Applying Large Language Model For Relevance Search In Tencent

Dezhi Ye
390959029@qq.com
dezhiye@tencent.com
Tencent
Beijing, China

Jie Liu
jesangliu@tencent.com
Tencent
Beijing, China

Junwei Hu
keewayhu@tencent.com
Tencent
Beijing, China

Jiabin Fan
robertfan@tencent.com
Tencent
Beijing, China

Bowen Tian
lukatian@tencent.com
Tencent
Beijing, China

Haijin Liang
hodgeliang@tencent.com
Tencent
Beijing, China

Jin Ma
daniellwang@tencent.com
Tencent
Beijing, China

Abstract

Relevance plays a crucial role in commercial search engines by identifying documents related to user queries and fulfilling their search needs. Traditional approaches employ encoder-only models like BERT, which process concatenated query-document pairs to predict relevance scores. While autoregressive large language models (LLMs) have revolutionized numerous NLP domains, their direct application to web-scale search systems presents significant challenges. On one hand, the relevance modeling capabilities of LLMs have not been fully explored. On the other, the high computational costs and inference times make deploying LLMs in online search systems, which demand extremely low latency, nearly impossible.

In this work, we address these challenges through two key contributions. First, we develop a comprehensive evaluation framework to systematically assess the effectiveness of LLMs in query-document relevance ranking. By conducting assessment experiments to LLMs in four perspectives: ranking objectives, model size, domain-specific continuous pre-training, and the integration of prior knowledge, we identify the best resource allocation strategy given a restricted budget and develop practical LLMs in a more efficient way. Second, we propose a novel framework to transfer the capabilities of LLMs in the ranking aspect to existing BERT models to avoid directly deploying LLMs. Finally, to fully leverage the improvements in relevance ranking brought by LLMs, we successfully nearline deploy LLMs in Tencent QQ Browser search engine using query-based on-demand computing and quantization. Experiments on real-world datasets and online A/B tests demonstrate that our approach significantly enhances search engine performance while maintaining practical operational efficiency. Our findings provide actionable insights for integrating LLMs into production search engines.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737193

CCS Concepts

• **Information systems** → **Language models**; **Web search engines**; *Rank aggregation.*

Keywords

Relevance search, Large language model, Search engines

ACM Reference Format:
Dezhi Ye, Jie Liu, Junwei Hu, Jiabin Fan, Bowen Tian, Haijin Liang, and Jin Ma. 2025. Applying Large Language Model For Relevance Search In Tencent. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737193

1 Introduction

Search relevance measures how well search results align with the intent behind a search query. Incorporating relevance ranking results allows search engines to ensure that the content displayed to users is genuinely pertinent to their information needs [40, 41]. Search relevance not only considers the literal meaning of query words but also encompasses understanding the user's intent and context to provide the most accurate and relevant search results, directly impacting user experience and business outcomes optimization.

Recent relevance modeling predominantly utilizes BERT [7] to delineate the relationship between queries and documents. BERT's profound understanding of language semantics and context enables it to provide more accurate relevance assessments [38, 49]. Additionally, to balance model performance and deployment costs, the Ranker Distill [10] study explores how knowledge for search within BERT large can be transferred to a smaller ranker through distillation. With the emergence of Large Language Models (LLMs), there has been a significant shift towards using LLMs in relevance ranking [1, 16]. For instance, unsupervised methods like RankGPT [33] utilize LLMs for reranking by treating text generation as a relevance ranking task [47], applying pairwise [28] or listwise [20, 48] approaches. However, challenges remain due to a misalignment between LLM pre-training and the specific demands of relevance ranking. Models such as RankLLaMA [21] and TSARankLLM [43]

use supervised fine-tuning to align LLMs more closely with relevance ranking tasks, thereby affirming that the capabilities of large language models significantly exceed those of their smaller counterparts. Despite the robust performance of LLMs in relevance retrieval, integrating recent advancements in LLMs into large-scale search engine systems, which manage trillions of documents and have stringent efficiency requirements, presents significant challenges [5, 34]. **C1: Fully leveraging the potential of LLMs remains a formidable challenge**. There is a persistent misalignment between LLM pre-training and relevance ranking. Current pre-training objectives, such as next token prediction, can not directly translate to modeling query-document relevance relations. **C2: The substantial improvements offered by LLMs entail high computational costs**. Notably, the parameter size of LLMs is several times larger than that of existing BERT models, leading to a proportional increase in latency. This substantial increase makes it nearly impossible to deploy LLMs directly in industrial-level search engines, presenting a difficult trade-off between efficiency and cost in real-world ranking systems.

This paper shares our experiences in applying LLMs for Relevance Search and introduces a series of practical techniques that have been successfully implemented and deployed to enhance the Tencent QQ Browser search engine. To address Challenge 1 (C1), we explore four key aspects of how to construct a robust relevance LLM. Rank Objective: We introduce the **Gen**erative **F**ine-grained **R**elevance Score approach (**GenFR**), which computes the final relevance score by performing a weighted sum of probabilities, as opposed to directly computing probabilities {0,1,2,3,4} at the last position. Model Size: by comparing seven different sizes of LLMs (0.5B, 1.5B, 3B, 7B, 13B, 30B, 70B), we find that the relationship between model performance and size is nonlinear, with performance gains becoming increasingly marginal as model size increases. Domain-Specific Continuous Pre-training: To address the dual challenges of preserving fine-grained relevance scoring capabilities and ordinal relationships in large language models, we propose a **R**elevance-**C**entric **C**ontinued **P**re-**T**raining paradigm (**RC-CPT**). Integrating Prior Knowledge: integrating prior query analysis features and document analysis features significantly boosts the relevance ranking capabilities of LLMs, with query analysis features showing particularly notable improvements. Our observations can be used to identify the best resource allocation strategy given a restricted budget and could provide important insights for the practical implementation of LLMs in search engines.

To overcome C2, we investigate methods to transfer the capabilities of LLMs to existing BERT models. These methods include: 1. LLM as Relevance Judge: Utilizing LLMs to sift through click data to isolate data that better aligns with relevance tasks, thereby enhancing the quality of BERT model samples. 2. LLM as Hard Sample Maker: Employing LLMs to generate challenging samples that specifically improve BERT models' ability to recognize and handle complex patterns, such as keyword mismatches. 3. Distilling LLMs into BERT: Applying Kullback-Leibler (KL) divergence and Margin MSE loss methods to distill the ranking capabilities of LLMs into BERT models. By employing these three methods, we have successfully transferred the ranking capabilities of LLMs to BERT, thereby circumventing the challenges associated with deploying LLMs online.

Finally, to fully leverage the relevance modeling capabilities of LLMs, we successfully deploy them nearline using quantization and On-Demand Computing. Offline data comparisons indicate that LLMs significantly outperform BERT models, particularly in handling long-tail queries. We segmented online requests into two categories: long-tail queries are processed using LLMs, while other requests are handled by BERT. This strategy allows us to process only about 14% of queries with LLMs, substantially reducing deployment costs. Additionally, to address the challenge posed by varying scoring distributions across different models, we implement a Bayesian [42] post-calibration method. This approach calibrates the scores of LLMs and BERT models to a unified distribution, allowing the models to operate independently while resolving the issue of adapting downstream modules to different scoring systems.

The key contributions of this paper are as follows:

- **Exploration of Building a Strong Relevance LLM**: We investigate four critical aspects of constructing a robust relevance LLM: Rank Objective, Model Size, Domain-Specific Continuous Pre-training, and Integration of Prior Knowledge.
- **Enhancement of BERT Models Using LLMs**: We employ three strategies to augment the relevance modeling capabilities of BERT models: using LLMs as relevance judges, as hard sample makers, and distilling LLMs into BERT. Our offline results indicate that these methods substantially improve the capabilities of BERT models.
- **Nearline Deployment of LLMs**: We detail the nearline deployment of LLMs, utilizing techniques such as quantization, and query-based on-demand computing. Extensive online and offline experiments demonstrate the effectiveness of our method.

2 Related Work

2.1 LLMs For Relevance Ranking

Recently, Large Language Models (LLMs) have increasingly been applied in relevance ranking tasks. Few-shot or zero-shot methods, which cast ranking as text generation [23, 32], leverage LLMs to directly generate a reordered list of candidates. Examples include LRL [22], RankGPT [33], and RankVicuna [26]. For instance, one could simply ask LLMs to determine the relevance of a query-document pair, yielding a rudimentary solution. These methods primarily utilize Pointwise [32], Listwise [20, 27], Setwise [48], and Pairwise [28] approaches, where documents are passed in the prompt. Their advantage lies in the fact that once you have obtained the pre-trained, instruction-tuned LLM, there is no need for further fine-tuning. Nevertheless, these prompt strategies often overlook the potential misalignment between LLMs and the specific requirements of relevance ranking tasks [15, 36]. To address this issue, task-specific LLMs through supervised fine-tuning adapt LLMs to relevance tasks and achieve more remarkable results[14, 18]. For example, RankLLaMA [21], which uses the last token as the ranking basis and trains with ranking losses. TSARankLLM [43] treats the generation probability of a query conditioned on the document as the relevance score. Although task-specific LLMs have achieved commendable performance, there is a lack of comprehensive research on adapting LLMs to relevance ranking.

Recent studies have also explored using LLMs for training data generation, which can enhance the modeling capabilities of systems. InPars [2, 13] utilizes GPT-3 [3] to generate synthetic queries, while Promptagator [6] introduces a few-shot dense retrieval approach to generate synthetic queries using demonstrations from the target domain. LLMJudge provides [17, 30] the results of a large-scale automatic relevance judgment evaluation. These offline generation methods alleviate the challenges of deploying LLMs online.

2.2 Knowledge Distillation For Relevance Ranking

Knowledge Distillation (KD) was recently proposed to distill a large and complex ranking teacher model into a smaller student model to alleviate computationally expensive issues [29, 31]. The Ranker Distill [10] study investigates how the ranking capabilities of BERT base can be transferred to BERT small. Margin-MSE [12] adapts knowledge distillation to accommodate the varying score output distributions of different BERT and non-BERT passage ranking architectures. Ensemble distillation [45] introduces a method to retain the performance of an ensemble of models at the inference cost of a single model by distilling the ensemble into a single BERT-based student ranking model. More recently, DisRanker [39] has employed knowledge distillation [12] to transfer the ranking capabilities of LLMs to BERT [4, 15, 35]. Although these methods have shown promising results, they have not fully explored how to deploy LLMs in industrial search engines.

3 Preliminary

3.1 Problem Statement

Given a query and a collection of documents, the goal of relevance ranking task is to rank documents based on their relevance to the query. Let q denote a query and d denotes a set of documents, we are required to find a relevance scoring function $\int(.)$. Utilizing function $\int(q, d)$ to discern the most pertinent document for the user.

3.2 Dataset Description

In our work, we define the relevance levels of query-document pairs as five ordered categories: Perfect/Highly Relevant (L4), Excellent/Relevant (L3), Good/Relatively Relevant (L2), Fair/Marginally Relevant (L1), and Bad/Irrelevant (L0). The data was collected from a real-world web search engine. We utilized a cloud platform for crowdsourced annotations, with each query-document (Q-D) pair annotated collaboratively by three individuals. The average number of tokens per document is 2,095, and the total number of documents is one million. We manually annotated the training, evaluation, and testing datasets resulting in 1,498,456, 209,425, and 291,452 query-document pairs, respectively. To ensure the completeness of our dataset and to accurately assess the performance of the relevance model, we included 50% head, 30% torso, and 20% tail queries based on their frequency distribution.

3.3 Language Model For Relevance Search

3.3.1 BERT based Relevance model. BERT, as a baseline model for search relevance, is widely used in industrial search engines. We can use the [CLS] embedding for classification and generating the relevance score of document.

3.3.2 LLM based Relevance model. Current mainstream Large Language Models (LLMs) utilized for relevance ranking tasks are generally categorized into two types: generative and discriminative. The discriminative approach like RankLLaMA [21, 39] employs the representation of the last position's </s> for both the query and document, which is subsequently used for ranking tasks in Eq.1 named **Dis**.

$$\begin{aligned} \text{input} =& < \text{s} > \text{query} : \text{title} : \text{summary} < /\text{s} > \\ \text{Dis}(q, d_i) =& \, Dense(Decoder(input)[-1]) \end{aligned} \tag{1}$$

Conversely, the generative approach capitalizes on the generative capabilities of LLMs, using the generation probability of labels [19, 28, 46] in Eq.3 named **GenL** or queries like TSARankLLM [43] in Eq.2 named **GenQ** as the basis for ranking. The generative approach uses next token prediction loss for training.

$$\begin{aligned} \mathcal{P}(d_i) =& \text{ 'Document: } d_i \text{ Query:'} \\ \text{GenQ}(q, d_i) =& \prod_j p\left(q_j \mid \mathcal{P}(d_i), q_{<j}\right). \end{aligned} \tag{2}$$

$$\begin{aligned} \text{input} =& < \text{s} > \text{query} : \text{title} : \text{summary} < /\text{s} > \\ \text{GenL}(q, d_i) =& \, P(Lable = \{0, 1, 2, 3, 4\} \mid (input)). \end{aligned} \tag{3}$$

4 Build a Strong Relevance LLM

4.1 Ranking Objective

Motivation: While **GenL** demonstrates capability in modeling relevance labels, it fails to capture the partial order relationships between relevance grades. For instance, L3 (Excellent) represents a higher relevance level than L2 (Good). Furthermore, even within the same grade category, there exists nuanced scoring distinctions that **GenL** cannot effectively model for fine-grained relevance scoring. In contrast, **Dis** approaches this by leveraging LLMs as feature extractors, enabling direct modeling of both fine-grained relevance scores and their ordinal relationships. However, **Dis** encounters alignment challenges between pretraining objectives and downstream relevance tasks, thereby limiting the full utilization of LLMs' potential. To address these limitations, we propose a Generative Fine-grained Relevance Score approach **GenFR** for relevance ranking as shown in Figure 1. **GenFR** first calculates the probability of each label {0,1,2,3,4} at the last position, then computes the final relevance score by performing a weighted sum of these probabilities. Finally, the model is trained using a pair-wise margin loss in Eq.4. In this way, the model can learn the ordinal relationships of relevance labels from the generation probabilities.

$$\begin{aligned} f(q, d_i) =& \sum p_{i,k} \cdot y_k \\ \text{where } p_{i,k} =& \frac{\exp(s_{i,k})}{\sum_{k'} \exp(s_{i,k'})}, y_k \in \{0, 1, 2, 3, 4\} \\ \text{GenFR} =& \, Max(0, f(q, d_i) - f(q, d_j), \mathcal{T}) \end{aligned} \tag{4}$$

Where $\exp(s_{i,k})$ is the log-likelihood obtained from the LLM, $\mathcal{T}$ is the margin. Besides, **GenFR** can be optimized jointly with **GenL**.

Experimental Setting: We conduct fine-tuning experiments from 0.5B to 70B models. To ensure a fair comparison, we trained for one

Model Size	PNR					NDCG@5				
	GenFR + GenL	GenFR	Dis	GenL	GenQ	GenFR + GenL	GenFR	Dis	GenL	GenQ
0.5B	2.07	**2.10**	2.05	1.89	1.81	65.89	**65.91**	65.61	64.29	64.01
1.5B	2.23	**2.27**	2.21	2.12	2.06	68.12	**68.22**	67.38	67.00	66.82
3B	2.50	**2.55**	2.48	2.40	2.35	72.45	**72.66**	72.38	72.03	71.89
7B	2.77	**2.81**	2.68	2.64	2.52	74.11	**74.23**	73.42	73.21	73.03
13B	**3.13**	3.07	3.01	3.00	2.94	**76.04**	75.99	75.51	75.42	75.19
30B	**3.29**	3.24	3.18	3.20	3.14	**77.01**	76.98	76.61	76.77	76.55
70B	**3.44**	3.42	3.32	3.35	3.30	**77.51**	77.44	77.13	77.22	77.03
BERT-Base (0.1B)		**2.34**					**68.42**			
BERT-Large (0.3B)		**2.50**					**71.99**			
DeBERTa-XLarge (0.7B)		**2.71**					**74.01**			

Table 1: PNR and NDCG@5 results under different rank objectives and model sizes with LLMs.

epoch on all supervised data, maintaining consistent batch sizes and other hyperparameters. For evaluation metrics, we selected NDCG@5 and PNR. The result is shown in Table 1.

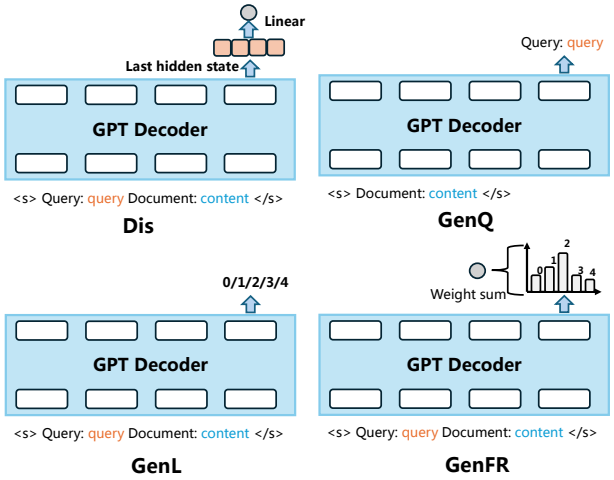


Figure 1: Comparison of four different ranking objectives for large language models.

Observations: 1. In terms of effectiveness, **GenFR** outperformed **GenQ**, **GenL**, and **Dis**. The likely reason is that GenFR can simultaneously model the fine-grained scoring of query-document pairs and the ordinal relationships of their relevance. 2. The combined optimization of **GenFR** and **GenL** has achieved better results on models larger than 7B. This improvement is attributed to the increased model size, which enhances the capability to distinguish between relevance categories.

4.2 Model Size

Motivation: To examine how model performance varies with model size and to explore the upper limits of LLMs in relevance search tasks, we fine-tune LLMs of different sizes and monitor the corresponding performance changes [9]. Such experiments could provide crucial insights for the practical implementation of large language models in relevance search, aiding in determining the optimal resource allocation strategy within a limited budget in an industrial setting.

Experimental Setting: We conduct experiments with seven models, utilizing different LLMs ranging from 0.5 billion to 70 billion parameters. The result is shown in Table 1.

Observations: 1. As the size of the model increases, the magnitude of result changes diminishes, suggesting diminishing marginal returns with increasing model size. This observation may be attributed to the fact that, for relevance search tasks, a 7B model already possesses sufficient knowledge, and larger models do not yield significantly better improvements. 2. Models larger than 13B that were directly fine-tuned performed better than a domain-fully trained BERT, but models smaller than 13B do not perform as well as a domain-fully trained 48 layer DeBERTa-XLarge [11]. This discrepancy could be due to LLMs pre-trained on general corpora lacking domain adaptation for relevance search tasks. Consequently, continuing to pre-train LLMs is essential to further unleash their potential.

4.3 Relevance-Centric Continued Pre-Training

Motivation: The tasks involved in LLM pre-training and relevance ranking differ significantly, with notable disparities in corpus distribution. These gaps between LLMs and ranking tasks may limit the capability of Supervised Fine-Tuning (SFT) to fully leverage the original knowledge embedded in LLMs. Therefore, we explore whether continued pre-training can enhance the performance of LLMs in relevance search tasks.

Method: Existing CPT (Continued Pre-Training) frameworks predominantly adopt Doc2query [43] or Query2doc [39] task formulations. These methods leverage high-engagement user query-document pairs to construct <query, document> training instances, optimizing language models through next-token prediction objectives. However, these approaches exhibit critical limitations in developing fine-grained relevance scoring capabilities during the CPT phase. To bridge this gap, we propose two innovative instructions for Relevance-Centric Continued Pre-Training (RC-CPT):

Point-wise Relevance-Centric CPT: Directly predict relevance labels through next token prediction. *Query: [X], Document: [Y]. Relevance level between this query and document: [Label].*

Pair-wise Relevance-Centric CPT: Jointly optimize for both absolute relevance labeling and comparative relevance judgments through contrastive learning. *Query: [X], Document 1: [Y1], Document 2: [Y2]. Relevance level for Document 1: [Label1]. Relevance level for Document 2: [Label2]. Document 1 is slightly less/more relevant than Document 2.*

Set-wise Relevance-Centric CPT: Select the most relevant document from the list of candidate documents. *Query: [X], Document 1: [Y1], Document 2: [Y2] ..., Document N: [Yn]. Document N is the most relevant in the candidate list.*

Specifically, we collect user query-document pairs to form <query, document> pairs and then annotate them using additional models. To ensure higher quality data, we employ a voting method involving multiple models, each fine-tuned with relevance-annotated samples. Finally, we compile the data into a format for Relevance-Centric Continued Pre-Training.

Experimental Setting: We conduct experiments using a 7B model for continued pre-training using **GenFR** method. After completing the continued pre-training, both groups underwent the same fine-tuning process. The result is shown in Table 2.

Method	PNR	NDCG
GenFR(7B)	2.81	74.23
w/ Doc2Query	3.04	75.21
w/ Query2Doc	3.08	75.33
w/ Point-wise RC-CPT	3.26	76.65
w/ Pair-wise RC-CPT	3.31	77.01
w/ Set-wise RC-CPT	3.34	77.12

Table 2: The performance comparison between four different CPT (Continued Pre-Training) tasks

Observations: 1. Continued pre-training significantly improved the ranking capabilities of the LLM, demonstrating that this approach can further unleash the performance of LLMs. 2. RC-CPT, compared to the Doc2Query and Query2Doc approaches, can further enhance the model's capability in relevance scoring.

4.4 Prior Knowledge Integration

Motivation: LLMs possess extensive world knowledge due to their training on massive text corpora. However, they often struggle to understand user search intentions for short and ambiguous queries in industrial search engines, which lack specialized and rapidly evolving domain knowledge. Additionally, the knowledge stored in LLMs is not always up-to-date, failing to capture new information. Therefore, supplementing LLMs with additional information is crucial for relevance discrimination, as external knowledge is continuously updated and can alleviate the LLMs' shortcomings in understanding new information. Fortunately, industrial-level search engines possess query understanding (QU) feature and document understanding modules that can provide insights into the user's intent behind a query and the entity relationships within documents. These features are updated in real-time, allowing LLMs to acquire new knowledge without model updates. Consequently, we introduce query understanding features and document understanding features to inject external knowledge into LLMs as shown in Figure 2. The QU and Doc features primarily consist of natural language tokens enhanced with specific numerical data to capture contextual nuances.

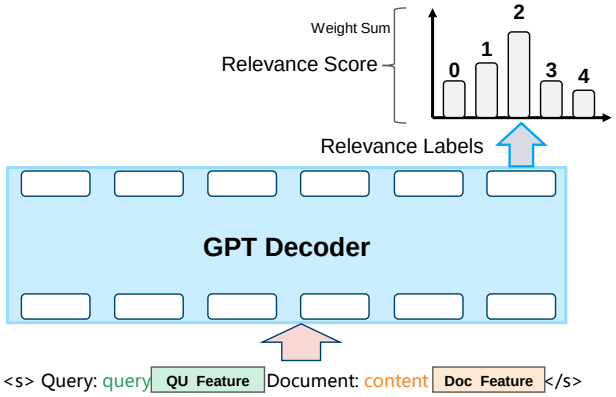


Figure 2: The overview of integrating prior knowledge into Large Language Models.

Experiment: We utilize a 7B model as a baseline to observe the performance after incorporating QU features and Doc analysis features. QU feature includes term analysis, intent information, and query expansion, while Doc feature include aggregated click queries for the document and its graph information. Additionally, we monitored the overall change in processing time after introducing this information. Enhancing LLMs with external knowledge increases the sequence length, presenting a trade-off since both processing time and performance are critical for an industrial search engine. The result is shown in Table 3.

Method	PNR	NDCG	inference time
GenFR(7B)	2.81	74.23	None
w/ QU Feature	2.94	75.45	+4%
w/ Doc Feature	2.87	74.91	+6%
w/ QU & Doc Feature	3.03	76.22	+10%

Table 3: The performance of integrating prior knowledge into Large Language Models.

Observations: The integration of prior knowledge significantly enhances the performance of LLMs, with the increase in processing time being almost negligible. The additional information is much shorter compared to the length of the query and document themselves, resulting in an average sequence length increase of about 12% and an inference time increase of around 10%. Compared to the integration of Doc feature, the integration of QU feature results in a greater improvement. This is because QU feature is more concentrated relative to Doc, and a more comprehensive understanding of the user's query can significantly enhance the model's performance.

5 Enhancing the Performance of the Existing Model Using LLMs

After achieving a strong relevance Large Language Model (LLM), the current challenge shifts to enhancing the performance of existing BERT using LLMs. In this paper, we primarily investigate three effective methods to improve existing BERT: using LLM as a relevance judge, LLM as hard sample maker, and distilling LLM into BERT.

5.1 LLM as Relevance Judge

Typically, BERT model training involves three stages: pre-training, post-pre-training, and fine-tuning. During the post-pre-training phase, a substantial amount of weakly supervised data, primarily derived from user click behavior, is utilized. However, click data can be problematic due to false positives, where a clicked document may not necessarily be relevant. To address this issue, an LLM can be used as a relevance judge to determine the true relevance of a document [8, 44].

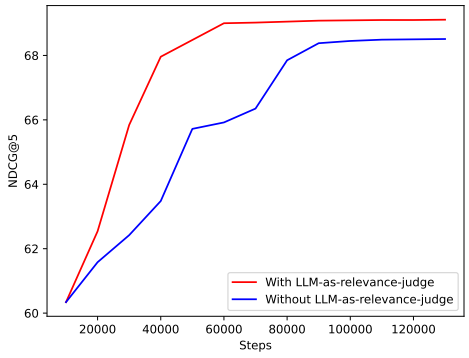


Figure 3: The performance of LLM as Relevance Judge in enhancing BERT's effectiveness with weakly supervised click data training process.

Specifically, we randomly sample from the training set and construct a batch of pairs. Then, we calculate the score differences between the output levels of the BERT model and the LLM model. Finally, we re-weight the loss based on score differences for back-propagation in the BERT model. If score differences is larger, we consider the sample to be more difficult and therefore assign a higher weight to such samples.

Experiment: We used a 7B GenFR as the judge model and a 12-layer BERT model for the post-pre-training phase. The baseline used the full volume of click data for training, while the test model utilized guidance from the LLM.

Observations: As illustrated in Figure 3, the LLM as Relevance Judge strategy not only improves the performance of the BERT model during the post-pre-training phase but also enhances training efficiency.

5.2 LLM as Hard Sample Maker

In routine iterations of relevance models, we collect problematic cases from model scoring and traditionally design manual rules to construct positive and negative samples for subsequent fine-tuning. While these samples can target specific issues, they generally exhibit poor generalization and often suffer from low quality, leading to semantic incoherence and contextual discontinuity. However, LLMs possess creative generative capabilities that can supplant manual rules to generate more effective positive and negative samples. Consequently, we utilize prompts to specifically generate challenging samples, which are then used to fine-tune BERT.

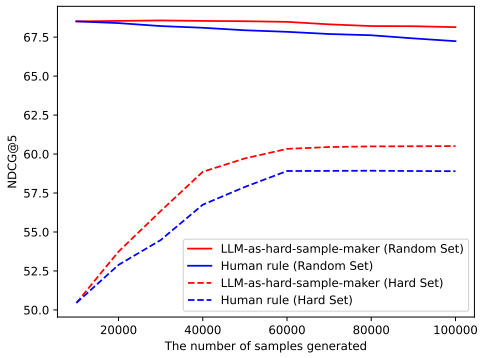


Figure 4: The performance of LLM as Hard Sample Maker on hard and random test sets.

Experiment: For high-frequency head queries, which typically perform well due to sample training data, we aim to enhance sample efficiency by first selecting difficult samples from the existing training set as seed samples. The seed samples for generating hard samples were selected based on specific criteria aimed at identifying challenging cases. These seed samples are then rewritten based on LLMs. The process involves: Generating samples based on keywords mismatch. Enhancing samples based on scattered hits. For manual rules, we primarily replace or delete certain terms in the document to construct positive and negative samples. Meanwhile, LLM experiments were conducted using a 30B LLM. The experimental results are displayed in Figure 4.

Observations: Samples constructed through manual rules, due to their low quality, degrade the performance of model on random samples, although they show improvement on difficult samples. In contrast, samples generated by LLM not only enhance performance on difficult samples but also maintain the capability of model on random samples. The introduction of synthetic samples needs to be limited, excessive inclusion can diminish the performance of model on random samples.

5.3 Distill LLM into BERT

Motivation: Knowledge distillation is an effective method for transferring the ranking capabilities of LLMs to student models, thereby alleviating the challenges associated with deploying LLMs online. A straightforward approach involves computing the Mean Squared

Error (MSE) loss between the logits of the teacher and student models. Margin MSE [12] loss to calculate the score differences between document pairs under a query. To maintain the multi-class relevance capabilities and the ordinal scoring relationships of LLMs, we integrate an additional dense layer after the [CLS] token of BERT for 5-class classification, while adopting the same strategy as LLMs for fine-grained relevance scoring. During knowledge distillation, we employ Kullback-Leibler (KL) divergence to distill classification information and Margin MSE to distill scoring information as shown in Figure 5.

$$\begin{aligned} \mathcal{L}_{MarginMSE} &= MSE(Sim_T(q, d^+) - Sim_T(q, d^-) \\ &\quad , Sim_S(q, d^+) - Sim_S(q, d^-)) \\ \mathcal{L}_{KL}(S \parallel T) &= \sum_{x \in (0,1,2,3,4)} S(x) \log\left(\frac{S(x)}{T(x)}\right) \\ \mathcal{L}_{hybrid} &= \mathcal{L}_{KL} + \beta * \mathcal{L}_{Margin} \end{aligned} \tag{5}$$

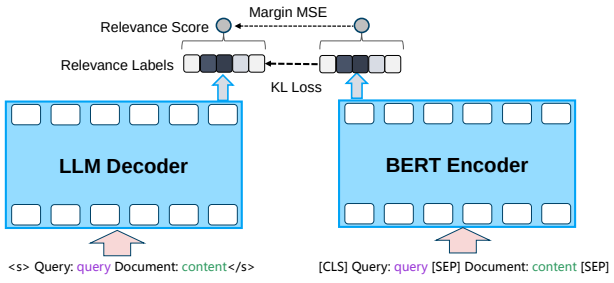


Figure 5: The overview distill LLM into BERT.

Experiment: We utilize 7B, 13B, and 30B GenFR as teachers and a 12-layer BERT as the student model to compare the performance of various distillation losses in Table 4. All teacher models are trained solely through supervised fine-tuning as shown in Table 1. The volume of distillation samples was maintained consistently across all strategies.

	PNR			NDCG@5		
Teacher	**Point**	**Margin**	**Hybrid**	**Point**	**Margin**	**Hybrid**
7B	2.47	2.54	2.55	**7**3.10	73.13	73.15
13B	**2**.80	2.85	2.89	75.02	75.13	75.20
30B	2.94	3.05	3.09	75.88	76.04	76.11
BERT(L)		2.44			71.73	
BERT(XL)		2.49			72.44	

Table 4: PNR and NDCG@5 Results across different teacher model sizes with LLMs using different distillation losses

Observations: For pointwise logits MSE loss, the distillation effectiveness decreases as the size of the teacher model increases. This may be attributed to the significant structural and knowledge differences between the BERT model and LLMs, which only widen as the parameters of LLMs increase. Margin MSE and mixed loss demonstrated consistent stability, unaffected by increases in LLM model size. Therefore, we recommend using a loss that incorporates Margin MSE for distillation loss selection, as it does not overly depend on the absolute values of scores of the teacher. Additionally, the reason BERT distilled from LLMs can perform better than the initial BERT model is that scores in [0,1] from the teacher can provide a confidence level for the sample scoring.

6 Efficient Online Deployment of LLMs

LLMs have demonstrated significant improvements in relevance modeling performance. However, their substantial computational requirements and latency during inference present challenges for deployment in real-world search scenarios. In this section, we discuss strategies to deploy LLMs online efficiently, aiming to balance performance with cost.

6.1 Model Quantization

In real-world search scenarios, deploying LLMs to handle all search traffic with acceptable cost and latency poses significant challenges. This challenge stems from the fact that LLMs have a parameter size orders of magnitude larger than that of online BERT models, while industrial search engines must operate within stringent latency constraints. Typically, the ranking phase can tolerate latencies of up to 30ms, but LLM inference times significantly exceed this threshold. We transfer 7B LLM from bf16 to int8 format using TensorRT's [1] SmoothQuant [25, 37]. Under conditions with a batch size of 32, this approach increased inference speed by 1.5 times, with a quantization loss of approximately 2% in PNR and NDCG. Additionally, we employ a calibration dataset of 10,000 entries during the quantization process to minimize accuracy loss.

6.2 On-Demand Computing

Motivation: In routine model iterations, we observe that beyond a certain size, the improvement in performance for popular queries becomes marginal. Consequently, these queries might not necessitate computation by LLMs, as switching to larger models does not demonstrate a significant advantage. Therefore, we explore the possibility of reducing the computational load of LLMs through query segmentation, employing LLMs only for query types where there is a substantial improvement. Specifically, we segment the dataset based on the frequency of queries and the number of terms, and assess the performance of BERT students and LLMs across different queries to determine where using LLMs is most cost-effective.

Experiment: We utilize a quantized 7B GenFR trained solely through SFT and distilled BERT students for comparison in Table 5. The test set was divided into Head, Torso, and Tail categories based on query frequency and the number of terms in the queries, with respective proportions of 50%, 30%, and 20%.

Observations: As the popularity of queries increases, the gap between the scores of BERT students and GenFR narrows. This phenomenon occurs because as the frequency of queries increases, the scoring patterns become simpler and more frequent in the training samples, making the model easier to fit. However, long-tail queries, which require stronger understanding capabilities, are where LLMs can truly demonstrate their strength. Employing LLMs solely for

[1] https://github.com/NVIDIA/TensorRT-LLM

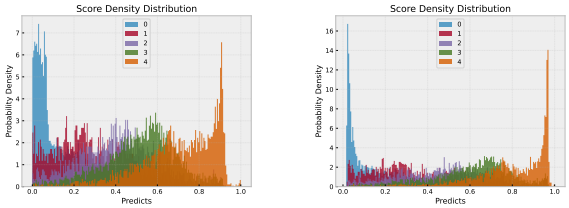
	PNR			NDCG@5		
Model Size	**Head**	**Torso**	**Tail**	**Head**	**Torso**	**Tail**
BERT	2.88	2.65	2.14	**75.14**	74.39	70.41
GenFR	**2.95**	2.78	2.62	75.33	74.54	73.42

Table 5: Comparison of quantized 7B GenFR and distilled BERT student across Head, Torso, and Tail test sets.

long-tail queries can achieve a better performance-cost trade-off, as the difference in effectiveness between BERT and LLMs is greatest for these queries. According to statistics, these queries account for about 14% of the total queries, resulting in a corresponding reduction in required resource consumption.

6.3 Model Calibration

While query-based computational segmentation effectively reduces the computational burden on models, it introduces a significant challenge: inconsistent scoring distributions between the two model types, necessitating distinct adaptation methods for downstream applications. In our system, relevance scores between queries and documents act as a one-dimensional feature that influences the final ranking. To manage the increased system complexity, we have implemented a score calibration method that aligns the scoring distributions of both models. To preserve the integrity of the training process and ranking outcomes, we adopted a Bayesian post-calibration approach [24, 42]. Specifically, we constructed an unbiased calibration set and calibrated the LLM and BERT models to the same distribution using Bayesian methods. The calibrated scores and their ranks are then utilized downstream. It is crucial to note that the calibration set is independent of both the training and testing sets.



(a) Before Calibration (b) After Calibration

Figure 6: The output score distribution of before and after model calibration.

Calibration Process: Steps 1, construct an unbiased calibration set and calculate the proportion of each rank to obtain its prior probability, P(label). Steps 2, based on the rank distribution in the dataset, obtain P(score | label) using Kernel Density Estimation (KDE), which directly estimates the input data for each rank, resulting in n sets of data for n models. Steps 3, for each score, calculate $P(label|score)$ using the Bayesian posterior derived from steps 1 and 2. Steps 4, the calibrated score for each score is the weighted average of $P(label|score)$ across all ranks.

Observations: Figure 6 illustrates the scoring distributions of the GenFR before and after calibration. It is clear from the figure that after Bayesian calibration, the model's scoring distribution becomes more uniform and meaningful. Through the various strategies outlined above, we achieve cost-effective efficiency and met resource requirements for utilizing LLMs to handle relevance judgments in industrial scenarios.

The final online architecture is depicted in Figure 7. When a query requests the ranking service, we first calculate the query length and page views (PV) to determine whether it belongs to the long tail. If it is classified as long tail, we asynchronously request the 7B GenFR and cache the relevance scores between the query and documents. Upon subsequent requests, we prefetch the scores from the cache. For query-document pairs without cached scores, we immediately invoke BERT for ranking while simultaneously making an asynchronous request to the 7B GenFR.

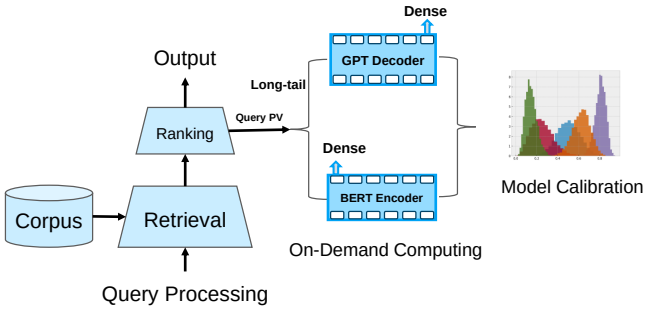


Figure 7: Overview of the online relevance model computation architecture.

7 Experiments

7.1 Production Baselines

Our production baselines utilize a 12-layer BERT model that undergone model distillation using a 7B LLM teacher (DisRanker) with Query2Doc CPT [39]. The training employs a hybrid loss combining point-wise and pair-wise approaches. It ranks dozen of candidates, each assigned a relevance score ranging from 0 to 1 and a categorical relevance value from {0,1,2,3,4}. As for the online A/B test model, we employed 7B GenFR as the teacher with Setwise RC-CPT and QU & Doc Feature, then used a hybrid loss to distill BERT. Table 6 shows the performance comparison between the production baseline and the proposed model.

Method	**PNR**	**NDCG**
GenFR (RC-CPT, QU & Doc Feature)	3.48	77.98
w/ Quantization	3.40	76.55
BERT-base (Distill from GenFR)	3.22	76.13
BERT-base (DisRanker)	3.03	75.06

Table 6: Performance comparison between the production baseline and the proposed model.

In terms of contribution, the greatest improvement is achieved by RC-CPT, followed by rank loss and QU & Doc Feature. These results indicate that continued pre-training of the LLM base model is advisable before tackling downstream relevance tasks.

7.2 Evaluation Metrics

We primarily utilize two categories of evaluation metrics to assess the performance of the ranking system: User Engagement and Expert Judgment.

User Engagement: This category measures user involvement, primarily using the following metrics: CTR: This metric assesses whether a user clicked on any result after a single query. Duration Time per Query: This measures the length of time a user spends on a document after clicking on it. Change Query Rate: This evaluates whether a user initiates another query shortly after the first, with both queries being similar. A lower rate indicates higher user satisfaction with the search results.

Expert Judgment: This category involves assessments by experts on how well the search results meet user needs. Good vs. Same vs. Bad (GSB): This metric compares two systems side-by-side, as measured by expert judgment.

7.3 Results of Improving the Performance of the Existing Model Using LLM

To validate the effectiveness of three methods for enhancing the current BERT model, we conducted comparative experiments in the production system over two weeks. Table 7 reports the performance comparison between different models regarding AB and GSB metrics. For random queries, the Click-Through Rate (CTR) increased by 1.4%, Duration Time per Query improved by 2.3%, and the Change Query Rate decreased by 0.33%. These results demonstrate that with the assistance of LLMs, the existing BERT model can better understand user queries, thereby increasing user engagement. The GSB of the expert assessment improved by 9%, revealing that with the help of LLMs, the BERT model not only retrieves relevant documents but also prefers high-quality results. Moreover, we also observe that our proposed schema significantly outperforms the online baseline system for long-tail queries. From the User Engagement metrics, the CTR increased by 2.5%, the duration time per query by 3.22%, and the change in the query rate decreased by 0.56%. From the Expert Judgment metrics, the improvement of GSB for long-tail queries is 11%. This indicates that our method has enhanced the BERT model's scoring ability for more challenging queries through the migration from LLMs.

Metric	Random	Tail
CTR	+1.4% ↑	+2.5% ↑
Duration Time	+2.3% ↑	+3.22% ↑
Change Query Ratio	-0.33% ↓	-0.56% ↓
$\triangle_{GSB}$	+9% ↑	+11% ↑

Table 7: Online A/B test results of Improving the Performance of the Existing Model Using LLM.

7.4 Results of Deploying LLMs Online

To investigate the effectiveness of relevance LLMs in a real-world commercial search engine, we deploy quantized GenFR to the online search system and compare it with the baseline model in a real production environment. Our experiment lasted two weeks, and Table 8 reports the performance comparison between different models regarding AB and GSB. Since LLMs are only effective for long-tail queries, we focus exclusively on these. Particularly, the CTR increased by 1.8%, Duration Time per Query by 3.4%, and Change Query Rate decreased by 0.74%, indicating that users are more satisfied with the search results and show higher engagement. Furthermore, we noticed an improvement of GSB for long-tail queries of 14%, which indicates that the search results are more relevant compared to the baseline model. Direct deployment of the LLM outperforms distillation, suggesting that future efforts will concentrate on maintaining effectiveness over the full query set and achieving real-time inference with the LLM.

Metric	△ Gain	P value
CTR	+1.8% ↑	0.01
Duration Time	+3.4% ↑	0.02
Change Query Ratio	-0.74% ↓	0.005
$\triangle_{GSB}$	+14% ↑	0.001

Table 8: Online A/B test results of Deploying LLMs Online. The p-value is less than 0.05

Cost Analysis: Although the deployment costs of LLMs are considerably high, we utilize LLMs for only 14% of queries. Consequently, the additional resource consumption involved approximately 50 A100 GPUs. Given the enhancements provided, this additional cost is deemed acceptable for industrial search engines.

8 Conclusion

In this paper, we share our experiences in applying large language models (LLMs) for relevance search. To explore how to adapt LLMs to relevance ranking, we conducted extensive experiments on large real-world datasets to assess the impact of ranking objectives, model size, domain-specific continued pre-training, and the integration of prior knowledge on ranking performance. These experiences can be used to identify the best resource allocation strategy given a restricted budget and could provide important insights for the practical implementation of LLMs. Additionally, to alleviate the challenges of deploying LLMs online, we explore three methods to enhance existing BERT models using LLMs: LLM-as-relevance-judge, LLM-as-hard-sample-maker, and distilling LLMs into BERT. Finally, our offline experiments observe that the improvements in relevance capabilities of LLMs over BERT are concentrated on long-tail queries. Therefore, we successfully deploy LLMs in a large-scale search engine using quantization and query on-demand computing, achieving a balance between cost and effectiveness. The comprehensive experiments, both offline and online, validate the superiority of our proposed solutions. In the future, we will explore the use of LLMs for relevance search under full traffic conditions, while simultaneously enhancing the model's reasoning capabilities.

References

[1] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961* (2024).

[2] 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* (2022).

[3] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[4] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[5] Zeyuan Chen, Haiyan Wu, Kaixin Wu, Wei Chen, Mingjie Zhong, Jia Xu, Zhongyi Liu, and Wei Zhang. 2024. Towards Boosting LLMs-driven Relevance Modeling with Progressive Retrieved Behavior-augmented Prompting. *arXiv preprint arXiv:2408.09439* (2024).

[6] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755* (2022).

[7] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Guglielmo Faggioli, Laura Dietz, Charles LA Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, et al. 2023. Perspectives on large language models for relevance judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*. 39–50.

[9] Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. 2024. Scaling laws for dense retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1339–1349.

[10] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding BERT rankers under distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 149–152.

[11] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

[12] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666* (2020).

[13] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv preprint arXiv:2301.01820* (2023).

[14] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. *arXiv preprint arXiv:2405.17428* (2024).

[15] Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327* (2024).

[16] Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024. Making text embedders few-shot learners. *arXiv preprint arXiv:2409.15700* (2024).

[17] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594* (2024).

[18] Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu-lee Wang, Qing Li, and Xiaoqin Zhong. 2023. Label supervised llama finetuning. *arXiv preprint arXiv:2310.01208* (2023).

[19] 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* (2022).

[20] Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2024. Leveraging passage embeddings for efficient listwise reranking with large language models. *arXiv preprint arXiv:2406.14848* (2024).

[21] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2421–2425.

[22] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156* (2023).

[23] Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904* (2022).

[24] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[25] Jiayi Pan, Chengcan Wang, Kaifu Zheng, Yangguang Li, Zhenyu Wang, and Bin Feng. 2023. Smoothquant+: Accurate and efficient 4-bit post-training weightquantization for llm. *arXiv preprint arXiv:2312.03788* (2023).

[26] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088* (2023).

[27] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724* (2023).

[28] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563* (2023).

[29] Zhen Qin, Rolf Jagerman, Rama Kumar Pasumarthi, Honglei Zhuang, He Zhang, Aijun Bai, Kai Hui, Le Yan, and Xuanhui Wang. 2023. RD-Suite: a benchmark for ranking distillation. *Advances in Neural Information Processing Systems* 36 (2023).

[30] Hossein A Rahmani, Clemencia Siro, Mohammad Aliannejadi, Nick Craswell, Charles LA Clarke, Guglielmo Faggioli, Bhaskar Mitra, Paul Thomas, and Emine Yilmaz. 2025. Judging the judges: A collection of llm-generated relevance judgements. *arXiv preprint arXiv:2502.13908* (2025).

[31] Sashank Reddi, Rama Kumar Pasumarthi, Aditya Menon, Ankit Singh Rawat, Felix Yu, Seungyeon Kim, Andreas Veit, and Sanjiv Kumar. 2021. Rankdistil: Knowledge distillation for ranking. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2368–2376.

[32] Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496* (2022).

[33] 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. *arXiv preprint arXiv:2304.09542* (2023).

[34] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. *arXiv preprint arXiv:2312.16098* (2023).

[35] Han Wang, Mukuntha Narayanan Sundararaman, Onur Gungor, Yu Xu, Krishna Kamath, Rakesh Chalasani, Kurchi Subhra Hazra, and Jinfeng Rao. 2024. Improving Pinterest Search Relevance Using Large Language Models. *arXiv preprint arXiv:2410.17152* (2024).

[36] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368* (2023).

[37] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR, 38087–38099.

[38] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: BERT and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*. 1154–1156.

[39] Dezhi Ye, Junwei Hu, Jiabin Fan, Bowen Tian, Jie Liu, Haijin Liang, and Jin Ma. 2024. Best Practices for Distilling Large Language Models into BERT for Web Search Ranking. *arXiv preprint arXiv:2411.04539* (2024).

[40] Dezhi Ye, Jie Liu, Jiabin Fan, Bowen Tian, Tianhua Zhou, Xiang Chen, and Jin Ma. 2024. Enhancing asymmetric web search through question-answer generation and ranking. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6127–6136.

[41] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 323–332.

[42] Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, Vol. 1. 609–616.

[43] Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2024. A two-stage adaptation of large language models for text ranking. In *Findings of the Association for Computational Linguistics ACL 2024*. 11880–11891.

[44] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2023), 46595–46623.

[45] Honglei Zhuang, Zhen Qin, Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Ensemble distillation for bert-based ranking models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. 131–136.

[46] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2023. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. *arXiv preprint arXiv:2310.14122* (2023).

[47] Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2024. PromptReps: Prompting Large Language Models to Generate Dense and Sparse Representations for Zero-Shot Document Retrieval. *arXiv preprint arXiv:2404.18424* (2024).

[48] Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 38–47.

[49] Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model based ranking in Baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4014–4022.