CROSS-JEM: Accurate and Efficient Cross-encoders for Short-text Ranking Tasks

Bhawna Paliwal [1*†] **Deepak Saini** [2*†] **Mudit Dhawan** [3‡]
Siddarth Asokan [1†] **Nagarajan Natarajan** [1] **Surbhi Aggarwal** [4]
Pankaj Malhotra [4] **Jian Jiao** [2] **Manik Varma** [1]

[1]Microsoft Research, India [2]Microsoft, USA [3]Carnegie Mellon University [4]Microsoft, India

{bhawna, desaini}@microsoft.com, mdhawan@cs.cmu.edu,
{siddarth.asokan, Nagarajan.Natarajan, Surbhi.Aggarwal}@microsoft.com,
{pamalhotra, Jian.Jiao, manik}@microsoft.com

Abstract

Ranking a set of items based on their relevance to a given query is a core problem in search and recommendation. Transformer-based ranking models are the state-of-the-art approaches for such tasks, but they score each query-item independently, ignoring the joint context of other relevant items. This leads to sub-optimal ranking accuracy and high computational costs. We address this by proposing Cross-encoders with Joint Efficient Modeling (CROSS-JEM), a novel ranking approach that enables transformer-based models to jointly score multiple items for a query, maximizing parameter utilization. CROSS-JEM leverages (a) redundancies and token overlaps to jointly score multiple items, that are typically short-text phrases arising in search and recommendations, and (b) a novel training objective that models ranking probabilities. CROSS-JEM achieves state-of-the-art accuracy and over 4x lower ranking latency over standard cross-encoders. Our contributions are threefold: (i) we highlight the gap between the ranking application's need for scoring thousands of items per query and the limited capabilities of current cross-encoders; (ii) we introduce CROSS-JEM for joint efficient scoring of multiple items per query; and (iii) we demonstrate state-of-the-art accuracy on standard public datasets and a proprietary dataset. CROSS-JEM opens up new directions for designing tailored early-attention-based ranking models that incorporate strict production constraints such as item multiplicity and latency.

1 Introduction

We consider the problem of ranking that arises in search and recommendation pipelines, wherein the goal is to rank a set of items based on their relevance to a given query. Our work is in the context of two-stage *retrieve-then-rank* pipelines in modern search engines [1–6] as depicted in Figure 1. Given a *query*, i.e., a search phrase such as *"patagonia japan"*, the retrieval stage pares the *items*, i.e., keywords such as *"clothing store japan"*, *"patagonia homes shimoda"* that are bid against for displaying ads, from billions to a few hundreds [7, 8] of potential interest. These items are subsequently provided as the input to the ranking stage. In this work, we focus on the ranking model, given a black-box retriever. We consider short-text items (as in the above example), which appear in a myriad of recommendation systems applications such as product recommendation, query to advertiser bid phrase recommendation, and Wikipedia category tagging [9–11]. In designing the

[*]Denotes equal contribution
[†]Corresponding authors
[‡]Work done while at Microsoft Research, India

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

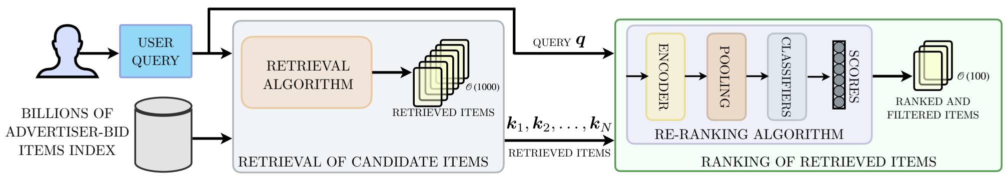


Figure 1: The two-stage large-scale search and recommendation pipeline comprises: (i) candidate selection from billions of items; (ii) re-ranking of retrieved items. CROSS-JEM works at stage (ii), to output the score for all N item in a single pass.

Table 1: A comparison of CROSS-JEM's listwise modeling and a pointwise ranking model [2]: Relatively more *generic* (but relevant) items are ranked higher in baseline predictions, owing to biases such as high frequency in the training data. Our listwise model evaluates all the shortlisted items in a single forward pass, and ranks more specific (and relevant) items higher.

Query	Top-5 ranked items in the Proposed Approach (CROSS-JEM)	Top-5 ranked items in the baseline Cross Encoder
different foods of oaxaca mexico	'the foods of oaxaca', 'oaxacan cuisine', 'exploring oaxacan food', 'authentic recipes from oaxaca, mexico', '6 things you'll love about oaxaca'	'culinary tales: the kinds of food mexicans eat', 'mexican christmas foods', 'mexican cuisine', 'culture: food and eating customes in mexico', 'popular food in mexico',
what is the bovine growth hormone	'recombinant bovine growth hormone', 'what is rbgh?', 'rbgh', 'bovine growth hormone and milk : what you need to know', 'what is rbst?'	'growth hormone', 'human growth hormone', 'alternative names for growth hormone', 'human growth hormone and insulin are friends', 'growth hormone (somatotropin)'

ranking model, two key axes are the model architecture, and the choice of the loss function, while the key performance metrics for such systems are accuracy and inference latency.

Traditionally, *encoders* that employ stacked attention layers to encode a query-item pair, followed by a *classifier* to predict ranking scores are widely adopted for ranking [2, 12, 5]. Recently, sequence-to-sequence models with encoder-decoder and decoder-only architectures have also been proposed for ranking. These models provide a ranking score based on particular vocabulary-token logits [13–15]. However, these approaches model ranking as a *pointwise* task, providing ranking scores for a given query and item pair. But ranking is inherently a list-based task that requires scoring query-item pairs relative to one another, and not in isolation. In particular, the other items in the list to be ranked for a given query provide crucial context for scoring. Pointwise models neglect the list context, produce independent scores that may not reflect the optimal ranking order, and are difficult to calibrate across items for sorting to provide final rankings [16]. Table 1 illustrates this by juxtaposing the top-5 ranked items obtained using our proposed approach (listwise modeling) and a baseline pointwise ranking model [2]. We observe that relatively more *generic* items *e.g.,* "*mexican cuisine*", although relevant to the query *"different foods of oaxaca mexico,"* are ranked higher in baseline predictions, owing to (a) their frequency in training data, (b) token-level matching and other biases which are difficult to mitigate in pointwise modeling. On the other hand, our proposed listwise approach evaluates all the items to be ranked holistically, and subsequently ranks more specific (not just relevant) items higher.

Furthermore, pointwise transformer based models [2, 14] are computationally expensive and impractical for real-time ranking systems that need to handle large-scale traffic requiring low latency and high throughput. Therefore, many industrial systems resort to using simpler sparse neural networks [17] or late-interaction models [18, 19] for online ranking, sacrificing accuracy for latency.

Research along listwise loss functions [20, 14, 21] aim to improve ranking accuracy by optimizing training objective for the whole list of items, not just query-item pairs. Yet, architecturally, they score items independently, ignoring inter-item dependencies and query context. Recent works also use pre-trained LLMs for listwise ranking [22–24, 15]. However, these models have a huge parameter count (a few billions), limiting their scalability and efficiency. **In this work, we bridge this gap by proposing a ranking model that works at the list level, explicitly models inter-item interactions, and achieves superior latency-accuracy tradeoff, making it deployable in real-time scenarios.**

The Proposed Approach: We propose an end-to-end joint ranking approach that models the listwise ranking of the query and given items, capturing both the query-item and the item-item interactions and satisfying the strict latency requirements of industry-scale recommendation systems. Our approach, entitled **CROSS-JEM** (**CROSS**-Encoder with **J**oint **E**fficient **M**odeling), leverages the list structure of the input in both the encoder and the classifier components, as well as in the training objective.

While any standard loss function (e.g., binary cross entropy) could be used to train a CROSS-JEM model, in this work, we propose a novel variant of the listwise loss functions [21, 25], Ranking Probability Loss (RPL); which can be interpreted as a divergence between the predicted probabilities and the estimated ranking probabilities (as a function of model logits and ground truth rankings) of items to be ranked. The proposed listwise loss works much better in conjunction with our joint modeling than standard pointwise or listwise losses [21, 25, 26]. To the best of our knowledge, we are the first to propose a joint ranking approach that can effectively model listwise ranking in both the model architecture and the training objective with real-time latency constraints.

In summary, our key contribution is the introduction of a novel **joint ranking approach CROSS-JEM**, that scores multiple items per query in a single pass, exploiting token interaction across items for better accuracy and token redundancies for better efficiency. We demonstrate the effectiveness of CROSS-JEM on two public benchmark datasets for short-text re-ranking, wherein it outperforms the best-performing baselines by at-least 3% in terms of MRR. When applied to large-scale search-based recommendation, CROSS-JEM demonstrated a 13% higher accuracy than state-of-the-art models, while being over 6× faster than standard cross-encoders [2]. We also deploy CROSS-JEM for real-time ranking on live traffic, where it reduces the quick-back-rate by 1.8%, indicating improved relevance of ads to users. Our work presents a general, scalable framework for joint ranking of multiple items across various domains and short-text tasks, accounting for ranking under task-specific constraints such as item multiplicity and latency.

2 Background and Related Work

Ranking Architecture: State-of-the-art transformer based models with stacked attention layers are typically encoder based, such as monoBERT and Birch [2, 27]. These models encode query-passage pairs with a bi-directional attention encoder and use a classifier to obtain a ranking score. Alternatively, sequence-to-sequence models, such as monoT5 [13] leverage the pre-training knowledge of generative models for ranking. These models generate a ranking score from a specific vocabulary token in the decoder output. Another line of work explores decoder-only models, such as llama- and GPT-based models and rely on their extensive pre-training knowledge and parameter count for ranking [15, 28]. Despite their advantages, decoder-only models fine-tuned on large ranking corpora [15] do not outperform the fine-tuned encoder-based and sequence-to-sequence models on short-text ranking tasks (cf. Section 5). CROSS-JEM is the first joint-ranking approach that effectively incorporates listwise ranking into the model architecture and training objective while maintaining latency constraints.

Joint (Listwise) Ranking: Ranking inherently involves comparing a list of items; thus, recent works employ listwise loss functions in encoder or encoder-decoder models to enhance ranking accuracy. Gao et al. [29] devised a listwise multi-class cross-entropy loss to optimize ranking probabilities in encoder models. Zhuang et al. [14] introduced a ranking-specific listwise cross-entropy loss to improve performance of Seq2Seq models for ranking. They also demonstrated that an expanded form of cross-entropy loss (poly-1) achieved superior performance across various ranking metrics. Although listwise losses improved ranking accuracy, the model architectures of these rankers remain pointwise and do not fully capture the ranking task. CROSS-JEM ranks multiple items per query in one pass, capturing item-item interactions in the ranking list to improve accuracy. It is optimized with Ranking Probability Loss (RPL), a novel variant of the listwise ranking loss (as considered in ListNet [21]) aligned with the CROSS-JEM architecture, and estimates the ranking probabilities using target relevance labels and predicted model logits.

LLMs for Ranking: LLMs have emerged as a powerful tool for ranking tasks; they can leverage pre-trained knowledge, large parameter count, and effective prompting to achieve superior performance. Existing works have adopted two main approaches to exploit LLMs for ranking: (a) Using LLMs directly as re-rankers by designing novel prompting schemes [30, 22, 23] and sorting strategies [24, 31]; the high-level idea is to encode the ranking items and the query into a single input and generate a ranked list as output, using a sliding window technique to handle long inputs; (b) Using LLMs for learning more accurate smaller ranking models [22, 32, 15, 33] via standard distillation or via augmenting the training data with synthetic samples generated by the LLM. These works have demonstrated that LLMs can outperform small-scale supervised methods [2, 13, 24] on various ranking benchmarks. Deploying LLMs at scale for real-time serving scenarios is still challenging due to their high computational costs and memory requirements. Using LLMs to enhance smaller ranking models (as in (b)), is practical and scalable, and can be applied to CROSS-JEM as well.

Efficiency and Scaling: Another closely related area is the focus on improving the efficiency and scaling of transformer based rankers (and retrievers) using light-weight architectures. Approaches employing early-attention (such as monoBERT [2]) have shown high ranking accuracies, but cannot support the low-latency requirements of online raking applications in industry-scale recommendation systems. This can be attributed to the requirement of making multiple calls to the expensive transformer-based encoder to rank a list of items per query, which is infeasible in a few milliseconds. Therefore, online production systems use a variation of sparse neural networks, namely MEB [17], for ranking thousands of items in real-time. Late-interaction models such as ColBERT [19], Baleen [34] and TwinBERT [18] are also used for online ranking due to their computational efficiency. These models reduce the computational costs by applying a late-interaction layer over query-item embeddings. This comes at the price of accuracy, owing to the lack of interactions between query and item tokens. They also incur high storage overheads in online settings, as they need to store and retrieve token-level embeddings. Another line of work that focuses on efficient retrieval and ranking is based on dual-encoder architectures, such as ANCE [35], DPR [36], and INSTRUCTOR [37]. These methods use contrastive-style training to learn a query and item encoder and metrics such as cosine similarity to rank query-item pairs. They can be made highly efficient via a nearest neighbor search [38, 39] over pre-computed embeddings, but lose out on accuracy compared to cross-encoders [2, 40].

3 CROSS-JEM: CROSS-encoders with Joint Efficient Modeling

We now present CROSS-JEM, our proposed approach to accurate and low-latency ranking via efficient scoring of multiple items. Before we describe CROSS-JEM in detail, we develop the notation used in the subsequent sections of this manuscript.

Notation: We denote queries by $\boldsymbol{q}$ and the corresponding set of N candidate items retrieved for $\boldsymbol{q}$ by $\mathbb{K}_{\boldsymbol{q}} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, ..., \boldsymbol{k}_N\}$. We denote the dataset of queries and items used for training by $\mathbb{Q}_{tr}$ and $\mathbb{I}_{tr}$ respectively, and that of the test datasets by $\mathbb{Q}_{te}$ and $\mathbb{I}_{te}$. We drop the subscripts when the meaning is clear from the context. The ground-truth scores are given by $\boldsymbol{y}_i \in \mathbb{R}^N$, $[\boldsymbol{y}_i]_j = y_{ij}$ denoting the score for the item $\boldsymbol{k}_j$ from set $\mathbb{K}_{\boldsymbol{q}_i}$, associated with the query $\boldsymbol{q}_i$. The queries $\boldsymbol{q}$ and items $\boldsymbol{k}_j$ are tokenized via $\mathcal{T}(\cdot)$ to obtain d-dimensional representations (tokens), given by $\mathbb{T}_{\boldsymbol{q}} = \left\{\mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^{L_{\boldsymbol{q}}}\right\}$, and $\mathbb{T}_{\boldsymbol{k}_j} = \left\{\mathbf{k}_j^1, \mathbf{k}_j^2, \ldots, \mathbf{k}_j^{L_{\boldsymbol{k}_j}}\right\}$, respectively, where $\mathbf{q}^\ell, \mathbf{k}_j^\ell \in \mathbb{R}^d$.

Research Problem: We seek to learn a ranking model that, given a query $\boldsymbol{q}$, assigns scores $\mathbb{S}_{\boldsymbol{q}} = \{s_1, s_2, ..., s_N\}$ for all items in $\mathbb{K}_{\boldsymbol{q}}$, such that the ranking induced by the scores is accurate.

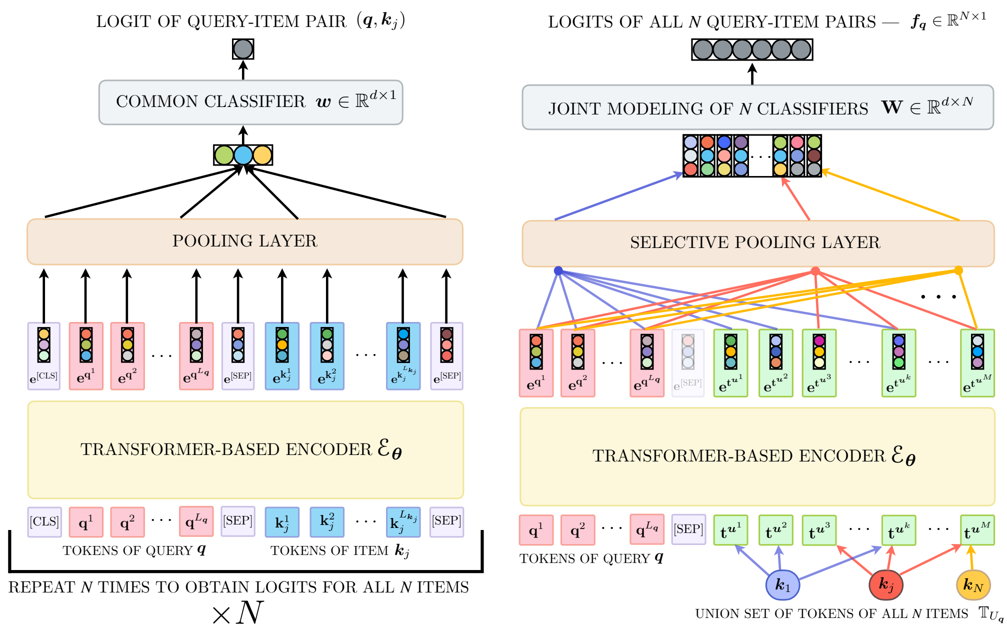
3.1 The CROSS-JEM Architecture

CROSS-JEM comprises an encoder to obtain representations of a given query $\boldsymbol{q}$ and all its candidate items $\mathbb{K}_{\boldsymbol{q}}$. The representations are pooled and passed to a classification head, that outputs a score corresponding to each item $\boldsymbol{k}_i \in \mathbb{K}_{\boldsymbol{q}}$ associated with the query.

CROSS-JEM's architecture is primarily inspired by the observation that the candidate items in the set $\mathbb{K}_{\boldsymbol{q}}$, for a given query $\boldsymbol{q}$, has significant token overlap amongst themselves. A more in-depth exploration of this phenomenon, in the context of efficient-scoring, is provided in Appendix G. Given a query $\boldsymbol{q}$ and its candidate items $\mathbb{K}_{\boldsymbol{q}}$, the core idea is we form the union of tokens $\mathbb{T}_{U_{\boldsymbol{q}}}$ of all items in $\mathbb{K}_{\boldsymbol{q}}$. Subsequently, the representations of query and set $\mathbb{T}_{U_{\boldsymbol{q}}}$ can be obtained in a *single pass* of the encoder, and scored via a *single pass* over the classifiers.

While at first glance, it might appear that using $\mathbb{T}_{U_{\boldsymbol{q}}}$ (with a potential loss of item token ordering) could adversely affect performance, we observed in preliminary experimentation that this is not the case when scoring short-text items. In particular, we compared the performance of two cross-encoder models on search engine logs, one with items as-is, and another comprising items with alphabetically sorted tokens. Both the mean average precision (MAP) and accuracy of the latter model was found to be within 1% of the score obtained when the sequence information is retained. Additional discussions on sequence information are provided in Appendix G. We now describe the CROSS-JEM encoder and classifier in detail.

Encoder: CROSS-JEM employs a trainable encoder $\mathcal{E}_{\boldsymbol{\theta}}$, which takes as input a sequence of tokens $\mathbf{T} = \left[\mathbf{t}^1, \mathbf{t}^2, \ldots, \mathbf{t}^n\right]$ (of length n), and generates as output another sequence of d-dimensional contextual embeddings $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{T}) = \mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]$. These embeddings provide context-



(a) **monoBERT** (b) **CROSS-JEM**

Figure 2: (Best viewed in color) Computing the relevance scores for query $\boldsymbol{q}$ and retrieved set of N items $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N\}$ using (a) monoBERT and (b) CROSS-JEM: (a) monoBERT (standard cross-encoders) infers the scores for each $(\boldsymbol{q}, \boldsymbol{k}_j)$ pair with query and item tokens individually (pointwise). The inference is repeated N times for N items. (b) CROSS-JEM jointly inputs query tokens and the set of union of tokens from all items to be ranked (listwise). The logits for an item $\boldsymbol{k}_j$ can be computed by selecting the contextual embeddings corresponding to its tokens. The selected embeddings are pooled, and input to the linear classifier jointly (*i.e.,* $\mathbf{W} = [\boldsymbol{w}, \boldsymbol{w}, \ldots, \boldsymbol{w}] \in \mathbb{R}^{d\times N}$), thus obtaining the N scores in a single encoder and classifier pass.

dependent representations of the input tokens, and can be used for downstream tasks such as classification, generation, and retrieval. Given the tokenization of the query $(\mathbb{T}_{\boldsymbol{q}})$, and that of an item $(\mathbb{T}_{\boldsymbol{k}_j})$, the contextual embeddings of the tokens $[\mathbf{t}^{[\text{CLS}]}, \mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^{L_{\boldsymbol{q}}}, \mathbf{t}^{[\text{SEP}]}, \mathbf{k}_j^1, \mathbf{k}_j^2, \ldots, \mathbf{k}_j^{L_{\boldsymbol{k}_j}}]$ in baseline variants is given by $[\mathbf{e}^{[\text{CLS}]}, \mathbf{e}^{\mathbf{q}^1}, \mathbf{e}^{\mathbf{q}^2}, \ldots, \mathbf{e}^{\mathbf{q}^{L_{\boldsymbol{q}}}}, \mathbf{e}^{[\text{SEP}]}, \mathbf{e}^{\mathbf{k}_j^1}, \mathbf{e}^{\mathbf{k}_j^2}, \ldots, \mathbf{e}^{\mathbf{k}_j^{L_{\boldsymbol{k}_j}}}]$, where $\mathbf{e}^{[\text{CLS}]}$ and $\mathbf{e}^{[\text{SEP}]}$ denote the embeddings of the $\mathbf{t}^{[\text{CLS}]}$ and $\mathbf{t}^{[\text{SEP}]}$ tokens, respectively. These contextual embeddings are pooled to obtain a single d-dimensional embedding for each pair $(\boldsymbol{q}, \boldsymbol{k}_j)$ by means of a sum or mean pooling layer, or by taking $\mathbf{e}^{[\text{CLS}]}$. This process is computationally expensive due to the need for N forward passes of the encoder to compute the scores for each item in $\mathbb{K}_{\boldsymbol{q}}$.

In CROSS-JEM, we leverage the short-text nature of the items, item-item interactions, and redundancy of tokens amongst items in $\mathbb{K}_{\boldsymbol{q}}$. This is done by computing the contextual embeddings for all *distinct tokens in the retrieved item set* $\mathbb{K}_{\boldsymbol{q}}$ in one pass over the sequence of the query tokens $\mathbb{T}_{\boldsymbol{q}}$ combined with *item token union set* $\mathbb{T}_{U_{\boldsymbol{q}}} = \left\{\mathbf{t}^{\boldsymbol{u}^1}, \mathbf{t}^{\boldsymbol{u}^2}, \ldots, \mathbf{t}^{\boldsymbol{u}^M}\right\}$, where M is the total number of tokens in the union set. The contextual embeddings of all input tokens in CROSS-JEM are computed as

$$\mathbf{E} = \mathcal{E}_{\boldsymbol{\theta}}\left(\left[\mathbf{q}^1, \ldots, \mathbf{q}^{L_{\boldsymbol{q}}}, \mathbf{t}^{[\text{SEP}]}, \mathbf{t}^{\boldsymbol{u}^1}, \ldots, \mathbf{t}^{\boldsymbol{u}^M}\right]\right) = \left[\mathbf{e}^{[\text{CLS}]}, \mathbf{e}^{\mathbf{q}^1}, \ldots, \mathbf{e}^{\mathbf{q}^{L_{\boldsymbol{q}}}}, \mathbf{e}^{[\text{SEP}]}, \mathbf{e}^{\mathbf{t}^{\boldsymbol{u}^1}}, \ldots, \mathbf{e}^{\mathbf{t}^{\boldsymbol{u}^M}}\right].$$

Since the number of tokens in the item union set is significantly smaller than the sum of tokens of all items in $\mathbb{K}_{\boldsymbol{q}}$ (cf. Section 5.2), the proposed token-union-based inference enables highly efficient computation of contextual embeddings. Figure 2 (a) illustrates the difference between the CROSS-JEM encoder, and standard encoders such as monoBERT.

Selective Pooling Layer: Given the contextual embeddings $\mathbf{E}$ for all query tokens $(\mathbb{T}_{\boldsymbol{q}})$ and union over keyword tokens $(\mathbb{T}_{U_{\boldsymbol{q}}})$, CROSS-JEM employs a selective pooling layer to *jointly model* the per-item relevance score. Given $\boldsymbol{k}_j \in \mathbb{K}_{\boldsymbol{q}}$, a pooled representation for pair $(\boldsymbol{q}, \boldsymbol{k}_j)$ is computed as the mean of the contextual embeddings for all tokens in $\mathbb{T}_{\boldsymbol{q}}$ and those tokens in $\mathbb{T}_{U_{\boldsymbol{q}}}$ which are present in $\mathbb{T}_{\boldsymbol{k}_j}$. The set of selected tokens for the pooling layer is given by $\mathbb{P}_{\boldsymbol{q}\boldsymbol{k}_j} = \mathbb{T}_{\boldsymbol{q}} \bigcup \left\{\mathbf{t}^{[SEP]}\right\} \bigcup \left\{\mathbb{T}_{U_{\boldsymbol{q}}} \bigcap \mathbb{T}_{\boldsymbol{k}_j}\right\}.$

Classifier: The final stage in CROSS-JEM is a d-dimensional shared linear classifier $\boldsymbol{w} \in \mathbb{R}^d$ which computes the relevance score associated with each pair $(\boldsymbol{q}, \boldsymbol{k}_j)$. The selectively pooled representations $\boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}_j}$ obtained for all $\boldsymbol{k}_j \in \mathbb{K}_{\boldsymbol{q}}$ are batched together ($\boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}} \in \mathbb{R}^{N \times d}$) allowing for the computation of all logits $[\boldsymbol{f}_{\boldsymbol{q}}]_j = \langle \boldsymbol{w}, \boldsymbol{e}^{\boldsymbol{q}\boldsymbol{k}_j} \rangle$ in a single shot. The scores are defined over these logits (cf. Section 4.1).

4 The CROSS-JEM Algorithm

The detailed inference algorithm for CROSS-JEM is provided in Appendix B, we now describe the Ranking Probability Loss (RPL), which is used to train CROSS-JEM. RPL is a novel variant of the list-based loss function, designed to take advantage of CROSS-JEM's, wherein all item scores are available jointly from a single forward pass.

4.1 The CROSS-JEM Ranking Probability Loss

The standard binary cross-entropy loss ($\mathcal{L}^{BCE}$), used to train cross-encoders, defined over $(\boldsymbol{q}_i, \boldsymbol{k}_j)$, is $-\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} \left[y_{ij} \log \left(\boldsymbol{f}_{\boldsymbol{w},\boldsymbol{q}_i,j} \right) + (1 - y_{ij}) \log \left(1 - \boldsymbol{f}_{\boldsymbol{w},\boldsymbol{q}_i,j} \right) \right]$, where $\boldsymbol{f}_{\boldsymbol{w},\boldsymbol{q}_i,j} = \langle \boldsymbol{w}, \mathcal{E}_{\boldsymbol{\theta}} \left(\boldsymbol{q}_i, \boldsymbol{k}_j \right) \rangle$ is the score of item j, associated with query i, computed by means of an inner product with the classifier $\boldsymbol{w}$. However, such cross-entropy-based pointwise losses fail to account for the list of items available for ranking. *List-based loss functions* [21], in contrast, leverage the task-specific ranking information, help learn a scoring function for a list of items to be ranked, rather than for individual query-item pairs. As an example, consider the ListNet [21] loss is given by:

$$\mathcal{L}^{LN}(\boldsymbol{\theta}, \boldsymbol{w}) = -\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} P_{\boldsymbol{y},j} \log \left(P_{\boldsymbol{f},j} \right), \text{ where } P_{\boldsymbol{x},j} = \frac{\Phi \left([\boldsymbol{x}]_j \right)}{\sum_{\ell=1}^{N} \Phi \left([\boldsymbol{x}]_\ell \right)}, \tag{1}$$

and $\boldsymbol{x}$ is set either to the targets $[\boldsymbol{y}_{\boldsymbol{q}_i}]$, or the output logits $[\boldsymbol{f}_{\boldsymbol{q}_i}]$, and Φ is a normalizing function, typically the exponential operation, leading to P being a $\mathrm{SoftMax}$ function. However, this formulation is still centered around obtaining the pointwise logits, and subsequently computing the top-one probability $P_{\boldsymbol{x},j}$ using a normalization term that accounts for all pairs.

In CROSS-JEM, we design Ranking Probability Loss, a novel variant of $\mathcal{L}^{LN}$ that factors in all logits $[\boldsymbol{f}_{\boldsymbol{q}_i}]$ computed by taking into account the item-item interactions. Given $\boldsymbol{f}_{\boldsymbol{q}_i}$, the ground-truth scores $\boldsymbol{y}_i$, and a candidate item $\boldsymbol{k}_j$, consider the set $\mathbb{L}_j = \{k \in \{1, N\} : [\boldsymbol{y}_i]_k < [\boldsymbol{y}_i]_j\}$, *i.e.,* $\mathbb{L}_j$ comprises indices k for which the ground-truth score at k is **lower** than $[\boldsymbol{y}_i]_j$, the ground-truth score at j. Then,

$$\mathcal{L}^{RPL} = -\sum_{i=1}^{|\mathbb{Q}_{tr}|} \sum_{j=1}^{N} \Big(\sum_{k \in \mathbb{L}_j} [\boldsymbol{y}_i]_k \Big) \log \Big(\mathrm{SoftMax} \Big(\sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_k \Big) \Big). \tag{2}$$

The loss $\mathcal{L}^{RPL}$ represents a cross-entropy loss over functions of the target $\boldsymbol{y}_i$ and scores computed as a function of the logits $\boldsymbol{f}_{\boldsymbol{q}_j}$. The following Lemma relates RPL to the ranking probability distribution.

Lemma 1. ***(Ranking Probability Loss)*** *Assume without loss of generality that $\boldsymbol{f}_{\boldsymbol{q}_i} \in [0, 1]^N$. Let $\mathbf{P} \in \mathbb{R}^{N \times N}$ denote a matrix with entries p_{jk} given by $p_{jk} = \mathrm{Prob}\left(\text{ranking item } \boldsymbol{k}_j \text{ at location } k\right) \triangleq C \sum_{\ell \in \mathbb{L}_k} [\boldsymbol{f}_{\boldsymbol{q}_i}]_\ell$, where C is a normalizing constant. Then, the Ranking Probability Loss maximizes the probability of ranking items $\mathbb{K}_{\boldsymbol{q}_i}$ in the ordering of the ground-truth $\boldsymbol{y}_i$.*

The proof is given in Appendix A. The following Corollary presents an equivalence between the ListNet loss [21] and the proposed Ranking Probability Loss.

Corollary 2. ***(RPL and the ListNet loss)*** *Minimizing the Ranking Probability Loss is equivalent to optimizing for the ListNet top-1 probability loss (Equation 1 [21]) defined over modified scores $\sum_{k \in \mathbb{L}_j} [\boldsymbol{f}_{\boldsymbol{q}_i}]_j$ and modified ground-truth scores $\tilde{y}_{ij} = \sum_{k \in \mathbb{L}_j} [\boldsymbol{y}_i]_j$.*

Intuitively, defining the modified scores in terms of the sum of all logits in $\mathbb{L}_j$, the set of indices of ground-truth scores lower than the logit at j, ensures that the loss takes into account the interactions between the contextual embeddings contribution to the different logits. This is unique to the CROSS-JEM setting, and Corollary 2 shows that all guarantees derived for ListNet loss also hold for RPL.

We now show that CROSS-JEM, trained with RPL yields significantly better ranking accuracy than existing pointwise and listwise loss functions, leading to state-of-the-art performance.

5 Experimental Validation

Datasets: We evaluate CROSS-JEM on the publicly available on **Stack Overflow Duplicate Questions** [41] **(SODQ)** and a short-text version of **MS MARCO** [42] datasets. While the SODQ dataset is used as is, for MS MARCO, a 'short-text' variant of the query-webpage click dataset is constructed by exclusively considering webpage titles (subsequently referred to as **MS MARCO-Titles**). This narrows the dataset's length distribution, aligning it more closely with the typical item lengths seen in sponsored search. We note that the standard metrics for MS MARCO passage ranking do not apply to MS MARCO-Titles, as they rely on the passage content as well as the title for ranking. Therefore, we report updated numbers for MS MARCO-Titles in Table 2, as appropriate. See Appendix C for more details on the datasets.

Evaluation Metrics: To validate the efficacy of CROSS-JEM for ranking tasks, we consider Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as the evaluation metrics on both MS MARCO and SODQ datasets (see Appendix D for metric definitions). Note that MAP is a more comprehensive version of MRR, which is specifically designed for scenarios where the test set contains only one positive item per test query. Hence, on MS MARCO, where there is only one relevant item per query, MAP@K and MRR@K are equivalent for any K.

Baselines: We compare CROSS-JEM with transformer based ranking approaches as well as sparse BoW methods such as **BM25**. Within transformer based ranking baselines, we consider state-of-the-art encoder based methods such as **monoBERT** [2]. We also compare CROSS-JEM against more efficient encoders such as **ColBERT** [19]. Dual Encoders such as **ANCE** [35], **DPR** [36], and **INSTRUCTOR** [37] use contrastive learning to train the encoder and obtain dense representations of queries and items. Following Su et al. [37], we use pre-trained and instruction-tuned INSTRUCTOR model for zero-shot evaluation on MS MARCO and SODQ for baseline. We also compare against state-of-the-art sequence-to-sequence class of models containing encoder-decoder architecture, **RankT5** [14]-24 layers. For a fair comparison with 6 layer CROSS-JEM model, we also fine-tune our own RankT5 model with 6 layers (encoder+decoder) on both MS MARCO-Titles and SODQ datasets. For completeness, we also compare against a decoder only ranking model with Llama2-7B architecture, referred to as **RankingGPT-Llama2-7B**, available from Zhang et al. [15].

Hyper-parameters: CROSS-JEM's tunable hyper-parameters include maximum item union length, L_u and number of items per query, N. These hyper-parameter values are dictated by the application requirements and efficiency constraints. We use $N = 10$ and $L_u = 360$ on MS MARCO and $N = 30$ and $L_u = 242$ on SODQ. Hyperparameters used for the baselines are given in Appendix E.

5.1 Accuracy Results on Public Benchmarks

Table 2 shows that CROSS-JEM achieves superior performance over the encoder based methods on both datasets, demonstrating the effectiveness of its listwise ranking approach. Encoder-decoder based methods such as RankT5-6L fine-tuned with a listwise ranking loss, outperform similar sized encoder based methods (monoBERT and ColBERT), highlighting the benefits of listwise modeling for ranking. However, we find that RankT5-base and RankingGPT-llama2-7b, large Seq2Seq models fine-tuned on a long-text passage ranking task [14], perform worse than smaller models that are specifically fine-tuned on short-text ranking. These results suggest that the Seq2Seq models are sensitive to the domain and length of the ranking tasks, and require careful fine-tuning and adaptation. To verify this, we fine-tune a RankT5-base model on the short-text ranking datasets and observe a significant improvement in performance (cf. Appendix E). We also report that CROSS-JEM, which uses a 6-layer BERT as the base encoder, has the same number of parameters as monoBERT, but is much faster and more accurate. Specifically, CROSS-JEM can perform joint (listwise) inference for ranking over 4× faster than monoBERT, which requires multiple pointwise computations (cf. Section 5.2). We perform further experiments on the comparison of pointwise and listwise loss functions in CROSS-JEM in Appendix F.

5.2 Interpreting CROSS-JEM's Performance

We characterize the **time complexity** of CROSS-JEM in terms of number of query tokens $L_{\boldsymbol{q}} = |\mathbb{T}_{\boldsymbol{q}}|$, number of item tokens $L_{\boldsymbol{k}} = |\mathbb{T}_{\boldsymbol{k}}|$, number of transformer layers L, number of candidate items N, and item union compression factor C (approximated as $\frac{L_{\boldsymbol{k}} \times N}{|\mathbb{T}_{U_{\boldsymbol{q}}}|}$). The time complexity for scoring

Table 2: Performance of CROSS-JEM and the baseline methods on the SODQ and MS MARCO-Titles ranking datasets: All baselines and our method, CROSS-JEM, use ∼60-100M parameter base models and are fine-tuned on the corresponding datasets, except for the large pre-trained models (indicated with asterisk (*)), which are used as is without any further fine-tuning on the two datasets. CROSS-JEM surpasses similar-sized state-of-the-art methods fine-tuned for short-text ranking as well as large pre-trained models by at least 3%.

Method			**Parameters**	**SODQ**		**MS MARCO-Titles**	
				MAP@5	MAP@10	MRR@5	MRR@10
Sparse		BM25	–	32.80	39.26	23.71	24.57
Encoder	Early-interaction	monoBERT	66M	46.79	48.04	30.89	32.47
	Late-interaction	ColBERT	109M	36.10	37.68	30.25	32.00
	Dual Encoders	DPR	66M	47.32	48.48	28.78	30.87
		ANCE	66M	48.31	49.41	28.48	30.53
		INSTRUCTOR*	335M	49.47	50.81	28.84	30.55
Encoder-Decoder	Seq2Seq	RankT5-6L	74M	49.50	50.75	30.73	32.52
		RankT5-base*	223M	45.66	49.47	27.87	29.75
Decoder	Ranking LLMs	RankingGPT-Llama2-7B*	7B	47.64	50.62	28.66	30.47
Ours	**Joint Ranking**	**CROSS-JEM-6L**	**66M**	**52.40**	**53.05**	**33.82**	**35.45**

Table 3: A comparison of latency between CROSS-JEM and various baselines. The mean latency for scoring 700 items per query was computed on A100 GPUs for all models. We observe that CROSS-JEM takes about $4\times$ lower inference time than standard cross-encoders (monoBERT).

Method	ANCE	ColBERT	monoBERT	RankT5-6L	**CROSS-JEM**
Latency (ms) ↓	4.0	4.5	41.3	41.3	9.8

all N items jointly is $(L_{\boldsymbol{q}} + L_{\boldsymbol{k}}N/C)^2 L$. On the other hand, for the standard cross-encoder, the corresponding time complexity is $(L_{\boldsymbol{q}} + L_{\boldsymbol{k}})^2 LN$. In practice, the inference time depends on factors such as the implementation of the model, the hardware employed, and optimizations used (such as quantization). For the sponsored search setting detailed in Appendix G, assuming $L_{\boldsymbol{q}} \approx L_{\boldsymbol{k}}$, letting $N = 100$, the maximum tokens in an item is 24 and maximum item union length used is 220, we have $C \approx (24 * 100)/220 \approx 11$. Then, the standard cross-encoder time complexity is $400L_q^2L$, while that of CROSS-JEM is $101.8L_q^2L$, which is 3.9× lower. These inference time gains are also reflected in the latency comparison of CROSS-JEM against pointwise approaches, *e.g.,* monoBERT (cf. Table 3).

6 Conclusion: Limitations and Future Work

We introduced CROSS-JEM, an accurate and efficient approach for joint ranking of a set of short-text items for a given query based on relevance. CROSS-JEM effectively addresses the two major challenge in existing ranking architectures – sub-optimal accuracy due to pointwise inference, and significantly higher computational cost. Our extensive evaluations on publicly available ranking benchmarks as well as large-scale sponsored search datasets reveal that CROSS-JEM significantly outperforms the baselines, establishing a new state of the art. The scope of this work is primarily focused on the ranking of short texts, a common requirement in both industrial Sponsored Search applications and academic benchmarks, including tasks like matching queries with webpage titles and ranking duplicate questions. While the current work demonstrates significant gains on such short-text ranking tasks, the proposed approach could be adapted for long-text ranking by incorporating post-hoc positional encodings (cf. Appendix F) and more sophisticated attention mechanisms that account for longer document lengths. Exploring these adaptations is an area for future research. CROSS-JEM opens up new directions for designing accurate ranking architectures and algorithms, accounting for application-specific constraints.

References

[1] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1557–1565, 2017.

[2] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.

[3] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4), feb 2024. ISSN 1046-8188. doi: 10.1145/3637870. URL `https://doi.org/10.1145/3637870`.

[4] J. Lin, R. Nogueira, and Yates A. Pretrained Transformers for Text Ranking: BERT and Beyond. In *NAACL*, 2021.

[5] Yucheng Zhou, Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Binxing Jiao, and Daxin Jiang. Towards robust ranker for text retrieval. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5387–5401, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.332. URL `https://aclanthology.org/2023.findings-acl.332`.

[6] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, Jiafeng Guo, et al. Pre-training methods in information retrieval. *Foundations and Trends® in Information Retrieval*, 16(3):178–317, 2022.

[7] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–42, 2022.

[8] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 437–444, 2006.

[9] K. Dahiya, D. Saini, A. Mittal, A. Shaw, K. Dave, A. Soni, H. Jain, S. Agarwal, and M. Varma. DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents. In *WSDM*, 2021.

[10] H. Zhou, M. Huang, Y. Mao, C. Zhu, P. Shu, and X. Zhu. Domain-constrained advertising keyword generation. In *WWW*, 2019.

[11] Peter Schonhofen. Identifying document topics using the wikipedia category network. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 456–462, 2006. doi: 10.1109/WI.2006.92.

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

[13] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL `https://aclanthology.org/2020.findings-emnlp.63`.

[14] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2308–2313, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592047. URL `https://doi.org/10.1145/3539618.3592047`.

[15] Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. A two-stage adaptation of large language models for text ranking. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 11880–11891, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.706.

[16] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.97. URL https://aclanthology.org/2024.findings-naacl.97.

[17] Junyan Chen, Frédéric Dubut, Jason (Zengzhong) Li, and Rangan Majumder. Make every feature binary: A 135b parameter sparse neural network for massively improved search relevance, 2023. URL https://tinyurl.com/y9amxjau.

[18] Wenhao Lu, Jian Jiao, and Ruofei Zhang. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 2645–2652, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412747. URL https://doi.org/10.1145/3340531.3412747.

[19] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[20] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of bert rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 280–286, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-72239-5. doi: 10.1007/978-3-030-72240-1_26. URL https://doi.org/10.1007/978-3-030-72240-1_26.

[21] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL http://doi.acm.org/10.1145/1273496.1273513.

[22] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 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*, pages 14918–14937, 2023.

[23] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*, 2023.

[24] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, 2024.

[25] Tao Qin, Tie-Yan Liu, and Hang Li. Query-level loss functions for information retrieval. *Information Processing Management*, 44(2):838–855, January 2008. URL https://www.microsoft.com/en-us/research/publication/query-level-loss-functions-for-information-retrieval/.

[26] Wei Chen, Tie-yan Liu, Yanyan Lan, Zhi-ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/2f55707d4193dc27118a0f19a1985716-Paper.pdf.

[27] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. 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)*, pages 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1352. URL `https://aclanthology.org/D19-1352`.

[28] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657951. URL `https://doi.org/10.1145/3626772.3657951`.

[29] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of bert rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43*, pages 280–286. Springer, 2021.

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

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

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

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

[34] Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *Advances in Neural Information Processing Systems*, 34: 27670–27682, 2021.

[35] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*, 2021.

[36] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-T. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[37] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.71. URL `https://aclanthology.org/2023.findings-acl.71`.

[38] S. J. Subramanya, Devvrit, R. Kadekodi, R. Krishnaswamy, and H. Simhadri. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS*, 2019.

[39] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[40] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 3715–3734, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272.

[41] Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 2–5, 11 2018. doi: 10.1145/3283812.3283815.

[42] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020.

[43] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[45] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

[46] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, page 58–65, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330008. doi: 10.1145/2682862.2682863. URL https://doi.org/10.1145/2682862.2682863.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.