Self-Attentive Document Interaction Networks for Permutation Equivariant Ranking

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

ABSTRACT

How to leverage cross-document interactions to improve ranking performance is an important topic in information retrieval (IR) research. However, this topic has not been well-studied in the learning-to-rank setting and most of the existing work still treats each document independently while scoring. The recent development of deep learning shows 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 and show that it satisfies the permutation-equivariant 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 three ranking datasets: the benchmark Web30k, a Gmail search, and a Google Drive Quick Access dataset. Experimental results show that our proposed methods are both more effective and efficient than baselines.

KEYWORDS

Learning to rank; document interaction networks, self-attention

1 INTRODUCTION

Ranking is a central problem in many applications of information retrieval (IR) such as search, recommender systems, and question answering. The purpose of a ranking algorithm is to sort a set of items into a ranked list such that the utility of the entire list is maximized. For example in search, a set of documents are to be ranked to answer a user’s query. The utility of the entire list highly depends on the top ranked documents.

Learning-to-rank employs machine learning techniques to solve ranking problems. The common formulation is to find a function that can produce scores for the list of documents of a query. The scores can then be used to sort the documents. Many early attempts to learning-to-rank cast a ranking problem as regression or classification [10, 27]. In such methods, the loss function being minimized incurs a cost for an incorrect prediction of relevance labels ("pointwise" loss) or pairwise preferences ("pairwise" loss). Such formulations are, however, misaligned with the ranking objective where the utility is often defined over the entire list of documents. Indeed, the so called “listwise” methods that optimize a loss function defined over the entire list have been shown to learn better ranking functions [9, 11, 45].

While much research has been devoted to the evolution of loss functions, 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 the motivation behind several previous works [1, 2, 4, 15, 16, 38]. Early methods such as the score regularization technique [16] and the conditional random field based models [38] use the similarity between documents to smooth or regulate ranking scores. These methods, however, assume the existence of document similarity information from another source such as document clusters. More recently, neural learning-to-rank algorithms [1, 4] and click models [5] capture document interactions using recurrent neural networks over document lists. These methods, however, belong to the re-ranking setting because they assume that the input is an ordered list, but not a set.

Another work that investigates the effect of document interactions on ranking quality is RankProb [15]. It is a bivariate neural scoring function that takes a pair of documents as input and predicts the preference of one over the other. More recently, a more general framework was proposed in [2] to learn multivariate “groupwise” scoring functions (GSFs). Though being able to model document interactions, both methods are highly inefficient at inference time. These models suffer from a training-serving discrepancy: the function learned during training is different from the scoring function used in serving. For example, average pooling over the bivariate function learned during training is used as the scoring function in RankProb during serving. For higher-order interaction models (such as GSFs), the pooling is over an intractable number of permutations, and hence, approximation via sub-sampling is used, which worsens the training-serving discrepancy and makes the inference unstable.

In this paper, we identify a generic requirement for scoring functions with document interactions: permutation-equivariance. We analyze the existing approaches with respect to this requirement. Based on the this requirement, we propose a class of neural network models and show that they not only satisfy this requirement precisely, but are also more efficient in modeling document interactions and do not have the training-serving discrepancy. Our proposed method is based on self-attention on the document level. It naturally captures the cross-document interactions via the self-attention mechanism. To the best of our knowledge, our work is the first to use it to model document interactions for learning-to-rank.

Our contributions can be summarized as follows:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

arXiv:1910.09676v2 [cs.LG] 23 Oct 2019
• We propose the permutation-equivariance requirement for any document interaction model and analyze existing methods with respect to this requirement.
• We identify a generic class of permutation-equivariant functions, instantiate it using a self-attentive document interaction network, and incorporate it into learning-to-rank.
• We empirically demonstrate the effectiveness and efficiency of our proposed methods on both search and recommendation tasks using three data sets.

This paper is organized as follows. We begin with a review of the literature in Section 2, and formalize the problem we wish to study in Section 3. In Section 4, we present a detailed description of our proposed methods. We examine the effectiveness of our methods empirically and summarize our findings in Section 5. Finally, we conclude this work and discuss future directions in Section 6.

2 RELATED WORK
In learning-to-rank literature, a common approach is called “score and sort”. For capturing the loss between the list of scores for documents and relevance labels, pointwise [12, 19], pairwise [8, 10] and listwise [7, 46, 47] losses have been extensively studied. Scoring functions have been parameterized by boosted decision trees [29], SVMs [26], and neural networks [10, 35, 36].

In the context of scoring query-document pairs, the recent neural ranking models have been broadly classified [20] into two categories: representation-focused and interaction-focused. The methods that are representation-focused [22, 34, 35] look at learning optimal vector space representations for queries and documents, and then combine them using dot product or cosine similarity. The interaction-focused methods learn a joint representation based on interaction networks between queries and documents. These approaches, along with hybrid variants between representation and interaction focused [32], are univariate approaches, i.e., they deal with scoring a query-document pair, and do not capture cross-document interactions. Please note that attention mechanism [41] has also been explored in this line of work, but it is mainly used in the word or paragraph level, not the document level.

Recent work in modeling document interactions in learning-to-rank have focused on the re-ranking scenario [1, 4, 37], where the input is an ordered list of documents, not a set. These are not applicable to full set ranking, which is the focus of our work. Regularizing scores [16], and a CRF approach [38] using document cluster information to augment the training loss have been explored, which are complementary to our proposed approach.

3 PROBLEM FORMULATION
In this section, we formulate our problem in the learning-to-rank setting.

3.1 Learning-to-Rank
Learning-to-rank solves ranking problems using machine learning techniques. In such a setting, the training data consists of a set of queries with each having a list of documents that we would like to rank. Formally, let \( D = \{(q, d, y)\} \) be a training data set where \( q \) is a query, \( d \) is the list of documents for \( q \), and \( y \) is the relevance labels for \( d \). We use \( d_i \) and \( y_i \) to refer to the \( i \)-th elements in \( d \) and \( y \) respectively. A scoring function \( s \) takes both \( q \) and \( d \) as input and computes a vector of scores \( \hat{y} \):

\[
\hat{y} = s(q, d).
\] (1)

A loss function \( \ell \) for query \( q \) can be defined between the predicted scores and the labels:

\[
\ell(q, d, y) = \ell(y, \hat{y}).
\]

The goal of a learning-to-rank task is to find a scoring function \( s^* \) that minimizes the empirical loss over the training data:

\[
s^* = \arg\min_{s \in \mathcal{H}} \frac{1}{|D|} \sum_{(q, d, y) \in D} \ell(q, d, y).
\] (2)

Typical examples of the hypothesis space \( \mathcal{H} \) for a scoring function \( s \) are support vector machines [27, 28], boosted weak learners [48], gradient-boosted trees [10, 18], and neural networks [8].

3.2 Ranking Loss Functions
Given a formulation of the scoring function, there are various definitions of ranking loss functions [31]. In this paper, we focus on the following two listwise loss functions as they have been shown to be closely related to the ranking metrics [6, 7, 39]. The first one is the Softmax Cross-Entropy loss (denoted as Softmax) and has been shown to be a proper bound of ranking metrics over binary relevance labels like MRR [6]:

\[
\ell(y, \hat{y}) = \sum_i \frac{y_i}{\sum_{i'} y_{i'}} \log \left( \frac{\exp(\hat{y}_i)}{\sum_{i'} \exp(\hat{y}_{i'})} \right).
\] (3)

where the subscript \( i \) and \( i' \) means the \( i \)-th and \( i' \)-th element in a vector.

The second one is the Approximate NDCG loss (denoted as ApproxNDCG) [7, 40]. It is more suitable for graded relevance labels, and is derived from the NDCG metric, but uses scores to approximate the ranks to make the objective smooth:

\[
\ell(y, \hat{y}) = \frac{1}{\text{DCG}(y)} \sum_i 2^{y_i} - 1 \log(1 + \hat{r}_i),
\] (4)

where \( \text{DCG}(y) \) is the normalization term of NDCG and \( \hat{r}_i \) is the approximate rank defined as

\[
\hat{r}_i = 1 + \sum_{i': i' \neq i} \frac{\exp(-\eta \hat{y}_i)}{\exp(-\eta \hat{y}_{i'}) + \exp(-\eta \hat{y}_{i'})}.
\]

where \( \eta \) is the parameter that controls the closeness of the approximation. When \( \hat{r}_i \) is replaced by the rank \( r_i \) sorted by scores \( y_i \), Equation 4 becomes the NDCG metric. A larger \( \eta \) makes \( \hat{r}_i \) closer to \( r_i \), but it also makes ApproxNDCG less smooth and thus harder for optimization. We tune \( \eta \) in our experiments and set \( \eta = 0.1 \) since it gives the optimal results.

3.3 Permutation-Equivariance Requirement
Our focus in this paper is on scoring functions. We postulate that it is preferable that the scoring function is permutation equivariant, so that the resulting ranking will be independent of the original ordering induced over the items by the candidate generation process (e.g., a base ranker, or a retrieval mechanism). This ensures
that the learned ranker will not be biased by any errors in the candidate generation process. We first give the general mathematical definition of permutation-equivariant functions.

**Definition 3.1 (Permutation-Equivariant Functions).** Let \( x \) be a vector of elements \([x_1, \ldots, x_n]\), where \( x_i \in X \), and \( \pi \) is a permutation of indices \([1, \ldots, n]\). A function \( f : X^n \rightarrow Y^n \) is permutation-equivariant iff

\[
f(\pi(x)) = \pi(f(x)).
\]

That is, a permutation applied to the input vector will result in the same permutation applied to the output of the function.

For a scoring function \( s(q, d) \), the input domain \( X \) is defined by the representation of \( q \) and \( d_i \) (e.g., \( \mathbb{R}^{k_q + k_d} \)) where \( k_q \) and \( k_d \) are the dimension of their vector representation) and the output domain is \( Y = \mathbb{R} \). It is permutation-equivariant iff

\[
s(q, \pi(d)) = \pi(s(q, d)).
\]

We analyze some existing work in term of this requirement.

The vast majority of learning-to-rank algorithms assume a univariate scoring function that computes a score for each document independently of others. With slight abuse of notation, we also use \( s \) to represent the scoring function on each individual document:

\[
\hat{y}_i = s(q, d_i)
\]

where \( d_i \) is an individual document in the list \( d \) and \( \hat{y}_i \) is the \( i \)-th value of the score vector \( \hat{y} \). A univariate scoring function is permutation-equivariant because

\[
\pi(s(q, d)) = \pi([s(q, d_1), s(q, d_2), \ldots]) = [s(q, d_{\pi(1)}), s(q, d_{\pi(2)}), \ldots] = s(q, \pi(d)).
\]

The Groupwise Scoring Functions (GSFs) [2] boil down to univariate scoring functions when the group size is 1. A larger group size is needed to model cross-document interactions. For example, for groups of size 2, the scoring of the \( i \)-th document is:

\[
\hat{y}_i = \frac{1}{2(n-1)} \sum_d g(q, d_i, d_j) + g(q, d_j, d_i),
\]

where \( g \) is the sub-scoring function in GSF and is implemented using feed forward networks. Higher-order interactions are explicitly captured when the group size is larger, but it becomes impractical to implement a GSF precisely due to the combinatorial number of groups. Monte Carlo sampling methods are used to approximate and this can make GSFs unstable. In this sense, GSFs are approximately permutation-equivariant.

The RankProb approach in [15] trains a bivariate interaction scoring function \( g(q, d_i, d_j) \) by concatenating the features as the input for a feed forward network. The loss function is a logistic regression on the pairwise preference of the two documents. For inference, it uses the average pooling in Equation 6. This model is similar to the GSFs with group size 2. It has a training-serving discrepancy. The average pooling makes the scoring function permutation-equivariant but directly using it has a \( O(n^2) \) time complexity, which is not scalable.

![Figure 1: Self-Attentive Document Interaction Network.](image)

#### 4 PROPOSED METHODS

In this section, we first present a general class of permutation-equivariant functions and outline how we build a permutation equivariant scoring function using deep neural networks for our proposed approach.

#### 4.1 A Class of Permutation-Equivariant Functions

Our permutation-equivariant functions are based on permutation-invariant functions. We start with the formal definition of permutation-invariant functions.

**Definition 4.1 (Permutation-Invariant Functions).** Let \( x \) be a vector of elements \([x_1, \ldots, x_n]\), where \( x_i \in X \) and \( \pi \) be a permutation of indices \([1, \ldots, n]\). A function \( f : X^n \rightarrow Y \) is permutation-invariant iff

\[
f(\pi(x)) = f(x).
\]

That is, any permutation of the input \( x \) has the same output.

The work in [49] provides a general characterization of permutation-invariant functions as follows:

**Theorem 4.2.** A function \( f(x) \) is permutation-invariant iff it can be decomposed in the form \( \rho (\sum_{x_i \in X} \phi(x_i)) \), for a suitable choice of \( \rho \) and \( \phi \).

Though simple, Theorem 4.2 is less constructive. Ilse et al. [23] proposed a mechanism to extend the form in Theorem 4.2 (called pooling function) to a weighting pooling, based on the attention mechanism. We shall refer this as attention pooling function. Given a generic context \( c \), a pooling function can be extended to attention pooling as follows:

\[
f(x; c) = \rho \left( \sum_{x_i \in X} \alpha(x_i, c) \phi(x_i) \right)
\]

Here, \( \alpha(\) is the popular attention mechanism, which is used to capture the similarity between the context and the item.
We instantiate the permutation-equivariant functions using the $W$ features:

to document $d$ and $s$, equivariant scoring function whose output is $s(d, q)$ and potentially capture higher-order interactions between document interactions. Since permutation-equivariance is preserved for function composition $G \circ F(x) = G(F(x))$, we can stack multiple self-attention layers. Multiple layers can enhance and potentially capture higher-order interactions better.

4.2.2 Scoring Layers. However, our goal is to derive a permutation-equivariant scoring function whose output is $Y^n = \mathbb{R}^n$. We propose to use a univariate scoring function $s$ on top of the output of self-attention layers. Let $F(q, d)$ be the output of self-attention layers and $e(d_i; d) = F(q, d)_i$ be the $i$-th row of the output, corresponding to document $d_i$. We propose a "wide and deep" scoring function to combine self-attention based features with query and document features:

$$\hat{y}_i = s(q, d_i, d) = s(q, d_i, e(d_i; d))$$

We refer to this as a "wide and deep" architecture, where the output of "deep" layers, a stack of self-attention layers, is combined in a "wide" univariate scoring function with query and document features to generate a score per document.

We show that this "wide and deep" scoring function (denoted as $s_{DI N}$) is still permutation-equivariant, while capturing cross-document interactions.

$$\pi(s_{DI N}(q, d)) = \pi([s(q, d_1, e(d_1; d)), s(q, d_2, e(d_2; d)), ...])$$

$$= [s(q, d_{\pi(1)}, e(d_{\pi(1)}; d)), s(q, d_{\pi(2)}, e(d_{\pi(2)}; d)), ...]$$

$$= [s(q, d_{\pi(1)}, F(q, d)_{\pi(1)}), s(q, d_{\pi(2)}, F(q, d)_{\pi(2)}), ...]$$

$$= s_{DI N}(q, \pi(d))$$

We call our method Self-Attentive Document Interaction Networks (denoted as att-DIN) and the structure of the score for a given document is shown in Figure 1. The self-attention layer can be stacked sequentially, without losing the permutation equivariance property. In "wide and deep" fashion, the output of this layer, document interaction embeddings, is combined with query and document features and fed as an input to a univariate scoring function. The specific univariate scoring function captures interactions between features using a stack of feedforward layers. Specifically, for each feedforward layer, the input is passed through a dropout regularization [42] (to prevent overfitting), and the output is passed through a batch normalization layer [24], followed by a non-linear ReLU [33] activation, where $ReLU(x) = \max(x, 0)$. We refer to this combination as $FC-BN-ReLU$ in Figure 1. The final output is projected to a single score for a document.

5 EXPERIMENTS

In this section, we first outline several learning-to-rank datasets and baseline methods used in our experiments. We then report the comparisons on both model effectiveness and inference efficiency.

5.1 Datasets

5.1.1 MSLR Web30k. The Microsoft Learning to Rank (MSLR) Web30k [9] public dataset comprises of 30,000 queries. We use Fold1, which has 3 partitions: train, validation, and test. For each query-document pair, it has 136 dense features. Each query has a variable number of documents, and we use at most 200 documents per query for training baseline and proposed methods. During evaluation, we use the test partition and consider all the documents present for a query. We discard queries with no relevant documents both during training and evaluation.

5.1.2 Quick Access. In Google Drive, Quick Access [43] is a zero-state recommendation system, that surfaces relevant documents that users might click on when they visit the Drive home. The features are all dense, as described in Tata et. al. [43], and user clicks are used as relevance labels. We collect around 30 million recommended documents and their click information. Each session has up to 100 documents, along with user features as contextual information for training and evaluation. We use a 90%-10% train-test split on this dataset.

5.1.3 Gmail Search. In Gmail, we look at search over e-mails, where a user types in a query, looks for a relevant e-mail, and clicks on one of the six results returned by the system. The list
of six e-mails are considered as the candidate set, 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 and sparse features. The sparse features comprise of character and word level n-grams with k-anonymization applied.

5.2 Baselines

On the public Web30k dataset, we compare with LambdaMART’s implementation in RankLib [14] and lightGBM [29], and with multivariate Groupwise Scoring Functions [2]. Since the labels consist of graded relevance, for evaluation measures, we use Normalized Discounted Cumulative Gain (NDCG) [25] for top 1, 5, and 10 documents ordered by the scores.

On the private datasets of Quick Access and Gmail, we compare only with Groupwise Scoring Functions. Given the massive scale of the datasets, and the heterogeneous nature of features (dense and sparse), the open source implementations of LambdaMART do not scale on these datasets. Furthermore, prior work demonstrated that GSFs are superior to LambdaMART when sparse features are not scale on these datasets. Furthermore, prior work demonstrated that GSFs are superior to LambdaMART when sparse features are present [2]. We evaluate using Mean Reciprocal Rank [13] and Average Relevance Position [50], as the labels are binary clicks, for which these two measures are most suitable.

5.3 Hyperparameters

On Web30k dataset, to encode document interactions, we use one self-attention layer with 100 neurons, and with a single attention head. The univariate scoring function to combine the output of self-attention with query and document features comprises of an input batch normalization layer, followed by 7 feedforward fully-connected layers (FC-BN-ReLU) layers, as shown in Figure 1) of sizes 1024, 512, 256, 128, 64, 32, 16. The model is trained using a training batch size of 128, and Adagrad [17] optimizer, with a learning rate of 0.005 to minimize the ApproxNDCG ranking loss. We use a similar setup for Gmail and Quick Access, with Softmax loss minimized using Adagrad Optimizer, trained for 10 million and 5 million steps respectively. For Gmail, we use 5 layers of self-attention with 100 neurons each, with 4 heads for encoding document interactions. For Quick Access, we use 3 layers of self-attention with 100 neurons each, with 5 heads for encoding document interactions.

5.4 Model Effectiveness

In Table 1, on Web30k data, we compare the proposed attn-DIN approach with LambdaMART and GSFs. For LambdaMART, we consider the lightGBM implementation and the older RankLib implementation, and list the best reported metrics on test data. For the GSFs, we list the best reported metrics, and also an improved model based on our finetuning experiments. In Figure 2, we compare attn-DIN model with multivariate GSF models for varying group sizes. Since the list size is large for Web30k dataset (around 200), we increase the group size on an exponential scale from 1 to 128. We also show 95% bootstrapped confidence intervals for each of the models.

We observe that the proposed approach significantly outperforms both the best reported and finetuned GSFs, giving around 1 point improvement for NDCG@5 (measured from 0-100), which is statistically significant (+1.03 ± 0.35), measured using paired t-test with p-value threshold of 0.05. These gains are not just from using a deeper network or from more neural network parameters, as shown in Figure 3. The increase in number of parameters over the univariate scoring is the smallest for attn-DIN model, compared to any of the GSF models, while the improvement in ranking measure is significant. Our attn-DIN model tries to capture similarity using dot product attention mechanism and pooling to combine feature values, while GSFs explicitly model cross-document interactions using feedforward networks. As the group size increases, the number of parameters needed to capture cross-document interactions also increase. This also leads to increase in inference time, as discussed in Section 5.5.

The proposed approach outperforms RankLib’s LambdaMART, but not the lightGBM implementation. We believe this is due to the fact that Gradient Boosted Decision Trees are very powerful class

| Method | NDCG@1 | NDCG@5 | NDCG@10 |
|--------|--------|--------|---------|
| GSF(m=64) with Softmax (best reported [2]) | 44.21 | 44.46 | 46.77 |
| GSF(m=1) with ApproxNDCG (best reported [7]) | 46.64 | 45.38 | 47.31 |
| GSF(m=1) with ApproxNDCG (finetuned) | 46.81 | 45.59 | 47.39 |
| attn-DIN with ApproxNDCG (proposed approach) | 48.16 △ | 46.62 △ | 48.21 △ |
| LambdaMART (RankLib) | 45.35 | 44.59 | 46.46 |
| LambdaMART (lightGBM) | 50.33 | 49.20 | 51.05 |

Table 1: Comparison of NDCG between GSF models and attn-DIN for cross-document interactions on Web30k data. * indicates the best GSF model. △/▽ indicate statistically significant increase/decrease compared to best GSF model (p-value<0.05).

| (a) Quick Access | △MRR | △ARP |
|-----------------|------|------|
| GSF(m=1) * (univariate scoring) | – | – |
| GSF(m=4) | -0.440 ± 0.177 △ | -0.659 ± 0.141 △ |
| attn-DIN (proposed approach) | +0.312 ± 0.113 △ | +0.413 ± 0.124 △ |

| (b) Gmail Search | △MRR | △ARP |
|-----------------|------|------|
| GSF(m=1) * (univariate scoring) | – | – |
| GSF(m=3) * | +1.006 ± 0.247 | +1.308 ± 0.246 |
| attn-DIN (proposed approach) | +1.245 ± 0.228 △ | +1.430 ± 0.247 |

Table 2: Model performance on Quick Access and Gmail data. Note that △MRR and △ARP denote % relative improvement. The best performance per column is in bold. * indicates the best GSF model. △/▽ indicate statistically significant increase/decrease compared to the best GSF model (p-value<0.05).
of machine learning models when the feature set consists of purely dense features, and smaller training datasets.

In most real-world scenarios, input features tend to have both dense and sparse features. Query, document titles and metadata tend to naturally have textual description, which play a key role during user’s relevance judgment, and hence are powerful signals for training ranking models. We look at two real-world datasets, on Gmail Search and Quick Access, with a large amount of data and a variety of features, as described in Section 5.3. In Table 2, we report relative improvements in MRR, due to private nature of these datasets. For statistical significance, we use paired t-test over relative improvements in MRR, with p-value threshold of 0.05.

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, which is in fact, the best performing GSF model. While the GSF models fail to produce any improvements from cross-document interactions on this dataset, our proposed approach effectively captures them.

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). We conducted a paired t-test between attn-DIN and the best GSF model, and we observe a relative improvement in MRR, $+0.237 \pm 0.206$, which is a statistically significant improvement. Note that in Gmail, we consider smaller document candidate sets (6 document per query), whereas in Web30k and and Quick Access, we use much larger candidate sets (200 documents per query and 100 documents per user request, respectively). For larger group sizes (≥ 8), the performance of GSF models deteriorates, whereas the proposed approach is able to capture cross-document interactions effectively.

5.5 Model Efficiency

In Figure 3, we compare the inference time and number of parameters for various GSF models, normalized with the value for the proposed approach. Over univariate scoring functions, the proposed approach has an increase in inference time similar to GSF model of group size 8, despite capturing interactions across the entire document set of sizes 200 for Web30k. For the GSF models, the inference time increases with the increase of group size. GSFs use an approximation during inference. For group size 2, it uses a rolling window of 2 over a shuffled list to reduce the time complexity to $O(n^2)$.

In our experiments, it takes around 2, 240 ms for inference per query, and is drastically slower than the attn-DIN approach, which takes around 13 ms for inference per query.

In Figure 3, we compare the inference time and number of parameters for various GSF models, normalized with the value for the proposed approach. Over univariate scoring functions, the proposed approach has an increase in inference time similar to GSF model of group size 8, despite capturing interactions across the entire document set of sizes 200 for Web30k. For the GSF models, the inference time increases with the increase of group size. GSFs use an approximation during inference. For group size 2, it uses a rolling window of 2 over a shuffled list to reduce the time complexity to $O(n^2)$.

However, it is not guaranteed to be permutation-equivariant, and may be unstable during inference. The exact inference is using Equation 6, the same as RankProb [15], which has $O(n^2)$ time complexity. In our experiments, it takes around 2, 240 ms for inference per query, and is drastically slower than the attn-DIN approach, which takes around 13 ms for inference per query.

For the Web30k dataset, from Figures 2 and 3, we can observe that the proposed approaches are significantly better than univariate approaches, and are faster during inference than GSF models at large group sizes while capturing cross-document interactions across the list. From Figure 3, we can also observe that the proposed model has fewer parameters than multivariate GSF models; hence the gain in ranking metrics is not from using larger number of parameters, but from effectively capturing similarity via cross-document attention pooling of the document features.

6 CONCLUSION

In this paper, we study how to leverage document interactions in the learning-to-rank setting. We proposed the permutation-equivariance requirement for a scoring function that takes document interactions into consideration. We show that self-attention mechanism can be used to implement such a permutation-equivariant function, and that any univariate query-document scoring function can be extended to capture cross-document interactions using this proposed self-attention mechanism. We choose the attention method used in Transformer [44] in our paper and combine the output of self-attention layers with a feed forward network in a wide and deep architecture. We conducted our experiments on three datasets and the results show that our proposed methods can capture document interactions effectively in a statistically significant manner, and can efficiently scale to large document sets.
[50] Mu Zhu. Recall, precision and average precision. Technical report, Department of Statistics and Actuarial Science, University of Waterloo, 2004.