Analyzing the Effectiveness of Listwise Reranking with Positional Invariance on Temporal Generalizability

Soyoung Yoon[1,†], Jongyoon Kim[1,†] and Seung-won Hwang[1,*]

[1]*Seoul National University (SNU), 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea*

Abstract

Benchmarking the performance of information retrieval (IR) methods are mostly conducted within a fixed set of documents (static corpora). However, in real-world web search engine environments, the document set is continuously updated and expanded. Addressing these discrepancies and measuring the temporal persistence of IR systems is crucial. By investigating the LongEval benchmark, specifically designed for such dynamic environments, our findings demonstrate the effectiveness of a listwise reranking approach, which proficiently handles inaccuracies induced by temporal distribution shifts. Among listwise rerankers, our findings show that ListT5, which effectively mitigates the positional bias problem by adopting the Fusion-in-Decoder architecture, is especially effective, and more so, as temporal drift increases, on the test-long subset.

Keywords

information retrieval, listwise reranking, temporal misalignment, positional bias, fusion-in-decoder, reranking, generative retrieval

1. Introduction

The majority of studies on information retrieval systems are concentrated on benchmarks that target static snapshots of knowledge. This leaves a gap in our understanding of how these models fare in dynamic environments where knowledge is temporal and constantly accumulating, and the value of adaptiveness to new information is underscored. Moreover, unlike statistical retrieval systems such as BM25, neural-based retrieval models have been found to underperform on unseen data without prior training, posing a challenge for direct application to temporal updates [1]. Attempting to navigate these issues with naïve full fine-tuning is computationally expensive, prone to excessive forgetting, and ultimately impractical. Regarding these circumstances, improving the temporal persistence of the retrieval model, i.e., improving the robustness of the model with respect to time change, is an important research field we should gain attention to. The LongEval Retrieval Challenge [2] aims to specifically target this problem, aligning more closely with real-world retrieval applications and scenarios. Meanwhile, we believe that the temporal change can also be viewed as one specific form of a distribution shift, and we believe applying retrieval methods that are effective in handling out-of-domain data could also be effective for handling the temporal persistence. BEIR [3] is regarded as one of the well-known benchmarks that evaluate the model's out-of-distribution retrieval performance. Until now, findings suggest [4] that re-ranking is very effective to handle ood retrieval. Specifically, a line of work on *listwise* reranking, a format that sees multiple passages at once when conducting reranking [5, 6, 7, 8, 9, 10] has shown to be effective on achieving SOTA performance on BEIR. Listwise rerankers can condition on and compare multiple passages to calibrate the relevance scores better, and can reduce the inaccuracy of predictions arising from domain shift, as theoretically supported by [11], and empirically evidenced by a line of works such as [5, 10].
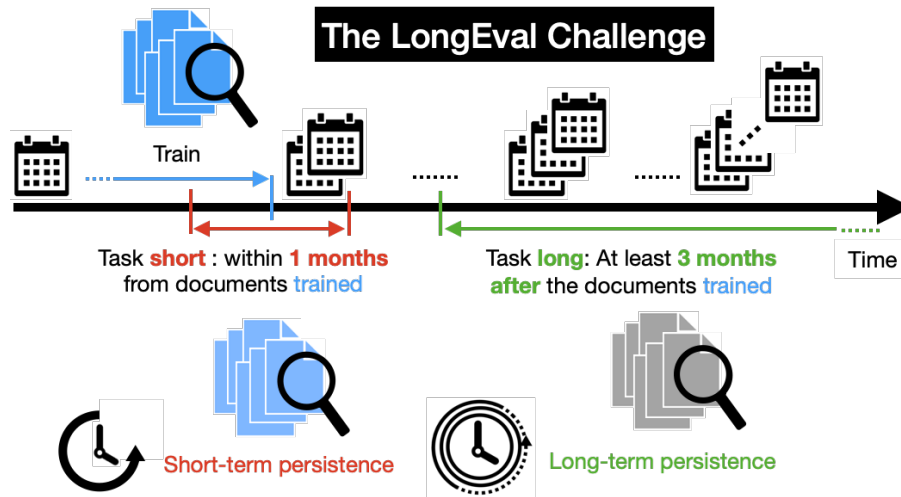
Figure 1: Overview of the longeval retrieval challenge. The task is evaluated into two parts: short-term persistency and long-term persistency.

However, naive application of listwise reranking with LLMs are known to face the Lost-in-the middle problem [2], favoring passages presented in the first and last position of the listwise input. Also, the large parametric size of the model itself requires high computational cost and affects negatively on the efficiency. Recently, a model named ListT5 [7] which uses the FiD architecture to conduct listwise reranking has been shown to be effective in both efficiency and performance, mitigating the positional bias by FiD and performing well despite its relatively small model size.

In this paper, we aim to bridge this gap by participating in the competition for temporal knowledge retrieval called the LongEval Retrieval Challenge [2], which aligns more closely with real-world retrieval applications and scenarios, illustrated in Figure 1. In particular, we frame temporal knowledge adaptation as another form of zero-shot domain retrieval and investigate the effectiveness of ListT5, listwise reranking with positional invariance, on the LongEval task. Findings show that ListT5, despite its smaller model size, is more effective than the RankZephyr [12] or other reranking variants, and is especially effective as the temporal shift becomes longer, showing superior performance on the long subset.

Our experiments on this benchmark reveal that applying listwise reranking greatly helps generalizing in temporal misalignment, ensuring flexibility for temporal knowledge accumulation, even without further training.

2. The LongEval Challenge

Started at 2023, The LongEval Challenge [2] in Figure 1, is a shared task designed to evaluate the temporal persistence of information retrieval systems. It addresses the challenge of maintaining model performance over time as test data becomes increasingly distant from the training data. LongEval sets itself apart from traditional IR tasks by focusing on the development of systems that can adapt to dynamic temporal text changes, introducing time as a new dimension for ranking model performance.

Our method was evaluated using LongEval retrieval benchmark datasets from 2023 competitions. The LongEval retrieval task aims to address the distribution shift between the training and test datasets, which occurs due to differences in the timing of data collection. To assess the resilience and temporal alignment of the retrieval system, the task offers two test datasets: **test-short**, which is the one with a small time shift from the training dataset, and **test-long**, the other with a more significant time shift. In 2023, the test-short dataset exhibited a 1-month shift from the training dataset, while the test-long dataset showed a 3-month shift. In 2024, the test-short dataset had a 5-month shift, and the test-long dataset had a 7-month shift.

Throughout the whole train, test-short, and test-long datasets in both two years, queries are having

Table 1

Detailed statistics of the six datasets from two year competitions in LongEval retrieval. This table presents the number of queries, number of documents, average length of query and document, and the number of relevant documents.

Year	2023			2024		
	Train	Test-Short	Test-Long	Train	Test-Short	Test-Long
Time Shift (months)	0	+1	+3	0	+5	+7
Total # Queries	672	882	923	599	407	1518
Total # Documents	1.57M	1.59M	1.08M	2.05M	1.79M	2.53M
Average Query Length (words)	2.76	2.71	2.55	2.45	2.41	2.48
Average Document Length (words)	793.1	792.9	806.1	770.5	624.8	429.1
Total # Relevance Annotations	9655	12217	13467	9785	88301	156170
Relevant Document / Query	4.0	3.96	4.32	7.28	4.15	5.48

length of 2 words approximately. This can easily state that almost all queries are keyword queries. Unlike query, as the documents are being crawled and collected from Qwant search logs, documents are often long, having around 800 words. The number of documents are mostly 1.5 millions.

The LongEval retrieval benchmark provides the relevance annotations, which are constructed by utilizing attractiveness probability of Dynamic Bayesian Network (DBN) click model trained on Qwant data. As click model uses search log as implicit feedback, which the user clicked the document or even stayed on the document with certain threshold time, the relevant annotation per query is roughly around 4. All the statistics can be found on Table 1.

3. Related Work

3.1. Temporal Retrieval

While there is a large body of prior work regarding temporal update of the language models itself [13] or real-time QA models [14, 15], there are relatively few works [16] regarding the development of adaptive retrieval systems. In the era where retrieval-augmented generation systems are well knonw and widely used for its effectiveness and performance [17], along with developing the adaptivity for the generation model for downstream tasks, it is crucial to develop retrieval models to better adapt to new information.

3.2. Listwise Reranking

Pointwise v.s. Listwise reranking Until now, the field of zero-shot reranking has been generally driven by cross-encoder models [4] such as MonoT5 [18]. As shown in Figure 2, MonoT5 is a *pointwise* reranker, which leverages pre-defined token probabilities (i.e., `true` / `false`) as relevance scores at inference time. However, while being efficient (requiring $O(n)$ number of forward passes to rank n passages), these models rely on *pointwise* reranking of each passage, and thus lacks the ability to compare between passages relatively at inference time. This could lead to a sub-optimal solution in the task of reranking, where the discrimination and ordering between passages are crucial. Unlike MonoT5, listwise rerankers consider the *relative* relevance between documents, thus being robust to domain shift [11]. The comparison as to how pointwise and listwise rerankers differ is illustrated at Fig.2.

Listwise reranking with LLM prompting Listwise reranking [5, 6, 7, 8, 9, 10, 12] is a line of work that gives multiple passages at once to the model and output the permutations, or orderings between passages by the relevance to the input query. Specifically, the prompts are formulated in a similar way as follows:

```
I will provide you with 20 passages, each indicated by numerical identifier [].
```

Q. How to make money easily
D1. Applicable online surveys by 2017 to make money …

Q. How to make money easily
D2. The history of making money. From 1984, farmers …

Q. How to make money easily
D3. How to make money in 2024? Recent trends show …

MonoT5 (pointwise) → 0.8
MonoT5 (pointwise) → 0.1
MonoT5 (pointwise) → 0.7

argsort → D1 D3 D2

Pointwise reranking with MonoT5

Listwise reranking with ListT5

Q. How to make money easily
D1. Applicable online surveys by 2017 to make money …

Q. How to make money easily
D2. The history of making money. From 1984, farmers …

Q. How to make money easily
D3. How to make money in 2024? Recent trends show …
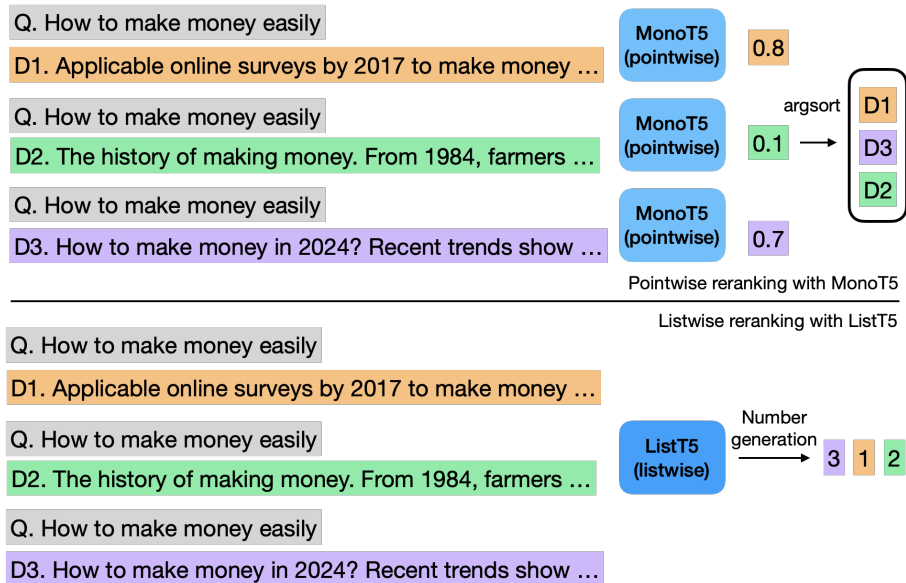
ListT5 (listwise) → Number generation → 3 1 2

Figure 2: Explanation of listwise reranking models with respect to the pointwise ranking variants. Pointwise reranking individually assigns relevance scores to each documents, where listwise reranking feeds a list of documents at once to the model and let the model generate the relative order of documents.

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

[1] {passage_1}
[2] {passage_2}
[3] {passage_3}
[4] {passage_4}
...
[20] {passage_20}

Search Query: {query}

Rank the 20 passages above based on their relevance to the search query.
All the passages should be included and listed using identifiers, in descending
order of relevance. The output format should be [] > [], e.g., [4] > [2].
Only respond with the ranking results, do not say any word or explain.
```

Given the above prompt, listwise reranking models are trained to output orderings of k=20 passages, i.e., `[1] > [3] > [20] > .. [19]`. Then, the orderings are parsed into list of numbers and the passages are sorted by the output orderings. A sliding window approach is usually adopted to rank top-n passages where n is bigger than the window size the model can accept (k), which is explained at the next paragraph. Due to its ability to utilize the generative capability of large language models by zero-shot prompting, listwise reranking is widely gaining attention. However, this format presents a challenge due to the monolithic and lengthy input size, since the model must process the information of multiple passages at once. Due to its lengthy input size, applying listwise reranking has been limited to LLMs trained with large context lengths. Therefore, application of listwise reranking to smaller models such as T5 [19] has been limited to seeing a pair of passages at once [20], which exhibits quadratic time complexity, far from being practical.

Listwise reranking with ListT5 The ListT5 [7] model jointly considering the relevancy of multiple candidate passages at both training and inference time based on the Fusion-in-Decoder (FiD) [21] architecture. By using the FiD architecture ListT5 effectively addresses this issue of positional bias, as each passage is processed by the encoder with identical positional encoding. Consequently, **the decoder cannot exploit positional bias**. Also, ListT5 effectively reduces the input length on the

Illustration of different strategies used for Listwise Reranking

window = 4, stride = 2, rerank top-2

window = 4, r=2, rerank top-2

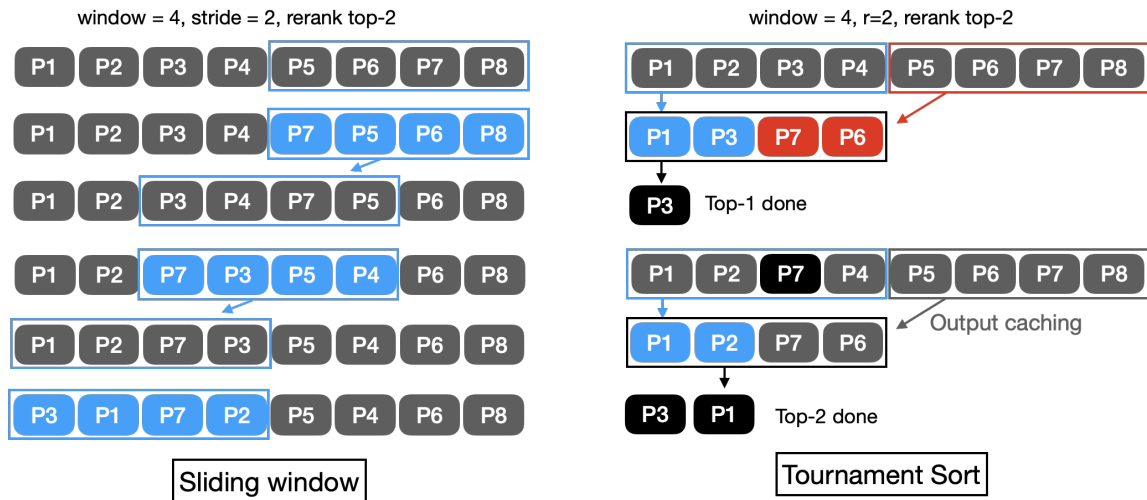Sliding window

Tournament Sort

Figure 3: Illustration of different sorting strategies for listwise reranking, mainly the sliding window approach used for listwise reranking models for with LLMs and the tournament sort approach used for ListT5. In the example, the number of total candidate passages n is 8, window size is 4, and stride is 2 where the hyperparameter r for ListT5 is 2. Please refer to the ListT5 paper [7] for more detailed explanation about tournament sort.

encoder level, by computing listwise reranking at the decoder part. Additionally, to better extend to diverse scenarios, e.g., to rerank k passages given a much larger number of n candidate passages than the model can see at once, ListT5 adopts a hierarchical tournament sort approach rather than a sliding window approach, to efficiently cache output and does not require multiple evaluations over the entire n passages.

The illustration of the two strategies for listwise reranking, as to how the sliding window approach and the tournament sort (used for ListT5) works, is illustrated at Fig.3. Typically, they rank top-n (n=100) passages with the window size of 20 and stride of 10.

ListT5 [7] uses listwise reranking with list size of 5 with FiD and conducts tournament sort for efficient reranking. Unlike other listwise reranking models which needs large sized models with context length, ListT5 is much more computationally efficient, using the relatively small-sized T5 architecture. This was made possible by utilizing the Fusion-in-Decoder [21] architecture with tournament sort [22]. ListT5 also effectively mitigates the lost-in-the middle problem, thereby having superior performance on zero-shot retrieval [3].

4. Baseline Models

In this section, we explain about the details on the models we used for both the submission and additional ablation experiments (both baseline and ours) for the LongEval challenge.

First-stage retrieval model: BM25, RepLLaMA   We mainly used two different first-stage retrieval models: neural-based bi-encoder models, or lexical-based statistical methods. Lexical-based statistical models like BM25 [23] measures the relevance between query and document based on the words. Bi-Encoder models, such as the DPR [24] model, measures the similarity of output embeddings between the query and the document, typically with cosine similarity or dot product. Since we can dump the embeddings of given documents asynchronously and use them on inference time, the Bi-Encoder approach can be computed very efficiently using the aid of optimized similarity search engines such as FAISS. While neural-based first-stage retrievers like ColBERT [25] could be a good option for its effectiveness and efficiency and its ability to capture semantic similarities, BM25 [23], which is a statistical retriever, consistently demonstrates robust performance, especially for zero-shot retrieval
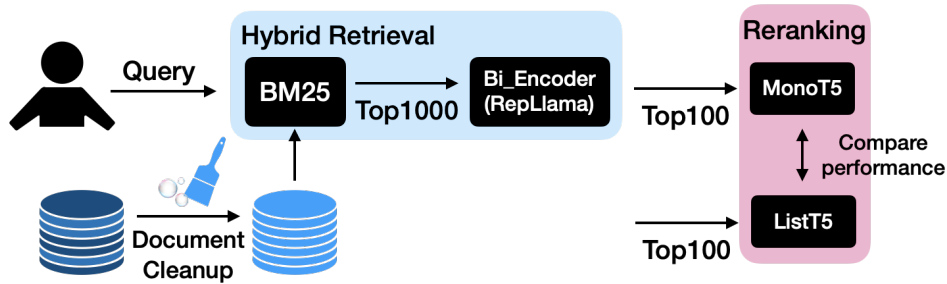
Figure 4: Overview of the retrieval process of the submission to the LongEval challenge.

benchmarks [3]. Therefore, in this project, we experiment with both neural-based and statistical-based retrievers as the first-stage retrieval system, and represent them as the weakest baseline to be experimented with various reranking models. Specifically, we used both retrieval models as the hybrid approach for submission, and independently use them for further ablation experiments to see the impact of first-stage retrievers for final model reranking performance.

Pointwise reranking - MonoT5   To compare the effectiveness of listwise reranking with respect to pointwise reranking on temporal misalignment, we experiment with MonoT5 [18]. MonoT5 is widely known for its effectiveness on zero-shot retrieval. We used the model with the huggingface identifier of `castorini/monot5-3b-msmarco-10k`, with maximum input size of 1024.

Listwise reranking - RankZephyr, RankVicuna, Llama-3-8B zeroshot, and ListT5   Among the listwise reranking models, we experiment with works that are gaining attention for its superior performance with respect to the BEIR[3] benchmark, and that could be fully reproducible by using open-sourced models, which was RankZephyr [12], and RankVicuna [9]. In addition, to test the effectiveness of zero-shot listwise reranking, we experiment with the Llama-3-8B-Instruct [26] model, which was not fine-tuned on the listwise reranking task (and thus are completely zero-shot.) All listwise reranking models except ListT5 were conducted by modifying code provided from the RankLLM repository [1]. The RankLLM repository provides codes to evaluate RankVicuna and RankZephyr models on the pyserini-indexed dataset (such as TREC-DL, MS MARCO, or BEIR). We applied slight modifications to the original code to accept custom datasets (The LongEval test set) and accept the Llama-3-8b model to the system (The codebase to inference models are based on the FastChat[2] library, and they already accept Llama-3 family models). The results are saved in the same output format as other models (MonoT5, ListT5) and then go into the same evaluation process as the other models. We used maximum input size of 4096 for RankZephyr, RankVicuna, and Llama-3-8B, and 1024 for ListT5-3B.

5. Submission Details

5.1. Overview

In this section, we describe the detailed process of our submission to the challenge. Fig 4 is the overview of our system. We first describe the selection choice of the corpus, data cleanup process and explain about the hybrid retrieval process followed by reranking. To this end, we have submitted 2 versions: one with the MonoT5 and one with ListT5. In summary, the submitted run used **BM25** as the first-stage retrieval model to select top-1000 documents, **RepLLaMa** to select top-100 documents among them, and used either **MonoT5** or **ListT5** for the final reranking process, as described at Fig 4.

5.2. Dataset Selection

Selection-language   The LongEval challenge provides a dataset with two languages, French and English. As the dataset was originally collected from the search log of Qwant, the French search engine, they provided the French version of documents and queries. For the non-French researchers, LongEval challenge organizers designed automatic French-to-English translation that utilizes fasttext [27] to detect the language of each sentence in the document and French-English CUBBITT [28] to ensure the high quality of translation [29]. Galuscáková et al. [29] state that they limited the translation length by 500 bytes at once to reduce the catastrophic error propagation, there may be some unintended translation caused due to the fault inherited in the translator. Even though there might be unknown translator errors in the English dataset, as we are non-French researchers, we selected the English dataset to qualitatively analyze the results in detail.

Selection-documents   The challenge dataset included URLs, but we only focused on the text field of the corpus and maintained the content without any additional post-processing, except document cleanup. The participants in 2023 showed some methods utilizing the URL fields to crawl some additional information from the original document or simulate the search. As we assume that these methods cannot be further applied to the general circumstance of temporal shift, we decide not to collect or simulate additional data. We only focused on the text field provided on the corpus to ensure that ours can be applied to other temporal shift tasks that receive query and corpus as input.

5.3. Evaluation

5.3.1. Proxy Metric

To validate our method, we utilized the dataset released in the previous round (2023), as the relevance annotation for 2024 was not released before the challenge submission reached the end. We initially set up the evaluation logic with 2023 test datasets and employed it as a preliminary performance measurement for various experiments to estimate performance on 2024 test datasets.

5.3.2. Detailed Explanation of Evaluation

For ease of analysis, we recorded all the possible information on each stage of retrieval including:

- query id
- query string
- retrieved document id
- retrieved document string
- retrieved document model score
- true relevance annotation (if available)

This information is recorded in the form of jsonline, where each line indicate retrieval results and true relevance annotation of each query from one test dataset. The jsonline files are then passed to evaluation logic written with TREC code[3].

5.4. Data Preprocessing - Data Cleanup

For the document, we have cleaned the document before any other experiments. Note that we did not apply any modification techniques for the query itself (e.g., query expansion). We used the original query without modification, and only cleaned up the documents. The LongEval retrieval dataset was constructed by extracting search logs from Qwant, from selected topics. Because their corpus was extracted from the SERPs of Qwant, its text includes HTML tags, cracked encoded strings, and unwanted

---

[3]We used `pytrec_eval` (https://github.com/cvangysel/pytrec_eval), the python wrapper version of `trec-eval`

emails and URLs, such as line-break characters (\n) or encoding cracked characters (\u00b), as shown on the following box.

> **Uncleaned Document**
>
> Top Message by razzmouette \u00bb 30 Oct 2012, 17:33\nGiven the MTO I will be comfirme Grevenlingendam tomorrow from 15 to 17 with claques of 25-27 it is quite a lot.\nTop Location:\nSt. Gilles Message by Xav \u00bb Oct.\n2012\n, 17:41 Pareil, Grevelingendam !\nWe organise a departure from BXL ...\nI have a place in my small car, leaving St Gilles.\nTop Message by Dimitri \u00bb 30 Oct 2012, 18:22 Salute to all, I am looking to sail tomorrow in Grevelingendam too!\nI am leaving from Brussels (Montgommery), if possible to reach a car I would be ultra boiling.\nThe following table shows the results of the survey.\nThank you in advance!\nDim \u00a0 PS: on this website via Manu LVR Top\nResponding\nDeveloped\nby phpBB \u00ae\nForum\nSoftware\n\u00a9 phpBB Limited\n ...

This makes the document difficult to read even for the human, which also impacts the language model since it is trained with cleaned data. Therefore, we apply the dataset cleanup code [4], which preprocess the corpus by:

- replacing line-break character to the empty string.
- transliterating to the closest ASCII representation.
- fixing Unicode errors.
- replacing phone numbers, URLs, and emails to the empty string.
- replacing HTML tags with the empty string.

For cleantext, we used the following code to remove and replace the first 4 types.

```python
from cleantext import clean

clean(
    text,
    fix_unicode=True,  # fix various unicode errors
    to_ascii=True,  # transliterate to closest ASCII representation
    lower=False,  # lowercase text
    no_line_breaks=True,  # fully strip line breaks as opposed to only normalizing them
    no_urls=True,  # replace all URLs with a special token
    no_emails=True,  # replace all email addresses with a special token
    no_phone_numbers=True,  # replace all phone numbers with a special token
    no_numbers=False,  # replace all numbers with a special token
    no_digits=False,  # replace all digits with a special token
    no_currency_symbols=False,  # replace all currency symbols with a special token
    no_punct=False,  # remove punctuations
    replace_with_punct="",  # instead of removing punctuations you may replace them
    replace_with_url="",
    replace_with_email="",
    replace_with_phone_number="",
    replace_with_number="<NUMBER>",
    replace_with_digit="0",
    replace_with_currency_symbol="<CUR>",
    lang="en"  # set to 'de' for German special handling
)
```

---

[4]https://github.com/prasanthg3/cleantext

Then we removed HTML tags with simple regex that matches all text that is surrounded by angled brackets.

After this pre-processing, we can obtain the more readable document which will also make the retrieval model not to suffer from unintended characters, as written on the following box.

> **Cleaned Document**
>
> Top Message by razzmouette 30 Oct 2012, 17:33 Given the MTO I will be comfirme Grevenlingendam tomorrow from 15 to 17 with claques of 25-27 it is quite a lot. Top Location: St. Gilles Message by Xav Öct. 2012 , 17:41 Pareil, Grevenlingendam ! We organise a departure from BXL ... I have a place in my small car, leaving St Gilles. Top Message by Dimitri 30 Oct 2012, 18:22 Salute to all, I am looking to sail tomorrow in Grevelingendam too! I am leaving from Brussels (Montgommery), if possible to reach a car I would be ultra boiling. The following table shows the results of the survey. Thank you in advance! Dim PS: on this website via Manu LVR Top Responding Developed by phpBB R Forum Software C phpBB Limited

This process reduces the length of document about 0.1% as described on the Table 2. This statistics indicates that the majority of contents remain while successfully removing or replacing unreadable characters.

Table 2
The document length in characters before and after pre-processing. The pre-processing incorporates replacing, removing, and transliterating the unreadable characters.

Dataset	Original Document Length	Cleaned Document Length
Train	4687.3	4655.0
Test-Short	3802.5	3775.6
Test-Long	2611.6	2593.1

To verify that human readable document will improve the language model performance, we conducted simple experiment with BM25 on 2023 dataset. BM25 indexed both uncleaned corpus and cleaned corpus and the performances are measured with nDCG@10 and nDCG@100, which is reported on Table 3.

Table 3
Comparative evaluation of **nDCG@10** and **nDCG@100** scores between BM25 index on cleaned and uncleaned LongEval 2023 corpus. We can see that applying document cleanup improves ranking scores on average for both NDCG@10 and NDCG@100.

Metric	Method	Train	Test-Long	Test-Short	Average
nDCG@10	BM25 (uncleaned)	16.88	17.37	17.47	17.24
	BM25	16.76	17.38	17.66	17.27
nDCG@100	BM25 (uncleaned)	24.23	25.27	24.72	24.74
	BM25	24.20	25.46	25.02	24.89

The result showing that applying document cleanup by filtering out unwanted values had a slight improvement on performance. Compared with the BM25 (uncleaned) version, the BM25 version had +0.03 for nDCG@10 in average, and +0.15 gain of nDCG@100, compared without pre-processing. Seeing this improvement, we decided to use the same cleaned data for the 2024 test set.

5.5. First Stage Retrieval: BM25

BM25 is a lexical based sparse retrieval model utilizes statistics of the corpus. It is often used by search engines to estimate the relevance of documents to a given search query. It is part of the family of probabilistic information retrieval models. BM25 incorporates several heuristics to balance term frequency and document length normalization, making it effective in practical applications.

For the implementaiton, we utilized python library `castorini/pyserini` [30]. We followed default argument for the `pyserini.index.lucene` command.

```
1  python -m pyserini.index.lucene \
2  --collection JsonCollection \
3  --input ${input_corpus_dir}$ \
4  --index ${index_save_dir}$ \
5  --generator DefaultLuceneDocumentGenerator \
6  --threads 36 \
7  --storePositions --storeDocvectors --storeRaw
8
9  python -m pyserini.search.lucene \
10 --index ${index_save_dir} \
11 --topics ${query_path} \
12 --output ${ranking_save_path} \
13 --bm25
```

Using code and parameters, we retrieved the top 1000 documents with BM25. We verified that the recall was already 0.9 when top 1000 documents are sorted in the order of true relevance annotations. This indicates that reranking the top 1000 document can improve the retrieval result.

5.6. Hybrid Retrieval: RepLLaMA-7B

RepLLaMA [31] is a fine-tuned version of the LLaMA language model tailored for multi-stage text retrieval tasks. As a RepLLaMA is a dense bi-encoder retrieval model, it encodes query and document by taking each as separate input and dot-product each other to measure the similarity between the query and the document. The models demonstrate strong zero-shot performance on out-of-domain dataset.

Using the huggingface parameters `castorini/repllama-v1-7b-lora-passage` and the implementation [32], we tested RepLLaMA in parallel with BM25. The RepLLaMA tokenizes the first 512 tokens for both document and query, but it takes the hidden state on the index of EOS token, which may not be located on the 512th output. For each query, given the top 1000 documents indexed by BM25 with RepLLaMA, we conduct additional hybrid retrieval by RepLLaMA-7B and selected the top 100 from them, which were then handed on to the candidate documents for reranking models. All the commands and hyperparameters to index and retrieve documents are followed the description on [32] [5].

5.7. Reranking

We performed reranking with pointwise (MonoT5) and listwise (ListT5) rerankers on the top 100 documents retrieved by BM25 and RepLLaMA.

MonoT5   MonoT5 is widely known for its effectiveness on zero-shot retrieval and uses pointwise reranking. MonoT5 is the model that takes a concatenated string of a query q and a document d as input, `Query:` q `Document:` d. The model is fine-tuned to return a word either "true" or "false" to determine whether the document is relevant to the query or not. The returned logit is softmaxed to calculate the probability of a "true" token to be assigned, which is used as the relevance score. As the monoT5 is

---

[5]https://github.com/texttron/tevatron/tree/main/examples/repllama

designed for re-ranking, the model iteratively takes each of the documents, that is concatenated with a query, from the top k ranking, and outputs its relevance score.

For a fair comparison between MonoT5 and ListT5, we specify k as 30, since the competition measured not only metrics@10 but also beyond @10. The model parameter we used for MonoT5 can be found on `castorini/monot5-3b-msmarco-10k`.

ListT5 In our official submission to the Codalab, we used the ListT5-3b model (with the huggingface identifier of `Soyoung97/ListT5-3b`). While the default setup of ListT5 uses $r=2$ and reranks top-10 passages, we modify the setup to use $r=2$ and run the model to rerank top-30 passages, to see improvements with NDCG@100 along with NDCG@10. Due to the time limitation and the deadline schedule, we used the top-100 retrieved results from BM25 for the first-stage retrieval model, reranks top 20 passages by ListT5, and appended the top 1000 results from RepLLama.

5.8. Other Details

For the hardware specifications, each model runs on a different system.

- BM25 (Lucene) utilized CPU only with around 100GB RAM for about 30 minutes to index in the size of 3GB (short), 8GB (long) on SSD.
- RepLLaMA used approximately 30GB of NVIDIA A6000 48GB (with batch size of 16) for about 17 hours to index 4 shards in the size of 3GB (short), 8GB (long) on SSD.
- MonoT5-3B used full memory (with batch size of 25) of a single NVIDIA H100 80GB.
- ListT5-3B used full memory (with batch size of 16) of a single NVIDIA H100 80GB.
- Other listwise rerankers (using the rankllm.ai repository) are run on a single NVIDIA H100 80GB. It took about 20 hours to finish inference on approx. 2000 short and long queries.

5.9. Submission Results

We conducted multiple experiments and submitted the results utilizing MonoT5 and ListT5. Our method only reranks the top 100 for efficiency, but we found that the challenge measures some metrics @all, where @all indicates @1000 as the submission detail states the results are taken up to 1000 ranking results. Therefore, we filled up the remaining top 1000 results based on the BM25 results. The evaluation of submitted results can be found on Table 4. The result confirms the effectiveness of the ListT5 outperforming in all metrics compared to MonoT5. The ListT5 outperforms MonoT5 by +5.29 (short), +3.8 (long) on nDCG@10, and +1.51 (short) and +1.66 (long) on nDCG@100. From the results, we conclude that listwise reranking also helps to mitigate the temporal misalignment of information retrieval systems, compared with pointwise reranking methods.

Moreover, to compare the retrieval performance in fair conditions, we conjectured the language usage on most of the submissions on the leaderboard. We found that the highest ranked participant who used English is *mam10eks* from team *OWS*. As far as we can verify when compared with teams that used only the English dataset, we achieved top 1 in all metrics except for nDCG@all, MAP@all, and P@10 in test-short, where we ranked top 2.

6. What Makes ListT5 Effective on Temporal Shift?

After the competition, the gold relevance annotations for the 2024 test short / long subset were released, and by utilizing them, we conducted additional experiments to further analyze the effectiveness of ListT5. Initially, we experiment on verifying whether temporal shift is related to domain shift, and that test-long is more shifted than test-short. Subsequently, we conducted experiments to answer three specific research questions.

Table 4

The official evaluations of our submissions and best performance with the English dataset (conjecture, the mam10eks team) on test-short (denoted as **S**) and test-long (denoted as **L**). The best results on each metric are highlighted in **bold**.

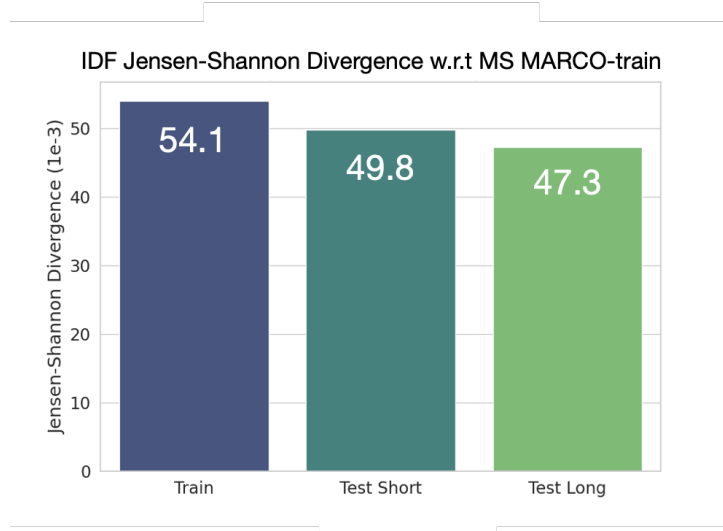|	nDCG				MAP		P		Recall	
	@10		@all		@all		@10		@1000	
	S	L	S	L	S	L	S	L	S	L
(Ours) ListT5	**33.45**	**25.07**	23.00	**19.33**	19.67	**14.23**	18.09	**14.29**	**59.92**	**40.18**
(Ours) MonoT5	28.16	21.27	21.49	17.67	17.67	12.74	16.78	13.13	42.45	29.45
(Other team) mam10eks	33.32	24.26	**24.06**	19.02	**19.94**	13.87	**18.31**	13.96	55.29	36.59

Figure 5: IDF similarity between datasets and MS-MARCO is measured with Jensen-Shannon divergence. The larger value indicates a more similar vocabulary distribution.

6.1. Hypothesis: Invariance of ListT5 pronounced in bigger shift

We hypothesized the most important factor for ListT5 compared with other listwise reranking variants is its permutation invariance property, and that this property is pronounced in bigger shift of domains. In order to show this property along with other analysis, we first analyzed the degree of domain shift between short and long subsets.

Temporal shift correlated to temporal shift. The distribution shift in documents is a primary concern in information retrieval. Neural models demonstrates strong performance on test datasets that follow similar data attributes of training datasets, but models struggle to retrieve documents from the test corpus with different distributions than the training corpus. This attribute is also pointed out in the BEIR paper [3], concluding that there are no correlation between models that excel at in-domain test set and out-of-domain test sets.

In real search scenarios, since continuously training the retrieval model with respect to the corpus update is computationally expensive and difficult, we train the retrieval model up to a certain point, fix the model, and use it. The retrieval model then experiences a performance drop as the time elapses, as the model struggles to retrieve recent documents that show distribution changed compared to the training dataset. This scenario is much alike the LongEval challenge, where the document is updated through time, and this is the reason why we assume that temporal shift is one format of distribution shift, and solving the temporal shift can be attained by resolving the distribution shift.

To measure the extent of the distribution shift between the train, test-short and test-long subset
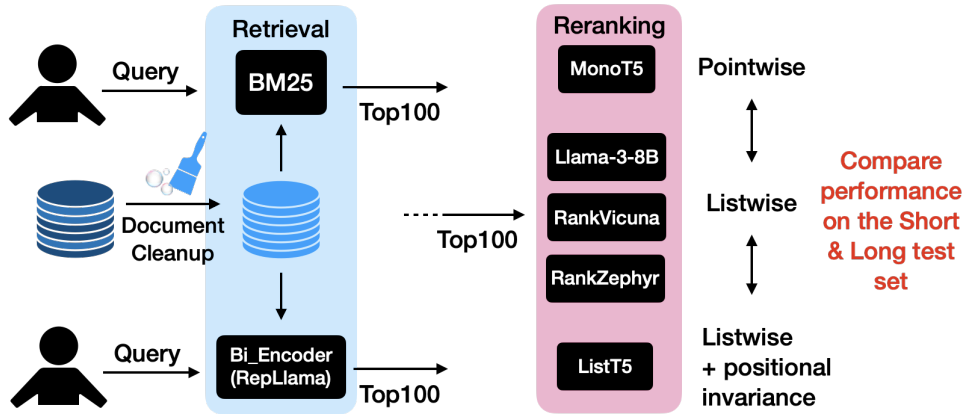
Figure 6: Overview of the additional experiment setup. Compared with the hybrid retrieval process we used on the actual submission (Fig. 4), we independently use 2 different first-stage retrievers (BM25 v.s. RepLLaMA), and compare with diverse different rerankers, including MonoT5(pointwise), RankVicuna, RankZephyr, Llama-3-8b (zero-shot prompting), and ListT5-3B, on the test-short and test-long of the LongEval 2024 test set.

Table 5

Comparative evaluation of **nDCG@10** and **nDCG@100** scores between baselines and ours on the test-short and test-long subset. The best performance on each dataset for each method is highlighted in bold.

	First-stage Retrieval: BM25				First-stage Retrieval: RepLlama			
	NDCG@10		NDCG@100		NDCG@10		NDCG@100	
	short	long	short	long	short	long	short	long
Initial	16.09	13.74	24.40	18.57	21.04	17.17	29.82	22.29
MonoT5-3B	21.49	17.67	28.16	21.27	21.72	18.77	30.35	23.27
Llama3-8B	15.03	12.72	23.49	17.65	20.19	16.22	29.08	21.41
RankVicuna-7B	18.68	15.28	26.41	19.91	19.74	16.46	29.22	22.19
RankZephyr-7B	**22.72**	17.70	**29.06**	21.47	**23.96**	19.18	**31.90**	23.75
ListT5-3B	22.41	**18.21**	28.62	**21.65**	23.11	**19.45**	31.27	**23.80**

of LongEval-2024, we employ inverse document frequency (IDF) and measure the similarity with the Jensen-Shannon divergence. Each document is tokenized with T5 tokenizer [6] and truncated the document by the length of 1024 tokens. [7] In addition to the LongEval-2024 training corpus, MS-MARCO is considered as the in-domain, a dataset to compare, as we target a zero-shot setting. Results from Figure 5 show that the test-long subset (47.3) were more out-of-domain compared to the test-short subset (49.8), since lower values indicate that they are more far away from each domain. From this experiment, we conclude that temporal shift is correlated with domain shift, and **test-long is more shifted than test-short**, which follows the original conjecture.

6.2. Research Questions & Experimental Setup

Figure 6 is the experimental overview on testing the effectiveness of ListT5 on various aspects. After finding out that temporal shift is related with domain shift, we try to answer 3 research questions regarding the effectiveness of ListT5 applied on the temporal domain:

1. Is listwise reranking more effective than pointwise rerankers for temporal persistence?
2. Will the invariance property of ListT5 (by FiD) be pronounced in bigger shift?
3. Does this effect hold when using different first-stage retrievers?

---

[6] `Soyoung97/ListT5-3b`

[7] Special tokens are neglected.

In order to answer the 3 research questions, we additionally conduct experiments and report results on different baseline models. As shown on Fig. 6, we experiment with 2 different first-stage retrieval models, 5 different reranking models including pointwise, listwise reranking models, and ListT5, and report the NDCG, MAP, Precision, and Recall performance. For each evaluation metric, we report both measures at @10 and @100 for through analysis. Mainly, we use and compare with the NDCG@10 and NDCG@100 results on Table.5, and use results on other metrics (Table. 6, 7, 8) as supporting evidence.

6.2.1. RQ1. Listwise Reranking vs Pointwise Reranking

Looking at the tables, we can see that generally, listwise reranking is much more effective than pointwise rerankers. The performance on RankVicuna, RankZephyr, and ListT5 is much more higher than MonoT5-3B models, for both short and long subsets. However, it seems that fine-tuning is crucial for rerankers, since zero-shot results from Llama3-8B was not as effective as the fine-tuned counterparts.

6.2.2. RQ2. Invariant Listwise Reranking (ListT5)

On comparing ListT5 with other listwise reranking models, we notice an interesting finding. On the short subset (with less temporal shift), RankZephyr was more effective. On the **long** subset, however, ListT5, which has invariant properties, was more effective. This property was consistent across all metrics including NDCG, MAP, Precision, and Recall, regardless of @k (@10, @100). We believe that the robustness to positional bias helped ListT5 perform better than other counterparts on the long subset, where the initial ordering was not very trustable. We hypothesize that ListT5 with invariant properties will be much more effective as the temporal shift becomes even larger.

6.2.3. RQ3. Impact of First-Stage Retrievers

While we used hybrid retrieval - a combination of BM25 and RepLLaMA for submission, we believed it was necessary to compare with statistical and neural retrievers, to see the impact on the reranking performance depending on the effectiveness of the initial retriever. Thus, we experiment and report results using either BM25 or RepLLaMA. The results show that while RepLLaMA is a better first-stage retrieval model than BM25 (higher precision, higher recall), the orderings between ListT5 and baseline models doesn't change much, and found that most properties held without significant differences.

6.3. Summary of findings

By investigating on 3 research questions, we conclude that permutation-invariant listwise reranking (ListT5) is an effective method not only for general out-of-domain data but also to mitigate the temporal drift, and the findings hold regardless of the choice of the first-stage retrievers. It is particularly interesting that it performs better than RankZephyr-7B in test-long subsets where distribution shift is more pronounced.

7. Conclusion and Future Work

In this paper, we focus on analyzing the effectiveness of listwise reranking (`ListT5`) on the LongEval Challenge set, to investigate its effectiveness on the temporal misalignment scenario. Our findings (on the Jensen-Shannon divergence) suggest that temporal misalignment could be viewed as one form of out-of-distribution scenario. Compared with other listwise reranking methods such as RankZephyr, we find that applying permutation-invariant listwise reranking becomes more effective as temporal drift increases, ListT5 achieving higher performance on the test-long subset with half the parametric model size. To this end, we aim to develop a search engine capable of delivering robust and stable results, even as the available document sets change over time.

Table 6

Comparative evaluation of **MAP@10** and **MAP@100** scores between baselines and ours on the test-short and test-long subset. The best performance on each dataset for each method is highlighted in bold.

	First-stage Retrieval: BM25				First-stage Retrieval: RepLlama			
	MAP@10		MAP@100		MAP@10		MAP@100	
	short	long	short	long	short	long	short	long
Initial	10.39	8.24	13.44	9.70	13.94	10.66	17.63	12.35
MonoT5-3B	14.74	11.41	17.67	12.74	14.86	11.95	18.46	13.54
Llama3-8B	9.21	7.29	12.35	8.76	12.71	9.70	16.45	11.40
RankVicuna-7B	13.06	9.64	16.14	11.12	13.52	10.30	17.32	12.17
RankZephyr-7B	**15.94**	11.50	**18.74**	12.79	**16.43**	12.49	**20.04**	14.04
ListT5-3B	15.71	**12.01**	18.32	**13.23**	16.26	**12.74**	19.51	**14.24**

Table 7

Comparative evaluation of **Precision@10** and **Precision@100** scores between baselines and ours on the test-short and test-long subset. The best performance on each dataset for each method is highlighted in bold. Notice that Precision@100 scores are same, since the model variants are all rerankers initialized from first-stage retrieval models.

	First-stage Retrieval: BM25				First-stage Retrieval: RepLlama			
	Precision@10		Precision@100		Precision@10		Precision@100	
	short	long	short	long	short	long	short	long
Initial	12.82	10.30			16.88	12.82		
MonoT5-3B	16.78	13.13			17.35	14.15		
Llama3-8B	12.57	10.15	2.91	1.98	16.68	12.65	3.41	2.29
RankVicuna-7B	14.79	10.92			15.99	11.87		
RankZephyr-7B	**17.23**	12.57			**18.94**	13.96		
ListT5-3B	17.56	**13.40**			18.17	**14.38**		

Table 8

Comparative evaluation of **Recall@10** and **Recall@100** scores between baselines and ours on the test-short and test-long subset. The best performance on each dataset for each method is highlighted in bold. Notice that Recall@100 scores are same, since the model variants are all rerankers initialized from first-stage retrieval models.

	First-stage Retrieval: BM25				First-stage Retrieval: RepLlama			
	Recall@10		Recall@100		Recall@10		Recall@100	
	short	long	short	long	short	long	short	long
Initial	19.23	15.78			25.29	19.65		
MonoT5-3B	25.45	20.10			26.29	21.90		
Llama3-8B	18.90	15.53	42.45	29.45	24.99	19.41	49.28	33.81
RankVicuna-7B	22.23	16.67			23.88	18.13		
RankZephyr-7B	25.74	19.28			**28.16**	21.25		
ListT5-3B	**26.56**	**20.52**			27.40	**22.19**		

8. Limitation

The LongEval Challenge dataset is being collected from the French search engine, Qwant. Therefore, the dataset has been collected in French, (e.g., mostly used Euro(€) to represent currency). However, as we are not native French, it is difficult to utilize the provided French document subset. Therefore, we conducted all experiments in English. However, as we referred to the challenge leaderboard, the

methods that utilized French showed a (much better) huge performance gap between English results. For future work, we hope to make a multilingual version of ListT5, where it is currently only limited to English. We believe ListT5 with the multilingual version would drastically improve its applicability to a much wider domain. Also, investigating and improving the temporal shift in a multi-lingual setting seems to be an interesting next future step. Lastly, due to time limitations, we had to choose the hybrid approach on the submission in the challenge. Our results would have been better if we had used RepLLaMA as the single first-stage retrieval model and rerank top-1000 passages instead of 100. However, due to time and computing resource limitations, we were unable to submit the results on time. Acknowledging these limitations, we would like to participate once again, if the competition holds, in the future.

References

[1] P. Röttger, J. Pierrehumbert, Temporal adaptation of BERT and performance on downstream document classification: Insights from social media, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 2400–2412. URL: https://aclanthology.org/2021.findings-emnlp.206. doi:10.18653/v1/2021.findings-emnlp.206.

[2] P. G. R. Deveaud, G. Gonzalez-Saez, P. Mulhem, L. Goeuriot, F. Piroi, M. Popel, Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation, 2023. arXiv:2303.03229.

[3] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021. arXiv:2104.08663.

[4] G. Rosa, L. Bonifacio, V. Jeronymo, H. Abonizio, M. Fadaee, R. Lotufo, R. Nogueira, In defense of cross-encoders for zero-shot retrieval, 2022. arXiv:2212.06121.

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

[6] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, Z. Ren, Is chatgpt good at search? investigating large language models as re-ranking agents, 2023. arXiv:2304.09542.

[7] S. Yoon, E. Choi, J. Kim, Y. Kim, H. Yun, S. won Hwang, Listt5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval, 2024. arXiv:2402.15838.

[8] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, M. Bendersky, Rankt5: Fine-tuning t5 for text ranking with ranking losses, 2022. arXiv:2210.10634.

[9] R. Pradeep, S. Sharifymoghaddam, J. Lin, Rankvicuna: Zero-shot listwise document reranking with open-source large language models, 2023. arXiv:2309.15088.

[10] W. Sun, Z. Chen, X. Ma, L. Yan, S. Wang, P. Ren, Z. Chen, D. Yin, Z. Ren, Instruction distillation makes large language models efficient zero-shot rankers, 2023. arXiv:2311.01555.

[11] R. Xian, H. Zhuang, Z. Qin, H. Zamani, J. Lu, J. Ma, K. Hui, H. Zhao, X. Wang, M. Bendersky, Learning list-level domain-invariant representations for ranking, 2023. arXiv:2212.10764.

[12] R. Pradeep, S. Sharifymoghaddam, J. Lin, Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!, 2023. arXiv:2312.02724.

[13] B. Dhingra, J. R. Cole, J. M. Eisenschlos, D. Gillick, J. Eisenstein, W. W. Cohen, Time-aware language models as temporal knowledge bases, Transactions of the Association for Computational Linguistics 10 (2022) 257–273. URL: https://aclanthology.org/2022.tacl-1.15. doi:10.1162/tacl_a_00459.

[14] K. Luu, D. Khashabi, S. Gururangan, K. Mandyam, N. A. Smith, Time waits for no one! analysis and challenges of temporal misalignment, 2022. arXiv:2111.07408.

[15] J. Kasai, K. Sakaguchi, Y. Takahashi, R. L. Bras, A. Asai, X. Yu, D. Radev, N. A. Smith, Y. Choi, K. Inui, Realtime qa: What's the answer right now?, 2024. arXiv:2207.13332.

[16] Z. Zhang, M. Fang, L. Chen, Retrievalqa: Assessing adaptive retrieval-augmented generation for short-form open-domain question answering, 2024. arXiv:2402.16457.

[17] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, 2024. `arXiv:2312.10997`.

[18] R. Nogueira, Z. Jiang, R. Pradeep, J. Lin, Document ranking with a pretrained sequence-to-sequence model, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 708–718. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.63. doi:`10.18653/v1/2020.findings-emnlp.63`.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of Machine Learning Research 21 (2020) 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[20] R. Pradeep, R. Nogueira, J. Lin, The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models, 2021. `arXiv:2101.05667`.

[21] G. Izacard, E. Grave, Leveraging passage retrieval with generative models for open domain question answering, 2021. `arXiv:2007.01282`.

[22] K. McLuckie, A. Barber, Tournament Sort, Macmillan Education UK, London, 1986, pp. 68–86. URL: https://doi.org/10.1007/978-1-349-08147-9_5. doi:`10.1007/978-1-349-08147-9_5`.

[23] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Foundations and Trends in Information Retrieval 3 (2009) 333–389. URL: https://doi.org/10.1561/1500000019. doi:`10.1561/1500000019`.

[24] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 6769–6781. URL: https://www.aclweb.org/anthology/2020.emnlp-main.550. doi:`10.18653/v1/2020.emnlp-main.550`.

[25] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 39–48.

[26] AI@Meta, Llama 3 model card (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[27] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, arXiv preprint arXiv:1607.01759 (2016).

[28] M. Popel, M. Tomkova, J. Tomek, Ł. Kaiser, J. Uszkoreit, O. Bojar, Z. Žabokrtský, Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals, Nature communications 11 (2020) 1–15.

[29] P. Galuscáková, R. Deveaud, G. González Sáez, P. Mulhem, L. Goeuriot, F. Piroi, M. Popel, Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 3086–3094.

[30] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, R. Nogueira, Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations, in: Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), 2021, pp. 2356–2362.

[31] X. Ma, L. Wang, N. Yang, F. Wei, J. Lin, Fine-tuning llama for multi-stage text retrieval, arXiv preprint arXiv:2310.08319 (2023).

[32] L. Gao, X. Ma, J. J. Lin, J. Callan, Tevatron: An efficient and flexible toolkit for dense retrieval, ArXiv abs/2203.05765 (2022).