# Permutation Equivariant Document Interaction Network for Neural Learning-to-Rank

Rama Kumar Pasumarthi, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, Marc Najork
{ramakumar,hlz,xuanhui,bemike,najork}@google.com
Google Research

## ABSTRACT

How to leverage cross-document interactions to improve ranking performance is an important topic in information retrieval research. The recent developments in deep learning show strength in modeling complex relationships across sequences and sets. It thus motivates us to study how to leverage cross-document interactions for learning-to-rank in the deep learning framework. In this paper, we formally define the permutation equivariance requirement for a scoring function that captures cross-document interactions. We then propose a self-attention based document interaction network that extends any univariate scoring function with contextual features capturing cross-document interactions. We show that it satisfies the permutation equivariance requirement, and can generate scores for document sets of varying sizes.

Our proposed methods can automatically learn to capture document interactions without any auxiliary information, and can scale across large document sets. We conduct experiments on four ranking datasets: the public benchmarks WEB30K and Istella, as well as Gmail search and Google Drive Quick Access datasets. Experimental results show that our proposed methods lead to significant quality improvements over state-of-the-art neural ranking models, and are competitive with state-of-the-art gradient boosted decision tree (GBDT) based models on the WEB30K dataset.

## CCS CONCEPTS

• **Information systems → Learning to rank**.

## KEYWORDS

Learning-to-Rank; Information Retrieval; Machine Learning

**ACM Reference Format:**
Rama Kumar Pasumarthi, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, Marc Najork. 2020. Permutation Equivariant Document Interaction Network for Neural Learning-to-Rank. In *Proceedings of the 2020 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '20), September 14–17, 2020, Virtual Event, Norway*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3409256.3409819

## 1 INTRODUCTION

Ranking is a central problem in many applications of information retrieval such as search and recommender systems. Given some query or context, the goal of a ranking algorithm is to sort a *set* of documents into a ranked list so that the utility of the entire list is maximized. Learning-to-rank (LTR) employs machine learning techniques to solve ranking problems. The common formulation is to find a function that can produce scores for the set of documents given a query. The scores can then be used to sort the documents.

While much of the research in LTR has been devoted to the evolution of ranking loss functions [11], the nature of the learned scoring function has largely remained the same: a univariate scoring function that computes a relevance score for a document in isolation. How to capture cross-document interactions is a promising but less well-studied research topic in LTR. Naturally, when cross-document interactions are considered in a model, the score of each individual document is influenced by other documents that are scored together. A desired property for such a model is being *permutation equivariant*, that is, the score of each document should not be affected by the order of the input documents, and shuffling the input documents produces an identical shuffle on the output scores.

Recently, neural network based approaches have proven effective for LTR applications [4, 14, 15]. In this context, we formally define the permutation equivariance requirement for a scoring function that models cross-document interactions. We propose a novel self-attentive Document Interaction Network (*attn-DIN*) that extends any univariate scoring function to combine query-document features with contextual cross-document features generated from a self-attention mechanism [17], and show that it not only satisfies the permutation equivariance requirement, but also applies to the ranking setting where queries may have varying number of documents. We conduct our experiments on four ranking datasets: benchmarks WEB30K and Istella, a Gmail search dataset, and a Google Drive Quick Access dataset. The first three are in a search setting, and the last one is in a recommendation setting. On all of them, our proposed method[1] significantly improves over neural network baselines.

## 2 RELATED WORK

Most of the previous work in LTR [11] focuses on designing loss functions, ranging from pointwise to pairwise to listwise ones. Gradient Boosted Decision Trees (e.g., [10]) are regarded as the state-of-the-art models for LTR on benchmark datasets. Recently, neural network based models have attracted considerable attention [8, 12].

There are two settings for modeling cross-document interactions: *re-ranking* and *full ranking*. In the former setting, a base ranking is provided and the documents are reordered using the ranker. For example, [1] applies sequence modeling on the top $k$ documents of the base ranking and then uses the final state vector to enrich each document for the re-ranking scoring. In the latter setting, we do not have a base ranking but start with a set of documents. For example,

[1] The implementation will be open-sourced at https://github.com/tensorflow/ranking.


*ICTIR '20, September 14–17, 2020, Virtual Event, Norway*

ACM ISBN 978-1-4503-8067-6/20/09.
https://doi.org/10.1145/3409256.3409819

RankProb [6] takes a pair of documents as input and uses a DNN to produce a preference score for the input documents, while Ai et al. [2] propose a groupwise scoring function to model document interactions, sampling a subset of all permutations of each group and thus not guaranteeing permutation equivariance.

Using attention mechanisms for ranking has been explored by several works. Romeo et al. [16] and Wang and Klabjan [18] use RNN-based approaches, whereas this paper uses Transformer self-attention [17]. AttRN [18] also uses an attention mechanism to capture listwise interactions, but the final ranking is generated by selecting documents one by one sequentially, making it sensitive to the input document order, and hence not applicable to full set ranking. SetRank [13] is a recent approach that uses attention for cross-document interactions, and is subsumed by our method.

## 3 PERMUTATION EQUIVARIANCE

In LTR, each training example consists of query $q$ and a list of documents $\boldsymbol{d}$ to be ranked w.r.t their relevance to $q$. Let $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$ be a training data set where $\boldsymbol{x} := (q, \boldsymbol{d})$, and $\boldsymbol{y}$ represents the relevance labels for $\boldsymbol{d}$. A scoring function $s : \mathcal{X}^n \rightarrow \mathbb{R}^n$ maps $\boldsymbol{x}$ to a vector of scores $\hat{\boldsymbol{y}}$. We overload $s$ to take $\boldsymbol{x}$ or equivalently $q$ and $\boldsymbol{d}$ as inputs.

$$\hat{\boldsymbol{y}} = s(\boldsymbol{x}) = s(q, \boldsymbol{d}). \tag{1}$$

A loss function $\ell(.)$ can be defined between the predicted scores and the labels. The LTR training objective is to find a scoring function $s^*$ that minimizes the empirical loss over the training data:

$$s^* = \underset{s:\mathcal{X}^n \rightarrow \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}} \ell(\boldsymbol{y}, s(\boldsymbol{x})). \tag{2}$$

*Definition 3.1 (permutation equivariant functions).* Let $\boldsymbol{x} \in \mathcal{X}^n$ be a vector of $n$ items to be ranked. Let $\pi \in \Pi^n$ be a permutation of indices of $[1, ..., n]$ which represents the rank of the items, where $\Pi^n$ is the universe of bijections from $[1, ..., n]$ to itself. A function $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ is *permutation equivariant* iff

$$f(\pi(\boldsymbol{x})) = \pi(f(\boldsymbol{x})).$$

For such a function, a permutation applied to the input vector $\boldsymbol{x}$ will result in the same permutation applied to the output of the function. Thus, a scoring function $s(\boldsymbol{x}) : \mathbb{R}^{k \times n} \rightarrow \mathbb{R}^n$ (where $k$ is the dimension of query-document pair vector representation, and $n$ is the number of scored documents) is permutation equivariant iff

$$s(\pi(\boldsymbol{x})) = s(q, \pi(\boldsymbol{d})) = \pi(s(q, \boldsymbol{d})) = \pi(s(\boldsymbol{x})).$$

## 4 DOCUMENT INTERACTION NETWORK

We propose a novel model, Document Interaction Network (*attn-DIN*), to extend a univariate scoring function with features based on self-attention mechanism to capture cross-document interactions, and show that it satisfies the permutation equivariance requirement.

### 4.1 Self-Attention Layers

Our main building blocks are self-attention layers. Let $D \in \mathbb{R}^{n \times k}$ be an input matrix corresponding to $n$ documents represented by $k$- dimensional vectors. The attention layer in Transformer [17] is defined based on three projection matrices: $W^Q \in \mathbb{R}^{k \times h}, W^K \in \mathbb{R}^{k \times h}, W^V \in \mathbb{R}^{k \times k}$ (where $h$ is the projection size), that project $D$ into a query[2] matrix $Q = DW^Q$, a key matrix $K = DW^K$ and a value matrix $V = DW^V$. At a high level, attention is a pooled combination of value output $V$ across documents, weighted by pairwise scaled-dot product similarity matrix $A(D)$ between query matrix $Q$ and key matrix $K$:

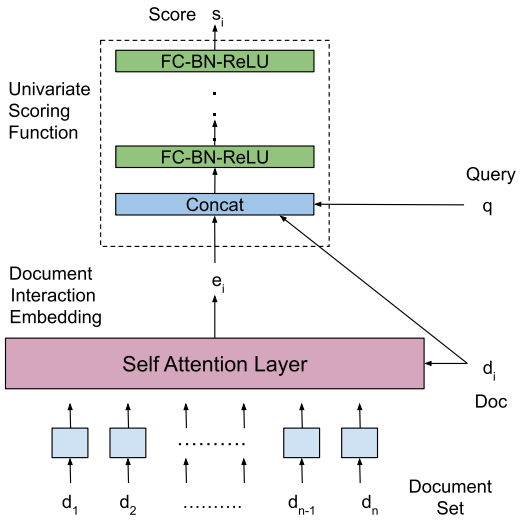


**Figure 1: Self-Attentive Document Interaction Network.**

$$A(D) := \frac{QK^T}{\sqrt{h}} \tag{3}$$

Using these weights, a self-attention layer computes a weighted sum of $V$, for any document for index $i \in [n]$ and $j \in [k]$ as follows:

$$\text{SA}(D)_{i,j} = \sum_{q=1}^{n} \frac{\exp(A(D)_{i,q})}{\sum_{p=1}^{n} \exp(A(D)_{i,p})} V_{q,j} \tag{4}$$

Multi-headed self-attention, first explored in Transformer [17], shows that having multiple heads, which attend on different parts of the input, can be beneficial. For $m$ heads, the output of several self-attention layers per head are concatenated and projected using a linear transformation using matrices $W_{out} \in \mathbb{R}^{mk \times k}$ and bias term $b_{out} \in \mathbb{R}^k$ to form the output $MHSA(D) \in \mathbb{R}^{n \times k}$:

$$\text{MHSA}(D) := \text{concat}_{j \in [m]}[\text{SA}_j(D)]W_{out} + b_{out} \tag{5}$$

It will be shown in Sec. 4.3 that such an attention mechanism is permutation equivariant. Additionally, we apply residual connections and layer normalization [17]. These are element-wise operations and preserve the permutation equivariant property.

We note that such a self-attention mostly takes the pairwise document interactions. Since permutation equivariant property is preserved for function composition $(G \circ F)(\boldsymbol{x}) = G(F(\boldsymbol{x}))$, we can stack multiple self-attention layers to capture higher-order interactions.

### 4.2 Scoring Function

Our goal is to derive a permutation equivariant multivariate scoring function that captures cross-document interactions. We propose a "wide and deep" scoring function that extends a univariate scoring

[2]Please note that the query here is different from the search query $q$.

function $s(.)$ to combine self-attention output with query and document features through concatenation:

$$s_{DIN}(q, \boldsymbol{d}) := s(q, \text{concat}(\boldsymbol{d}, \text{MHSA}(\boldsymbol{d}))). \quad (6)$$

In this architecture, the output of a stack of self-attention layers (the "deep" part) is combined in a "wide" univariate scoring function with query and document features to generate scores. We refer to this scoring method as Document Interaction Network (*attn-DIN*), and show the scoring architecture for a single document $d_i$ in Figure 1.

Our model handles varying result set sizes by padding $d$ to a maximum set size of $n$ and masking out the padding in the pooled weighted combination of value output in self-attention.

## 4.3 Theoretical Analysis

To show that *attn-DIN* is permutation equivariant, we start by showing that self-attention is permutation equivariant. Since $A(D)$ is computed by pairwise scaled dot product attention (see Eq. 3), it follows that $A(\pi(D))_{i,j} = A(D)_{\pi(i),\pi(j)}$. Applying the permutation while computing self-attention (SA) for any $i \in [n]$ and $j \in [k]$,

$$\begin{aligned}
\text{SA}(\pi(D))_{i,j} &= \sum_{q=1}^{n} \frac{\exp(A(\pi(D))_{i,q})}{\sum_{p=1}^{n} \exp(A(\pi(D))_{i,p})} \pi(V_{q,j}) \\
&= \sum_{q=1}^{n} \frac{\exp(A(D)_{\pi(i),\pi(q)})}{\sum_{p=1}^{n} \exp(A(D)_{\pi(i),\pi(p)})} V_{\pi(q),j} \\
&= \sum_{q'=1}^{n} \frac{\exp(A(D)_{\pi(i),q'})}{\sum_{p'=1}^{n} \exp(A(D)_{\pi(i),p'})} V_{q',j} \\
&= \text{SA}(D)_{\pi(i),j} = \pi(\text{SA}(D)_{i,j}).
\end{aligned}$$

Hence, $\pi(\text{SA}(D)) = \text{SA}(\pi(D))$. This proves that self-attention is permutation equivariant. As MHSA is concatenation of several self-attention layers followed by a linear projection, it follows that MHSA is permutation equivariant. Additionally, the univariate scoring function and concat(.) are permutation equivariant trivially. Using these, we can then show that the "wide and deep" scoring function (denoted as $s_{DIN}$) is permutation equivariant:

$$\begin{aligned}
& s_{DIN}(q, \pi(\boldsymbol{d})) \\
&= s(q, \text{concat}(\pi(\boldsymbol{d}), \text{MHSA}(\pi(\boldsymbol{d})))) && \text{(using Eq. 6)} \\
&= s(q, \text{concat}(\pi(\boldsymbol{d}), \pi(\text{MHSA}(\boldsymbol{d})))) && \text{(using MHSA is PE)} \\
&= s(q, \pi(\text{concat}(\boldsymbol{d}, \text{MHSA}(\boldsymbol{d})))) && \text{(using concat is PE)} \\
&= \pi(s(q, \text{concat}(\boldsymbol{d}, \text{MHSA}(\boldsymbol{d})))) && \text{(using s(q, .) is PE)} \\
&= \pi(s_{DIN}(q, \boldsymbol{d})) && \text{(using Eq. 6)}
\end{aligned}$$

The recently proposed SetRank [14] can be seen as a special case of *attn-DIN*, where only the deep cross-document attention is passed to the scoring function. *attn-DIN* is more general: it augments query-document features with these contextualized features and supports context features (user, session, query) in the scoring function via $q$.

# 5 EXPERIMENTS

## 5.1 Datasets

For our experiments, we use two public learning-to-rank datasets with numerical features, and two large-scale proprietary datasets.

[3]We discard queries with no relevant documents, similar to evaluation in [4].

**Table 1: Comparison of NDCG[3] between various ranking models on the Web30K and Istella datasets. $\triangle/\triangledown$ indicate statistically significant increase/decrease of *attn-DIN* compared to best neural ranking baseline (p-value<0.05).**

| (a) WEB30K | NDCG@1 | NDCG@5 | NDCG@10 |
|---|---|---|---|
| LambdaMART (RankLib) | 0.4535 | 0.4459 | 0.4646 |
| LambdaMART (lightGBM) | 0.5057 | 0.4991 | 0.5183 |
| LambdaMART + DLCM [1] | 0.4630 | 0.4500 | 0.4690 |
| GSF(m=64) with Softmax loss [2] | 0.4421 | 0.4446 | 0.4677 |
| FFNN with $\mathbb{E}$[ApproxNDCG] [3] | 0.4951 | 0.4820 | 0.4996 |
| SetRank with Softmax Loss [14] | 0.4904 | 0.4885 | 0.5101 |
| attn-DIN with Softmax Loss | 0.5005$^{\triangle}$ | 0.5014$^{\triangle}$ | 0.5218$^{\triangle}$ |
| (b) Istella | NDCG@1 | NDCG@5 | NDCG@10 |
| LambdaMART (RankLib) | 0.6571 | 0.6118 | 0.6591 |
| LambdaMART (lightGBM) | 0.7264 | 0.6883 | 0.7356 |
| LambdaMART + DLCM [1] | 0.6272 | 0.5848 | 0.6310 |
| FFNN with Softmax Loss | 0.6645 | 0.6422 | 0.6962 |
| SetRank with Softmax Loss [14] | 0.6702 | 0.6419 | 0.6958 |
| attn-DIN with Softmax Loss | 0.6747 | 0.6455$^{\triangle}$ | 0.6999$^{\triangle}$ |

*WEB30K & Istella.* WEB30K comprises of 30K queries with 136 dense features per query-document pair, and similarly Istella full dataset comprises of 33K queries with 220 dense features, both labeled with relevance judgments from 0 (not relevant) to 4 (highly relevant). We use train, validation and test split provided for the datasets (Fold1 for WEB30K). Each query has a variable number of documents, and we use at most 200 documents per query while training baseline and proposed methods, but consider all documents during evaluation.

*Quick Access.* In Google Drive, Quick Access is a zero-state recommendation system, surfacing relevant documents for easy user access. The features are all dense and each session has up to 100 documents, with user clicks as relevance labels. Around 30 million recommended documents are collected, with a 90%-10% train-test split.

*Gmail Search.* A list of up-to 6 emails is considered as the candidate set for each query, and the clicks are used as the relevance labels. To preserve privacy, we remove personal information, and apply $k$-anonymization. Around 200 million queries are collected, with a 90%-10% train-test split. The features comprise of both dense features and sparse character and word level n-gram features.

## 5.2 Baselines

On the public datasets, we compare *attn-DIN* with the RankLib[4] and LightGBM [10] implementations of LambdaMART, and state-of-the-art neural ranking algorithms: SetRank [14], Deep Listwise Context Model (DLCM) [1], Groupwise Scoring Functions (GSF) [2], and Feed-Forward Neural Network (with ReLU activations) with Gumbel Approximate NDCG loss [3]. We tune the hyperparameters of LightGBM and set both the number of iteration and the number of leaves to be 2,000 for WEB30K and 500 for Istella. Since the labels consist of graded relevance, for evaluation we use Normalized Discounted Cumulative Gain (NDCG) [9].

[4]https://sourceforge.net/p/lemur/wiki/RankLib/

**Table 2: Model performance on Quick Access and Gmail data. Note that ΔMRR and ΔARP denote % relative improvement. △/▽ indicate statistically significant increase/decrease compared to the best reported listwise neural ranking model [15] (p-value<0.05).**

| (a) Quick Access | ΔMRR | ΔARP |
|---|---|---|
| GSF(m=4) | -0.440 ± 0.177$^{\triangledown}$ | -0.659 ± 0.141$^{\triangledown}$ |
| attn-DIN | **+0.312 ± 0.113**$^{\triangle}$ | **+0.413 ± 0.124**$^{\triangle}$ |
| **(b) Gmail Search** | **ΔMRR** | **ΔARP** |
| GSF(m=3) | +1.006 ± 0.247 | +1.308 ± 0.246 |
| attn-DIN | **+1.245 ± 0.228**$^{\triangle}$ | **+1.430 ± 0.247** |

On the proprietary datasets, we compare *attn-DIN* and the best Groupwise Ranking model with a baseline approach of listwise neural ranking model [15] (FFNN with Softmax loss). The open source implementations of LambdaMART do not scale due to the massive scale of these datasets and the heterogeneous nature of features (dense and sparse). As the labels are binary clicks, we evaluate using Mean Reciprocal Rank and Average Relevance Position [15], and only report relative percentage improvements due to the proprietary nature of these datasets.

### 5.3 Hyperparameters

We train the neural ranking models to minimize Softmax Cross-Entropy loss [5] using Adagrad [7] optimizer with a batch size of 128. which are shown to be effective for neural ranking models, and focus on the choice of a scoring function. We tune the number of self-attention layers, number of heads and number of neurons per layer and report the best models on validation split.

For WEB30K, we use two self-attention layers with 100 neurons and two heads for *attn-DIN*, and one self-attention layer with 200 neurons and one head for SetRank. For Istella, we use two self-attention layers with 200 neurons and two heads for attn-DIN, and one self-attention layer with 200 neurons and two heads for SetRank. The univariate scoring function, shown in Figure 1, comprises of an input batch normalization layer, followed by 3 feedforward fully connected layers of sizes [1024, 512, 256] with batch normalization and ReLU activations, and is used for both *attn-DIN* and SetRank. These models are trained to 250k steps with early stopping. For Gmail, we use 5 self-attention layers with 100 neurons and 4 heads and train for 10 million steps. For Quick Access, we use 3 self-attention layers with 100 neurons and 5 heads and train for 5 million steps.

### 5.4 Model Effectiveness

In Table 1, we compare the proposed (*attn-DIN*) approach with the baselines outlined in Section 5.2. For feature scaling, we apply a $sign(x)\, log(1 + |x|)$ transformation on the features for WEB30K and Istella datasets as it is shown to be effective [19] on web-related data sets. On public datasets, we observe that the proposed approach significantly outperforms neural network baselines, and is competitive with state-of-the-art gradient boosted decision trees on WEB30K. On the Quick Access dataset (Table 2(a)), we analyze the relative improvements in MRR, and observe that the proposed approach does significantly better than the univariate model, while the GSF models fail to produce any improvements from cross-document interactions. On the Gmail dataset (Table 2(b)), the proposed approach is significantly better than the univariate model, and is superior to the best GSF model ($m = 3$). For both datasets, we observe a statistically significant improvement in relative MRR over both the best reported listwise ranking model [15] as well as the best GSF model [2].

## 6 CONCLUSION

In this paper, we study cross-document interactions for learning-to-rank. We propose the permutation equivariance requirement for a scoring function that considers document interactions. We further show that (a) self-attention mechanism can be used to implement such a permutation equivariant function, and (b) any univariate scoring function can be extended to capture cross-document interactions using the proposed self-attention mechanism. We conduct experiments on four datasets. The results show that our proposed methods can effectively capture document interactions, outperform state-of-the-art neural ranking models, and are competitive with state-of-the-art tree based models on the WEB30K dataset.

## REFERENCES


[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. Learning a deep listwise context model for ranking refinement. In *SIGIR*, pages 135–144, 2018.

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Mike Bendersky, and Marc Najork. Learning groupwise multivariate scoring functions using deep neural networks. In *ICTIR*, pages 85–92, 2019.

[3] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. A stochastic treatment of learning to rank scoring functions. In *WSDM*, pages 61–69, 2020.

[4] Sebastian Nima Bruch, Masrour Zoghi, Mike Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In *SIGIR*, pages 1241–1244, 2019.

[5] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[6] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *SIGIR*, pages 65–74, 2017.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, July 2011.

[8] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *arXiv:1903.06902*, 2019.

[9] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*, pages 3146–3154, 2017.

[11] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[12] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *WWW*, pages 1291–1299, 2017.

[13] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *AAAI*, pages 2793–2799, 2016.

[14] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. Setrank: Learning a permutation-invariant ranking model for information retrieval. *arXiv:1912.05891*, 2019.

[15] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. TF-Ranking: Scalable tensorflow library for learning-to-rank. In *KDD*, pages 2970–2978, 2019.

[16] Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeno, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. Neural attention for learning to rank questions in community question answering. In *COLING*, pages 1734–1745, 2016.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[18] Baiyang Wang and Diego Klabjan. An attention-based deep net for learning to rank. *arXiv:1702.06106*, 2017.

[19] Honglei Zhuang, Xuanhui Wang, Mike Bendersky, and Marc Najork. Feature transformation for neural ranking models. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1649–1652, 2020.