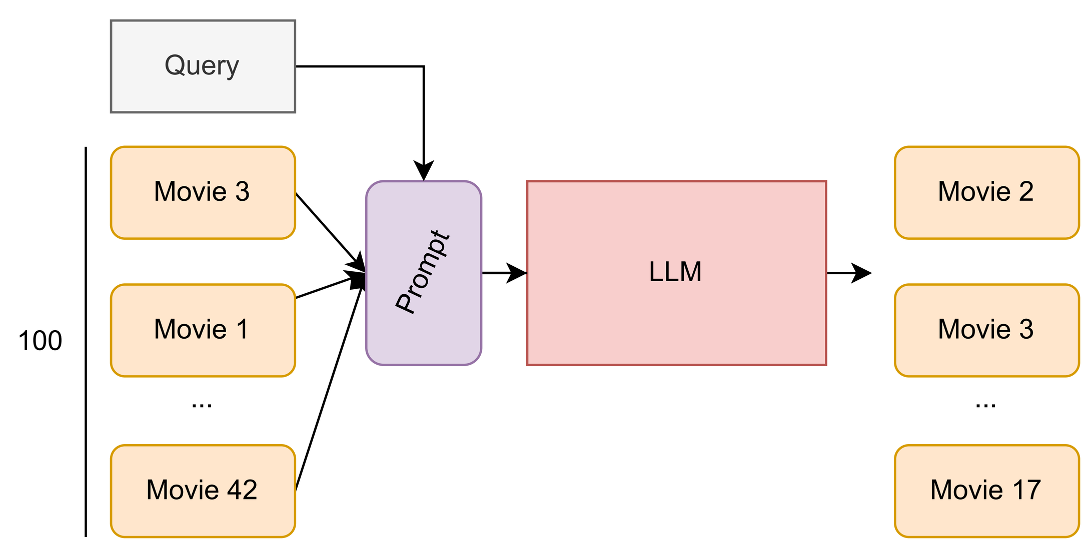
- Prompt an LLM to **decompose** one large query into multiple smaller ones. Retrieve the top-100 movies for each smaller query, and perform **fusion** on the results:



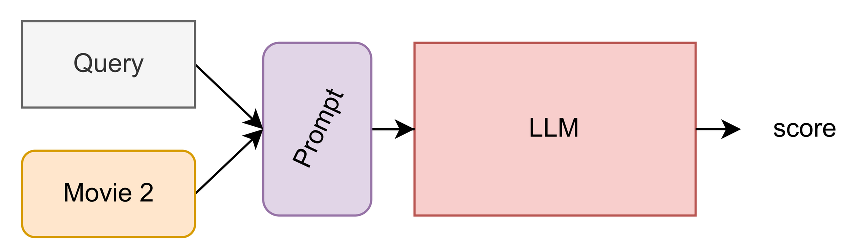
- Comparison of **sparse and dense retrieval**. Usage of a **self-supervised warm-up task** and **public data** to overcome low training data for dense retrieval.
- Identification of **irrelevant snippets** in the original query (e.g., "Help if you can!", uncertainty). **Retain** important details from previous to later subqueries.

LLM Re-rankers

- Two steps: top-100 followed by top-10 re-ranking:
  - **Listwise top-100:** build a prompt with the query and the top-100 movie titles. Ask the LLM to **re-order**:



  - **Pointwise top-10:** build a prompt containing the query, one movie title, and its description. LLM **scores** the top-10 movies after listwise ranking:



Findings

Retrieval

- Our techniques improve the TREC dense baseline by 33% in terms of NDCG and by 38% in terms of R@100.
  - Query decomposition is beneficial for sparse retrieval. Limited benefits for the dense model.
  - The dense retrieval warm-up strategy helps recall.
  - More supervised data further boosts the results.

Re-ranking

- We are able to achieve a **final NDCG of 39.6%**, improving the TREC GPT-4 baseline by 65%.
  - Listwise prompt was robust to input order.
  - Pointwise re-ranking further improved results.
  - Movie description improves the pointwise approach.

		P@1	R@10	R@100	R@1000	NDCG@10	NDCG@100	**NDCG@1000**
Baselines	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	0.040	0.147	0.360	0.660	0.085	0.127	0.163
Query Decomposition (for sparse and dense retrieval)	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
Retrieval train data (self-supervised / supervised)	warm-up + distil-bert	0.027	0.153	0.353	0.706	0.086	0.127	0.170
	reddit data + distil-bert	0.073	0.233	0.433	0.696	0.145	0.185	0.217
	warm-up + reddit data + distil-bert	0.046	0.213	**0.500**	**0.713**	0.126	0.184	0.210
Top-100 Re-ranking	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
Final Top-10 Re-ranking (bottom-90 order kept)	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**

Carnegie Mellon University