

# RANKFLOW : A Multi-Role Collaborative Reranking Workflow Utilizing Large Language Models

Can Jin<sup>1\*†</sup> Hongwu Peng<sup>2\*</sup> Anxiang Zhang Nuo Chen<sup>3</sup> Jiahui Zhao<sup>2</sup> Xi Xie<sup>2</sup>  
Kuangzheng Li Shuya Feng<sup>2</sup> Kai Zhong Caiwen Ding<sup>2</sup> Dimitris N. Metaxas<sup>1</sup>

<sup>1</sup>Rutgers University, USA <sup>2</sup>University of Connecticut, USA

<sup>3</sup>Waseda University, Japan

can.jin@rutgers.edu

{hongwu.peng, jiahui.zhao, xi.xie, shuya.feng, caiwen.ding}@uconn.edu

{adamzhang1679, likuangzheng0633, kaizhong}@gmail.com

pleviumtan@toki.waseda.jp dnm@cs.rutgers.edu

## Abstract

In an Information Retrieval (IR) system, reranking plays a critical role by sorting candidate passages according to their relevance to a specific query. This process demands a nuanced understanding of the variations among passages linked to the query. In this work, we introduce RANKFLOW, a multi-role reranking workflow that leverages the capabilities of Large Language Models (LLMs) and role specializations to improve reranking performance. RANKFLOW enlists LLMs to fulfill four distinct roles: the query **Rewriter**, the pseudo **Answerer**, the passage **Summarizer**, and the **Reranker**. This orchestrated approach enables RANKFLOW to: (1) accurately interpret queries, (2) draw upon LLMs’ extensive pre-existing knowledge, (3) distill passages into concise versions, and (4) assess passages in a comprehensive manner, resulting in notably better reranking results. Our experimental results reveal that RANKFLOW outperforms existing leading approaches on widely recognized IR benchmarks, such as TREC-DL, BEIR, and NovelEval. Additionally, we investigate the individual contributions of each role in RANKFLOW. Code is available at <https://github.com/jincan333/RankFlow>.

## 1 Introduction

The integration of large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Lewis et al., 2020) into Information Retrieval (IR) systems has revolutionized user interactions with information and knowledge (Hou et al., 2024; Fan et al., 2023; Xi et al., 2023). LLMs enhance the IR process, including query rewriting and retrieval, through advanced linguistic understanding, semantic representation, context management, and encyclopedic knowledge (Wang et al., 2023; Sachan et al., 2022; Qin et al., 2023).

The application of LLMs to zero-shot text ranking has seen increasing interest. Based on the type of instruction employed, Ranking strategies utilizing LLMs can be categorized into Pointwise (Sachan et al., 2022; Liang et al., 2022), Pairwise (Qin et al., 2023; Sun et al., 2023a), and Listwise methods (Sun et al., 2023b; Pradeep et al., 2023). Among existing methods, listwise approaches achieve superior performance by enabling concurrent relevance judgment across a list of passages (Sun et al., 2023b; Pradeep et al., 2023).

Although LLMs exhibit strong semantic understanding abilities, retrieval performance can still be compromised by queries that are short, ambiguous, or lack context (Wang et al., 2023; Nogueira et al., 2019b). Moreover, the listwise reranking approach, which involves lengthy contexts, faces the challenge of decreased LLM instruction following and reasoning capabilities as context length increases (Sun et al., 2023b; Levy et al., 2024; Bai et al., 2023), potentially affecting reranking performance. Motivated by the notion that structured workflows enhance task execution consistency and accuracy (Wooldridge and Jennings, 1998; Belbin and Brown, 2022; Hong et al., 2023), we introduce RANKFLOW—a multi-role workflow utilizing LLMs for reranking. As illustrated in Figure 1, RANKFLOW applies enriched queries with LLMs’ prior knowledge and summarized information from passages to overcome existing challenges in LLM-based reranking systems.

We assess the performance of RANKFLOW across a broad range of datasets, including TREC-DL (Craswell et al., 2020), BEIR (Thakur et al., 2021), and NovelEval (Sun et al., 2023b). Our empirical findings consistently demonstrate RANKFLOW’s superior performance. Notably, RANKFLOW outpaces current state-of-the-art (SoTA) methods, achieving higher scores than RankGPT (Sun et al., 2023b) on four BEIR datasets—Covid, NFCorpus, SciFact, and Ro-

\* Equal contribution.

† Corresponding author.

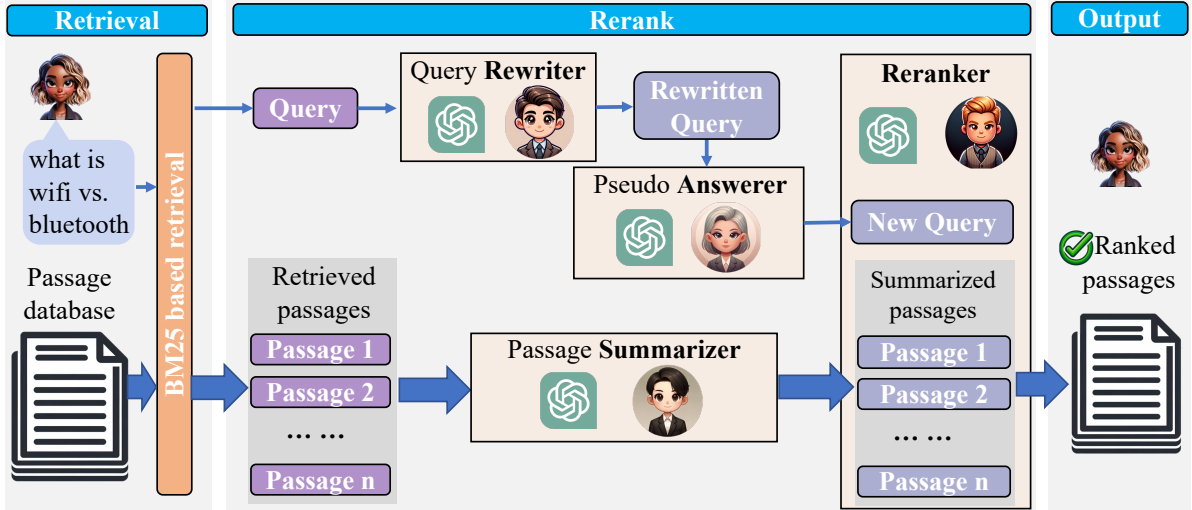


Figure 1: Overview of RANKFLOW. RANKFLOW is composed of four well-defined expert roles: **Rewriter**, **Answerer**, **Summarizer**, and **Reranker**, each designed to address specific issues in passage reranking. These roles work sequentially to handle the ranking task.

bust04—by an average of 2.5% in nDCG@10, and surpassing RankZephyr (Pradeep et al., 2023) on NovelEval by 5% in nDCG@{1, 5, 10}.

In a nutshell, our contributions can be highlighted from four perspectives:

1. We introduce a unique multi-role reranking workflow, denoted as RANKFLOW, which is grounded in LLMs and employs well-defined role specializations. This workflow exhibits remarkable adaptability, allowing for the dynamic alteration of roles to enhance reranking efficacy.
2. Our approach innovatively addresses the challenges of semantic ambiguity and context length constraints in listwise zero-shot reranking by incorporating query rewriting and passage summarization to augment clarity.
3. We carry out extensive experiments on a variety of datasets, evidencing that RANKFLOW consistently surpasses SoTA methods (Sun et al., 2023b; Pradeep et al., 2023).
4. We thoroughly examine the impact of individual roles within RANKFLOW, providing valuable insights for further investigation.

## 2 Related Works

### 2.1 LLMs for Information Retrieval

Text retrieval is a key component in a multitude of knowledge-driven Natural Language Processing (NLP) applications (Jin et al., 2024a,b; Wu

et al., 2024). In practice, this task is approached with a multi-stage ranking pipeline, typically consisting of an initial, cost-effective retriever followed by a more sophisticated reranker to refine the results (Ma et al., 2023b; Craswell et al., 2020; Nogueira et al., 2019a). Large language models (LLMs) have shown remarkable efficacy in information retrieval tasks (Zhu et al., 2023; Sun et al., 2023b; Pradeep et al., 2023). Supervised reranking methods (Nogueira et al., 2020; Zhuang et al., 2023; Pradeep et al., 2023) have traditionally relied on fine-tuning transformer-based models with copious training data, such as the MS MARCO v1 passage ranking dataset (Bajaj et al., 2016). However, recent explorations involve LLMs in zero-shot unsupervised reranking. Pointwise approaches evaluate passage relevance individually (Sachan et al., 2022; Liang et al., 2022), whereas pairwise strategies compare two documents’ relevancies for a given query (Qin et al., 2023; Sun et al., 2023a). Listwise methods, which directly reorder document relevance collectively, have achieved state-of-the-art performance (Sun et al., 2023b; Ma et al., 2023b). This study introduces a novel multi-role reranking framework, RANKFLOW, which significantly enhances listwise reranking performance.

### 2.2 Query Rewriting

Original queries in traditional IR systems are often short or ambiguous, leading to vocabulary mismatch issues. Classic query rewriting techniques refine the original query iteratively by analyzing top-retrieved documents (Abdul-Jaleel et al., 2004;

Metzler and Croft, 2005; Zhai and Lafferty, 2001; Metzler and Croft, 2007). These methods, however, largely depend on term frequency statistics and may not grasp the true query intent. LLMs, with their advanced linguistic capabilities, support the generation of query rewrites that more accurately reflect the complex and varied information needs of users (Mao et al., 2023; Gao et al., 2023; Jagerman et al., 2023; Ma et al., 2023a). HyDE (Gao et al., 2023) utilizes dense retrievers to generate pseudo-documents, while Query2doc (Wang et al., 2023) and InPars (Bonifacio et al., 2022), along with Promptagator (Dai et al., 2022), harness LLMs for producing synthetic queries through zero-shot or few-shot prompting.

### 2.3 Prompt Engineer

Prompt engineering is a critical technique for efficiently tailoring models to specific downstream tasks without fine-tuning (Liu et al., 2023; Brown et al., 2020; Zhou et al., 2022; Jin et al., 2023; Zhou et al., 2024; Zhang et al., 2023, 2022; Zhao et al., 2024). The chain-of-thought (CoT) prompting method was introduced to encourage LLMs to generate intermediate reasoning steps before reaching a final answer (Kojima et al., 2022; Wei et al., 2022). In-context learning (ICL) leverages a few examples within the input to guide LLMs towards the intended task (Radford et al., 2019; Liu et al., 2022). Expert prompting (Xu et al., 2023) designs prompts that emulate an expert’s reasoning, tailored to the input query’s context. Multi-persona prompting (Du et al., 2023) employs a range of ‘personas’ to tackle specific tasks. In RANKFLOW, we engage LLMs with various roles outlined in a standard operating procedure (SOP) for retrieval, yielding empirically validated improvements in reranking.

## 3 RANKFLOW

RANKFLOW is a novel reranking framework for LLMs. Section 3.1 presents the role specialization and overall procedure in RANKFLOW. Sections 3.2, 3.3, 3.4, and 3.5 elucidate the detailed design for each role.

### 3.1 Role Specialization and Overall Procedure

Unambiguous role specialization facilitates the decomposition of intricate work into smaller, distinct tasks. In RANKFLOW, we divide the reranking task into four parts, assigning four agents with specific skills and expertise for each subtask. Defining the

LLMs’ roles and operational abilities establishes a workflow, allowing the LLMs to work sequentially.

As depicted in Figure 1, the Rewriter, adept at rephrasing sentences, reformulates the user query into a clearer and well-interpreted version. The rewritten query is then passed to the Answerer, which offers a standard answer. Subsequently, the rewritten query is concatenated with the generated answer, forming a new query containing more comprehensive information about the original query. During ranking, a passage Summarizer creates a concise summary for each candidate passage, effectively capturing the essential information more efficiently than the original passage. Finally, the Reranker receives the new query and the summarized candidate passages, outputting a relevance-based reranking.

### 3.2 LLMs as Rewriters

Original queries often exhibit brevity or ambiguity. For example, a query in TREC-DL19 (Craswell et al., 2020) is ‘what is wifi vs bluetooth’, where the desired passage should emphasize the distinctions and similarities between ‘wifi’ and ‘bluetooth’. Nevertheless, most pertinent documents in the search results utilize the term ‘comparative’ instead of ‘vs’. Existing query rewriting techniques employ document corpora to supply domain-specific knowledge for subject areas (Gao et al., 2023; Ma et al., 2023a). These approaches concentrate on examining query rewriting in the initial retrieval stage, while its application in the subsequent reranking stage remains underexplored. In RANKFLOW, we concentrate on harnessing LLMs’ language abilities for query rewriting in passage reranking tasks, employing role specialization without reliance on any corpora base.

Let  $q$  be a query from the query distribution  $\mathcal{Q}$ . We specialize the LLM as an expert in refining user queries to enhance their suitability for ranking tasks. Next, we use a zero-shot prompt  $c_{rew}$ , devoid of any corpus or specific domain knowledge, for query rewriting to obtain a clearer and well-interpreted query  $q_{rew}$ .

$$q_{rew} = f(q; c_{rew}) \quad (1)$$

where  $f$  represents the LLM.

The specialization and prompting for Rewriter can be found in Appendix C. Generally, the rewritten query  $q_{rew}$  conveys more precise information and avoids ambiguous expressions. Examples of

the original query  $q$  and the rewritten query  $q_{rew}$  are displayed in Table 9.

### 3.3 LLMs as Answerers

Query expansion enhances retrieval systems by enriching query representation with additional terms, expressing identical concepts or information needs, and improving lexical or semantic alignment with corpus documents (Datta et al., 2008; Huang and Efthimiadis, 2009). Early research on query expansion focused on using lexical knowledge bases (Robertson et al., 1995) or Pseudo-Relevance Feedback (PRF) (Borgeaud et al., 2022). Wang et al. (2023) propose expanding the original query during the previous sparse or dense retrieval stage. Our Answerer adopts a similar approach to Wang et al. (2023), emphasizing reranking performance enhancement without any knowledge base or feedback, necessitating more refined relevance judgment.

We specialize the LLM as an assistant adept at providing detailed and pertinent responses to user queries. Through carefully crafted prompts, the Answerer generates a pseudo-passage  $P_{gen}$  that addresses the query, serving as a ‘standard answer’ for the given query  $q_{rew}$ :

$$P_{gen} = f(q_{rew}; c_{gen}) \quad (2)$$

where  $c_{gen}$  represents the zero-shot prompt for ‘standard answer’ generation.

Owing to the LLMs’ proficient language ability and world knowledge, the generated passage offers abundant information about the given query. Table 9 showcases some examples for the answer. The complete specialization for our Answerer is illustrated in Appendix D.

We then define a new query  $q_{new}$  as follows:

$$q_{new} = \text{Concat}(q_{rew} * m, P_{gen}) \quad (3)$$

where  $*$  denotes string repetition,  $m$  represents the number of repetitions, and  $\text{Concat}$  is the string concatenation operator. Our experiments demonstrate that repeating the query an appropriate number of times strengthens the query’s ‘weights’, leading to improved passage reranking performance.

### 3.4 LLMs as Summarizers

Candidate passages  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  from the preceding retrieval stage are often lengthy, and their contained information may be vague, ambiguous, or redundant, complicating concise relevance

judgments. To tackle this issue, we suggest summarizing candidate passages before utilizing them for reranking. These summaries effectively preserve essential information without redundancy and are typically much shorter than the original passages, facilitating improved relevance judgment.

To generate enhanced summaries of the original passages, we devise a Summarizer skilled in condensing passages for better information retrieval. For each candidate passage  $P \in \mathcal{P}$ , we obtain a summarized passage  $\hat{P}$  as follows:

$$\hat{P} = f(P; c_{sum}) \quad (4)$$

where  $c_{sum}$  represents the zero-shot prompt for passage summarization. The complete prompt can be found in Appendix E.

Following the summarization process, we acquire a list of summarized candidate passages  $\hat{\mathcal{P}} = \{\hat{P}_1, \hat{P}_2, \dots, \hat{P}_n\}$ . Table 10 offers some examples of summarized passages, which are generally shorter and clearer than their original counterparts.

### 3.5 LLMs as Reranker

Sun et al. (2023b) propose listwise permutation generation to directly output a ranked list given a set of candidate passages. However, the listwise approach necessitates a considerable number of tokens, potentially degrading instruction following and reasoning abilities (Levy et al., 2024; Bai et al., 2023; Sun et al., 2023a) and negatively affecting ranking performance. In RANKFLOW, we mitigate this limitation by employing summarized passages for listwise reranking, which are shorter and clearer than the original ones. To enable more precise relevance judgments and further enhance ranking performance through increased reasoning, we adopt a distinct prompting strategy from Sun et al. (2023b), encompassing three aspects:

- **Relevance Standard.** To facilitate more accurate relevance judgments, we instruct the LLM to adhere to a detailed relevance standard, as proposed by Craswell et al. (2020). For instance, a passage is deemed perfectly relevant if it is dedicated to the query and contains the exact answer, whereas a passage is considered irrelevant if it bears no relation to the query.
- **CoT.** CoT prompting can elicit the reasoning ability of LLMs (Wei et al., 2022). Prior studies have applied CoT in query rewriting (Jagerman et al., 2023; Alaofi et al., 2023), while the

impact of CoT in passage reranking remains underexplored. In RANKFLOW, we prompt the LLM to rank passages thoughtfully and systematically, enabling more reasoning in the relevance judgment process.

- **Format Requirement.** After employing the relevance standard and CoT, the LLM’s output format becomes more diverse (e.g., containing rationales). To minimize malformed outputs, we instruct the LLM to conform to the ranking format and ensure that no passages are omitted or repeated in the ranking results.

In RANKFLOW reranking, we specialize the LLM as an adept intelligent assistant for ranking passages based on query relevance. Our carefully designed prompt strategy enables the Reranker to generate superior ranking results. The specification and prompt for our Reranker are illustrated in Appendix F.

Assuming there are  $n$  summarized candidate passages  $\hat{\mathcal{P}} = \{\hat{P}_1, \hat{P}_2, \dots, \hat{P}_n\}$  from the Summarizer, we rerank these passages in a back-to-first order from  $\hat{P}_n$  to  $\hat{P}_1$  using a sliding window of length  $w$ . Initially, we attain a rank list  $l_1$  by prompting the LLM to rank the passages  $[\hat{P}_{n-w+1}, \dots, \hat{P}_{n-1}, \hat{P}_n]$  according to their relevance to  $q_{new}$ :

$$l_1 = f(q_{new}, [\hat{P}_{n-w+1}, \dots, \hat{P}_{n-1}, \hat{P}_n]; c_{list}) \quad (5)$$

where  $c_{list}$  represents our zero-shot listwise prompt, employing the relevance standard, CoT, and format requirement.

Subsequently, we reorder the passage order based on the rank list  $l_1$  to obtain a ranked passage set, still denoted as  $\hat{\mathcal{P}} = \{\hat{P}_1, \hat{P}_2, \dots, \hat{P}_n\}$  for simplicity. We then slide the window in steps of length  $s$  and rank the passages  $[\hat{P}_{n-s-w+1}, \dots, \hat{P}_{n-s-1}, \hat{P}_{n-s}]$  following equation 5, resulting in a rank list  $l_2$ . We reorder the passage order in  $\hat{\mathcal{P}}$  based on  $l_2$ . This procedure repeats until all passages are ranked.

## 4 Experiments

To assess the efficacy of RANKFLOW, we perform extensive experiments to: (1) showcase the superior performance of RANKFLOW across various benchmarks; (2) examine the influence of each component in our approach, namely, Rewriter, Answerer, Summarizer, and Reranker; and (3) evaluate the impact of distinct prompts in the listwise Reranker.

### 4.1 Models and Benchmarks

We choose GPT-4 (Achiam et al., 2023) as our primary model, employing the Azure API, which features a GPT-4-0613 version. Our experiments are evaluated on three benchmark datasets, including TREC-DL (Craswell et al., 2020), BEIR (Thakur et al., 2021), and NovelEval (Sun et al., 2023b).

**TREC** is a widely adopted benchmark dataset in IR research. We use the test sets from the 2019 and 2020 competitions, which employed the MS MARCO v1 passage corpus: (i) TREC-DL19 comprises 43 queries, and (ii) TREC-DL20 consists of 54 queries.

**BEIR** encompasses diverse retrieval tasks and domains. We select the test sets of eight tasks in BEIR to evaluate our approach: (i) *Covid*: Retrieves scientific articles related to COVID-19. (ii) *NFCorpus*: A biomedical information retrieval dataset. (iii) *SciFact*: Retrieves evidence for claims verification. (iv) *Robust04*: Assesses challenging topics. (v) *Touche*: An argument retrieval dataset. (vi) *DBpedia*: Retrieves entities from the DBpedia corpus. (vii) *Signal*: Retrieves relevant tweets for a given news title. (viii) *News*: Retrieves relevant news articles for headlines.

**NovelEval** features queries not learned by GPT-4-0613 (Sun et al., 2023b). Questions in current benchmarks (e.g., TREC-DL) are typically collected years ago, raising concerns that existing LLMs may already possess knowledge of these questions (Yu et al., 2023). Moreover, since many LLMs do not disclose information about their training data, there is a potential risk of contamination in existing benchmark test sets (Achiam et al., 2023). To mitigate these concerns, we evaluate RANKFLOW on NovelEval-2306.

### 4.2 Baselines and Evaluation Metrics

**Baselines.** We select several representative SoTA passage reranking methods as our baselines: (1) *BM25* (Lin et al., 2021) serves as a fundamental sanity check in reranking, directly using the rank results after the previous retrieval stage; (2) *monoT5* (Nogueira et al., 2020) is a sequence-to-sequence reranker employing T5 (3B) to compute the relevance score with pointwise ranking loss, trained on MS MARCO; (3) *RankT5* (Zhuang et al., 2023) is a reranker utilizing T5 (3B) and listwise ranking loss, trained on MS MARCO; (4) *RG* (Liang et al., 2022) is a pointwise reranking approach based on relevance generation using FLAN-

| Method              | LLM            | TREC-DL19    |              |              | TREC-DL20    |              |              |
|---------------------|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                     |                | nDCG@1       | nDCG@5       | nDCG@10      | nDCG@1       | nDCG@5       | nDCG@10      |
| BM25                | -              | 54.26        | 52.78        | 50.58        | 57.72        | 50.67        | 47.96        |
| <b>Supervised</b>   |                |              |              |              |              |              |              |
| monoT5              | T5 (3B)        | 79.04        | 73.74        | 71.83        | 80.25        | 72.32        | 68.89        |
| RankT5              | T5 (3B)        | 77.38        | 73.94        | 71.22        | 80.86        | 72.99        | 69.49        |
| RankZephyr          | Zephyr (7B)    | -            | -            | 74.20        | -            | -            | 70.86        |
| <b>Unsupervised</b> |                |              |              |              |              |              |              |
| RG                  | FLAN-UL2 (20B) | 70.93        | 66.81        | 64.61        | 75.62        | 66.85        | 65.39        |
| PRP                 | FLAN-UL2 (20B) | 78.29        | 75.49        | 72.65        | <b>85.80</b> | 75.35        | 70.46        |
| RankGPT-4           | GPT-4          | 80.62        | 77.83        | 74.89        | 79.73        | 73.15        | 70.14        |
| RANKFLOW            | GPT-4          | <b>83.33</b> | <b>79.44</b> | <b>76.65</b> | 82.41        | <b>75.68</b> | <b>71.80</b> |

Table 1: **Results (nDCG@{1,5,10}) on TREC.** The performance of eight reranking methods on TREC-DL19 and TREC-DL20. All the unsupervised methods use zero-shot prompts. The best performances are marked in bold.

| Method          | COVID        | NFCorpus     | SciFact      | Robust04     | Touche       | DBpedia      | Signal       | News         | Avg          |
|-----------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| BM25            | 59.47        | 30.75        | 67.89        | 40.70        | 44.22        | 31.80        | 67.89        | 39.52        | 43.42        |
| monoT5          | 80.71        | 38.97        | 76.57        | 56.71        | 32.41        | 44.45        | 32.55        | 48.49        | 51.36        |
| RankT5          | 80.71        | 38.10        | 74.99        | -            | 44.01        | 44.22        | 32.00        | -            | -            |
| RankZephyr      | 83.78        | -            | -            | -            | -            | -            | -            | 51.84        | -            |
| RankGPT-4       | 83.98        | 38.83        | 75.61        | 59.74        | 40.72        | 47.12        | 33.90        | 52.82        | 54.09        |
| <b>RANKFLOW</b> | <b>85.77</b> | <b>39.74</b> | <b>77.73</b> | <b>64.88</b> | <b>42.08</b> | <b>47.43</b> | <b>34.54</b> | <b>52.97</b> | <b>55.64</b> |

Table 2: **Results (nDCG@10) on BEIR.** The performance of six reranking methods on eight BEIR datasets. RankT5 reranks the top 1000 passages returned by BM25 while other methods rerank top 100 passages retrieved by BM25.

UL2 (20B); (5) *PRP* (Qin et al., 2023) is a pairwise reranking approach employing a sliding window strategy with 10 passes, using the FLAN-UL2 (20B) model; (6) *RankZephyr* (Pradeep et al., 2023) is a recent reranker leveraging the 7B parameter Zephyr<sub>β</sub> (built on Mistral), distilled from GPT-3.5 and GPT-4 on MS MARCO; and (7) *rankGPT-4* (Sun et al., 2023b) is our most crucial baseline, adopting a listwise reranking strategy with GPT-4.

**Implementation and Metrics.** All baselines and RANKFLOW rerank the top 100 passages retrieved by BM25 using pyserini (Lin et al., 2021) unless specified otherwise. We employ normalized Discounted Cumulative Gain (nDCG) at rank cutoffs of {1, 5, 10} (nDCG@{1, 5, 10}) to evaluate performance. For RANKFLOW and rankGPT-4, we utilize the Azure API with a context size setting of 8192, employing the GPT-4 version GPT-4-0613, which differs from the one used in Sun et al. (2023b). Additionally, rankGPT-4 employs GPT-4 to rerank the top 30 passages reranked by GPT-3.5

| Method          | nDCG@1       | nDCG@5       | nDCG@10      |
|-----------------|--------------|--------------|--------------|
| BM25            | 33.33        | 45.96        | 55.77        |
| monoT5          | 83.33        | 78.38        | 84.62        |
| RankZephyr      | 92.86        | 86.15        | 89.34        |
| RankGPT-4       | 92.86        | 86.10        | 89.18        |
| <b>RANKFLOW</b> | <b>97.62</b> | <b>91.79</b> | <b>94.21</b> |

Table 3: **Results (nDCG@{1,5,10}) on NovelEval.** The performance of five reranking methods on NovelEval-2306.

(which reranks the top 100 passages by BM25) on BEIR. These differences result in discrepancies between the rankGPT-4 outcomes in our paper and those in Sun et al. (2023b). In our Answerer, we set a repeat time  $m$  of 3. In our Reranker, we use a window size  $w$  of 20 and a step size  $s$  of 10, following Sun et al. (2023b). We set the temperature to 0 for the GPT-4 API to reduce randomness. All our results are averaged over 3 runs. Finally, cost of our experiments are indicated in Appendix A.

| Method               | TREC-DL20                | COVID                    | NovelEval                |
|----------------------|--------------------------|--------------------------|--------------------------|
| RankGPT-4            | 79.73/73.15/70.14        | 88.25/86.67/83.98        | 92.86/86.10/89.18        |
| RankGPT-4 + Rewriter | 80.56/74.06/70.10        | 91.33/87.06/84.50        | 91.27/86.77/91.50        |
| RANKFLOW             | <b>82.41/75.68/71.80</b> | <b>96.00/89.48/85.77</b> | <b>97.62/91.79/94.21</b> |

Table 4: Results (nDCG@{1,5,10}) of utilizing the Rewriter.

| Method                         | TREC-DL19                | COVID                    | NovelEval                |
|--------------------------------|--------------------------|--------------------------|--------------------------|
| RankGPT-4                      | 80.62/77.83/74.89        | 88.25/86.67/83.98        | 92.86/86.10/89.18        |
| RankGPT-4 + Answerer, $m = 1$  | 81.79/77.98/74.21        | 87.67/86.68/84.00        | 97.62/90.51/92.38        |
| RankGPT-4 + Answerer, $m = 3$  | 82.17/78.41/74.70        | 89.67/88.40/85.00        | 97.62/91.63/93.85        |
| RankGPT-4 + Answerer, $m = 10$ | 80.82/78.09/74.63        | 89.00/87.87/84.14        | 97.62/91.49/93.83        |
| RANKFLOW                       | <b>83.33/79.44/76.65</b> | <b>96.00/89.48/85.77</b> | <b>97.62/91.79/94.21</b> |

Table 5: Results (nDCG@{1,5,10}) of utilizing the Answerer.

### 4.3 Main Results

**Results on TREC.** To demonstrate the superior performance of RANKFLOW on TREC datasets, we compare it with seven baselines. The nDCG@{1, 5, 10} results are presented in Table 1. We can draw the following positive observations: ❶ RANKFLOW exhibits superior performance compared to RankGPT-4 on TREC datasets, surpassing RankGPT-4 by an average nDCG@{1, 5, 10} of {2.70, 2.07, 1.71} on TREC-DL19 and TREC-DL20. ❷ RANKFLOW achieves the best performance among both supervised and unsupervised methods in terms of nDCG@5 and nDCG@10. It not only surpasses all supervised methods but also outperforms SoTA zero-shot unsupervised methods, including the pointwise method RG, the pairwise method PRP, and the listwise method RankGPT-4. This further indicates the effectiveness of our reranking framework.

**Results on BEIR.** We further evaluate the performance of RANKFLOW on eight BEIR datasets, which contain more queries and heterogeneous topics than TREC-DL19 and TREC-DL20. The results are displayed in Table 2, from which we can observe that: ❶ RANKFLOW outperforms the baselines on BEIR, achieving the best nDCG@10 across all baselines and datasets. ❷ RANKFLOW is robust to diverse topics and queries. It surpasses RankGPT-4 by an average of 1.55 on wight BEIR datasets, which contain more queries than the TREC datasets. Notably, RANKFLOW achieves a 5.14 nDCG@10 improvement over RankGPT-4 on Robust04, which consists of 249 queries and

diverse topics in news articles.

**Results on NovelEval.** To address the concern of data contamination in LLM reranking, we further evaluate the performance of five reranking methods on NovelEval-2306. The results are shown in Table 3, from which we can observe that: ❶ RANKFLOW maintains significant reranking performance on unlearned datasets, achieving the best performance among all reranking methods. ❷ RANKFLOW exhibits a substantial performance enhancement compared to RankGPT-4 and RankZephyr, with an improvement of 5 points in terms of nDCG@1, nDCG@5, and nDCG@10.

### 4.4 Ablation Study

**Rewriter.** First, we investigate the effect of the Rewriter in our multi-role reranking workflow. We incorporate the Rewriter into RankGPT-4, which rewrites the original queries in the benchmarks and utilizes the rewritten queries for listwise passage reranking. As shown in Table 4, the Rewriter demonstrates a capability to slightly enhance ranking performance by using well-interpreted queries. On TREC-DL20, COVID, and NovelEval, it achieves improvements in the majority of instances and yields an average nDCG@{1, 5, 10} improvement of {0.76, 0.66, 0.93}.

**Answerer.** We further investigate the effects of our Answerer in reranking by integrating it with RankGPT-4. For an original query  $q$  from the benchmarks, we generate a passage  $P$  answering the query using the Answerer. We then form a new query by repeating  $q$  for  $m$  times and con-

| Method                 | TREC-DL20                | COVID                    | Robust04                 |
|------------------------|--------------------------|--------------------------|--------------------------|
| RankGPT-4              | 79.73/73.15/70.14        | 88.25/86.67/83.98        | 75.30/66.07/59.74        |
| RankGPT-4 + Summarizer | 81.07/74.46/71.34        | 92.00/87.06/85.26        | 78.58/70.44/63.69        |
| RANKFLOW               | <b>82.41/75.68/71.80</b> | <b>96.00/89.48/85.77</b> | <b>80.92/70.99/64.88</b> |

Table 6: Results ( $nDCG@\{1,5,10\}$ ) of utilizing the Summarizer.

catenating the repeated queries with  $P$ , which is utilized in listwise reranking. The empirical results in Table 5 yield several positive observations: ❶ Our Answerer can enhance the overall ranking performance. For  $m \in \{1, 3, 10\}$ , incorporating the Answerer leads to performance gains in most cases on TREC-DL19, COVID, and NovelEval. ❷ A moderate value of  $m$  in the Answerer results in the best performance gains. As shown in Table 5,  $m = 3$  consistently outperforms  $m = 1$  and  $m = 10$ . ❸ The Answerer is capable of generating valuable feedback, even on datasets unlearned by GPT-4. On NovelEval, it achieves performance comparable to RANKFLOW. These observations demonstrate that the answers generated by the Answerer contain rich information that can improve semantic-level matching and relevance judgment in reranking.

**Summarizer.** Next, we evaluate the effects of Summarizer. We integrate the Summarizer into RankGPT-4, which uses the original query from benchmarks but replaces the candidate passages retrieved by BM25 with the summarized passages generated by the Summarizer. The results in Table 6 reveal that: ❶ The Summarizer is capable of enhancing reranking performance, achieving consistent performance gains in terms of  $nDCG@\{1, 5, 10\}$  on TREC-DL20, COVID, and Robust04. ❷ The Summarizer particularly improves the performance of  $nDCG@10$ , achieving an average  $nDCG@10$  improvement of 2.14 on three datasets and comparable  $nDCG@10$  with RANKFLOW. Notably, it attains an  $nDCG@10$  improvement of **3.95** on Robust04. The findings indicate that substituting vague, ambiguous, and redundant passages with concise, clear, and well-structured ones improves reranking performance.

**Reranker.** We conduct additional experiments to investigate the effect of different prompt designs in the Reranker. As indicated in Section 3.5, our prompting strategy comprises three aspects: Relevance Standard, CoT, and Format Requirement. We add each type of prompt to RankGPT-4 sep-

arately, resulting in ‘RankGPT-4 w. Relevance Standard’, ‘RankGPT-4 w. CoT’, and ‘RankGPT-4 w. Format Requirement’. Detailed information on the prompts can be found in Appendix G. The experimental results are presented in Table 8.

*Relevance Standard.* We prompt RankGPT-4 to follow a detailed four-level relevance standard: Perfectly relevant, Highly Relevant, Relevant, and Irrelevant, as described by Craswell et al. (2020). The results in Table 8 show that this prompting strategy yields consistent performance gains across all three datasets, indicating that detailed relevance standards can enhance relevance judgments.

*CoT.* We utilize zero-shot CoT prompts in RankGPT-4, instructing it to think thoughtfully and systematically during ranking. We observe that CoT leads to significant performance gains by enabling more reasoning in relevance judgments. ‘RankGPT-4 w. CoT’ achieves consistent performance improvements, with an average gain of  $\{1.99, 2.15, 1.97\}$  in terms of  $nDCG@\{1, 5, 10\}$  compared to RankGPT-4 on the three datasets.

*Format Requirement.* We incorporate format requirement prompts to instruct RankGPT-4 to adhere to the specific rank format and ensure no repeated or missing passages in the rank list, facilitating a more convenient extraction of the final rank results. As shown in Table 8, the format requirement instructions marginally improve reranking performance in most cases, yielding an average gain of  $\{1.26, 0.69, 1.05\}$  in terms of  $nDCG@\{1, 5, 10\}$  on TREC-DL20, COVID, and NovelEval.

## 5 Conclusion

In this paper, we introduce RANKFLOW, a novel multi-role collaborative reranking workflow for Large Language Models (LLMs), comprising four roles: Rewriter, Answerer, Summarizer, and Reranker. Our extensive empirical results showcase the significant effectiveness of RANKFLOW, which consistently surpasses state-of-the-art methods on various datasets. Furthermore, we investigate the individual contributions of each role and the impacts of prompt designs in listwise passage rerank-

ing, providing valuable insights for future research. The robust performance of RANKFLOW highlights the potential of an effective reranking workflow in the era of LLMs.

## 6 Limitations

A limitation of this paper is that the RANKFLOW framework is comprehensively investigated only on the GPT4 model, leaving other closed-source and open-source models less explored. Future work should include investigating the effect of the RANKFLOW framework on models such as Claude 3, recently released by Anthropic.

## 7 Ethics Statement

In our research, we strictly adhere to the ACM Code of Ethics by ensuring that all datasets and models used are publicly accessible. However, we recognize the potential risks and harms associated with LLMs, such as the generation of harmful, offensive, or biased content. Furthermore, LLMs are known to generate incorrect information, a phenomenon commonly referred to as hallucinations. It is important to note that the models investigated in this study are not exempt from these limitations. Furthermore, the GPT4 used in this work are licensed under the terms of OpenAI.

## References

- Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, page 189.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Marwah Alaofi, Luke Gallagher, Mark Sanderson, Falk Scholer, and Paul Thomas. 2023. Can generative llms create query variants for test collections? an exploratory study. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1869–1873.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.
- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- R Meredith Belbin and Victoria Brown. 2022. *Team roles at work*. Routledge.
- Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.
- Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*.
- Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2):1–60.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.
- Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.
- Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

- Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.
- Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.
- Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 77–86.
- Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.
- Can Jin, Tong Che, Hongwu Peng, Yiyuan Li, and Marco Pavone. 2024a. [Learning from teaching regularization: Generalizable correlations should be easy to imitate](#). *Preprint*, arXiv:2402.02769.
- Can Jin, Tianjin Huang, Yihua Zhang, Mykola Pechenizkiy, Sijia Liu, Shiwei Liu, and Tianlong Chen. 2023. [Visual prompting upgrades neural network sparsification: A data-model perspective](#). *Preprint*, arXiv:2312.01397.
- Can Jin, Hongwu Peng, Shiyu Zhao, Zhenting Wang, Wujiang Xu, Ligong Han, Jiahui Zhao, Kai Zhong, Sanguthevar Rajasekaran, and Dimitris N Metaxas. 2024b. Apeer: Automatic prompt engineering enhances large language model reranking. *arXiv preprint arXiv:2406.14449*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.
- Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.
- Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohhta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.
- Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023a. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.
- Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023b. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*.
- Kelong Mao, Zhicheng Dou, Fengran Mo, Jiewen Hou, Haonan Chen, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1211–1225.
- Donald Metzler and W Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479.
- Donald Metzler and W Bruce Croft. 2007. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 311–318.
- Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.
- Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019a. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.
- Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

- Ronak Pradeep, Sahel Sharifmoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.
- Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.
- Devendra Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3781–3797.
- Weiwei Sun, Zheng Chen, Xinyu Ma, Lingyong Yan, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023a. Instruction distillation makes large language models efficient zero-shot rankers. *arXiv preprint arXiv:2311.01555*.
- Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023b. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Michael Wooldridge and Nicholas R Jennings. 1998. Pitfalls of agent-oriented development. In *Proceedings of the second international conference on Autonomous agents*, pages 385–391.
- Huiwen Wu, Xiaohan Li, Deyi Zhang, Xiaogang Xu, Jiafei Wu, Puning Zhao, and Zhe Liu. 2024. Cg-fedllm: How to compress gradients in federated fine-tuning for large language models. *arXiv preprint arXiv:2405.13746*.
- Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933*.
- Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. Expertprompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*.
- Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, S Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *International Conference on Learning Representations*.
- Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410.
- Qixin Zhang, Zengde Deng, Zaiyi Chen, Haoyuan Hu, and Yu Yang. 2022. Stochastic continuous submodular maximization: Boosting via non-oblivious function. In *International Conference on Machine Learning*, pages 26116–26134. PMLR.
- Qixin Zhang, Zengde Deng, Zaiyi Chen, Kuangqi Zhou, Haoyuan Hu, and Yu Yang. 2023. Online learning for non-monotone dr-submodular maximization: From full information to bandit feedback. In *International Conference on Artificial Intelligence and Statistics*, pages 3515–3537. PMLR.
- Puning Zhao, Lifeng Lai, Li Shen, Qingming Li, Jiafei Wu, and Zhe Liu. 2024. A huber loss minimization approach to mean estimation under user-level differential privacy. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825.

Tong Zhou, Jiahui Zhao, Yukui Luo, Xi Xie, Wujie Wen, Caiwen Ding, and Xiaolin Xu. 2024. Adapi: Facilitating dnn model adaptivity for efficient private inference in edge computing. *CoRR*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313.

## A Cost Analysis

We compared the average token cost, running time, and USD cost per query of RANKFLOW to RankGPT. In RANKFLOW, we can preserve the new queries and summarized passages locally and reuse them in later reranking, helping to save running time and financial budget, which we name “RANKFLOW w. Local Passage”. The results are shown in Table 7. While RANKFLOW costs 10.63% more tokens and USD than RankGPT, it is 7.32% faster and achieves an average nDCG@10 improvement of nearly 2% on TREC, BEIR, and NovelEval as shown in Tables 1, Table 2, and Table 3. Moreover, using “RANKFLOW w. Local Passages” can further reduce computational cost and financial budget, saving 26.83% tokens and USD and being 32.26% faster than RankGPT.

| Dataset                    | Tokens | Time (s) | USD   |
|----------------------------|--------|----------|-------|
| RankGPT-4                  | 19890  | 82       | 0.596 |
| RANKFLOW                   | 21938  | 76       | 0.658 |
| RANKFLOW w. Local Passages | 12027  | 60       | 0.361 |

Table 7: Average token cost, running time, and USD cost per query on TREC-DL.

## B Additional Results

**Additional Results of Reranker.** The effects of different prompt designs in the RANKFLOW Reranker on three datasets are shown in Table 8. Each component enhances performance, while RANKFLOW consistently achieves the best results.

| Method                          | TREC-DL20                | COVID                    | NovelEval                |
|---------------------------------|--------------------------|--------------------------|--------------------------|
| RankGPT-4                       | 79.73/73.15/70.14        | 88.25/86.67/83.98        | 92.86/86.10/89.18        |
| RankGPT-4 w. Relevance Standard | 80.45/73.94/70.62        | 91.67/88.04/85.38        | 92.89/89.57/92.49        |
| RankGPT-4 w. CoT                | 80.25/74.13/70.42        | 93.33/88.51/85.51        | 95.24/89.74/93.28        |
| RankGPT-4 w. Format Requirement | 79.63/73.68/70.38        | 91.33/87.11/84.70        | 93.65/87.19/91.24        |
| RANKFLOW                        | <b>82.41/75.68/71.80</b> | <b>96.00/89.48/85.77</b> | <b>97.62/91.79/94.21</b> |

Table 8: Results (nDCG@{1,5,10}) of utilizing different prompting strategies in the Reranker.

## C Specilization and Prompt for Rewriter

### system

You are an AI retrieval assistant, skilled at rewriting user queries to enhance their suitability for retrieval tasks and optimizing compatibility with retrieval systems like BM25.

### user

Rewrite the following user query into a clear, specific, and formal request suitable for retrieving relevant information from a list of passages. Keep in mind that your rewritten query will be sent to rerank system, which does relevance search for retrieving documents.

### assistant

Kindly provide the query you would like me to rewrite.

### user

{query}

## D Specilization and Prompt for Answer

**system**

You are an AI retrieval expert, skilled at providing detailed and relevant answers to user queries.

**user**

Compose a passage to address the following user query effectively.

**assistant**

Please provide the query for which you would like an answer.

**user**

{query}

## E Specilization and Prompt for Summarizer

**system**

You are an AI assistant who is good at summarizing passages the user provides you.

**user**

I will provide you a passage. Summarize the passage to make it suit for a passage retrieval task which means the summarized passages can better reflect the information and the relevance to a giving query than the original passage.

Passage: {passage}

## F Specilization and Prompt for Reranker

### **system**

You are RankGPT, an intelligent assistant that ranks passages based on their relevance to a given query. Apply the following relevance criteria when ranking passages:

1. Perfectly relevant: The passage directly addresses the query and contains the exact answer.
2. Highly relevant: The passage contains information related to the query, but the answer may be unclear or surrounded by unrelated details.
3. Related: The passage is related to the query but does not provide an answer.
4. Irrelevant: The passage is not connected to the query.

### **user**

Please rank the {num} passages I will provide, each identified by a number in brackets []. Evaluate the passages based on their relevance to the following query: {query}. List the passages in descending order of relevance, with the most relevant passages at the top. Use [rankstart] to begin the ranking and [rankend] to conclude it. Ensure that no passages are missed or repeated in the ranking. The output format should be:

[rankstart] [] > [] [rankend],

For example,

[rankstart] [1] > [2] [rankend]. Follow the ranking format diligently and avoid missing or repeating passages. Approach the task systematically and thoughtfully.

### **assistant**

Understood, I will adhere to the ranking format. Please provide the passages for evaluation and ranking.

### **user**

[1] {Passage1}

### **assistant**

Received passage [1]

### **user**

[2] {Passage2}

### **assistant**

Received passage [2]

(more passages) ...

### **user**

Search Query: {query}.

Rank the {num} passages above based on their relevance to the search query.

## G Different prompt strategies for the Reranker

### G.1 Relevance Standard

**system**

You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query. Your relevance judgment should follow the following standard:

Perfectly relevant: The passage is dedicated to the query and contains the exact answer.

Highly relevant: The passage has some answer for the query, but the answer may be a bit unclear, or hidden amongst extraneous information.

Related: The passage seems related to the query but does not answer it.

Irrelevant: The passage has nothing to do with the query.

**user**

I will provide you with {num} passages, each indicated by number identifier [].

Rank the passages based on their relevance to query: {query}.

**assistant**

Okay, please provide the passages.

**user** [1] {Passage1}

**assistant** Received passage [1]

(more passages) ...

**user**

Search Query: {query}.

Rank the {num} passages above based on their relevance to the search query. The passages should be listed in descending order using identifiers. The most relevant passages should be listed first. The output format should be [] > [], e.g., [1] > [2]. Only response the ranking results, do not say any word or explain.

### G.2 CoT

**system**

You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

**user**

I will provide you with {num} passages, each indicated by number identifier [].

Rank the passages based on their relevance to query: {query}.

**assistant**

Okay, please provide the passages.

**user** [1] {Passage1}

**assistant** Received passage [1]

(more passages) ...

**user**

Search Query: query.

Please evaluate the {num} passages provided and rank them based on their relevance to the search query. Arrange the passages in descending order of relevance by using their identifiers. Place the most relevant passage at the beginning of the list. Start your ranking with [rankstart] and end with [rankend]. The output format should be [rankstart] [] > [] [rankend],

For example,

[rankstart] [1] > [2] [rankend]. Proceed thoughtfully and systematically.

### G.3 Format Requirement

**system**

You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

**user**

I will provide you with {num} passages, each indicated by number identifier [].

Rank the passages based on their relevance to query: {query}.

**assistant**

Okay, please provide the passages.

**user** [1] {Passage1}

**assistant** Received passage [1]

(more passages) ...

**user**

Evaluate and rank the passages based on their relevance to the query:

{query}.

List the passages in descending order of relevance using their identifiers, placing the most relevant passages at the top. Begin the ranking with [rankstart] and end with [rankend]. Ensure that no passages are missed or repeated in the ranking. The output format should be:

[rankstart] [] > [] [ranked],

For example,

[rankstart] [1] > [2] [ranked].

Adhere to the ranking format and avoid missing or repeating passages while ranking.

## H Examples

### H.1 Query

The qualitative examples of RANKFLOW Rewriter and Answerer are shown in Table 9.

### H.2 Passage

The qualitative examples of RANKFLOW Summarizer are shown in Table 10.

|                 |   |
|-----------------|---|
| original query  | what is wifi vs bluetooth   |
| rewritten query | Provide a comparative analysis of Wi-Fi and Bluetooth, detailing their differences and similarities.  |
| answer          | Wi-Fi and Bluetooth are both wireless communication technologies but they are designed for different purposes and have distinct characteristics. Wi-Fi, short for Wireless Fidelity, is primarily used to provide high-speed internet access. It allows devices like computers, smartphones, or smart TVs to connect to the internet or each other wirelessly within a particular area, known as a Wi-Fi network. Bluetooth, on the other hand, is used for connecting devices to each other over short distances, typically up to 30 feet. It's commonly used for connecting peripherals like keyboards, mice, or headphones to a computer or for connecting a hands-free headset to a mobile phone. In summary, while both Wi-Fi and Bluetooth are wireless technologies that allow devices to communicate without physical connections, they differ in their range, power consumption, data transfer speed, and typical use cases.   |
| new query       | Provide a comparative analysis of Wi-Fi and Bluetooth, detailing their differences and similarities.<br>Provide a comparative analysis of Wi-Fi and Bluetooth, detailing their differences and similarities.<br>Provide a comparative analysis of Wi-Fi and Bluetooth, detailing their differences and similarities.<br>Wi-Fi and Bluetooth are both wireless communication technologies but they are designed for different purposes and have distinct characteristics. Wi-Fi, short for Wireless Fidelity, is primarily used to provide high-speed internet access. It allows devices like computers, smartphones, or smart TVs to connect to the internet or each other wirelessly within a particular area, known as a Wi-Fi network. Bluetooth, on the other hand, is used for connecting devices to each other over short distances, typically up to 30 feet. It's commonly used for connecting peripherals like keyboards, mice, or headphones to a computer or for connecting a hands-free headset to a mobile phone. In summary, while both Wi-Fi and Bluetooth are wireless technologies that allow devices to communicate without physical connections, they differ in their range, power consumption, data transfer speed, and typical use cases. |
| original query  | ms symptoms ms  |
| rewritten query | Provide detailed information on the symptoms associated with Multiple Sclerosis (MS).   |
| answer          | Multiple Sclerosis (MS) is a chronic disease that affects the central nervous system, particularly the brain and spinal cord. Symptoms of MS can vary widely and may change over time, but some of the most common include: 1. Fatigue: This is one of the most common symptoms of MS, affecting about 80% of people. It can significantly interfere with a person's ability to function at work or at home. 2. Difficulty Walking: MS can affect coordination and balance, making walking difficult. This is often due to numbness in the legs, muscle weakness, or problems with balance and coordination. 3. Numbness or Tingling: MS often starts with the sensation of numbness or tingling in various parts of the body, most commonly in the face, body, or extremities. It's important to note that these symptoms can also be caused by conditions other than MS, and experiencing one or more of these symptoms does not necessarily mean you have MS.  |
| new query       | Provide detailed information on the symptoms associated with Multiple Sclerosis (MS).<br>Provide detailed information on the symptoms associated with Multiple Sclerosis (MS).<br>Provide detailed information on the symptoms associated with Multiple Sclerosis (MS).<br>Multiple Sclerosis (MS) is a chronic disease that affects the central nervous system, particularly the brain and spinal cord. Symptoms of MS can vary widely and may change over time, but some of the most common include: 1. Fatigue: This is one of the most common symptoms of MS, affecting about 80% of people. It can significantly interfere with a person's ability to function at work or at home. 2. Difficulty Walking: MS can affect coordination and balance, making walking difficult. This is often due to numbness in the legs, muscle weakness, or problems with balance and coordination. 3. Numbness or Tingling: MS often starts with the sensation of numbness or tingling in various parts of the body, most commonly in the face, body, or extremities. It's important to note that these symptoms can also be caused by conditions other than MS, and experiencing one or more of these symptoms does not necessarily mean you have MS.                   |

Table 9: Examples of the original query, rewritten query, answer, and new query.

|                    |   |
|--------------------|---|
| original passage   | 5. Cancel. A flea can live up to a year, but its general lifespan depends on its living conditions, such as the availability of hosts. Find out how long a flea's life cycle can last with tips from a pet industry specialist in this free video on fleas and pest control.Part of the Video Series: Flea Control.ancel. A flea can live up to a year, but its general lifespan depends on its living conditions, such as the availability of hosts. Find out how long a flea's life cycle can last with tips from a pet industry specialist in this free video on fleas and pest control. Part of the Video Series: Flea Control.   |
| summarized passage | The lifespan of a flea, which can be up to a year, is influenced by its living conditions, particularly the availability of hosts. This information, along with tips on flea control, is provided by a pet industry specialist in a free video series.  |
| original passage   | Surprise! The federal government's Digital Analytics Program (DAP) has found that Windows is the most popular end-user operating system with 58.4 percent of all government web site visitors. What's more interesting though is what versions of Windows we're using and what's number two and three behind Windows. The government's data, derived from 1.39 billion web visits to more than 37 government agencies over the last 90 days, reveals that the second place operating system is Apple iOS with 16.4 percent. It's followed close by Android with 13.9 percent.   |
| summarized passage | The Digital Analytics Program (DAP) of the federal government reveals that Windows is the most used operating system by government website visitors, accounting for 58.4% of all users. The data, collected from 1.39 billion web visits across 37 government agencies in the past 90 days, also shows that Apple iOS and Android are the second and third most popular operating systems, with 16.4% and 13.9% usage respectively.   |
| original passage   | Pitch In For The Big Three: You know the 80/20 rule, right? Basically, 80 percent of your wedding's costs will come from 20 percent of the decisions. So the trick is to help your fiancée with the right 20 percent. That means pitching in on the Big Three: the guest list, the date and the venue.For these, you need to do more than just say, "Sure, honey, that sounds fine..itch In For The Big Three: You know the 80/20 rule, right? Basically, 80 percent of your wedding's costs will come from 20 percent of the decisions. So the trick is to help your fiancée with the right 20 percent. That means pitching in on the Big Three: the guest list, the date and the venue. |
| summarized passage | The passage emphasizes the importance of the 80/20 rule in wedding planning, where 80% of the costs are determined by 20% of the decisions. It advises the reader to assist their fiancée with these crucial decisions, specifically focusing on the "Big Three": the guest list, the date, and the venue.  |
| original passage   | 1 Pelvic inflammatory disease many not produce any symptoms. 2 In other cases it can cause fever, abdominal or pelvic pain, vaginal discharge, painful urination, or painful sexual intercourse.3 Pelvic inflammatory disease is treated with antibiotics. Pelvic inflammatory disease many not produce any symptoms. 2 In other cases it can cause fever, abdominal or pelvic pain, vaginal discharge, painful urination, or painful sexual intercourse.   |
| summarized passage | Pelvic inflammatory disease (PID) may be asymptomatic or cause symptoms like fever, abdominal or pelvic pain, vaginal discharge, painful urination, and discomfort during sexual intercourse. The treatment for PID typically involves antibiotics.   |

Table 10: Examples of the original passage and summarized passage.