information

MDPI

Article

MultiLTR: Text Ranking with a Multi-Stage Learning-to-Rank Approach

Hua Yang [1,2,*] **and Teresa Gonçalves** [2,3]

[1] School of Artificial Intelligence, Zhongyuan University of Technology, Zhengzhou 450007, China
[2] VISTA Lab, Algoritmi Center, University of Évora, 7000-671 Évora, Portugal; tcg@uevora.pt
[3] Department of Computer Science, University of Évora, 7000-671 Évora, Portugal
* Correspondence: huayang@zut.edu.cn

Abstract: The division of retrieval into multiple stages has evolved to balance efficiency and effectiveness among various ranking models. Faster but less accurate models are used to retrieve results from the entire corpus. Slower yet more precise models refine the ranking within the top candidate list. This study proposes a multi-stage learning-to-rank (MultiLTR) method. MultiLTR applies learning-to-rank techniques across multiple stages. It incorporates text from different fields such as titles, body content, and abstracts to produce a more comprehensive and accurate ranking. MultiLTR iteratively refines ranking accuracy through sequential processing phases. It dynamically selects top-performing rankers from a diverse candidate pool at each stage. Experiments were carried out on benchmark datasets, MQ2007 and MQ2008, using three categories of learning-to-rank algorithms. The results demonstrate that MultiLTR outperforms state-of-the-art ranking approaches, particularly in field-based ranking tasks. This study improves ranking accuracy and offers new insights into enhancing multi-stage ranking strategies.

Keywords: learning-to-rank; multi-stage ranking; field-based; text re-ranking

check for updates

Academic Editor: Shmuel Tomi Klein

Received: 13 February 2025
Revised: 4 April 2025
Accepted: 11 April 2025
Published: 13 April 2025

Citation: Yang, H.; Gonçalves, T. MultiLTR: Text Ranking with a Multi-Stage Learning-to-Rank Approach. *Information* **2025**, *16*, 308. https://doi.org/10.3390/info16040308

1. Introduction

The effectiveness of learning-to-rank (LTR) techniques within a multi-stage retrieval framework remains underexplored. This paper introduces a multi-stage learning-to-rank (MultiLTR) approach. MultiLTR integrates LTR techniques into a multi-stage retrieval process. This approach leverages field-based information and addresses ranking challenges in text re-ranking tasks. While existing methods demonstrate the value of field-based feature aggregation [1], their static weighted-sum approach to combining field signals exhibits some limitations, such as rigid interaction modeling that cannot adapt to query-dependent feature importance; error propagation from separate field-level optimization to final ranking; and context blindness in handling multi-stage retrieval dependencies.

The methodology presented in this article builds on previous research introduced in [1]. In that work, we proposed the field learning-to-rank (fLTR) framework and trained rankers using field-based features; results from various text fields were aggregated. In the final phase of the fLTR method, the ranking problem was reformulated as an aggregation task, and the aggregation techniques significantly impacted the overall training outcomes. In contrast, this article defines the task as an ordinal LTR problem and incorporates multistage retrieval principles. We refer to this approach as the MultiLTR method.

In summary, this paper makes the following contributions:

- We propose a multi-stage ranking approach. The initial stage constructs a set of retrieval models and utilizes features derived from a single text field. The subsequent re-ranking stages build local and global rankers using learning-to-rank techniques.
- We introduce a mid-evaluation strategy to select the best-performing model. This strategy evaluates a set of candidate models before generating the final ranking list. The model achieving the highest performance acts as the final ranking model.
- We conduct empirical experiments on two publicly available benchmark datasets and use three different LTR algorithms. We evaluate the results using widely adopted assessment metrics.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 describes the proposed approach, Section 4 presents the experiments and results, Section 5 discusses the key findings. Finally, Section 6 concludes this paper and outlines potential directions for future research.

2. Literature Review

This section reviews the multi-stage ranking approach, focusing on re-ranking techniques. Additionally, it surveys neural ranking models and provides an overview of the LTR techniques.

2.1. Multi-Stage Ranking Architectures

A multi-stage ranking architecture typically consists of an initial ranking phase and one or more re-ranking stages [2–5]. Traditionally, the initial ranking uses classical term-based retrieval models. The retrieved results are subsequently passed to the re-ranking stage [6,7].

Recent studies have explored various approaches to multi-stage ranking tasks. In [8], a hybrid re-ranking framework was introduced. It uses BM25 as the initial ranker and a cross-attention neural model as the re-ranker. They are evaluated on two datasets, namely, the MS MARCO passage retrieval dataset and the BEIR corpus for zero-shot retrieval. The results demonstrate its effectiveness compared to baseline models and other approaches.

Wang et al. [9] proposed a multi-stage information search architecture to generate semantic embeddings. It employs a post-fusion strategy to project items retrieved from different stages into a common space. It utilizes multi-grained learning objectives to preserve multi-level similarity. The authors conducted experiments on a billion-scale corpus from the JD E-commerce website. They compared the results with those from DSSM [10], DPSR [11], and RSR [12], demonstrating its effectiveness.

Hai et al. [13] developed a lightweight learning technique to train a first-stage ranker. The paper used T5Mono for the second-stage ranking. They trained the two-stage model on the CANARD dataset and evaluated the results on the TREC CAsT 2020–2021 datasets. This approach is effective in conversational search tasks. In [14], Yang et al. proposed a mixed-initiative query reformulation module within a two-stage retrieval pipeline. BM25 serves as the first-stage ranker. MonoT5 and DuoT5 are used as re-rankers. Experiments were carried out on the TREC CAsT 2021–2022 datasets. The results showed that this method achieved results comparable to two popular reformulators, namely, CANARD-T5 and the historical query reformulator.

Finally, the LCE (localized contrastive estimation) method proposed by Gao et al. [15] re-ranks documents in a deep two-stage model. It aims to enhance initial rankings and experiments on the MS MARCO dataset. The results demonstrate that LCE outperforms the Vanilla method. The method achieves even higher scores when combined with HDCT as the initial retriever. The findings suggest that LCE has strong potential for integrating learning techniques with existing language models for re-ranking tasks.

2.2. Neural Ranking Models

In recent years, neural ranking models have been increasingly used for document re-ranking [6,16–18]. In text re-ranking tasks, existing research demonstrates that pre-trained neural network models can effectively learn text representations and text–pair interactions for various applications. The scenario includes document ranking and re-ranking, document summarization, query suggestion, and query parsing [7,19–21].

Neural ranking models can be categorized in two ways. On the one hand, they can be classified based on whether they utilize transformer-based BERT models. And this classification results in two groups, namely, pre-BERT- and BERT-based neural ranking models [21]. BERT is one of the most significant transformer-based pre-trained models [22]. In text re-ranking, BERT and its variants have been widely adopted to estimate the relevance between queries and documents [7,21]. Instead of relying on a single input representation, dynamic memory networks [23] have been used to model sentence-level representations and aggregate BERT output. They carry out the experiments on three passage-ranking datasets, namely, ANTIQUE, InsuranceQA, and TREC-DL 2019. The results demonstrate that incorporating sentence-level representations improves re-ranking performance over fine-tuned baseline BERT models. Additionally, researchers found that training only the DMN layer could achieve comparable performance to the full model while significantly enhancing training efficiency [23]. Two BERT-based ranking variants, monoBERT, and duoBERT, are proposed to address ranking problems [6]. monoBERT employs a point-wise classification approach, while duoBERT follows a pairwise classification strategy. These models are incorporated into a multi-stage ranking framework and evaluated on TREC CAR and MS MARCO datasets. This method achieves comparable results when compared to the state-of-the-art techniques.

On the other hand, neural ranking models can also be classified based on neural network techniques' applications: during the representation or interaction phase. This results in three categories, namely, representation-based models, interaction-based models, and hybrid models [7,21]. Representation-based models independently learn dense vector representations for queries and documents. These representations are then compared using similarity measures such as cosine similarity or inner products to determine relevance [7,21]. In contrast, interaction-based models compare the representations of individual query terms with document terms. The interaction-based models construct a similarity matrix that captures word interactions. This matrix further processes to derive a relevance score [7,21]. Hybrid models incorporate elements from both representation-based and interaction-based approaches. A well-known example is Duet [24]. It integrates a representation-learning component with an interaction-based component responsible for detecting exact term matches.

Furthermore, Fan et al. [7] classified neural ranking models into two broader generations. The first generation, emerging around 2010, focused on word embedding techniques for learning word representations. The second generation leveraged transformer-based methods to learn text representations and interactions. Consequently, research that uses classical word embedding methods—such as Word2Vec [25] and GloVe [26]—falls into the pre-BERT category [7].

2.3. Learning-to-Rank Techniques

Semantic retrieval models and neural ranking models demonstrate their effectiveness across various ranking tasks [27]. A key distinction between these models and LTR methods is that they typically do not rely on additional manually crafted features to estimate document relevance to a query [28]. However, their relative performance compared to feature-based LTR methods remains an open question [29]. Additionally, feature-based

LTR methods continue to attract research interest, particularly in areas such as efficiency, diversification, and permutation-invariant models [29]. Notably, recent efforts research on bridging the performance gap between neural and ensemble-based models [30].

Learning-to-rank techniques and their corresponding models play a crucial role in re-ranking within multi-stage ranking systems. These techniques are typically classified into three categories, namely, pointwise, pairwise, and listwise approaches. Classical LTR models include the following: MART [31], RankNet [32], RankBoost [32], AdaRank [33], Coordinate ascent [34], LambdaMART [35], ListNet [36], and random forest [37]. In recent years, several new LTR models have emerged, including the following: DirectRanker [38], PairRank [39], DeepPLTR [40], DLCM [41], DeepQRank [42], SetRank [43], PiRank [44], PoolRank [45], and ListMAP [46].

A study by Chen et al. [47] explored the use of a neural-based multi-layer perceptron model in combination with the BM25 retrieval model for document ranking. Their experiments on LambdaMART demonstrated that supervised learning-based re-rankers could enhance ranking performance. Han et al. [48] introduced a machine learning algorithm for document re-ranking. A learning-to-rank model was applied after encoding queries and documents using BERT. Their approach led to a 4.3% improvement over their previous best-performing re-ranking method. Additionally, Awan et al. [49] proposed a deep learning approach to address named entity normalization as a pairwise learning-to-rank problem. Their method uses BM25 to generate candidate entities and then applies BERT representations for re-ranking. Experimental results across various species entity types have demonstrated the superiority of their method over existing state-of-the-art techniques. One study highlighted the limitations of the traditional probability ranking principle (PRP) in multi-stage information retrieval systems. This study proposes a generalized probability ranking principle (GPRP). It integrates both stage-specific selection bias and user interests. It implements GPRP through a full-stage LTR framework [4].

3. Methodology

This section introduces the proposed MultiLTR approach. It presents the overall architecture and explains the MultiLTR framework in detail.

3.1. Architecture of the Proposed MultiLTR Approach

Given a query q and a set of documents $t_1, t_2, \ldots, t_n$ from a document collection T, the objective of a ranking task is to order these documents based on their relevance to the query. Documents with higher relevance scores appear at the top of the ranking list [8]. In a multi-stage ranking system, the initial retrieval stage identifies a set of candidate documents, subsequent re-ranking phases refine their order, and the final stage returns the ranked list to the user. Our primary goal is to enhance ranking performance through effective re-ranking techniques.

The initial ranking layer employs classical text retrieval models to rank documents. For example, if a user searches for “effective treatments for diabetes” in a medical database, the initial retrieval might return all documents containing the keywords “diabetes” and “treatments”. The re-ranking phase would then apply a model to refine the ranking. For example, a TF-IDF-based algorithm can be used to prioritize documents based on keyword frequency. However, this approach does not consider the context of the information. This approach potentially ranks documents that mention “diabetes” and “treatments”. However, it misses detailed or reliable information about effective treatments.

In contrast, our approach improves this by multiple re-ranking layers. The initial layer generates preliminary results using BM25. And the local re-ranking layer applies supervised LTR techniques. Each re-ranker focuses on a specific field, such as the title or abstract. The global re-ranking layer integrates information from different fields. It mitigates correlation issues by delaying feature combinations. Additionally, the mid-evaluation layer assesses the ranker performance using standard evaluation metrics. We select the best-performing model to construct the final re-ranked list. This ensures that the most relevant and reliable documents are ranked at the top. This can provide users with high-quality, contextually relevant information about effective treatments for diabetes.

Figure 1 illustrates the hierarchical structure of the MultiLTR and the flow of information from input features to the final re-ranked output. The system begins with an initial ranking stage. A series of re-ranking layers then refine and filter the results from the initial retrieval. At each stage, we process and refine a ranked list of candidates from the previous phase before passing to the next stage. Finally, we select the best-performing model for the final process and return the ultimate ranking list to the user.

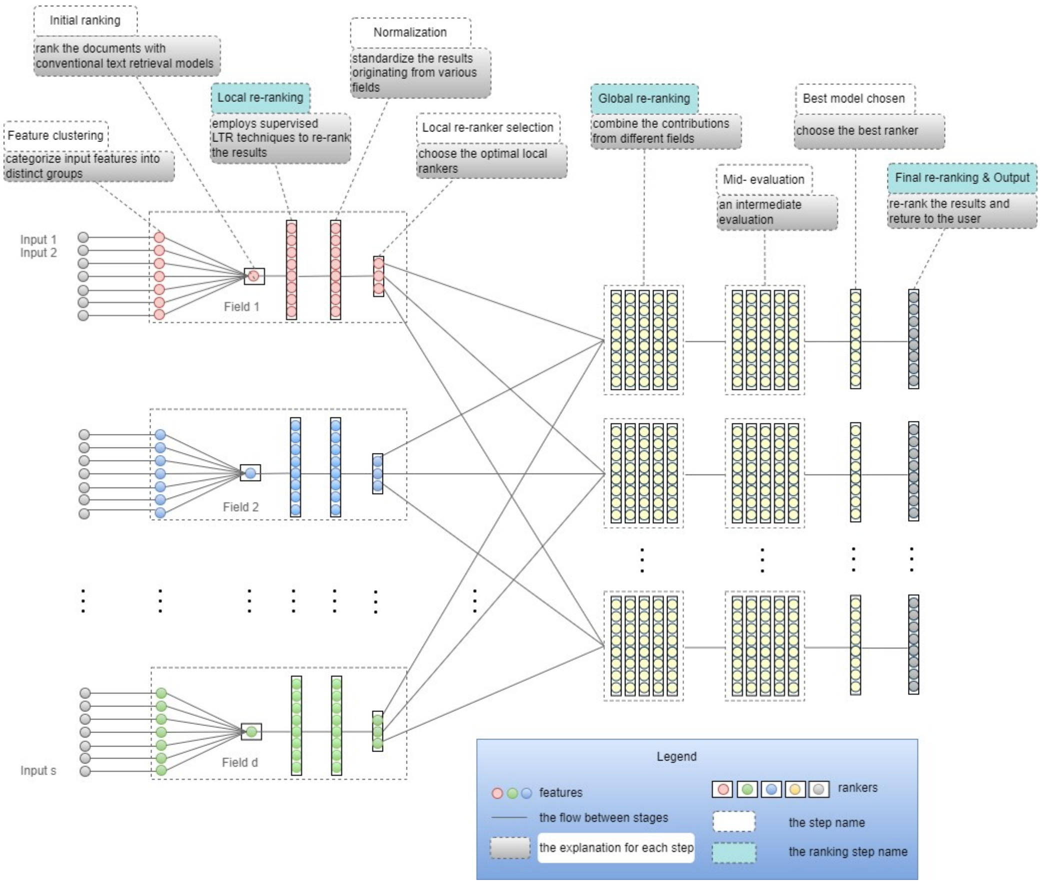

Initial ranking
rank the documents with conventional text retrieval models
Normalization
standardize the results originating from various fields
Local re-ranking
employs supervised LTR techniques to re-rank the results
Feature clustering
categorize input features into distinct groups
Local re-ranker selection
choose the optimal local rankers
Global re-ranking
combine the contributions from different fields
Best model chosen
choose the best ranker
Mid- evaluation
an intermediate evaluation
Final re-ranking & Output
re-rank the results and reture to the user
Input 1
Input 2
Field 1
Field 2
Field d
Input s
Legend
features
rankers
the flow between stages
the step name
the explanation for each step
the ranking step name

Figure 1. MultiLTR architecture overview. Key stages: Input → Feature clustering → Initial ranking → Local re-ranking (noise filtering) → Normalization → Dynamic model selection → Global re-ranking → Mid-evaluation → Final re-ranking → Output.

To clarify the interactions between different ranking layers, we present the inputs and outputs of each layer in Figure 2. Refining ranking performance is crucial in each stage, ensuring an efficient and effective retrieval process. We define the key notations used in Table 1. Algorithm 1 provides the detailed pseudocode for the proposed MultiLTR approach.

Algorithm 1: Pseudocode for the MultiLTR Approach.

```
Input: $\{f_1, f_2, \ldots, f_s$, where $f$ = E($q$,$t$)$\}$
for $i = 1, 2, \ldots, d$ do
    /* Feature_clustering: Cluster features based on relevance to
       specific fields                                              */
    Extract query-dependent or independent features from the given data
      collection $T$;
    Select features from the field $i$;
    Filter features according to the set rules and maintain an equal and consistent
      number of features to $k$ across all fields;
    /* Rank documents employing conventional text retrieval models  */
    $r_1^{(d)}$ = Initial ranking using features from field $d$ $\{f_1^{(d)}, f_2^{(d)}, \ldots, f_k^{(d)}\}$;
    /* Local re-ranking: Employ supervised LTR techniques to re-rank
       the results from the initial ranking layer                   */
    select a set of LTR algorithms from previous research work;
    for each LTR algorithm, build an LTR ranker using the features from field $d$
      $\{f_1^{(d)}, f_2^{(d)}, \ldots, f_k^{(d)}\}$;
    $r_2^{(d)}$ = Local re-ranking;
    /* Standardize data by rescaling values to a consistent scale or
       distribution                                                 */
    $r_3^{(d)}$ = Normalization ($r_2^{(d)}$);
    /* Choose the optimal local rankers                             */
    for $j = pointwise, pairwise, listwised$ do
        $r_4^{(d-j)}$ = Local re-ranker selection;
    end
end
for $d, j = pointwise, pairwise, listwised$ do
    for $n = 1, 2, \ldots, 8$ do
        for $m$ = NDCG@5, 10, 15, 20, 30 do
            /* Combine the contributions from different fields      */
            $r_5^{(j-n-m)}$ = Global re-ranking;
            /* Select the best-performing model to build the final
               re-ranking list                                      */
            Mid-evaluation;
            Evaluate the LTR ranker for each field $d$;
        end
    end
end
/* Choose the top-performing model and re-rank the results          */
for $d, j = pointwise, pairwise, listwised$ do
    $r_6^{j-n-best}$ = Best of ($r_5^{(j-n-best)}$) ;
    H = ranking results using model $r_6^{j-n-best}$;
end
Output: final ranking results H
```

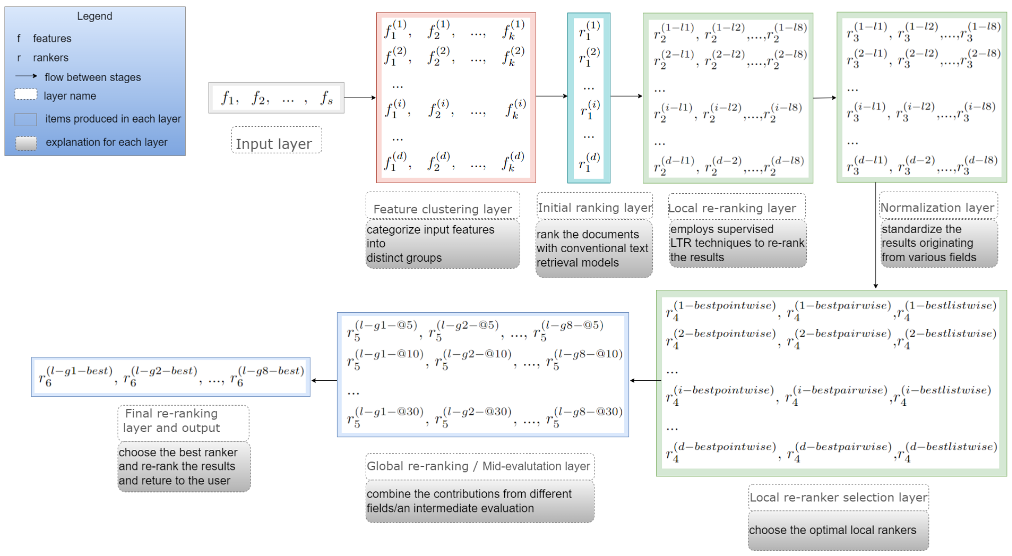

Legend
f features
r rankers
flow between stages
layer name
items produced in each layer
explanation for each layer
Input layer
Feature clustering layer
categorize input features into distinct groups
Initial ranking layer
rank the documents with conventional text retrieval models
Local re-ranking layer
employs supervised LTR techniques to re-rank the results
Normalization layer
standardize the results originating from various fields
Local re-ranker selection layer
choose the optimal local rankers
Global re-ranking / Mid-evulation layer
combine the contributions from different fields/an intermediate evaluation
Final re-ranking layer and output
choose the best ranker and re-rank the results and reture to the user

Figure 2. Data flow in the MultiLTR approach. Key stages: Input → Feature Clustering → Initial Ranking → Local Re-ranking → Normalization → Global Re-ranking → Mid-Evaluation → Final Re-ranking → Output.

Table 1. Key notations for the MultiLTR approach.

Notation	Meaning
T	data collection
t	a document in D
q	a query in D
f	original and unclustered features provided or extracted from a data collection T
s	the total number of features extracted from a data collection T
d	the number of fields where features are extracted from
k	the number of features included in a field
$f_j^{(i)}$	the feature j from the field i
r_m	the ranker of the mth layer
H	final result

3.2. Layers in the Proposed MultiLTR Approach

This section provides a detailed explanation of the components illustrated in Figure 2, including the input layer, feature clustering layer, initial ranking layer, local re-ranking layer, normalization layer, global re-ranking layer, mid-evaluation layer, final re-ranking layer, and output layer.

3.2.1. Input Layer

The dataset undergoes pre-processing through feature engineering techniques to ensure compatibility with the MultiLTR approach. The dataset can also be transformed using pre-trained language models like BERT to generate embeddings. These refined features are then used as input for the ranking architecture.

Formally, we extract a feature f and represent it as follows:

$$f = E(t, q) \tag{1}$$

where E represents a feature extractor, q denotes the given queries, and t corresponds to the associated documents. As a result, the input layer receives an array of features $\{f_1, f_2, \ldots, f_s\}$. These features serve as the foundation for subsequent ranking processes.

3.2.2. Feature Clustering Layer

When training ranking models, highly correlated features can negatively impact performance. Empirical studies [1] show that features originating from different domains or fields often exhibit significant correlations. Traditionally, ranking models are trained using a combined set of extracted features without considering their relationships. However, the model may inadvertently rely on redundant information if highly correlated features are not handled properly. And this can lead to suboptimal performance [50].

In machine learning, blindly aggregating strongly correlated attributes can degrade model effectiveness. Incorporating diverse and discriminative features with minimal inter-correlation can help to improve performance. Previous research demonstrates that using field-based features outperforms naive feature aggregation [1]. Following this principle, the proposed MultiLTR approach clusters feature into distinct groups. This ensures that each ranker is trained using field-specific data. The feature clustering layer organizes input features into groups based on their respective fields, such as the title, abstract, URL, or body.

As illustrated in Figure 2, the approach assumes we have s features extracted from d fields. The feature clustering layer classifies these features into d distinct groups. To maintain balanced contributions from each field, we retain an equal number of features for every field. Thus, each group consists of k field-specific features. They are represented as $f_1^{(i)}, f_2^{(i)}, \ldots, f_k^{(i)}$, and i indicates the field from which the features are extracted.

3.2.3. Initial Ranking Layer

The initial ranking layer is responsible for ranking documents using traditional models. Its primary function is to generate preliminary ranking results from the dataset, which are then passed to subsequent stages for further refinement.

Each initial ranker, denoted as $r_1^{(i)}$, is trained exclusively using features $\{f_1^{(i)}, f_2^{(i)}, \ldots, f_k^{(i)}\}$ extracted from a specific field i. As a result, this process produces a collection of initial rankers, represented as $\{r_1^{(1)}, r_1^{(2)}, \ldots, r_1^{(d)}\}$, each based on field-specific characteristics.

3.2.4. Local Re-Ranking Layer

After the initial ranking layer, the local re-ranking layer applies supervised LTR techniques to refine the ranking results. We term this layer as the "local" re-ranking layer because each ranker focuses exclusively on information from a single field in this phase. This ensures a more field-specific ranking process.

As outlined in [51], LTR techniques rely on training data that consists of the following: a set of queries $\{q_1, q_2, \ldots, q_m\}$; a corresponding collection of documents $\{x_{1(1)}, x_{2(1)}, \ldots, x_{n(1)}\}$; and their associated relevance judgments $\{y_{(1)}, y_{(2)}, \ldots, y_{(m)}\}$. These relevance judgments can be formulated in the following ways: (1) binary or graded relevance, indicating whether a document is relevant to a query or the degree of relevance; (2) pairwise preference, specifying the relative relevance between two documents for a given query; and (3) listwise ranking, determining the optimal order of documents for a query. Consequently, LTR algorithms are categorized into three types: pointwise, pairwise, and listwise approaches. When we input a test query q into the trained LTR re-ranker, the relevant documents $\{x_1, x_2, \ldots, x_n\}$ are re-ranked, producing a refined ranking list $h(x)$ [51].

In the proposed MultiLTR framework, a group of LTR rankers is constructed. The training dataset for these rankers is derived from the results of the initial ranking layer. This ensures that only candidates are retrieved for further ranking refinements.

We apply eight local LTR algorithms for each field and generate eight distinct local re-rankers per field. As illustrated in Figure 2, this process results in a collection of local re-rankers $\{r_2^{(i-l1)}, r_2^{(i-l2)}, \ldots, r_2^{(i-l8)}\}$ for each field i. This ensures a structured and comprehensive ranking refinement process.

3.2.5. Normalization Layer

Each model is trained exclusively on field-specific features in previous layers. The characteristics of each field heavily influence the re-ranking outcomes. And this often leads to significant variations in scale. Features with larger value ranges tend to dominate the results. Standardizing the data through normalization ensures each field contributes equally to the final ranking.

Normalization mitigates the influence of varying feature magnitudes. Normalization plays a crucial role in standardizing results before passing to the global re-ranking layer. Specifically, we apply two normalization methods, namely, min–max normalization and Z-score normalization [52].

Min–max normalization. Min–max normalization typically adjusts numerical data to fit within a predefined range between 0 and 1. It is computed using the following formula:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{2}$$

where X is the original value, X_{norm} is the normalized value, and X_{min} and X_{max} are the minimum and maximum values in the dataset or feature.

Z-score normalization. Z-score normalization is a technique used to scale numerical data and has a mean of 0 and a standard deviation of 1. It is computed using the following formula:

$$Z = \frac{X - \mu}{\sigma} \tag{3}$$

where X is the original data value, Z is the standardized value, and μ and σ represent the mean and the standard deviation of the dataset or feature, respectively.

Min–max normalization ensures uniform scaling across all features but may struggle to handle outliers effectively. In contrast, Z-score normalization is well-suited for managing outliers but does not produce data with a consistent scale. To address these limitations, the MultiLTR method integrates the two normalization techniques to balance uniform scaling and robustness to outliers. As illustrated in Figure 2, a set of normalized results $\{r_3^{(i-l1)}, r_3^{(i-l2)}, \ldots, r_3^{(i-l8)}\}$ is generated for each field i,

3.2.6. Local Re-Ranker Selection Layer

This layer aims to select the most effective local rankers. We use normalized discounted cumulative gain (NDCG) [53] to evaluate the normalized outputs. NDCG, a widely recognized ranking metric, was applied with cut-off values set at 5, 10, 15, 20, and 30.

Each field contains eight local LTR algorithms, resulting in eight corresponding local re-rankers. We chose the top-performing rankers to advance to the global re-ranking layer and discarded the remaining ones. The selection process ensures coverage across all three categories of LTR algorithms. For each category, the best-performing LTR algorithm is identified. And each field ultimately retains three top-performing local re-rankers. As illustrated in Figure 2, this layer selects a set of rankers $\{r_4^{(i-bestpointwise)}, r_4^{(i-bestpairwise)}, \ldots, r_4^{(i-bestlistwise)}\}$ for each field i.

3.2.7. Global Re-Ranking Layer

The preceding layers process the information independently for each field without considering their combined impact. Combining all features from different fields at the beginning of model training could introduce correlation issues and affect ranking performance. This layer is a delayed fusion mechanism and integrates contributions from different fields. By deferring the combination, this approach mitigates such risks and allows a more effective integration of field-specific insights.

Traditional fusion techniques are often used to merge ranking results from multiple sources. The MultiLTR approach introduces an additional re-ranking process to enhance ranking quality further. At this stage, the same LTR techniques applied in the local re-ranking layer are utilized to construct global rankers. The results from the previous layer are treated as features within the LTR algorithms. This enables the model to learn and refine the final ranking.

As illustrated in Figure 2, a set of global re-rankers $\{r_5^{(l-g1)}, r_5^{(l-g2)}, \ldots, r_5^{(l-g8)}\}$ is generated. Each ranker is trained using evaluation metrics at multiple cut-off points (5, 10, 15, 20, and 30).

3.2.8. Mid-Evaluation Layer

In the preceding layer, each model is trained using a specific evaluation metric at different cut-off values. This layer acts as an intermediate evaluation stage. We assess the trained rankers using widely accepted ranking metrics. The best-performing model is identified and then used to construct the final re-ranking list. We employ the normalized discounted cumulative gain (NDCG) metric with various cut-off values (5, 10, 15, 20, and 30).

To maintain consistency, we apply the same evaluation metric for both training and assessment. For instance, if a model is trained using NDCG@5, its evaluation is also conducted using NDCG@5. We select the model that achieves the highest improvement over the baseline as the best-performing model, denoted as r_{best}. We then use this model to re-rank the documents and generate the final ranked list.

3.2.9. Final Re-Ranking Layer and Output

The top-performing models $\{r_5^{(l-g1-best)}, r_5^{(l-g2-best)}, \ldots, r_5^{(l-g8-best)}\}$ are selected and utilized for re-ranking the final results. The output layer then delivers the optimized ranked list to the system users. This ensures that the most relevant documents are presented at the top.

4. Experiments and Results

This section first presents the datasets used in the experiments. Then, we discuss the evaluation metrics. Next, we provide a detailed description of the constructed rankers and conclude the analysis of the results.

4.1. Datasets

The MultiLTR technique implements and evaluates using benchmark datasets from Microsoft LETOR 4.0 [54]. Specifically, the MQ2007 and MQ2008 datasets are derived from the TREC 2007–2008 Million Query Track, which utilizes the Gov2 web page collection, containing approximately 25 million pages. MQ2007 consists of around 1700 queries with labeled documents, and MQ2008 includes approximately 800 queries. Relevance scores range from 0 to 2, where 2 signifies high relevance and 0 indicates low relevance.

Table 2 presents a summary of the key statistics for both datasets.

Table 2. Statistics for the MQ2007 and MQ2008 datasets.

Dataset	Selected Features	Queries	Labeled Query–Document Pairs
MQ2007	40	1700	69,623
MQ2008	40	800	15,211

Each dataset includes a predefined set of 46 standard features [54]. We exclude certain features to better align with the objectives of our research experiments. We group the features into six categories based on the field from which they originate: *anchor*, *url*, *title*, *body*, *whole document*, and *other place*. The *other place* category is omitted since it lacks field-specific information. Consequently, features such as *number of child pages*, *number of inlinks*, *number of outlinks*, and *PageRank* are removed. To maintain an equal and consistent number of features across all fields, we also exclude the following features: *length of URL* and *number of slashes in URL*.

The final dataset consists of five fields, each containing eight different types of features. The experimental features, totaling 40, are summarized in Table 3.

Table 3. Selected and discarded fields and features. ○: selected feature, ⊗: rejected features, −: not provided in the dataset.

Feature	**Field**					
	Body	**Anchor**	**Title**	**URL**	**Wholedoc**	**Other Place**
TF	○	○	○	○	○	⊗
IDF	○	○	○	○	○	⊗
TF × IDF	○	○	○	○	○	⊗
DL	○	○	○	○	○	⊗
BM25	○	○	○	○	○	⊗
LMIR.ABS	○	○	○	○	○	⊗
LMIR.DIR	○	○	○	○	○	⊗
LMIR.JM	○	○	○	○	○	⊗
Length of URL	−	−	−	⊗	−	−
Number of slash in URL	−	−	−	⊗	−	−
Number of child page	−	−	−	−	−	⊗
Number of inlinks	−	−	−	−	−	⊗
Number of outlinks	−	−	−	−	−	⊗
PageRank	−	−	−	−	−	⊗

The datasets are pre-split into five folds. Each fold consists of three subsets, namely, training, validation, and test sets. The data distribution typically follows a ratio of 60% for training, 20% for validation, and 20% for testing. This ensures a balanced and effective evaluation of the models.

4.2. Evaluation Measures

We evaluate the effectiveness of the proposed method using two widely adopted assessment metrics, namely, mean average precision (MAP) [55] and NDCG [53]. These metrics provide a comprehensive measure of ranking quality and relevance.

Mean average precision. In a text retrieval system, the average precision (AP) for a single query is computed as the mean precision obtained after retrieving each relevant document from a ranked list of top documents [56]. The mean average precision (MAP) is derived as the arithmetic mean of the AP values across all query topics. MAP has strong discriminative power and stability [57].

Formally, MAP is defined as follows:

$$\mathrm{MAP} = \sum_{q=1}^{Q} \frac{\text{Average Precision}}{Q} \tag{4}$$

where each query, q, is part of Q, the total number of queries [58].

Normalized discounted cumulative gain. Another widely used metric for evaluating ranking performance is NDCG [57]. It penalizes highly relevant documents that appear lower in the ranking. Conceptually, NDCG is defined as follows:

$$\mathrm{NDCG@n} = \frac{\mathrm{DCG_n}}{\mathrm{IDCG_n}} \tag{5}$$

where discounted cumulative gain (DCG) represents the accumulated gain from ranking results, with higher-ranked documents receiving greater weight, and ideal discounted cumulative gain (IDCG) refers to the document ordering based on their actual relevance (the most relevant documents appear first). The NDCG value at a specific rank position n is denoted as NDCG@n.

4.3. Built Rankers

The ranking models are constructed using the RankLib tool (https://sourceforge.net/p/lemur/wiki/RankLib/, accessed on 12 January 2024). A diverse set of LTR algorithms is employed. They include two pointwise algorithms (MART and random forest), three pairwise algorithms (RankBoost, RankNet, and LambdaMART), and three listwise algorithms (ListNet, coordinate ascent, and AdaRank). For ease of reference, we abbreviate the algorithms as follows: MART (MR), random forest (RF), RankBoost (RB), RankNet (RN), LambdaMART (LM), ListNet (LN), coordinate ascent (CA), and AdaRank (AR).

The experiments adhered to the LETOR benchmark dataset configuration [54], utilizing five-fold cross-validation. Hyperparameter tuning was conducted using the validation dataset. The final performance was evaluated based on average scores from the test dataset at cut-offs (5, 10, 15, 20, and 30) for the NDCG and MAP metrics.

Following the framework illustrated in Figures 1 and 2, the experimental process begins with the construction of primary rankers in the initial ranking layer. Each ranker is trained exclusively on features from a specific field. The local re-ranking layer then applies LTR algorithms to refine the rankings of the retrieved candidates. Next, the normalization layer standardizes the ranking scores across different fields to ensure fair comparison and combination. The intermediate evaluation results, obtained after normalization, are reported in Tables 4 and 5.

Table 4. Normalization NDCG@k performance on MQ2007 after normalization. Models are assessed using NDCG@k at cutoffs of 5, 10, 15, 20, and 30.

Alg.	**Features**	**@5**	**@10**	**@15**	**@20**	**@30**	**Alg.**	**Features**	**@5**	**@10**	**@15**	**@20**	**@30**
MR	baseline	0.4150	0.4429	0.4721	0.5030	0.5604	RN	baseline	0.3951	0.4243	0.4560	0.4870	0.5457
	body	0.3958	0.4243	0.4551	0.4856	0.5436		body	0.3416	0.3734	0.4053	0.4382	0.5030
	anchor	0.3477	0.3764	0.4109	0.4415	0.5064		anchor	0.3457	0.3777	0.4122	0.4433	0.5094
	title	0.4073	0.4345	0.4625	0.4935	0.5533		title	0.4054	0.4344	0.4664	0.4968	0.5545
	url	0.4055	0.4329	0.4639	0.4931	0.5500		url	0.3988	0.4298	0.4630	0.4934	0.5504
	wholedoc	0.3977	0.4286	0.4576	0.4890	0.5495		wholedoc	0.3618	0.3977	0.4321	0.4639	0.5254
RB	baseline	0.4046	0.4333	0.4665	0.4968	0.5527	AR	baseline	0.4013	0.4300	0.4601	0.4909	0.5480
	body	0.3682	0.3991	0.4299	0.4608	0.5215		body	0.3439	0.3717	0.4052	0.4386	0.5044
	anchor	0.3454	0.3823	0.4145	0.4476	0.5111		anchor	0.3308	0.3640	0.3958	0.4268	0.4958
	title	0.4048	0.4353	0.4680	0.4980	0.5539		title	0.3023	0.3353	0.3672	0.4004	0.4657
	url	0.4018	0.4327	0.4667	0.4963	0.5519		url	0.3079	0.3475	0.3865	0.4213	0.4893
	wholedoc	0.3405	0.3796	0.4153	0.4485	0.5122		wholedoc	0.3094	0.3451	0.3806	0.4157	0.4837
CA	baseline	0.4087	0.4386	0.4704	0.4997	0.5560	LM	baseline	0.4197	0.4478	0.4781	0.5084	0.5642
	body	0.3656	0.3922	0.4222	0.4571	0.5194		body	0.3919	0.4210	0.4546	0.4863	0.5440
	anchor	0.3444	0.3798	0.4145	0.4462	0.5105		anchor	0.3468	0.3783	0.4131	0.4465	0.5095
	title	0.4071	0.4345	0.4669	0.4957	0.5531		title	0.4062	0.4364	0.4655	0.4951	0.5530
	url	0.4059	0.4371	0.4688	0.4985	0.5545		url	0.4046	0.4352	0.4667	0.4954	0.5523
	wholedoc	0.3534	0.3895	0.4258	0.4596	0.5223		wholedoc	0.4030	0.4353	0.4661	0.4968	0.5537
LN	baseline	0.3890	0.4185	0.4482	0.4795	0.5390	RF	baseline	0.4129	0.4389	0.4694	0.4995	0.5569
	body	0.3412	0.3650	0.3964	0.4304	0.4965		body	0.3941	0.4212	0.4511	0.4818	0.5423
	anchor	0.3382	0.3733	0.4036	0.4349	0.5022		anchor	0.3444	0.3725	0.4076	0.4385	0.5048
	title	0.3957	0.4252	0.4542	0.4849	0.5432		title	0.4065	0.4342	0.4613	0.4901	0.5496
	url	0.3876	0.4202	0.4533	0.4854	0.5428		url	0.4036	0.4327	0.4629	0.4921	0.5485
	wholedoc	0.3939	0.4209	0.4525	0.4846	0.5453		wholedoc	0.3352	0.3687	0.4026	0.4360	0.5033

Table 5. Normalization NDCG@k performance on MQ2008 after normalization. Models are assessed using NDCG@k at cutoffs of 5, 10, 15, 20, and 30.

Alg.	Features	@5	@10	@15	@20	@30	Alg.	Features	@5	@10	@15	@20	@30
MR	baseline	0.4586	0.5036	0.5179	0.5257	0.5328	RN	baseline	0.4345	0.4843	0.5010	0.5072	0.5157
	body	0.4469	0.4943	0.5077	0.5165	0.5257		body	0.4055	0.4575	0.4771	0.4858	0.4948
	anchor	0.4299	0.4812	0.4989	0.5061	0.5153		anchor	0.4122	0.4688	0.4859	0.4940	0.5036
	title	0.4496	0.4977	0.5096	0.5167	0.5258		title	0.4446	0.4931	0.5084	0.5161	0.5254
	url	0.4616	0.5086	0.5229	0.5308	0.5383		url	0.4604	0.5076	0.5214	0.5299	0.5375
	wholedoc	0.4517	0.5005	0.5155	0.5230	0.5308		wholedoc	0.4224	0.4726	0.4910	0.4988	0.5085
RB	baseline	0.4550	0.5003	0.5157	0.5230	0.5301	AR	baseline	0.4364	0.4850	0.5004	0.5075	0.5161
	body	0.4055	0.4575	0.4771	0.4858	0.4948		body	0.4085	0.4667	0.4839	0.4912	0.5011
	anchor	0.4251	0.4763	0.4933	0.5011	0.5118		anchor	0.4099	0.4664	0.4842	0.4934	0.5032
	title	0.4521	0.5020	0.5153	0.5225	0.5307		title	0.3618	0.4118	0.4291	0.4356	0.4470
	url	0.4681	0.5107	0.5239	0.5322	0.5403		url	0.3550	0.4164	0.4409	0.4506	0.4636
	wholedoc	0.4481	0.4922	0.5089	0.5162	0.5255		wholedoc	0.4343	0.4813	0.4957	0.5049	0.5140
CA	baseline	0.4553	0.5016	0.5150	0.5224	0.5304	LM	baseline	0.4619	0.5050	0.5180	0.5267	0.5338
	body	0.4289	0.4792	0.4960	0.5029	0.5127		body	0.4457	0.4944	0.5108	0.5184	0.5279
	anchor	0.4351	0.4851	0.5019	0.5101	0.5196		anchor	0.4267	0.4779	0.4963	0.5044	0.5127
	title	0.4686	0.5142	0.5284	0.5350	0.5432		title	0.4587	0.5014	0.5163	0.5240	0.5331
	url	0.4692	0.5134	0.5267	0.5348	0.5435		url	0.4612	0.5063	0.5212	0.5308	0.5386
	wholedoc	0.4509	0.4970	0.5110	0.5182	0.5277		wholedoc	0.4532	0.5002	0.5164	0.5248	0.5318
LN	baseline	0.4341	0.4851	0.5011	0.5080	0.5156	RF	baseline	0.4513	0.4985	0.5127	0.5200	0.5282
	body	0.4080	0.4585	0.4778	0.4869	0.4957		body	0.4459	0.4985	0.5123	0.5201	0.5298
	anchor	0.4128	0.4678	0.4855	0.4934	0.5030		anchor	0.4277	0.4778	0.4966	0.5040	0.5137
	title	0.4346	0.4864	0.5029	0.5103	0.5189		title	0.4556	0.5040	0.5170	0.5248	0.5340
	url	0.4545	0.5038	0.5182	0.5262	0.5348		url	0.4630	0.5089	0.5241	0.5329	0.5401
	wholedoc	0.4177	0.4728	0.4898	0.4965	0.5067		wholedoc	0.4587	0.5054	0.5212	0.5280	0.5365

The optimal algorithm selection is based on the total number of rankers that outperform the baseline. This is referred to as the winning number [55], as shown in Table 6. We identify the top three performers for each dataset. In Table 6, RF emerges as the top performer on the MQ2007 dataset with 17 wins, followed by LN with 15. Additionally, RN, RB, and CA each achieve a count of 10. We observe a similar pattern in MQ2008, where RF again leads with 17 wins, LN follows with 15, and RN and CA each attain 10.

Table 6. Normalization NDCG@k performance on MQ2008 after normalization. Models are assessed using NDCG@k at cutoffs of 5, 10, 15, 20, and 30.

	MR	RF	RN	RB	LM	AR	CA	LN
MQ2007	5	17	10	10	4	0	10	15
MQ2008	5	17	10	7	4	0	10	15

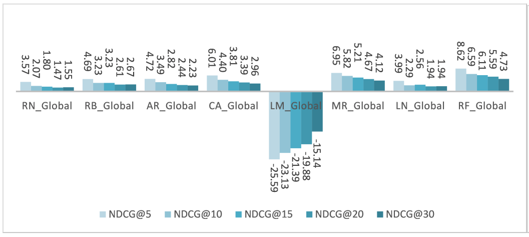
During the selection process, RF performs as the best-performing algorithm, representing the pointwise category. LN, as the second-highest performer, represents the listwise category. And RN represents the pairwise category.

In the global re-ranking layer, we only utilize these selected optimal LTR algorithms (RF, RN, and LN) to integrate results from the local re-ranking layers. For each selected algorithm, multiple rankers are trained on the training data. We select and retain the model performing best on the validation data. The training metric is NDCG at cutoff values. We chose the top-performing ranker as the final ranking model. It is then saved and used to generate the final ranking results.

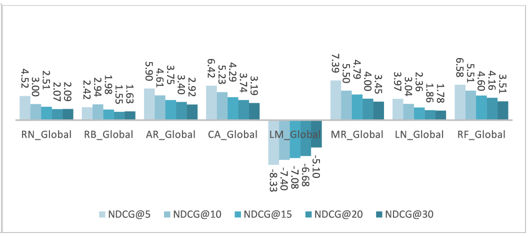
4.4. Results

The MultiLTR retrieval effectiveness is evaluated using the NDCG@n and MAP@n metrics, as shown in Figures 3 and 4.

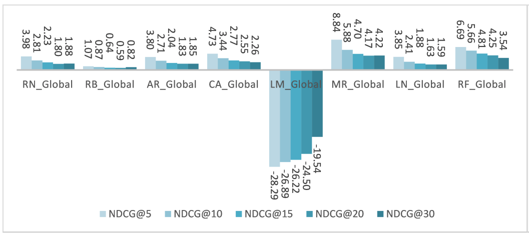
As outlined in Section 4.3, we select three optimal LTR algorithms. LN represents the pointwise approach, RN represents the pairwise approach, and RF represents the listwise approach. These algorithms are used to construct the local re-ranking rankers. In the global re-ranking layer, all eight LTR algorithms are incorporated. The reported results reflect performance after the completion of all layers.



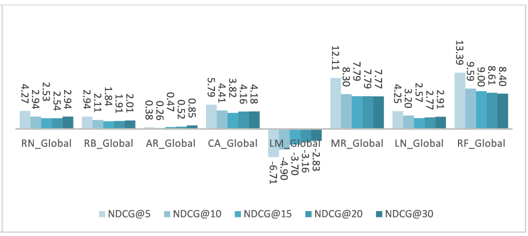
(**a**) LN_Local on MQ2007



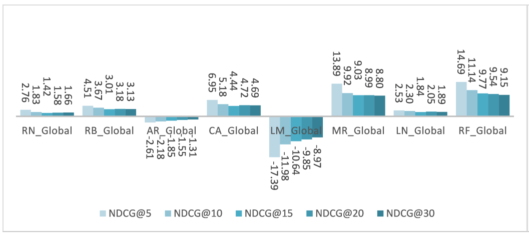
(**b**) RN_Local on MQ2007



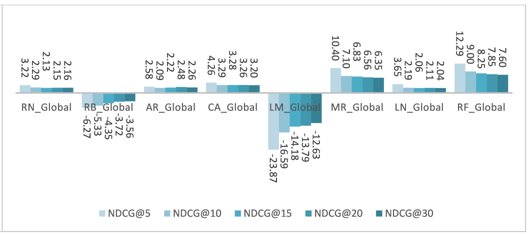
(**c**) RF_Local on MQ2007



(**d**) LN_Local on MQ2008



(**e**) RN_Local on MQ2008



(**f**) RF_Local on MQ2008

Figure 3. NDCG performance of MultiLTR vs. baselines. The local stage employs three optimal LTR algorithms (LN: pointwise, RN: pairwise, RF: listwise). The global stage integrates eight diverse LTR algorithms. Models are assessed using NDCG@k at cutoffs of 5, 10, 15, 20, and 30.

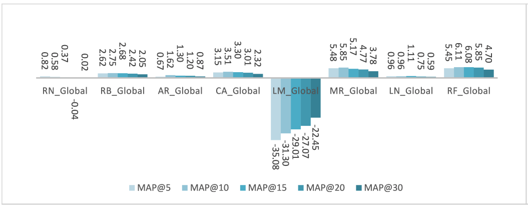
Analyzing the results in Figure 3, the MultiLTR approach consistently outperforms the baselines across all NDCG@n evaluations for the MQ2007 dataset, except for cases where LM_Global is used as the re-ranking algorithm. Among all three selected local re-ranker algorithms, the top-performing global re-rankers are RF_Global and MR_Global. They are both pointwise-based methods and consistently yield the best results:

- When employing LN_Local as the local re-ranker, the most effective model utilizes RF_Global as the global re-ranker. This achieves a notable improvement of 8.62%. Following closely, MR_Global as the global re-ranker yields an improvement of 6.95%;
- For RN_Local as the local re-ranker, the optimal model applies MR_Global as the global re-ranker. This leads to a significant improvement of 7.39%, with RF_Global following at 6.58%;
- In the case of RF_Local, the best-performing model integrates MR_Global as the global re-ranker. This produces a substantial improvement of 8.84%, while RF_Global ranks second with a 6.69% improvement.

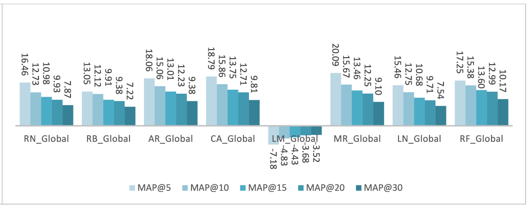
Similarly, for the MQ2008 dataset, models using LM_Global as the global re-ranker fail to surpass the baselines across all three selected local LTR algorithms:

- With LN_Local as the local re-ranker, RF_Global emerges as the most effective global re-ranker. It delivers a substantial 13.39% improvement in NDCG@5, closely followed by MR_Global with a 12.11% increase;
- For RN_Local, RF_Global remains the optimal choice. It achieves a remarkable 14.69% improvement, with MR_Global trailing slightly at 13.89%;
- When applying RF_Local, RF_Global again leads with a 12.29% improvement, while MR_Global secures the second-best performance at 10.40%.

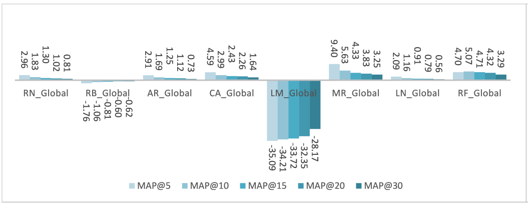
Figure 4 presents the MAP@n evaluation results. It exhibits a trend similar to that observed with the NDCG@n metric. This reinforces the consistency of the proposed MultiLTR approach in enhancing retrieval performance.



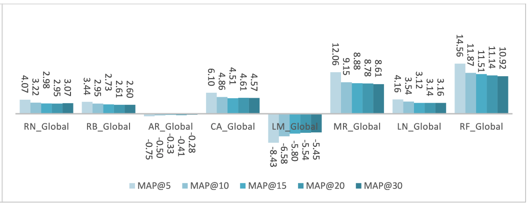
(**a**) LN_Local on MQ2007



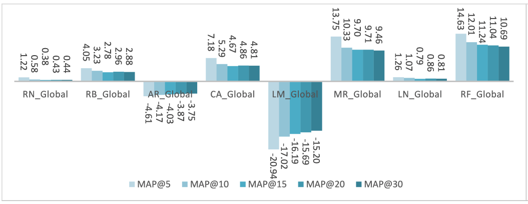
(**b**) RN_Local on MQ2007



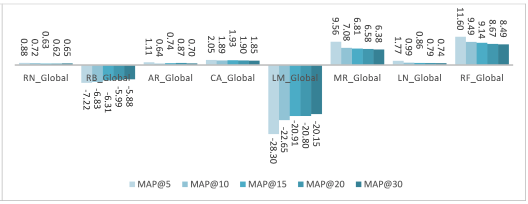
(**c**) RF_Local on MQ2007



(**d**) LN_Local on MQ2008



(**e**) RN_Local on MQ2008



(**f**) RF_Local on MQ2008

Figure 4. MAP Performance of MultiLTR vs. baselines. The local stage employs three optimal LTR algorithms (LN: pointwise, RN: pairwise, RF: listwise). The global stage integrates eight diverse LTR algorithms. Models are assessed using MAP@k at cutoffs of 5, 10, 15, 20, and 30.

5. Discussion

To better understand the results, we first conduct a comparative analysis of the MQ2007 dataset using NDCG and MAP metrics. Next, we assess the impact of three key components in MultiLTR through extensive ablation experiments. Additionally, we evaluate the computational efficiency of the proposed method.

5.1. Comparative Analysis

Tables 7 and 8 present a comparative analysis of the MQ2007 dataset, using the NDCG and MAP metrics, respectively. We highlight and evaluate the top-performing models for each approach.

To thoroughly assess the effectiveness of our models against existing ones, we include the performance of these models: the classic BM25 and three neural-based ranking models (KNRM, HiNT, and DeepTileBars). KNRM [59] is a neural retrieval model that captures term-level interactions via embedded representations. It utilizes kernel-based pooling to extract multi-level soft-matching features, and computes a final relevance score to align documents with queries. HiNT [60] is a hierarchical neural retrieval model and processes segment-level interaction matrices as input. It leverages a local matching layer and a global decision layer to determine document relevance. DeepTileBars [61] segments documents into topic-based sections and applies Convolutional Neural Networks of varying sizes to analyze these sections.

Table 7. Comparison between MultiLTR and fLTR on MQ2007 with NDCG@k, with cutoffs at 5, 10, 15, 20, and 30.

LTR Alg.	Method	@5	@10	@15	@20	@30
Neural Model	KNRM	0.3790	0.4120	0.4256	0.4309	0.4324
	HiNT	0.4630	0.4900	0.5102	0.5253	0.5358
	DeepTileBars	0.3980	0.4340	0.4507	0.4605	0.4651
LN	BM25	0.3890	0.4185	0.4482	0.4795	0.5390
	fLTR	0.3957	0.4193	0.4482	0.4796	0.5413
	MultiLTR	0.4225	0.4460	0.4756	0.5063	0.5645
RN	BM25	0.3951	0.4243	0.4560	0.4870	0.5457
	fLTR	0.4051	0.4298	0.4603	0.4893	0.5504
	MultiLTR	0.4243	0.4477	0.4779	0.5072	0.5649
RF	BM25	0.4129	0.4389	0.4694	0.4995	0.5569
	fLTR	0.4286	0.4489	0.4775	0.5063	0.5657
	MultiLTR	0.4494	0.4647	0.4920	0.5207	0.5804

Table 8. Comparison between MultiLTR fLTR on MQ2007 using MAP@k, with cutoffs at 5, 10, 15, 20, and 30.

LTR Alg.	Method	@5	@10	@15	@20	@30
Neural Model	KNRM	0.1567	0.2254	0.2683	0.2941	0.3193
	HiNT	0.1916	0.2683	0.3218	0.3583	0.3953
	DeepTileBars	0.1647	0.2377	0.2840	0.3140	0.3430
LN	BM25	0.1610	0.2292	0.2824	0.3273	0.3981
	fLTR	0.1566	0.2232	0.2748	0.3187	0.3897
	MultiLTR	0.1698	0.2432	0.2996	0.3465	0.4168
RN	BM25	0.1461	0.2137	0.2670	0.3099	0.3822
	fLTR	0.1650	0.2344	0.2887	0.3323	0.4039
	MultiLTR	0.1754	0.2476	0.3037	0.3501	0.4210
RF	BM25	0.1672	0.2432	0.3006	0.3481	0.4202
	fLTR	0.1710	0.2455	0.3020	0.3489	0.4212
	MultiLTR	0.1829	0.2569	0.3148	0.3631	0.4340

As shown in Tables 7 and 8, the MultiLTR rankers consistently outperform KNRM and DeepTileBars across all evaluation metrics and surpass HiNT at the @30 metric. Furthermore, the results demonstrate that the MultiLTR approach provides additional improvements over the fLTR method introduced in our previous study [1]. Specifically, as Table 7 presents, even the modest improvement achieved by MultiLTR under the LN algorithm shows the superiority of MultiLTR compared to other approaches: MultiLTR outperforms both BM25 and fLTR across all cutoff points (@5–@30). For instance, with a cutoff @30, MultiLTR (0.5645) surpasses BM25 (0.5390) and fLTR (0.5413) by 2.55% and 2.32%, respectively; when compared with the neural baselines, MultiLTR (0.5645) substantially exceeds all neural models with cutoff @30, including HiNT (0.5358), KNRM (0.4324), and DeepTileBars (0.4651), achieving 2.87–13.21% improvements; MultiLTR still outperforms other neural baselines (KNRM, DeepTileBars) for early cutoffs (@5–@20) while slightly trailing HiNT. Similar results with the MAP evaluation can be observed in Table 8.

The fLTR method [1] is a two-stage re-ranking technique that employs aggregation algorithms like comSUM [62] to merge results from different fields. In contrast, the MultiLTR approach utilizes a multi-stage re-ranking strategy for enhanced flexibility and performance.

Aggregation methods such as comSUM effectively address field-based retrieval tasks. However, they are constrained by fixed formulas and ranking score calculations. And they often rely on pre-defined or equal-weighted scores that lack adaptability. Essentially, aggregation functions as a simple regression-based approach, and the scoring function depends on the specific aggregation technique used. For instance, comSUM calculates the final score by summing individual field-based scores.

Re-ranking with LTR can be seen as an extension of aggregation methods, as LTR algorithms refine results from the previous local ranking layers. Unlike traditional aggregation, LTR provides a broader selection of algorithms and greater flexibility. Its parameters are fine-tuned during model training, enabling more adaptive and effective ranking performance.

5.2. Ablation Results

We conducted extensive ablation experiments to evaluate the impact of three key components in MultiLTR. Specifically, MultiLTR w/o feature clustering removes the feature clustering module and groups similar features to enhance data representation. MultiLTR w/o local ranking eliminates the local ranking module and captures feature relationships to improve ranking performance. MultiLTR w/o normalization excludes the normalization module and standardizes feature scales to ensure equal contribution to the model's performance.

The results in Table 9 show that these components contribute to positive improvements for MultiLTR. Feature clustering has a moderate impact on the performance of MultiLTR, with the most significant improvements observed in the RF setting. The local ranking has the most significant impact on the performance of MultiLTR, with noticeable improvements across all LTR algorithms and evaluation metrics. Compared to the other two components, normalization has a smaller impact on the performance of MultiLTR. However, it still contributes to the overall performance.

Table 9. Ablation study on the impact of feature clustering, local ranking, and normalization. Models are assessed using NDCG@k at cutoffs of 5, 10, 15, 20, and 30.

LTR Alg.	Model	@5	@10	@15	@20	@30
LN	MultiLTR	0.4225	0.4460	0.4756	0.5063	0.5645
	MultiLTR w/o feature clustering	0.4181	0.4451	0.4722	0.5310	0.5600
	MultiLTR w/o local ranking	0.4112	0.4352	0.4650	0.4950	0.5550
	MultiLTR w/o normalization	0.4202	0.4441	0.4730	0.5030	0.5620
RN	MultiLTR	0.4243	0.4477	0.4779	0.5072	0.5649
	MultiLTR w/o feature clustering	0.4201	0.4430	0.4730	0.5030	0.5620
	MultiLTR w/o local ranking	0.4150	0.4401	0.4700	0.5012	0.5601
	MultiLTR w/o normalization	0.4220	0.4460	0.4761	0.5060	0.5630
RF	MultiLTR	0.4494	0.4647	0.4920	0.5207	0.5804
	MultiLTR w/o feature clustering	0.4451	0.4600	0.4870	0.5170	0.5770
	MultiLTR w/o local ranking	0.4350	0.4550	0.4801	0.5111	0.5703
	MultiLTR w/o normalization	0.4471	0.4620	0.4890	0.5190	0.5780

5.3. Time Consumption

Having established the effectiveness of the proposed method, it is equally important to evaluate its computational efficiency. We assess the time required to construct a ranking model for each method and present it in Figure 5. Compared to the fLTR method, the

MultiLTR approach demands more computational time. We conducted all experiments and evaluations on a machine equipped with an Intel i7-10700K CPU, 64GB of memory, and an Nvidia A5000 GPU, DELL, USA .

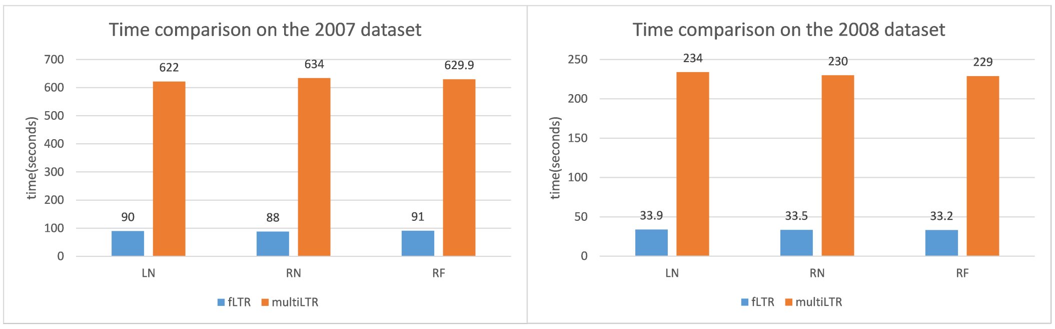


Figure 5. Comparison of time consumption between the fLTR and MultiLTR methods.

6. Conclusions and Future Work

This research introduces a MultiLTR approach and evaluates its effectiveness in text ranking tasks. We performed experiments on the MQ2007 and MQ2008 benchmark datasets and utilized a diverse set of learning-to-rank algorithms. The results demonstrate that MultiLTR consistently outperforms state-of-the-art baselines. ListNet, RankNet, and random forest are selected to construct the local re-ranking rankers. For all three local re-ranker algorithms, the top-performing global re-rankers are random forest and MART. They are both pointwise-based methods and consistently yield the best results. The results highlight that incorporating multi-stage ranking significantly enhances overall ranking performance.

MultiLTR proves superior to single-stage field-based LTR methods, underscoring the benefits of multiple re-ranking stages. However, its increased computational cost presents an opportunity for further optimization. This work lays the foundation for integration with neural ranking architectures through a structured three-tier approach [63]. First, we will augment MultiLTR's feature representation by hybridizing BERT-derived contextual embeddings with traditional features via attention-gated fusion modules, preserving backward compatibility while enhancing semantic expressiveness. Second, a reranking system will deploy efficiency-optimized BERT variants (e.g., distilled cross-encoders) on top-K candidates, coupled with asynchronous batch processing to balance latency and relevance. Finally, knowledge distillation will bridge BERT's document interaction insights into MultiLTR's core ranking model through attention pattern transfer and hybrid loss alignment. Additionally, to bridge the gap between LTR frameworks and neural ranking paradigms, we will adapt MultiLTR's multi-granularity optimization framework to transformer-based models. A potential approach involves employing MultiLTR to dynamically aggregate BERT-derived relevance signals at varying semantic levels. This integration could leverage attention-based gating mechanisms to weigh the contributions of features at different granularities. Moreover, we will rigorously validate this hybrid paradigm on both standardized large-scale benchmarks (MS MARCO, TREC DL) and industrial search systems. This ensures generalizability across diverse real-world scenarios. This systematic validation will quantify robustness against dataset shifts and operational constraints, bridging theoretical advances with practical deployment requirements.

Author Contributions: Conceptualization, methodology, software, data curation, validation, writing—original draft, funding acquisition, project administration, H.Y.; conceptualization, formal analysis, resources, supervision, writing—review and editing, T.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by the Key Scientific Research Project of Higher Education Institutions in Henan Province, (grant No. 24A520060).

Institutional Review Board Statement: Not applicable

Data Availability Statement: The data that support the findings of this study are available upon request.

Acknowledgments: We are thankful to Shilong Li for his work in parts of the experiments in this work.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LTR	learning-to-rank
MultiLTR	multi-stage learning-to-rank
fLTR	field learning-to-rank
DCG	discounted cumulative gain
IDCG	Idea discounted cumulative gain
NDCG	normalized discounted cumulative gain
AP	average precision
MAP	mean average precision
MR	MART
RF	random forest
RB	RankBoost as RB
RN	RankNet
LM	LambdaMART
LN	ListNet
CA	coordinate ascent
AR	AdaRank

References

1. Yang, H.; Gonçalves, T. Field features: The impact in learning to rank approaches. *Appl. Soft Comput.* **2023**, *138*, 110183. [CrossRef]
2. Clarke, C.L.; Culpepper, J.S.; Moffat, A. Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. *Inf. Retr. J.* **2016**, *19*, 351–377. [CrossRef]
3. Zhang, L.; Zhang, Y.; Long, D.; Xie, P.; Zhang, M.; Zhang, M. A two-stage adaptation of large language models for text ranking. In Proceedings of the Findings of the Association for Computational Linguistics ACL 2024, Bangkok, Thailand, 11–16 August 2024; pp. 11880–11891.
4. Zheng, K.; Zhao, H.; Huang, R.; Zhang, B.; Mou, N.; Niu, Y.; Song, Y.; Wang, H.; Gai, K. Full stage learning to rank: A unified framework for multi-stage systems. In Proceedings of the ACM Web Conference 2024, Singapore, 13–17 May 2024; pp. 3621–3631.
5. Liu, Z.; Li, C.; Xiao, S.; Li, C.; Lian, D.; Shao, Y. Matryoshka Re-Ranker: A Flexible Re-Ranking Architecture with Configurable Depth and Width. *arXiv* **2025**, arXiv:2501.16302.
6. Nogueira, R.; Yang, W.; Cho, K.; Lin, J. Multi-stage document ranking with BERT. *arXiv* **2019**, arXiv:1910.14424.
7. Fan, Y.; Xie, X.; Cai, Y.; Chen, J.; Ma, X.; Li, X.; Zhang, R.; Guo, J. Pre-training methods in information retrieval. *Found. Trends® Inf. Retr.* **2022**, *16*, 178–317. [CrossRef]
8. Lu, J.; Hall, K.; Ma, J.; Ni, J. HYRR: Hybrid Infused Reranking for Passage Retrieval. *arXiv* **2022**, arXiv:2212.10528.
9. Wang, B.; Li, M.; Zeng, Z.; Zhuo, J.; Wang, S.; Xu, S.; Long, B.; Yan, W. Learning Multi-Stage Multi-Grained Semantic Embeddings for E-Commerce Search. *arXiv* **2023**, arXiv:2303.11009.
10. Huang, P.S.; He, X.; Gao, J.; Deng, L.; Acero, A.; Heck, L. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, San Francisco, CA, USA, 27 October–1 November 2013; pp. 2333–2338.
11. Zhang, H.; Wang, S.; Zhang, K.; Tang, Z.; Jiang, Y.; Xiao, Y.; Yan, W.; Yang, W.Y. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 25–30 July 2020; pp. 2407–2416.

12. Qiu, Y.; Zhao, C.; Zhang, H.; Zhuo, J.; Li, T.; Zhang, X.; Wang, S.; Xu, S.; Long, B.; Yang, W.Y. Pre-training Tasks for User Intent Detection and Embedding Retrieval in E-commerce Search. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, 17–21 October 2022; pp. 4424–4428.
13. Hai Le, N.; Gerald, T.; Formal, T.; Nie, J.Y.; Piwowarski, B.; Soulier, L. CoSPLADE: Contextualizing SPLADE for Conversational Information Retrieval. In Proceedings of the European Conference on Information Retrieval, Dublin, Ireland, 2–6 April 2023; Springer: Cham, Switzerland , 2023; pp. 537–552.
14. Yang, D.; Zhang, Y.; Fang, H. An exploration study of mixed-initiative query reformulation in conversational passage retrieval. *arXiv* **2023**, arXiv:2307.08803.
15. Gao, L.; Dai, Z.; Callan, J. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, 28 March–1 April 2021, Proceedings, Part II 43*; Springer: Cham, Switzerland, 2021; pp. 280–286.
16. Nogueira, R.; Cho, K. Passage Re-ranking with BERT. *arXiv* **2019**, arXiv:1901.04085.
17. Yilmaz, Z.A.; Wang, S.; Yang, W.; Zhang, H.; Lin, J. Applying BERT to document retrieval with birch. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, Hong Kong, China, 3–7 November 2019; pp. 19–24.
18. Lin, S.C.; Yang, J.H.; Nogueira, R.; Tsai, M.F.; Wang, C.J.; Lin, J. Multi-stage conversational passage retrieval: An approach to fusing term importance estimation and neural query rewriting. *ACM Trans. Inf. Syst. (TOIS)* **2021**, *39*, 48. [CrossRef]
19. Guo, J.; Fan, Y.; Pang, L.; Yang, L.; Ai, Q.; Zamani, H.; Wu, C.; Croft, W.B.; Cheng, X. A deep look into neural ranking models for information retrieval. *Inf. Process. Manag.* **2020**, *57*, 102067. [CrossRef]
20. Craswell, N.; Mitra, B.; Yilmaz, E.; Campos, D.; Lin, J. Ms marco: Benchmarking ranking models in the large-data regime. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 11–15 July 2021; pp. 1566–1576.
21. Yates, A.; Nogueira, R.; Lin, J. Pretrained transformers for text ranking: BERT and beyond. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Virtual, 8–12 March 2021; pp. 1154–1156.
22. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
23. Leonhardt, J.; Beringer, F.; Anand, A. Exploiting Sentence-Level Representations for Passage Ranking. *arXiv* **2021**, arXiv:2106.07316.
24. Mitra, B.; Diaz, F.; Craswell, N. Learning to match using local and distributed representations of text for web search. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 1291–1299.
25. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
26. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
27. Ahmadi, K.; Gathwala, A.; Osajima, J.; Hsiao, D.; Das, P. SLLIM-Rank: A Multi-Stage Item-to-Item Recommendation Model using Learning-to-Rank. In Proceedings of the 2024 IEEE International Conference on Big Data (BigData), Washington, DC, USA, 15–18 December 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 2264–2268.
28. Lee, J.; Bernier-Colborne, G.; Maharaj, T.; Vajjala, S. Methods, Applications, and Directions of Learning-to-Rank in NLP Research. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, 16–21 June 2024; pp. 1900–1917.
29. Dato, D.; MacAvaney, S.; Nardini, F.M.; Perego, R.; Tonellotto, N. The Istella22 Dataset: Bridging Traditional and Neural Learning to Rank Evaluation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022; pp. 3099–3107.
30. Qin, Z.; Yan, L.; Zhuang, H.; Tay, Y.; Pasumarthi, R.K.; Wang, X.; Bendersky, M.; Najork, M. Are neural rankers still outperformed by gradient boosted decision trees? In Proceedings of the ICLR'2021, Virtual, 3–7 May 2021.
31. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Statist.* **2001**, *29*, 1189–1232. [CrossRef]
32. Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; Hullender, G. Learning to rank using gradient descent. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 89–96.
33. Xu, J.; Li, H. Adarank: A boosting algorithm for information retrieval. In Proceedings of the 30th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, 23–27 July 2007; ACM: New York, NY, USA, 2007; pp. 391–398.
34. Metzler, D.; Croft, W.B. Linear feature-based models for information retrieval. *Inf. Retr.* **2007**, *10*, 257–274. [CrossRef]
35. Wu, Q.; Burges, C.J.; Svore, K.M.; Gao, J. Adapting boosting for information retrieval measures. *Inf. Retr.* **2010**, *13*, 254–270. [CrossRef]
36. Cao, Z.; Qin, T.; Liu, T.Y.; Tsai, M.F.; Li, H. Learning to rank: From pairwise approach to listwise approach. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; ACM: New York, NY, USA, 2007; pp. 129–136.

37. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
38. Köppel, M.; Segner, A.; Wagener, M.; Pensel, L.; Karwath, A.; Kramer, S. Pairwise learning to rank by neural networks revisited: Reconstruction, theoretical analysis and practical performance. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Würzburg, Germany, 16–20 September 2019; Springer: Cham, Switzerland, 2019; pp. 237–252.
39. Jia, Y.; Wang, H.; Guo, S.; Wang, H. Pairrank: Online pairwise learning to rank by divide-and-conquer. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 146–157.
40. Yuan, K.; Kuang, D. Deep Pairwise Learning To Rank For Search Autocomplete. *arXiv* **2021**, arXiv:2108.04976.
41. Ai, Q.; Bi, K.; Guo, J.; Croft, W.B. Learning a deep listwise context model for ranking refinement. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 135–144.
42. Sharma, A. Listwise Learning to Rank with Deep Q-Networks. *arXiv* **2020**, arXiv:2002.07651.
43. Pang, L.; Xu, J.; Ai, Q.; Lan, Y.; Cheng, X.; Wen, J. 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, Virtual, 25–30 July 2020; pp. 499–508.
44. Swezey, R.; Grover, A.; Charron, B.; Ermon, S. PiRank: Scalable Learning To Rank via Differentiable Sorting. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA , 2021; Volume 34.
45. Chen, Z.; Eickhoff, C. PoolRank: Max/Min Pooling-based Ranking Loss for Listwise Learning & Ranking Balance. *arXiv* **2021**, arXiv:2108.03586.
46. Keshvari, S.; Ensan, F.; Yazdi, H.S. ListMAP: Listwise learning to rank as maximum a posteriori estimation. *Inf. Process. Manag.* **2022**, *59*, 102962. [CrossRef]
47. Chen, F.; Fang, H. An Exploration of Learning-to-re-rank Using a Two-step Framework for Fair Ranking. In Proceedings of the TREC, Online, 15–19 November 2022.
48. Han, S.; Wang, X.; Bendersky, M.; Najork, M. Learning-to-Rank with BERT in TF-Ranking. *arXiv* **2020**, arXiv:2004.08476.
49. Awan, Z.; Kahlke, T.; Ralph, P.; Kennedy, P. Bi-Encoders based Species Normalization–Pairwise Sentence Learning to Rank. *arXiv* **2023**, arXiv:2310.14366.
50. Richards, J.A. Feature reduction. In *Remote Sensing Digital Image Analysis*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 403–446.
51. Liu, T.Y. Learning to rank for information retrieval. *Found. Trends® Inf. Retr.* **2009**, *3*, 225–331. [CrossRef]
52. Cabello-Solorzano, K.; Ortigosa de Araujo, I.; Pe na, M.; Correia, L.; Tallón-Ballesteros, A.J. The impact of data normalization on the accuracy of machine learning algorithms: A comparative analysis. In Proceedings of the International Conference on Soft Computing Models in Industrial and Environmental Applications, Salamanca, Spain, 5–7 September 2023; Springer: Cham, Switzerland, 2023; pp. 344–353.
53. Järvelin, K.; Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst. (TOIS)* **2002**, *20*, 422–446. [CrossRef]
54. Qin, T.; Liu, T.Y. Introducing LETOR 4.0 datasets. *arXiv* **2013**, arXiv:1306.2597.
55. Qin, T.; Liu, T.Y.; Xu, J.; Li, H. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.* **2010**, *13*, 346–374. [CrossRef]
56. Zhang, E.; Zhang, Y. Average Precision. In *Encyclopedia of Database Systems*; Springer: Boston, MA, USA, 2009; pp. 192–193. [CrossRef]
57. Sanderson, M. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. ISBN-13 978-0-521-86571-5, xxi+ 482 pages. *Nat. Lang. Eng.* **2010**, *16*, 100–103. [CrossRef]
58. Beitzel, S.M.; Jensen, E.C.; Frieder, O. MAP. In *Encyclopedia of Database Systems*; Springer: Boston, MA, USA, 2009; pp. 1691–1692. [CrossRef]
59. Xiong, C.; Dai, Z.; Callan, J.; Liu, Z.; Power, R. End-to-end neural ad-hoc ranking with kernel pooling. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 55–64.
60. Fan, Y.; Guo, J.; Lan, Y.; Xu, J.; Zhai, C.; Cheng, X. Modeling diverse relevance patterns in ad-hoc retrieval. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 375–384.
61. Tang, Z.; Yang, G.H. Deeptilebars: Visualizing term distribution for neural information retrieval. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 289–296.

62. Fox, E.A.; Shaw, J.A. Combination of multiple searches. In *NIST Special Publications SP*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 1994; Volume 243 . Available online: https://trec.nist.gov/pubs/trec2/t2_proceedings.html (accessed on 12 February 2025).
63. Askari, A.; Abolghasemi, A.; Pasi, G.; Kraaij, W.; Verberne, S. Injecting the score of the first-stage retriever as text improves BERT-based re-rankers. *Discov. Comput.* **2024**, *27*, 15. [CrossRef]