LLM4PR: Improving Post-Ranking in Search Engine with Large Language Models

Yang Yan, Yihao Wang, Chi Zhang, Wenyuan Hou, Kang Pan, Xingkai Ren, Zelun Wu, Zhixin Zhai, Enyun Yu, Wenwu Ou and Yang Song

Kuaishou Technology

Beijing, China

ABSTRACT

Alongside the rapid development of Large Language Models (LLMs), there has been a notable increase in efforts to integrate LLM techniques in information retrieval (IR) and search engines (SE). Recently, an additional "post-ranking" stage is suggested in SE to enhance user satisfaction in practical applications. Nevertheless, research dedicated to enhancing the post-ranking stage through LLMs remains largely unexplored. In this study, we introduce a novel paradigm named Large Language Models for Post-Ranking in search engine (**LLM4PR**), which leverages the capabilities of LLMs to accomplish the post-ranking task in SE. Concretely, a Query-Instructed Adapter (QIA) module is designed to derive the user/item representation vectors by incorporating their heterogeneous features. A feature adaptation step is further introduced to align the semantics of user/item representations with the LLM. Finally, the LLM4PR integrates a learning to post-rank step, leveraging both a main task and an auxiliary task to fine-tune the model to adapt the post-ranking task. Experiment studies demonstrate that the proposed framework leads to significant improvements and exhibits state-of-the-art performance compared with other alternatives.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Large Language Model, Search Engine,Generative Post-ranking

1 INTRODUCTION

Conventional search engines (SE) and information retrieval (IR) systems are composed of multiple stages including matching and ranking, in which the relevant items are retrieved in matching
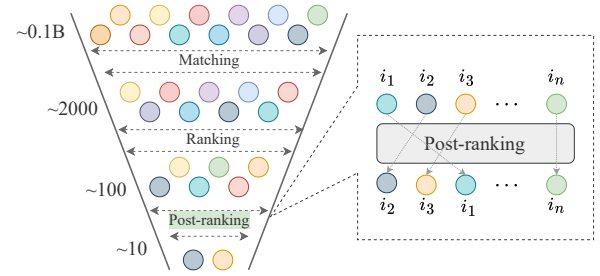
Figure 1: Illustration of the search process, encompassing matching, ranking and post-ranking stages.

stages and sorted by relevance in ranking stage. However, in practical applications, the relevance score of a single item is not the sole measure for practical search engines. On e-commerce platforms, the search engine is required not only to provide relevant results but also to maximize users' purchase rate by presenting the product list. Meanwhile, in the short video search scenario, the search platform also attempts to maximize users' play time during the entire search session. To this end, a post-ranking stage is suggested in practical applications. As depicted in Figure 1, the post-ranking stage severs as the final stage in search engines. This stage considers multiple attributes (item relevance score, user click/purchase rate, *etc.*) as well as mutual influences between items, and deliver the final list to optimally enhances user experience in the search session.

In IR and SE, the classic Learning-To-Rank (LTR) method formulates a composite objective encompassing item relevance and click/purchase to model the post-ranking stage. Recently, numerous additional efforts have been dedicated to studying this search topic[1, 3, 15, 19, 26, 34, 55]. Concretely, SetRank [34] proposes the self-attention method to model mutual influences between items, and DLCM [2] suggests a RNN-based approach. PRM [35], PRS [13] and PIER [42] suggest permutation-based two-stage processes for list generation and evaluation. Another pioneering stream of studies on post-ranking stages focus on the generative method. Seq2slate[4] and GRN [14] introduce the generative method in post-ranking stage in recommendation scenario[1]. Regarding the increasing popularity of generative methods, especially its significant success in Large Language Models (LLMs), the generative and LLM-based approaches emerge as a promising avenue for optimizing post-ranking stage in search engine.

The advances in Large Language Models (LLMs) have achieved remarkable success in Natural Language Processing (NLP) and IR

---

[1]In recommender system, "re-ranking" refers to "post-ranking" stage. While in information retrieval, "re-ranking" denotes the "ranking". In case of confusion, we call the last stage in search engine as "post-ranking" in this paper.

tasks [6, 11, 33, 46–48], and it has been prompting increasing integration between LLM and practical applications such as search engine (SE). Currently, most existing LLM methods for SE primarily focus on augmenting document retrieval [5, 10, 20, 28, 49], document ranking/re-ranking [36, 43] and modeling relevance between queries and items [7, 36, 41, 54]. However, despite significant efforts being directed towards matching (*i.e.*, item retrieval) and ranking within SE, the post-ranking stage has been overlooked. Consequently, the explorations of LLMs for the post-ranking stage in SE remain scarce. To implement LLMs for the post-ranking process in search engine, several challenges have yet to be overcome.

Issue 1: Heterogeneous Features Input. The input of post-ranking model commonly encompasses heterogeneous features including item descriptions, category ID features, item-level statistical features, and the output (*i.e.*, numerical embedding) from the upstream (*i.e.*, ranking) stage. However, current LLMs are specifically designed to process semantic textual input only. Directly inputting these feature embeddings into the LLM could potentially lead to confusion, as these embeddings lack semantic context. Therefore, developing a mechanism to seamlessly incorporate these diverse features into the LLM input and align the feature representation with the LLM remains a significant challenge.

Issue 2: Task Specification for Post-Ranking. Existing LLMs are designed for general purposes (*i.e.*conversation, question and answering), exhibiting limited capacities for the post-ranking task in practical applications. Thus, it is essential to explore appropriate adjustments to enhance the LLM's ability for optimizing the post-ranking task.

To address these issues, we propose an L̲LM-based framework for P̲ost-R̲anking in search engine, named **LLM4PR**, comprising a Query-Instructed Adapter (QIA) component and a backbone LLM. To handle the challenge of heterogeneous features (**Issue 1**), we propose a feature adaptation step to integrate diverse features. The QIAs take user/item side features as input, and combine various features in a query-instructed manner. Specifically, within the QIAs, the search query assigns distinct weights to diverse feature domains via the attention mechanism and derive the representation vectors for the user/item. Moreover, a template-based generation task is designed to achieve the semantic alignment between the user/item representations and the LLM. Particularly, we freeze the LLM and train the QIAs to generate semantic user/item representations that can be decoded by the LLM. To tackle the post-ranking task challenge (**Issue 2**), we design a main task and an auxiliary task to fine-tune the **LLM4PR** to learn to post-rank, named learning to post-rank step. The main task involves taking candidate item list as input and instructing **LLM4PR** to predict the target post-ranking order in a generative manner. To produce satisfying post-ranking lists, the model is required to assess the quality of various candidate item lists. To this end, we propose an auxiliary task that compares a pair of candidate lists and predicts the superior one. The auxiliary task serves as a supplement to the main task, supporting the model in developing an insight for judging the quality of candidate lists. In the training stage, both the main task and the auxiliary task are trained simultaneously. During the inference stage, we utilize the main task to generate the final post-ranking results.

Overall, the contributions of this paper can be summarized as follows:

- Firstly, to the best of our knowledge, our **LLM4PR** is the first LLM-based framework that optimizing the post-ranking stage for search engines. This framework leverages the robust capabilities of LLM and generates the final results straightforwardly.
- Secondly, the proposed **LLM4PR** suggests a novel QIA component and feature adaptation step to align heterogeneous features. Additionally, it introduces an efficient template-based learning to post-rank step for model tuning.
- Last but not least, the effectiveness of the proposed **LLM4PR** is fully validated through comprehensive experiment studies.

2 RELATED WORKS

Post-Raking in Search Engine. In the context of the post-ranking stage, where not only document relevance but also user satisfaction are taken into consideration in practical applications, [26] categorizes the methods from the perspectives of modeling objectives and learning signals. Early methods [2, 13, 34, 35, 42, 55] consider the single objective and learn from supervised signals. [1] proposes location diversity ranker loss to optimize the diversity issue in Airbnb Search. Among generative post-ranking models, ListCVAE [21] selects the Conditional Variational AutoEncoder (CVAE) to generate the slate straightforwardly, while Seq2slate [4] and GRN [14] predict the next item sequentially via pointer network and policy gradient, respectively. [19] further introduces a discriminator to ensure the generated results are close to real lists.

LLM for Information Retrieval. In recent years, research on Large Language Models (LLMs) has gained considerable attention. Various pre-trained models (BERT[11], T5[38, 48], BART[22]) and LLMs (LLaMA series [46, 47], GPT series [6, 33], Baichuan [51], GLM [12]) are proposed, and [52] provides a comprehensive survey reviewing the development of LLMs. In light of the remarkable successes achieved by LLMs for natural language problems, considerable efforts have been dedicated to integrating LLMs with information retrieval. [53] offers an insightful overview of potential applications of LLMs in information retrieval, including query rewriter, retriever, text ranking and so on. The text ranking capabilities of LLMs have been comprehensively explored through various methodologies, including the pointwise approach [7, 41, 54], the pairwise approach [36], and the listwise approach [29, 43]. [28] proposes a LLaMA-based retriever and ranker simultaneously. [43] investigates ChatGPT for passage ranking straightforwardly and suggests model distillation. Further works on LLMs for document ranking could be found in [17, 23, 24, 37, 50]. However, the aforementioned methods do not pay attention to the post-ranking stage, and LLM-based methods for post-ranking in search engine are urgently needed.

3 PRELIMINARY

In search engines, post-ranking refines the results list by accounting for both item quality and the mutual influences between items, and determines the ultimate outcomes to maximize users experience. Particularly, given a request $\mathcal{R}$ and its output list $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$ from previous stage (*i.e.*, ranking stage), the post-ranking algorithm aims to refine the order and produce a new ranking list $\mathcal{I}^* = \{i_1^*, i_2^*, \cdots, i_n^*\}$ that better matches the users preferences. Therefore, the post-ranking stage can be formulated as
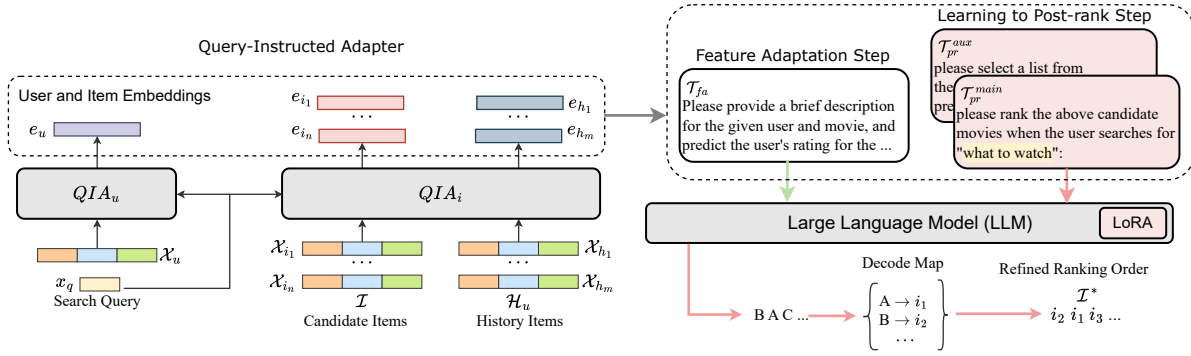
Figure 2: Overview of the proposed LLM4PR framework.

a function $f : \mathcal{R} \times \mathcal{I} \mapsto \mathcal{I}^*$, where the request $\mathcal{R} = \{q, \mathcal{X}_u, \mathcal{H}_u\}$ comprises the raw search query q, user profile features $\mathcal{X}_u$, and information about user history behavior items $\mathcal{H}_u$.

4 METHODOLOGY

4.1 Overview of LLM4PR

The overview structure of the proposed LLM4PR is illustrated in Figure 2. Specifically, QIA_u takes query embedding x_q and user feature embeddings $\mathcal{X}_u$ (*e.g.*, id embedding, gender embedding, *etc.*) as input, and produces a single embedding e_u to represent the user. Similarly, QIA_i takes query embedding x_q and candidate/history item feature embeddings $\mathcal{X}_i/\mathcal{X}_h$ as input, and generates a hidden vector for each item in $\mathcal{I}/\mathcal{H}_u$. Note that, the query embedding and feature embeddings of the user/item fed into the QIA are fetched from previous stage (*i.e.*, the ranking stage), instead of training from scratch. Subsequently, the user embedding e_u, candidate item embeddings e_i, behavior item embeddings e_h, and the raw query q are combined using a post-ranking template $\mathcal{T}_{pr}^{main}$. The template provides a hybrid input instruction, containing both texts and embeddings, and the LLM directly predicts the final list in a generative manner according to the given input instruction. Next, we will introduce the designed QIA and training tasks for LLM4PR.

4.2 Query-Instructed Adapter

Traditional post-ranking methods utilize diverse features to represent a specific user or item. A straightforward way to utilize these features in LLMs is to combine all texts into one sentence as the input for LLM (serving as one baseline method in Section 5). However, considering current ranking methods include hundreds or even thousands of different features, the combined sentence may become excessively long, causing significant increase in inference time. To this end, we propose the QIA component, which utilizes the attention mechanism to merge various features of the user/item and delivers a single embedding vector to represent the specific user/item.

Specifically, the QIA takes search query embedding $x_q \in \mathbb{R}^{1 \times d_q}$ and multiple user/item features $\mathcal{X} = \{x_1, x_2, \cdots, x_k\}$ as input, where $x_j \in \mathbb{R}^{1 \times d_j}$ is the j-th feature vector with dimension d_j. We concatenate all the numerical features (*e.g.*, category embeddings, predicted scores from the up-stream stage, *etc.*) and treat it as an embedding vector. It is noted that the query embedding

x_q and feature vectors $\mathcal{X}$ are fetched from the ranking stage and fixed during the training phase. The search query q reflects user demand and may align with specific characteristics of the items. For example, when a user searches for a *romance film*, the model should focus on the genre of the movies. Thus, we implement a feature field attention technique to automatically align various features based on the query. Concretely, the projection heads are first applied on the query and feature embeddings to produce the Query Q, Key K and Value V, which can be formulated as

$$Q = x_q W_Q, \tag{1}$$
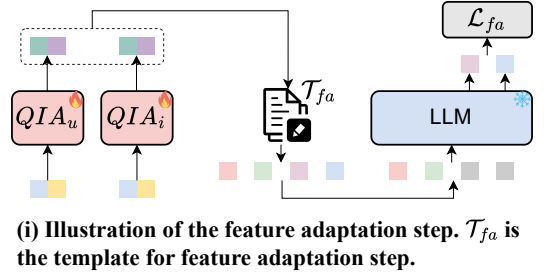$$K = \{x_1 W_{K_1}; x_2 W_{K_2}; \cdots; x_k W_{K_k}\}, \tag{2}$$
$$V = \{x_1 W_{V_1}; x_2 W_{V_2}; \cdots; x_k W_{V_k}\}, \tag{3}$$

where $Q \in \mathbb{R}^{1 \times d_h}$, $K, V \in \mathbb{R}^{k \times d_h}$, $W_Q \in \mathbb{R}^{d_q \times d_h}$, $W_{K_j}, W_{V_j} \in \mathbb{R}^{d_j \times d_h}$, d_h represents the attention hidden size, and k is the number of feature fields. Then, a standard target attention is employed to derive the hidden vector $h_w = \text{Softmax}(\frac{QK^\top}{\sqrt{d_h}})V$, where $h_w \in \mathbb{R}^{1 \times d_h}$. Note that, the QIA attention mechanism provides a fine-grained interaction approach at the feature level, which enriches the expression of the user/item embedding. Finally, an additional linear projection maps the size of the hidden feature vector to the LLM hidden size: $e = \text{Linear}(h_w)$, where $e \in \mathbb{R}^{1 \times d_{llm}}$ is the final embedding to represent the specific user/item. d_{llm} is the hidden size for the LLM.

4.3 Feature Adaptation Step

LLMs are initially proposed to handle the NLP tasks, which take semantic texts as input. However, the user/item representations provided by the QIAs lack semantic information and cannot be understood by the LLM. Thus, we introduce a feature adaptation step to align the user/item representation embeddings with the LLM, serving as the preparatory phase for subsequent learning to post-rank. In this step, we freeze the LLM and fine-tune the QIAs to produce user/item representations that are most concordant with their semantic information.

The process of feature adaptation step is illustrated in Figure 3. The QIA takes query q and user/item feature vectors $(\mathcal{X}_u/\mathcal{X}_i)$ as input and produces the representation vector for the user/item (e_u/e_i). A template $\mathcal{T}_{fa}$ is introduced to integrate the query, user representation and item representation into a prompt sentence.

(i) Illustration of the feature adaptation step. $\mathcal{T}_{fa}$ **is the template for feature adaptation step.**

$\mathcal{T}_{fa}$
Input: User: e_u Movie: e_i Please provide a brief description for the given user and movie, and predict the user's rating for the movie when the user seaches for "what to watch":
Output: This user is a male adult who is currently technician or engineer. He has rated a range of movies with an average rating of 3.6 .... This movie is a action & thriller movie and has an average rating of 4.1 .... He will rate this film 5 score.

(ii) Take MovieLens dataset as example to illustrate the template for the feature adaptation step.

Figure 3: Illustration of the feature adaptation step.

$\mathcal{T}_{fa}$, demonstrated in Figure 3(ii), is a hybrid template including both texts and numerical embeddings. There are two parts in the template, the input part is the input sentence for LLM and the output part is the target response that LLM is expected to produce.
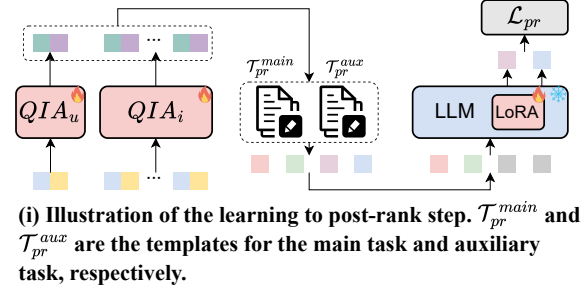
In template $\mathcal{T}_{fa}$, e_u and e_i are the user and item representations provided by QIA_u and QIA_i, respectively. Besides the user and item information, $\mathcal{T}_{fa}$ also contains the instructions for LLM. Take MovieLens[16] dataset as an example, $\mathcal{T}_{fa}$ requires the LLM to predict the user rating for the movie by providing descriptions for the input user and movie. The description generation aims to align the user/movie representation vector to its corresponding textual semantic. In this sense, the output of this instruction is constructed by organizing the various features of user/movie into a coherent sentence. Particularly, for categorical features such as gender, age group and occupation, we assemble the original meaning of the features as the description (*e.g.*, this user is a *male adult* who is currently *technician or engineer*). For numerical features, which typically are the predicted values from the previous stage or statistics features like average rating scores, we directly articulate these features by combining their semantic interpretations and numerical values (*e.g.*, the user has rated a range of movies with an *average rating of 3.6*). The user rating prediction instruction is designed to transfer the preference information from user/movie feature embeddings to its representation vector. In the output part of $\mathcal{T}_{fa}$, the response for predicting user ratings is straightforwardly given as the rating score like *he will rate this film 5 score.*

When constructing the input and output sentences for the LLM, we can train LLM4PR with such sentence pairs. Specifically, we freeze the LLM and train the parameters of the QIAs with the target of generating corresponding output sentence. We achieve this by maximizing the log likelihood for predicting target responses, and

the loss function can be formulated as

$$\mathcal{L}_{fa} = - \sum_{t=1}^{|y|} \log(P(y_t | q, \mathcal{X}_u, \mathcal{X}_i, y_{<t})), \qquad (4)$$

where $\mathcal{X}_u$ and $\mathcal{X}_i$ represent the user and item feature embeddings, respectively. $y_{<t}$ indicates the tokens predicted in the previous $t-1$ steps.

(i) Illustration of the learning to post-rank step. $\mathcal{T}_{pr}^{main}$ **and** $\mathcal{T}_{pr}^{aux}$ **are the templates for the main task and auxiliary task, respectively.**

$\mathcal{T}_{pr}^{aux}$
Input: User: e_u Candidate movies: (A)e_{i_1} (B)e_{i_2} (C)e_{i_3} ... The user has watched: e_{h_1} e_{h_2} ... We provide two ranking orders options: (I)ACB... (II)BAC... Based on the user and the user's history movie watching information, please select a list from the options that the user prefers when searches for "what to watch":
Response: I

$\mathcal{T}_{pr}^{main}$
Input: User: e_u Candidate movies: (A)e_{i_1} (B)e_{i_2} (C)e_{i_3} ... The user has watched: e_{h_1} e_{h_2} ... Based on the user and the user's history movie watching information, please rank the above candidate movies when searches for "what to watch":
Response: BAC .....

(ii) Take MovieLens dataset as example to illustrate the two task templates for the learning to post-rank step.

Figure 4: Illustration of the learning to post-rank step.

4.4 Learning to Post-rank Step

LLMs are originally designed for general purposes and necessitate proper modifications to effectively address the post-ranking task. Therefore, we introduce a learning to post-rank step that trains the generative LLM4PR framework. We construct the input for the LLM and instruct it to generate a suitable order for candidate items, which serves as the main task. In addition, we further incorporate an auxiliary task that evaluates the qualities of two candidate lists via pairwise comparisons. The auxiliary task determines the superior list, and this auxiliary task serves as a simplified version for generating final post-ranking outcomes.

The procedure of learning to post-rank step is described in Figure 4, QIA_u and QIA_i produce the user representation e_u and item representation vector e_{i_j} for item j in candidates I, respectively. The representation vector for the k-th item in user's history behaviors is denoted as e_{h_k}.

Two hybrid templates $\mathcal{T}_{pr}^{main}$ and $\mathcal{T}_{pr}^{aux}$ are proposed to implement the learning to post-rank step. According to Figure 4(ii), the input parts of both templates are comprised of query q, user e_u,

---

Algorithm 1 LLM4PR

Input: Parameters of QIA_u: θ_{QIA_u}; Parameters of QIA_i: θ_{QIA_i}; LoRA parameters: θ_{LoRA}; Post-ranking dataset: $\mathcal{D}$; Maximum iteration step: T_1, T_2
Output: Optimized learnable parameters

    // feature adaptation step
1: **for** $t = 0$ to $T_1 - 1$ **do**
2:     **for** sampled minibatch $\mathcal{D}_{batch}$ from $\mathcal{D}$ **do**
3:         construct training data using $\mathcal{T}_{fa}$
4:         update θ_{QIA_u} and θ_{QIA_i} using Equation 4
5:     **end for**
6: **end for**
    // learning to post-rank step
7: **for** $t = 0$ to $T_2 - 1$ **do**
8:     **for** sampled minibatch $\mathcal{D}_{batch}$ from $\mathcal{D}$ **do**
9:         construct training data using $\mathcal{T}_{pr}^{aux}$ and $\mathcal{T}_{pr}^{main}$
10:       update θ_{QIA_u}, θ_{QIA_i}, and θ_{LoRA} using Equation 5
11:     **end for**
12: **end for**
13: **return** θ_{QIA_u}, θ_{QIA_i}, θ_{LoRA}

---

candidates e_i and history behaviors e_h. In the auxiliary task template $\mathcal{T}_{pr}^{aux}$, two randomly generated candidate lists, denoted by the letters I and II, are provided for each training sample, prompting the LLM to identify the superior option. The quality of the generated ranking list is assessed by Normalized Discounted Cumulative Gain (NDCG). As a result, the target response indicates the list that achieves a higher NDCG value. Regarding the main task template $\mathcal{T}_{pr}^{main}$, it requires the LLM to post-rank all candidates. In this sense, the response to this instruction should be as plain as possible to simplify the generation process. Therefore, the output is formatted as a sequence of capital letters, with each letter symbolizing a specific item among the candidates. The final post-ranking list is determined by mapping letter A to i_1, letter B to i_2, and so forth.

When training the LLM4PR, the main task and auxiliary task are combined to concurrently fine-tune the QIAs and LLM. However, fine-tuning all the parameters of the LLM is time-consuming and requires massive computational resources. To this end, we utilize LoRA[18], which brings a separate set of trainable parameters and keeps the pre-trained parameters of LLM frozen, to reduce the costs of fine-tuning. In our implementation, LoRA parameters are exclusively integrated into the projection parts within the self-attention modules of the LLM. We train the LLM4PR to predict the corresponding response using next-token prediction, which maximizes the log likelihood of the subsequent token. The loss function of this stage is represented as

$$\mathcal{L}_{pr} = - \sum_{t=1}^{|y|} \log(P(y_t | q, \mathcal{X}_u, \mathcal{I}, \mathcal{H}_u, y_{<t})), \tag{5}$$

where $\mathcal{I}$ and $\mathcal{H}_u$ represent the set of candidate items and history behavior items, respectively. $y_{<t}$ indicates the tokens predicted in the previous $t - 1$ steps.

4.5 Training and Inference

In the training stage, the feature adaptation step and learning to post-rank step are performed sequentially, and the entire process for training LLM4PR is illustrated in Algo 1. In the inference stage, only $\mathcal{T}_{pr}^{main}$ is utilized to construct the inputs for the LLM, which instructs the LLM to generate the post-ranking list. The final post-ranking results are obtained by decoding the generated sequence produced by the LLM4PR.

5 EXPERIMENTS

Table 1: Statistics of the datasets.

Dataset	#Users	#Items	#Queries	#Actions
Covid	-	171,000	50	-
NFCorpus	-	3,600	323	-
Touche	-	382,000	49	-
DBPedia	-	4,630,000	400	-
SciFact	-	5,000	300	-
Signal	-	2,860,000	97	-
News	-	595,000	57	-
Robust04	-	528,000	249	-
MovieLens-1M	6,040	3,883	-	1,000,209
KuaiSAR(Search Part)	25,877	3,026,189	453,667	5,059,169

5.1 Experiment Settings

5.1.1 Datasets. To validate the post-ranking performance of the proposed LLM4PR, we conduct experiments on two kinds of datasets: information retrieval dataset and search dataset. For the information retrieval dataset, we select several sub-datasets from the BEIR dataset[45], which contain only pure text feature. Moreover, we select MovieLens-1M[16] and KuaiSAR[44] as the search datasets. Both datasets contain heterogeneous features and are modified to adapt the search post-ranking scenario. It is also noted that the KuaiSAR dataset is an industrial dataset collected from the Kuaishou App, the second largest short-video platform in China. The statistics of the datasets are shown in Table 1.

To adapt MovieLens-1M for the post-ranking scenario, following [13], we sort movies in ascending order based on the timestamp for each user, and split the rating sequence into segments with a size of 5. The target post-ranking list is the movies arranged in descending order based on their rating scores. Considering LLM4PR is designed the search scenario, we allocate a pseudo search query, such as *what to watch* or *movie recommendations*, for each segment. For KuaiSAR dataset, we conduct experiments on the search part of KuaiSAR, treating the items in each search session as candidates for post-ranking. To leverage the textual information, which is encrypted in this dataset, we have contacted the authors and acquired the raw data.

5.1.2 Baselines. For the information retrieval dataset, we compare LLM4PR with several state-of-the-art (SOTA) passage ranking methods. The baselines include monoBERT[32], monoT5[31], and Cohere Rerank[40]. For the search dataset, we compare LLM4PR with LTR methods, conventional post-ranking methods, and LLM-based approaches. Specifically, we choose the LTR-DNN and LTR-DCN as the LTR baselines. For conventional post-ranking methods, we

Table 2: Evaluation results (NDCG@10) on the information retrieval task. Bold indicates the best results. The values in the table are presented as percentages (%).

Method	Covid	NFCorpus	Touche	DBPedia	SciFact	Signal	News	Robust04	BEIR(Avg)
BM25	59.47	30.75	**44.22**	31.80	67.89	33.05	39.52	40.70	43.42
monoBERT(340M)	70.01	36.88	31.75	41.87	71.36	31.44	44.62	49.35	47.16
monoT5(220M)	78.34	37.38	30.82	42.42	73.40	31.67	46.83	51.72	49.01
monoT5(3B)	80.71	**38.97**	32.41	44.45	**76.57**	32.55	48.49	56.71	51.36
Cohere Rerank-v2	81.81	36.36	32.51	42.51	74.44	29.60	47.59	50.78	49.45
LLM4PR(7B)	**83.75**	38.18	36.83	**46.35**	74.77	**33.86**	**49.85**	**57.28**	**52.61**

Table 3: Evaluation results on the search post-ranking task. Bold indicates the best results. The values in the table are presented as percentages (%).

Dataset→	MovieLens-1M				KuaiSAR					
Model↓	NDCG@3	NDCG@5	MRR@3	MRR@5	NDCG@3	NDCG@5	NDCG@10	MRR@3	MRR@5	MRR@10
LTR-DNN	63.30	77.59	42.71	50.14	63.52	68.80	83.06	43.33	47.32	50.05
LTR-DCN	70.21	80.91	50.92	56.18	64.33	69.37	83.84	45.35	49.37	51.75
DLCM	71.21	81.33	52.17	57.15	64.93	69.73	84.06	47.29	50.99	53.31
Seq2Slate	73.47	82.18	55.15	59.14	65.17	70.19	84.22	47.40	51.02	53.46
SetRank	73.49	82.39	55.89	60.03	65.39	70.23	84.34	47.54	51.38	53.55
PRS	75.71	83.61	57.06	60.90	65.68	70.74	**84.96**	47.86	51.61	53.72
ChatGPT4RS	65.07	77.50	48.16	54.11	55.55	61.08	79.79	17.88	23.35	29.46
LLaRA	72.64	81.98	55.32	60.17	64.86	69.94	84.37	47.13	50.89	53.22
PTPR	71.63	81.43	57.60	61.63	65.27	70.05	84.58	47.35	50.57	53.08
LLM4PR(Ours)	**76.94**	**84.35**	**61.77**	**65.15**	**66.81**	**71.96**	84.91	**48.31**	**52.75**	**54.92**

Table 4: Ablation evaluation results on MovieLens-1M and KuaiSAR datasets. Bold indicates the best results. FA refers to the Feature Adaptation step and AT indicates the Auxiliary Task in the learning to post-rank step.

Dataset→	MovieLens-1M				KuaiSAR					
Model↓	NDCG@3	NDCG@5	MRR@3	MRR@5	NDCG@3	NDCG@5	NDCG@10	MRR@3	MRR@5	MRR@10
LLM4PR	**76.94**	**84.35**	**61.77**	**65.15**	**66.81**	**71.96**	**84.91**	**48.31**	**52.75**	**54.92**
w/o QIA	76.81	84.28	61.66	64.88	65.54	70.47	84.24	47.97	51.45	53.58
w/o FA	75.45	83.68	60.85	64.06	64.90	69.26	83.31	45.56	48.03	49.70
w/o AT	75.98	84.07	61.31	64.28	65.05	69.67	83.78	47.20	50.48	52.84

select DLCM[2], Seq2Slate[4], SetRank[34], and PRS[13] for comparisons. For LLM-based benchmark, we select ChatGPT4RS[9] and LLaRA[25] as alternatives. For ChatGPT4RS, we select the list-wise template to implement the post-ranking task. For LLaRA, we input all the candidate items into model, and the model chooses one item each time. Subsequently, the selected item is eliminated from candidates and the procedure is repeated for the remaining items until the ultimate list is acquired. Furthermore, as discussed in Section 4.2, we combine all features into a single sentence as input for the LLM and train it using post-ranking targets. This alternative is referred to as 'Pure Text Post-Ranking model' (PTPR).

5.1.3 Evaluation Metrics. For the information retrieval task, following [43], we utilize NDCG@10 to evaluate the retrieval performance. For the search post-ranking task, we use NDCG@k and MRR@k as the evaluation metrics.

5.1.4 Implementation Details. For the information retrieval dataset, following [43], BM25 is used to retrieve the top 100 candidate

passages. To simulate the post-ranking scenario, a pre-trained Sentence-BERT[39] is used as the ranking model and obtain the passage embeddings. In the post-ranking stage, we utilize LLaMA2-7B[47] as the backbone for LLM4PR. Following [43], our LLM4PR model is trained on the MS MARCO dataset[30] and evaluated on the sub-datasets of BEIR[45]. For the search dataset, the initial ranking lists and feature embeddings are produced by a DNN model[8], which serves as the ranking stage. The DNN leverages pre-trained BERT[11] to process the textual features. We utilize LLaMA2-7B[47] as the backbone for LLM4PR on the MovieLens dataset and Baichuan2-7B[51] on the KuaiSAR dataset. For both tasks, The hidden size d_h for QIA is set to 512. We employ AdamW[27] with learning rate annealing from 1e-3 to 1e-6 to optimize the trainable parameters for both the feature adaptation step and the learning to post-rank step. Furthermore, we implement a linear warm-up strategy during the initial 10% of all iterations. LoRA[18] is integrated into [q_proj,k_proj] and [W_pack] modules for LLaMA2 and Baichuan2, respectively. The parameters for LoRA, including

rank, alpha, and dropout, are configured as 32, 32, and 0.1, respectively. All experiments are conducted on machines equipped with 8× NVIDIA A100 GPUs, with a batch size set to 5 for each GPU. All experiments are repeated 10 times, and the average values are reported below.

5.2 Main Results

For the information retrieval dataset, the evaluation results are reported in Table 2. As shown in the table, the proposed LLM4PR achieves the best average results (Best performances in 5 out of 8 sub-datasets, and ranking second in the remaining 3 sub-datasets), which fully demonstrate the superiority of the proposed method. Moreover, the model is trained with the MS MARCO dataset and evaluated on the BEIR dataset, verifying its remarkable transferability and generalization capability.

For the search dataset, the evaluation results are reported in Table 3. We can see that, the proposed LLM4PR outperforms the baselines in most metrics in both datasets. For the NDCG@10 metric in the KuaiSAR dataset, the performance of LLM4PR is also close to the best results of PRS. Moreover, the ChatGPT4RS merely feeds query and item texts into LLM without specific adaptation to the post-ranking task, leading to an inferior performance, indicating that generically pre-trained LLMs have limited capabilities for post-ranking. Additionally, the proposed LLM4PR achieves better results than other fine-tuned LLM-based methods, such as LLaRA and PTPR, demonstrating the superiority of our method.

5.3 Ablation Studies

In LLM4PR, we propose the QIA component and various tasks to address the post-ranking challenge. To explore the influence of these components in LLM4PR, we conduct ablation studies on the datasets and report the results in Table 4.

QIA. QIA utilizes the attention mechanism to combine the multiple feature embeddings. We substitute it with DNN and report the results as *w/o* QIA in Table 4. For the MovieLens-1M dataset, all metrics show a slight average decrease of 0.14% in both NDCG and MRR. Since the search queries in MovieLens-1M dataset are synthetic and may convey limited information on user requirements, it is reasonable that replacing the QIA does not significantly affect the overall performance. In contrast, the KuaiSAR dataset is a real-world dataset in the search scenario, encompassing authentic search queries for diverse requests. In this context, the decline extended to an average of 1.14% for NDCG and 0.99% for MRR, which verifies the importance of QIA and usefulness of integrating feature embeddings guided by the search query.

Feature Adaptation (FA). The FA step is designed to align the user/item representation embeddings with the LLM. To evaluate the impact of this step, we exclude the FA step and train the LLM4PR directly on the post-ranking task. The results are presented as *w/o* FA in Table 4. As shown in the table, the performance of our model degrades significantly after removing the FA step. Specifically, in the MovieLens-1M dataset, the average NDCG decreases by 1.08%, and the MRR decreases by 1.00%. Similarly, the KuaiSAR dataset exhibits a notable average decrease of 2.07% in NDCG and 4.23% in MRR. Therefore, removing the feature adaptation step notably affects the post-ranking performance, underscoring the indispensableness

of FA step and confirming the significance of injecting semantic meanings into the representation embeddings through alignment with the LLM.

Auxiliary Task (AT). The AT aims to equip the model with the ability to identify the superior list order. We assess its effectiveness by removing the auxiliary task from the training stage and report the results as *w/o* AT in Table 4. Particularly, the NDCG and MRR decrease for both datasets, which indicates that the auxiliary task positively contributes to enhancing the post-ranking performance.

5.4 Further Analysis

5.4.1 Feature Embeddings with Different Qualities. The LLM4PR utilizes the feature embeddings obtained from the previous stage (*i.e.*, ranking stage). We explore how feature embeddings with different qualities influence the post-ranking performance. We extract embeddings from various stages of the training phase of the ranking model (*i.e.*, the DNN model), denoted as Poor Quality (PQ, validation ranking AUC score of 0.5), Medium Quality (MQ, validation ranking AUC score of 0.7), and High Quality (HQ, validation ranking AUC score exceeding 0.85). We utilize these embeddings to train the LLM4PR and present the results in Table 5. As shown in the table, the post-ranking performance consistently declines from high-quality to poor-quality embeddings, indicating that high-quality embeddings facilitate the learning to post-rank process for LLM4PR. We also find that even with poor-quality or medium-quality embeddings, LLM4PR is capable of delivering competitive results. In this context, the QIA component functions akin to an "embedding layer". The low-quality embeddings serve as the initialization, and during the training process, the QIA learns to map these low-quality vectors to meaningful embeddings for the LLM. Therefore, our model exhibits favorable compatibility for the input embeddings.

5.4.2 Inference Time Cost. As we mentioned in Section 4.2, directly concatenating all features into a single sentence as input for LLM would significantly increase the inference time cost. To verify this statement, we compare the inference time costs of PTPR and LLM4PR, with the results presented in Table 6. As shown in the table, the input tokens for PTPR, which simply concatenates all features into a single sentence, are 9 and 23 times greater than those for LLM4PR in MovieLens and KuaiSAR datasets, respectively. Consequently, the inference time cost for PTPR is 1.7 and 3.0 times higher than that of LLM4PR, respectively. The increase in token length primarily comes from the numerical features. For instance, one movie item in the MovieLens dataset has a feature named 'history average rating' with a value of 0.5. In LLM4PR, the rating value is converted into an embedding and serves as only one input token for the LLM. However, for PTPR, the same feature is input as plain text, 'history average rating is 0.5', which will be tokenized into 8 tokens ('_history', '_average', '_rating', '_is', '_', '0', '.', '5'). Thus, the proposed LLM4PR can efficiently processes diverse input features.
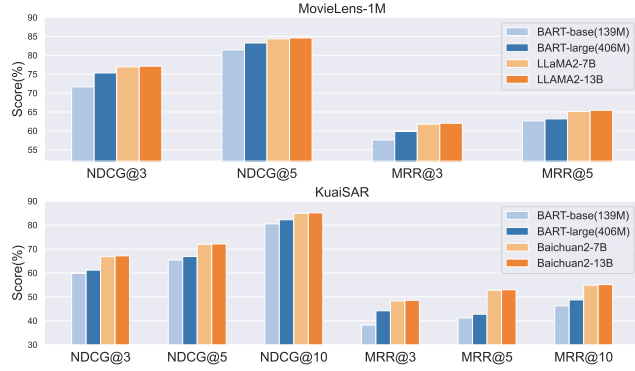
5.4.3 Backbone LLM with Different Sizes. To evaluate the impact of the backbone LLM size, we substitute the 7B LLM with different language models and provide the post-ranking performance in Figure 5. As shown in this Figure, the post-ranking performances

Table 5: Evaluation results of LLM4PR with feature embeddings of varying qualities. Bold indicates the best results. HQ, MQ, and PQ represent feature embeddings of High Quality, Medium Quality, and Poor Quality, respectively.

Dataset→	MovieLens-1M				KuaiSAR					
Model↓	NDCG@3	NDCG@5	MRR@3	MRR@5	NDCG@3	NDCG@5	NDCG@10	MRR@3	MRR@5	MRR@10
LLM4PR-HQ	**76.94**	**84.35**	**61.77**	**65.15**	**66.81**	**71.96**	**84.91**	**48.31**	**52.75**	**54.92**
LLM4PR-MQ	76.84	84.19	61.60	64.83	65.89	71.22	84.88	47.83	51.81	54.26
LLM4PR-PQ	76.43	83.96	61.23	64.53	65.27	70.05	84.58	47.35	50.57	53.08

Table 6: Inference Time Cost. The Token column indicates the input token length and the Cost column represents the inference time.

Dataset→	MovieLens-1M		KuaiSAR	
Model↓	Tokens	Cost(ms)	Tokens	Cost(ms)
PTPR	772	556	2135	1525
LLM4PR	80	318	92	501

Figure 5: Post-ranking performance comparison *w.r.t.* different sizes of backbone language models.

consistently improve with the increasing size of backbone models. This is consistent with the intuition that larger LLMs are inherently more powerful than smaller ones. However, when the model size reaches a sufficiently large scale, such as 7B, further enlarging the model size yields only marginal improvements. Therefore, to strike a balance between the training burden and overall performance, selecting a 7B version LLM as the backbone is a practical choice.

Table 7: Feature embedding alignment performance of the feature adaptation step for the MovieLens-1M dataset. Acc represents the accuracy.

Features	Acc (%)	Auc (%)
Categorical Features	100	-
Numerical Features	99.89	-
Score Predictions	48.70	64.78

5.4.4 Feature Adaptation Analysis. In this subsection, we validate the feature adaptation step and analyze its prediction results. We

Figure 6: Example of the input and output from the feature adaptation step. Yellow, blue, and green indicate the predicted categorical features (*e.g.*, gender, age, and occupation, *etc*), numerical features (*e.g.*, average rating scores, *etc*), and rating score, respectively.

utilize the template $\mathcal{T}_{fa}$ (shown in Figure 6) to generate the outputs, which include predicting categorical feature, numerical feature and rating scores. The generation results are evaluated accuracy and AUC values for the predicted features and rating scores, and the results are reported in Table 7. We observe that the LLM achieves exceptionally high accuracy scores for both categorical and numerical features, illustrating that the feature adaptation step successfully incorporates semantic meanings into the user/item representation vectors. Moreover, the LLM achieves 48.70% accuracy score and 64.78% AUC score for the rating score prediction, which is a 5-class classification task in MoiveLen-1M dataset. This demonstrates that the generated user/item representation vectors also encapsulate the user preference information, which is beneficial for learning to post-rank.

6 CONCLUSION

In this paper, we introduce LLM4PR, the first LLM-based post-ranking framework designed for search engines. We introduce a QIA component to fuse heterogeneous features of users and items, and a feature adaptation step to align the user/item embeddings with the LLM. Furthermore, we leverage both the main and auxiliary tasks to fine-tune the model for learning post-ranking. The extensive experiments on multiple datasets fully demonstrate the effectiveness and superiority of our proposed LLM4PR.

REFERENCES

[1] Mustafa Abdool, Malay Haldar, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Manaswala, Lynn Yang, Bradley C. Turnbull, Qing Zhang, and Thomas Legrand. 2020. Managing Diversity in Airbnb Search. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya

Prakash (Eds.). ACM, 2952–2960.

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 135–144.

[3] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 85–92.

[4] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Huai-hsin Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2Slate: Re-ranking and Slate Optimization with RNNs. *CoRR* abs/1810.02019. arXiv:1810.02019

[5] Luiz Henrique Bonifacio, Hugo Queiroz Abonizio, Marzieh Fadaee, and Rodrigo Frassetto Nogueira. 2022. InPars: Data Augmentation for Information Retrieval using Large Language Models. *CoRR* abs/2202.05144 (2022). arXiv:2202.05144

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[7] Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, and Jong C. Park. 2023. Discrete Prompt Optimization via Constrained Generation for Zero-shot Re-ranker. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*. Association for Computational Linguistics, 960–971.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*. ACM, 191–198.

[9] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering ChatGPT's Capabilities in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*. ACM, 1126–1132.

[10] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot Dense Retrieval From 8 Examples. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[12] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*. Association for Computational Linguistics, 320–335.

[13] Yufei Feng, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. 2021. Revisit Recommender System in the Permutation Prospective. *CoRR* abs/2102.12057 (2021). arXiv:2102.12057

[14] Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. 2021. GRN: Generative Rerank Network for Context-wise Recommendation. In *arXiv preprint arXiv:2104.00860*.

[15] Yu Gong, Ziwen Jiang, Yufei Feng, Binbin Hu, Kaiqi Zhao, Qingwen Liu, and Wenwu Ou. 2020. EdgeRec: Recommender System on Edge in Mobile Taobao. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2477–2484.

[16] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2016), 19:1–19:19.

[17] Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging Large Language Models for Sequential Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*. ACM, 1096–1102.

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[19] Guangda Huzhang, Zhen-Jia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qianying Lin, Qing Da, Anxiang Zeng, Han Yu, Yang Yu, and Zhi-Hua Zhou. 2023. AliExpress Learning-to-Rank: Maximizing Online Model Performance Without Going Online. *IEEE Trans. Knowl. Data Eng.* 35, 2 (2023), 1214–1226.

[20] Vitor Jeronymo, Luiz Henrique Bonifacio, Hugo Queiroz Abonizio, Marzieh Fadaee, Roberto de Alencar Lotufo, Jakub Zavrel, and Rodrigo Frassetto Nogueira. 2023. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *CoRR* abs/2301.01820 (2023). https://doi.org/10.48550/ARXIV.2301.01820 arXiv:2301.01820

[21] Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy A. Mann, and Danilo J. Rezende. 2019. Beyond Greedy Ranking: Slate Optimization via List-CVAE. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[22] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 7871–7880.

[23] Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4SRec: An Elegant Effective Efficient Extensible Solution of Large Language Models for Sequential Recommendation. *CoRR* abs/2312.02443 (2023). https://doi.org/10.48550/ARXIV.2312.02443 arXiv:2312.02443

[24] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, and Xiang Wang. 2023. LLaRA: Aligning Large Language Models with Sequential Recommenders. *CoRR* abs/2312.02445 (2023). https://doi.org/10.48550/ARXIV.2312.02445 arXiv:2312.02445

[25] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1785–1795.

[26] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural Re-ranking in Multi-stage Recommender Systems: A Review. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22) Survey Track*.

[27] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[28] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. *CoRR* abs/2310.08319 (2023). https://doi.org/10.48550/ARXIV.2310.08319 arXiv:2310.08319

[29] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot Listwise Document Reranking with a Large Language Model. *CoRR* abs/2305.02156 (2023). https://doi.org/10.48550/ARXIV.2305.02156 arXiv:2305.02156

[30] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *CoCo@NIPS (CEUR Workshop Proceedings, Vol. 1773)*. CEUR-WS.org.

[31] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 708–718.

[32] Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019).

[33] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023). https://doi.org/10.48550/ARXIV.2303.08774 arXiv:2303.08774

[34] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 499–508.

[35] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, and Wenwu Ou. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 3–11.

[36] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. *CoRR* abs/2306.17563 (2023). https://doi.org/10.48550/ARXIV.2306.17563 arXiv:2306.17563

[37] Junyan Qiu, Haitao Wang, Zhaolin Hong, Yiping Yang, Qiang Liu, and Xingxing Wang. 2023. ControlRec: Bridging the Semantic Gap between Language Model and Personalized Recommendation. *CoRR* abs/2311.16441 (2023). https://doi.org/10.48550/ARXIV.2311.16441 arXiv:2311.16441

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[39] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 3980–3990.

[40] Nils Reimers, Sylvie Shi, Lucas Fayoux, and Elliott Choi. 2023. Say Goodbye to Irrelevant Search Results: Cohere Rerank Is Here. https://cohere.com/blog/rerank.

[41] 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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022.* Association for Computational Linguistics, 3781–3797.

[42] Xiaowen Shi, Fan Yang, Ze Wang, Xiaoxu Wu, Muzhi Guan, Guogang Liao, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. PIER: Permutation-Level Interest-Based End-to-End Re-ranking Framework in E-commerce. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023.* ACM, 4823–4831.

[43] 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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023.* Association for Computational Linguistics, 14918–14937.

[44] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Dewei Leng, Yanan Niu, Yang Song, Xiao Zhang, and Jun Xu. 2023. KuaiSAR: A Unified Search And Recommendation Dataset. In *CIKM.* ACM, 5407–5411.

[45] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung (Eds.), Vol. 1. https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/65b9eea6e1cc6bb9f0cd2a47751a186f-Paper-round2.pdf

[46] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023). https://doi.org/10.48550/ARXIV.2302.13971 arXiv:2302.13971

[47] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/ARXIV.2307.09288 arXiv:2307.09288

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA.* 5998–6008.

[49] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving Text Embeddings with Large Language Models. *CoRR* abs/2401.00368 (2024). https://doi.org/10.48550/ARXIV.2401.00368 arXiv:2401.00368

[50] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A Survey on Large Language Models for Recommendation. *CoRR* abs/2305.19860 (2023). https://doi.org/10.48550/ARXIV.2305.19860 arXiv:2305.19860

[51] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open Large-scale Language Models. *CoRR* abs/2309.10305 (2023). https://doi.org/10.48550/ARXIV.2309.10305 arXiv:2309.10305

[52] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *CoRR* abs/2303.18223 (2023). https://doi.org/10.48550/ARXIV.2303.18223 arXiv:2303.18223

[53] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large Language Models for Information Retrieval: A Survey. *CoRR* abs/2308.07107 (2023). https://doi.org/10.48550/ARXIV.2308.07107 arXiv:2308.07107

[54] 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. *CoRR* abs/2310.14122 (2023). https://doi.org/10.48550/ARXIV.2310.14122 arXiv:2310.14122

[55] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.* ijcai.org, 3725–3731.