Refine, Compress, Re-rank: Improving Listwise Re-ranking with Large Language Models

Vineet Sunkavalli and **Veda Kailasam** and **Shagufta Anjum**
University of Illinois, Urbana-Champaign
vineets3@illinois.edu, vedak2@illinois.edu, sanjum8@illinois.edu

Abstract

Recent advancements in Large Language Models (LLMs) have significantly improved their application in information retrieval, particularly in listwise passage ranking. However, these models often face challenges due to initial data order sensitivity and constraints imposed by context window sizes. Our paper introduces a multi-stage re-ranking approach that begins by refining initial rankings using pre-trained models. It then employs natural language processing techniques such as summarization, keyword extraction, title generation, and topic modeling to compress text. This allows the entire passage set to fit within a single LLM context window, eliminating the need for a sliding window approach. This method addresses the core challenges of initial order sensitivity and context window limitations, thereby enhancing ranking accuracy. Our findings suggest that this approach could set a new standard for efficiency and effectiveness in passage ranking tasks using LLMs.

1 Introduction

Modern information retrieval (IR) systems rely on a multi-stage approach called passage re-ranking to first retrieve, then identify the most relevant passages and order them based on their relevance. This process has typically involved evaluating each document-query pair individually, utilizing what is known as a point-wise approach. Typically, these models are given a document-query pair and simply output relevance scoring for said pair directly. Some examples of point-wise rankers are MonoBERT(Nogueira et al., 2019) and MonoT5(Nogueira et al., 2020). There are also pair-wise approaches, such as DuoBERT and DuoT5, which compared two passages at a time in a fashion similar to a sorting algorithm; however, this is an expensive approach with respect to both time and compute.

In recent years, advancements in deep learning have led to the advent of Large Language Models (LLMs). Unlike prior language models, LLMs are highly adaptable and capable of performing at or near state of the art in most, if not all, Natural Language Processing(NLP) Tasks. More importantly, they are capable of doing so while being capable of performing tasks and outlining their reasoning and thought process in natural language as well. The use of LLMs has proliferated in several NLP tasks, with passage re-ranking being no exception.

Early approaches to LLM Re-ranking employed point-wise approaches, with some notable examples being UPR (Sachan et al., 2023a) and HELM (Liang et al., 2023). These approaches take a document and document-query pair respectively and attempt to form a natural language representation of their relevance. While this approach can be strong, it does not take other passages into account, leading to a potential area of improvement.

RankGPT (Sun et al., 2023) overcomes the flaws of prior point-wise approaches. It employs what is known as a list-wise approach, enabling the model to relatively rank documents with comparison to other passages. In addition, less calls are needed and more passages are read than a pairwise approach.

However, RankGPT itself suffers from several issues:

- The initial order of documents can greatly influence the final ranking due to position bias exhibited by LLMs. If the initial order is bad, that can negatively affect the ranking. (Tang et al., 2024)
- There are constraints on the maximum input length that LLMs can process at one time. In order to circumvent this limitation, the sliding window method is widely used. However, this can still affect accuracy as not all passages are available when re-ranking, leading to

inevitable oversights.

- It tends to be computationally quite intensive, as compared to point-wise reranking.
- Documents in the middle of a long list sometimes get "lost" or ignored as models tend to focus on the documents at the start and end of the list (Tang et al., 2023)

Our aim is to resolve the inefficiencies and issues caused by the aforementioned flaws present in listwise approaches. To do so, we plan to address the two largest issues: (i) initial ordering; and, (ii) sliding window length. We also aim to improve computational cost and contextual understanding.

To resolve these issues, we propose a multi-stage pipeline: Refine-Compress-Re-rank. Our key contributions with our pipeline are:

- **Refine:** A MonoBERT model used a simple pointwise re-ranker in order to improve the initial order of documents given to our re-ranker, with a goal of being computationally cheap as well.
- **Compress:** A Flan-T5-XL(Chung et al., 2022a) model used to compress passages to a single sentence, greatly decreasing the number of tokens per passage and enabling us to increase sliding window length as well as enable better contextual understanding.
- **Re-rank:** A GPT-3.5.-Turbo model used to rerank passages, similar to RankGPT but without a sliding window.

2 Related Work

The problem of improving passage ranking with Large Language Models (LLMs) has seen substantial research interest, particularly focusing on refining pre-trained models, re-ranking methodologies, and prompt engineering. Pre-trained models like BERT have been effectively utilized for passage ranking, leading to the development of architectures such as MonoBERT and DuoBERT. MonoBERT, which applies BERT in a pointwise manner, scores passages independently of each other, while DuoBERT employs a pairwise approach to compare two passages at a time (Nogueira and Cho, 2019). Despite their effectiveness, these models present trade-offs in latency and computational costs. MonoBERT scales linearly with the number of candidate passages, while DuoBERT exhibits exponential scaling (Nogueira et al., 2019), making it challenging to scale to large datasets.

To tackle the limitations of existing methods, Unsupervised Passage Retriever (UPR) (Gao et al., 2022) generates questions from passages using LLMs, then computes log probability scores to rank passages based on the generated question and query relevance. This innovative approach allows for improved passage retrieval in zero-shot settings, but incurs computational overhead due to the high complexity of question generation.

Recent work by (Sachan et al., 2023b) demonstrated that pairwise ranking prompting with LLMs can effectively address positional bias inherent in listwise approaches. By comparing each document pair in both possible orders, pairwise ranking reduces bias due to initial ordering. However, the $O(n^2)$ complexity of this approach can make it computationally prohibitive for large-scale passage ranking tasks.

Progressive re-ranking methods refine passage lists iteratively but suffer from initial document order bias and sliding window limitations (Yang et al., 2023). Relevant passages can become "trapped" in local blocks, reducing the chance of reaching the top ranks. This limitation is addressed by RankZephyr (Jain et al., 2023) and Rank-without-GPT (Xu and Lin, 2023), which advocate for high-quality listwise training data and open-source LLMs for effective re-ranking (Pradeep et al., 2023).

While pre-trained models like BERT have revolutionized passage ranking, challenges like initial ordering bias and context window limitations remain significant hurdles. Our proposed multi-stage approach addresses these issues by refining initial rankings with pre-trained models and compressing text using NLP techniques, setting a new benchmark for efficient passage ranking tasks with LLMs.

3 Methodology

Our proposed multi-stage re-ranking pipeline comprises three components: (1) Refinement, where we use a pre-trained, cost-effective ranking model to improve the initial ranking; (2) Compression, where we apply text-shortening techniques to fit all documents within a single LLM context window;

and (3) Re-ranking, which leverages the refined and compressed list to produce the final ranking. Figure 2 provides an illustrative overview of our proposed Multi-Stage Re-Ranking Pipeline.

3.1 Refine: Re-ordering with BERT

In our methodology, the first step involves refining the initial document ranking using MonoBERT, a BERT-based model optimized for passage ranking. MonoBERT is specifically adapted for query-based document reranking, offering a computationally efficient alternative to more resource-intensive Large Language Models (LLMs). Its ability to provide significant ranking improvements at a reduced cost makes it well-suited as a "pre-reranker" in our multi-stage approach.

To solve the issue of sub-optimal initial order of the retrieved passages, MonoBERT helps to improve the ordering given to our Listwise Ranker. MonoBERT, introduced by (Nogueira and Cho, 2019) in their seminal work Passage Re-ranking with BERT, is designed to refine the initial ranking of documents generated by a traditional IR model such as BM25. MonoBERT takes a single document-query pair and generates a relevance score, meaning that it is unaffected by the initial order of the documents given by BM25. The scoring function for a given query q and passage p can be expressed as:

$$\text{S}(q, p) = \text{MonoBERT}(q, p)$$

In addition, since MonoBERT is a smaller pre-trained model, it means that we aren't employing a solution that is much more expensive than a single-stage Listwise Ranker.

While DuoBERT, also introduced by (Nogueira et al., 2019) in their subsequent paper Multi-Stage Document Ranking with BERT, is another BERT-based model for passage ranking, it requires pairwise comparisons between document pairs, significantly increasing computational costs. The marginal performance gains from DuoBERT do not justify this higher expense, especially when compared to MonoBERT's efficiency. Empirically, MonoBERT has been shown to provide a 10% improvement in NDCG@10 over BM25, making it a more suitable choice for initial ranking refinement in our methodology. The pointwise scoring process by MonoBERT ensures that the subsequent compression and re-ranking stages are resilient to initial list order sensitivity.

Refine + Re-rank: After the initial refinement with MonoBERT, the next stage in our methodology involves re-ranking the refined list using a Large Language Model (LLM) in a listwise fashion. This stage leverages the improved ordering provided by MonoBERT and applies sophisticated natural language processing techniques to further enhance the ranking accuracy.

3.2 Compression: Compressing with FlanT5

To address the context window limitations of Large Language Models (LLMs) and ensure efficient re-ranking, we explored various text compression techniques before final re-ranking (Figure 4). By leveraging natural language generation models like FlanT5-XL (Chung et al., 2022b) and GPT-3.5 (OpenAI, 2022), we aimed to condense the essential information of each passage into a more manageable format. This stage ensures that all passages fit within a single context window, eliminating the need for a sliding window approach.

3.2.1 Compression Techniques

We experimented with a variety of compression techniques and compared their performance:

Title Generation: Ask the LLM to write a concise title describing the passage. Titles provide a quick summary of the main themes and guide the LLM's understanding during re-ranking. Title Compression involves condensing the main themes and content of a text into a short, informative title. This process enables the LLM to focus on the most relevant aspects of the document. After the initial retrieval, the top-K passages are processed through a prompt that summarizes or distills the main content. Compressed titles distill the passages to their essence, potentially eliminating the need for sliding windows since the inputs will not exceed the model's maximum input length. Figure 3 shows the precise input prompt used for title compression. For title generation using FlanT5-XL, the model generates a summary title T for each passage p given a query q:

$$T = \text{FlanT5-XL}(q, p)$$

Title + Topics: This approach incorporates topic modeling into title compression. Each passage is first processed to identify its main content, which is then tokenized and converted into a bag-of-words representation. Using an LDA model (Jelodar et al., 2018), we extract the top topics for each passage and format them into a string summarizing these

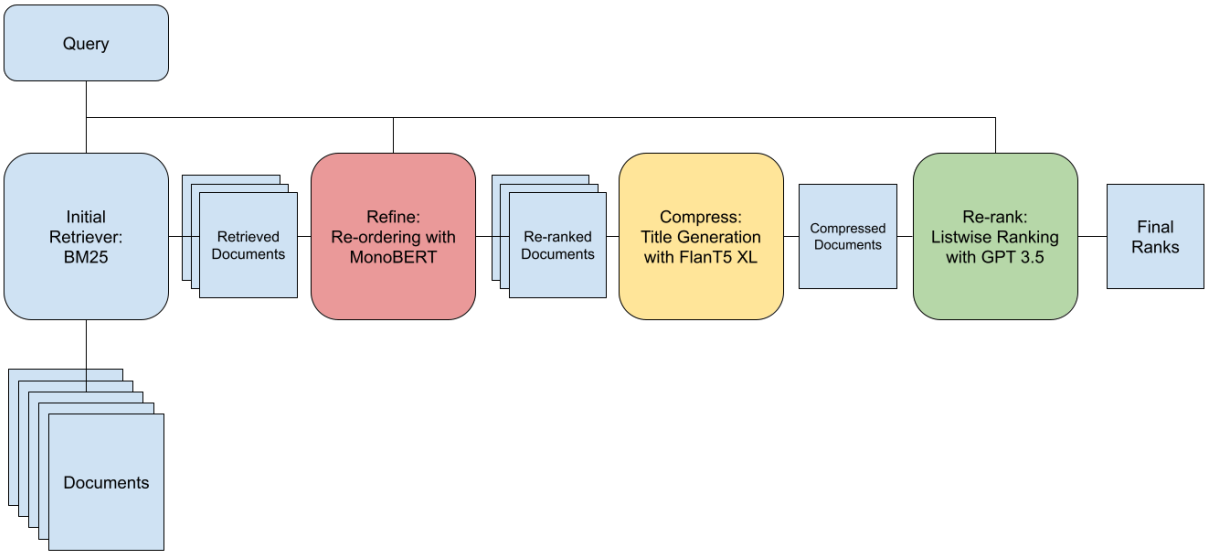

Figure 1: Multi-Stage Re-Ranking Pipeline

themes. This enriched passage information is submitted to GPT-3.5, which generates concise, informative titles based on both the passage content and the identified topics. The combination of titles and topics improves the LLM's understanding, leading to accurate re-ranking results.

Keyword Extraction: In the keyword extraction approach, we experimented with two methods: KeyBERT (Koloski et al., 2022), a pre-trained model that uses BERT embeddings, and FlanT5-XL, a generative model with a custom prompt. The LLM-based extraction proved more effective, especially for the TREC datasets. We tested extracting different numbers of keywords (1, 5, 10, and 20) and found that 10 keywords struck a good balance between ranking performance and fitting within the LLM's input constraints. The extracted keywords were then combined with titles to form new input "passages" for GPT-3.5, which significantly improved the re-ranking accuracy. Keyword extraction involves computing the cosine similarity between the passage embeddings e_p and keyword embeddinsgs e_k:

$$\cos(e_p, e_k) = \frac{e_p \cdot e_k}{\|e_p\|\|e_k\|}$$

3.2.2 Compress + Re-Rank

The compress + re-rank approach begins with a basic text retrieval method like BM25 and incorporates advanced neural models like FlanT5-XL and GPT-3.5 to enhance document understanding and contextual relevance. This multi-stage approach optimizes both the quality of search results and the efficiency of the retrieval process, transitioning from broad initial retrieval to fine-grained re-ranking.

By employing this method, we are able to circumvent the issue of token limits as well as improve the LLMs ability to attend to the context of 100 passages. By using an LLM other than GPT, we are able to minimize the number of calls made to GPT and reduce the cost of our pipeline as opposed to compressing the passages with GPT.

3.3 Re-Rank: Listwise Re-Ranking with GPT-3.5

The final stage of our pipeline involves re-ranking the compressed passages using GPT-3.5 in a listwise fashion. Inspired by the work of (Li et al., 2023) in their RankGPT paper, we leverage GPT-3.5's comprehensive understanding and contextual processing capabilities to deliver a highly accurate final ranking. In our approach, each passage is represented using the compressed information generated in the previous stages. This includes titles created with FlanT5-XL, along with either keywords or topics extracted from the original passages. The titles provide concise summaries, while the keywords and topics offer additional context that enhances GPT-3.5's understanding.

To guide GPT-3.5 in understanding the query, passage context, and relationships between passages, we carefully designed a listwise prompt. The prompt structure includes a brief description of the user query, a list of compressed titles with accompanying keywords or topics, and a clear instruction to rank the passages based on their relevance to the

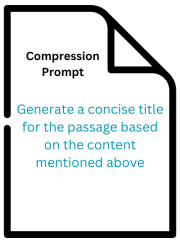

Figure 2: Title Generation Prompt

query.

4 Experiments

4.1 Datasets

We chose to run our experiments on two benchmark datasets, including TREC-DL (Craswell et al., 202(Craswell et al., 2021) and BEIR (Thakur et al., 2021). Both benchmark datasets are widely used in information retrieval research. TREC datasets consist of text documents along with relevance judgments, which are used to evaluate the effectiveness of different information retrieval systems. We use test sets from the 2019 and 2020 competitions: (i) TREC-DL19 which contains 43 queries, (ii) TREC-DL20 which contains 54 queries. BEIR contains data for retrieval tasks across diverse domains. We use four tasks from BEIR: (i) Covid: retrieves scientific articles for COVID-19 related questions; 50 queries. (ii) Touche is an argument retrieval dataset; 49 queries. (iii) Signal retrieves relevant tweets for a given news title; 97 queries. (iv) News retrieves relevant news articles for news headlines; 54 queries. Together, these six tasks cover a wide variety of passage topics and queries, enabling us to gather a holistic view of our model's performance.

4.2 Baselines

For our baselines, we chose to compare our results to two models, BM25 and RankGPT.

BM25 is a lexical retriever that is typically used as an initial retriever in several multistage pipelines. BM25 orders passages by weighing the term-frequency and inverse document frequency of the keywords found in both the question and passage (Robertson and Zaragoza, 2009). This configuration is show to be a robust method of retrieving (Ma, 2021) and is a staple baseline for several IR papers, which is why chose to include it in our results.

RankGPT is the initial implementation of Permutation Generation which we aimed to improve upon, which is why we include it as our second baseline. RankGPT performs relative re-ranking on a set of passages with a sliding window configuration, reading 20 passages at a time with a stride of 10. We use GPT-3.5-Turbo when evaluating with RankGPT.

4.3 Results

In this section, we provide a detailed evaluation of our multi-stage re-ranking methodology across several datasets. We compare the performance of different ranking strategies using nDCG scores at varying cutoff points (1, 5, and 10). The datasets used include DL-19, DL-20, COVID, News, Touche, and Signal. TODO: Include quantitative and qualitative results

4.3.1 Quantitative Evaluation

Table 1 presents the evaluation results obtained across different datasets, including DL-19, DL-20, COVID, News, Touche, and Signal. We compare several ranking strategies such as BM25 initial retrieval, MonoBERT, listwise re-ranking with sliding window (baseline), and our proposed combinations of MonoBERT and FLAN-T5 models. For each dataset, we report the nDCG scores at cutoff points 1, 5, and 10.

- **MonoBERT + Listwise Re-Ranking:** Incorporating MonoBERT generally enhances performance further, achieving significant improvements over BM25 initial retrieval across most datasets. For instance, in the DL-19 dataset, MonoBERT achieves an nDCG@1 score of 79.07, compared to BM25's score of 54.26. However, in some cases, MonoBERT falls slightly below the baseline listwise re-ranking, particularly in the News and Touche datasets.

- **Title Compression with FLAN-T5 + Listwise Re-Ranking:** This method works well in general and consistently outperforms BM25 initial retrieval. It also surpasses more sophisticated re-ranking strategies involving MonoBERT or listwise re-ranking with sliding window in some cases. For example, in the COVID dataset, Title Compression with FLAN-T5 achieves an nDCG@1 score of 94.00 and nDCG@5 score of 85.64, which is

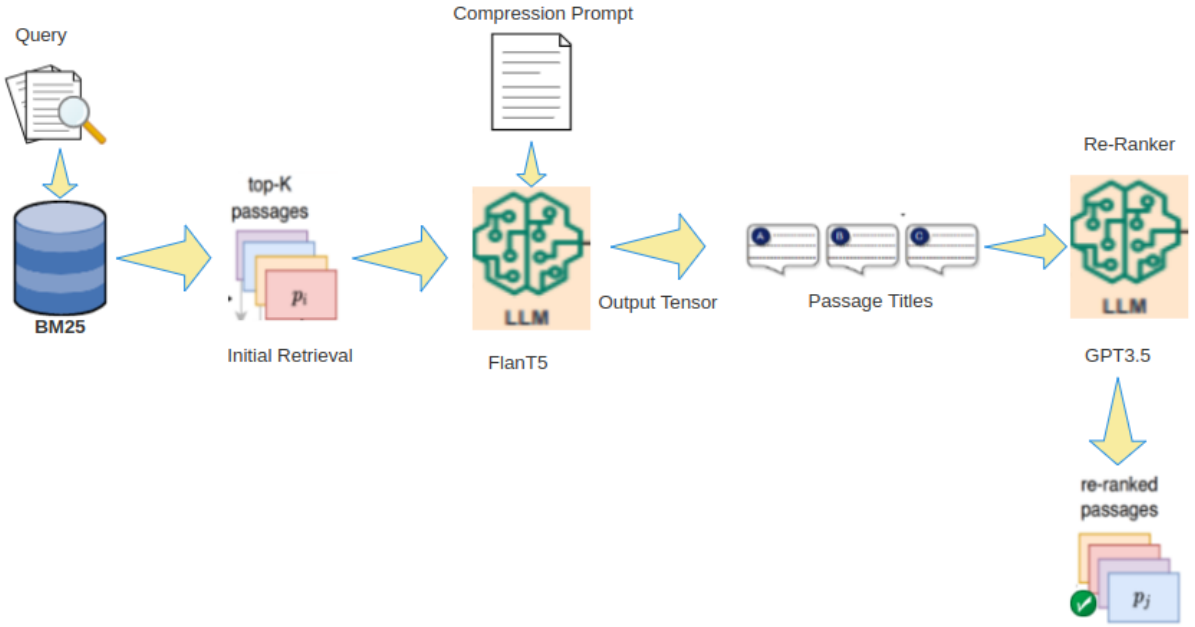

Figure 3: Compression Pipeline

the highest score among all methods, indicating that title compression works particularly well for this dataset.

- **MonoBERT + Listwise Re-Ranking with Sliding Window:** This combination shows significant improvements over BM25, particularly in the COVID dataset, where it achieves an nDCG@1 score of 92.00, compared to BM25's score of 68.00. However, it does not consistently outperform other methods across all datasets.

- **MonoBERT + Compression with FLAN-T5 + Listwise Re-Ranking:** This combination generally improves the ranking results but does not consistently outperform methods involving only MonoBERT or only title compression. For instance, in the DL-19 dataset, it achieves an nDCG@1 score of 65.12, which is lower than the score achieved by MonoBERT alone (79.07).

- **COVID Dataset:** This dataset shows the highest scores across nearly all methods, particularly when FLAN-T5 is involved, suggesting that title compression works especially well for this dataset. The combination of MonoBERT with Title Compression and Listwise Re-Ranking achieves an nDCG@1 score of 90.00.

- **DL-19 and DL-20:** In the DL-19 and DL-20 datasets, MonoBERT-enhanced methods generally perform well, indicating that the sophistication of MonoBERT suits the types of queries or document characteristics in these datasets. For instance, MonoBERT achieves an nDCG@5 score of 70.35 in DL-20, compared to BM25's score of 50.67.

- **News, Touche, and Signal Datasets:** These datasets appear more challenging for title compression methods, which sometimes underperform even the basic BM25 retrieval, particularly at lower ranks. In the News dataset, Title Compression with FLAN-T5 achieves an nDCG@1 score of 54.53, which is only marginally better than BM25's score of 42.69.

4.3.2 Qualitative Evaluation

Traditional approaches in this domain often involve fine-tuning the models on specific datasets or making architectural modifications, which fail to address the inherent limitations of the listwise ranking approach or the issue of positional bias that makes LLMs sensitive to input order. Our method is focused on solving these issues by prioritizing the preprocessing of data before introducing it to the LLM. The use of a cheaper ranking model to enhance the initial order of passages helps us present a better ranked list of passages to the LLM, while

Datasets	DL-19			DL-20			Covid			News			Touche			Signal		
nDCG Score	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10	1	5	10
BM25	54.26	52.78	50.58	57.72	50.67	47.96	68.00	63.24	59.47	42.69	41.28	39.52	56.12	48.11	44.22	41.75	37.04	33.05
RankGPT	77.52	75.06	69.51	78.09	68.48	65.42	76.00	75.27	73.06	52.05	49.91	47.21	53.06	40.37	37.29	47.94	36.55	32.00
Refine-Rank	74.42	73.39	70.77	78.09	70.86	68.46	92.00	81.91	76.40	47.66	49.96	46.96	43.88	33.64	30.03	41.24	33.70	30.59
Compress-Rank	51.55	56.91	53.17	62.96	50.88	46.52	94.00	85.64	77.69	54.53	51.08	45.98	37.76	33.61	30.44	48.45	35.10	29.24
Refine-Compress-Rank	65.12	60.61	57.23	65.74	53.78	47.58	90.00	83.43	78.46	49.71	49.59	46.66	42.86	31.04	27.80	43.30	32.80	28.30

Table 1: This table presents results performed on various datasets from BEIR and TREC. The models shown are either baselines or variations of our final pipeline.

keeping costs low, and the use of natural language-based text condensation to effectively reduce text verbosity while preserving its semantic essence allows the LLM to process the relevant information without the need for a sliding window.

The refine + rerank method demonstrated a major improvement in scores across four of the six datasets tested, suggesting that LLMs do in fact exhibit a strong positional bias, and enhancing the initial order of documents can significantly boost ranking performance. The compress + rerank method showed improvement over the baseline for the COVID and News datasets and yielded comparable results for the Signal dataset. While we attempted to condense passage information into a summarized form while retaining significant information, it appears that this method caused to a greater loss in information than we hoped for, subsequently impairing the LLM's ability to effectively rerank passages. Furthermore, combining these strategies into refine + rompress + rerank approach improved performance over the baseline in the COVID dataset and achieved comparable outcomes in the News dataset, performing similarly to the compress + rerank method, demonstrating that the loss of information causes due to the compression step has a measurable negative impact.

Our solution, which eliminates a sliding window completely, surpasses the performance of the previous state-of-the-art method which relies on a sliding window for one of the benchmark datasets, and performs relatively well for the other datasets as well. This indicates that our approach, while not perfectly successful in outperforming the existing methods, does hold promise for future developments and warrants further exploration into similar methods. Furthermore, this approach allows for the use of any off-the-shelf LLMs to produce good ranking results despite not being particularly tuned for the ranking task or domain.

4.3.3 Cost Breakdown

One aspect which several re-ranking papers fail to mention is the cost of running models for the task of re-ranking. This was a harsh limitation that we had to face, however, as running more expensive re-ranking methods was not a viable approach for us. The final methods we chose were not only some of the most performant, but were also determined to be much more cost effective than running RankGPT alone. To break down how this is the case, we will going over platform costs, time spent in inference, as well as the differing costs between using GPT with and without compression. For this, we will use Covid as our reference dataset for cost.

Platform Cost A majority of our re-ranking pipeline was run using a V100 GPU on Google Colab. Colab gives 100 credits per $10 spent and running a V100 GPU for an hour costed 4.3 credits, as detailed in their inline resource monitor. Given these two metrics, we can determine the cost per hour to be $0.43 per hour. It was also possible to use a T4 GPU for free on an unpaid account, which we took advantage of for our refinement stage, meaning the effective cost was 0. As for GPT, we tracked overall money spent using their API dashboard.

Compute Time The models we chose for Refinement and Compression are MonoBERT and FlanT5XL respectively. MonoBERT can be excluded from the cost analysis due to it's relative low cost. FlanT5XL typically ran for 2 minutes for every 100 passages compressed on a V100 GPU. Depending on the dataset, the cost can differ drastically. With Covid, this would amount to approximately 100 minutes of compression. Based on our Platform Costs, we can determine the overall cost to be $0.72.

GPT Costs OpenAI provides a billing dashboard, which we used to track the money spent. Without compression, we found that GPT would cost about $3.83 on the Covid dataset. With compression, we found that GPT would cost around $2.02. This is likely due to compression, reduc-

ing the number of tokens inputted into GPT. We also noticed that larger token inputs tend to result in timeouts and additional calls, which incur more cost and time spent, something more prevalent without compression.

Overall Costs Overall, we found that using Refinement and Compression Reduced Costs to a significant degree. With Refinement and Compression, the overall cost was $2.74 as opposed to RankGPT, which was $3.83. This marks a 29% decrease in cost, something which was much more noticeable when running over large datasets.

5 Conclusion

This paper presents a comprehensive multi-stage approach to improving listwise re-ranking in Large Language Models (LLMs), leveraging the strengths of MonoBERT, FlanT5-XL, and GPT-3.5. Our methodology addresses the inherent challenges in listwise re-ranking, such as sensitivity to initial data ordering and the limitations of the context window. By using the open-source FlanT5-XL model for title generation and compression, we significantly reduced the costs compared to proprietary models like GPT-3.5. Our findings underscore the importance of choosing the right strategies and the potential of open-source models like FLAN-T5 to offer competitive performance with cost-effective solutions.

5.1 Limitations

Refine: We found that the refinement stage of the pipeline did not work as intended. In some scenarios, it improved performance; however, in others, it exacerbated prior issues present in both the pretrained model and Listwise Generation.

Compress: We found that compression also presented mixed results. It struggled to accurately represent the passages. Since we are prompting an LLM, we found that part of the issue lies with the prompt not being ideal for all datasets as well as the LLM itself not being properly trained for the task at hand.

5.2 Future Ideas

Fine-tuning: We believe that fine-tuning the open parts of our pipeline, both individually and as part of the entire pipeline, would benefit greatly, as the models would be better able to re-order and compress the passages according to the need of the Listwise model.

Prompting: We also believe that modifying our prompt to better summarize and capture key details about the passage would help in compressing our passages effectively. In addition, we believe text extraction could pose a better solution if done on a sentence or multi-sentence level (i.e. Please select a sentence from the passage which accurately answers the query.)

Model: Both the pre-trained model and open LLM could be replaced with potentially better models. However, we have not found time to test these alternative strategies.

6 Acknowledgements

We would like to thank Professor Kevin Chang and Pritom Saha Akash at the University of Illinois Urbana-Champaign for their guidance with this project.

References

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022a. Scaling instruction-finetuned language models.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022b. Scaling instruction-finetuned language models.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track.

Luyu Gao, Sheng Ma, Jing Zhou, and Jamie Callan. 2022. Unsupervised passage retriever for open-domain question answering. *arXiv preprint arXiv:2212.09736*.

Mahima Jain, Yijin Niu, Abhishek Sharma, et al. 2023. Rankzephyr: Zero-shot learning for large-scale retrieval using iterative multi-task training. *arXiv preprint arXiv:2303.03212*.

Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2018. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey.

Boshko Koloski, Senja Pollak, Blaž Škrlj, and Matej Martinc. 2022. Out of thin air: Is zero-shot cross-lingual keyword detection better than unsupervised?

Chuhan Li et al. 2023. Rankgpt: Empowering large language models with retrieval and ranking abilities for cognitive search. *arXiv preprint arXiv:2303.11504*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models.

2021 Ma. 2021. Improving information retrieval with deep learning techniques. *Journal of Information Retrieval Research*, 12(3):10–50.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Rodrigo Frassetto Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *CoRR*, abs/2003.06713.

OpenAI. 2022. Introducing chatgpt.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2023a. Improving passage retrieval with zero-shot question generation.

Mrinmaya Sachan, Tao Yu, Hui Li, Panupong Pasupat, Percy Liang, and Mohit Iyyer. 2023b. Rank and rerank: Multistage ranking for compositional semantic parsing. *arXiv preprint arXiv:2303.13051*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents.

Fan Tang, Ronak Pradeep, Julius Owoicho, and Jimmy Lin. 2023. Document ranking with large language models. *arXiv preprint arXiv:2304.02284*.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. Found in the middle: Permutation self-consistency improves listwise ranking in large language models.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models.

Chenyan Xu and Jimmy Lin. 2023. Rank-without-gpt: Improving open-domain passage ranking with large language models. *arXiv preprint arXiv:2302.09708*.

Manling Yang, Jay Pujara, Moontae Lee, and Heng Ji Wang. 2023. Progressive reranking for zero-shot passage retrieval. *arXiv preprint arXiv:2301.11668*.