Learning Groupwise Scoring Functions Using Deep Neural Networks

Qingyao Ai
CICS, UMass Amherst
Amherst, MA, USA
aiqy@cs.umass.edu

Xuanhui Wang
Google Inc.
Mountain View, CA, USA
xuanhui@google.com

Nadav Golbandi
Google Inc.
Mountain View, CA, USA
nadavg@google.com

Michael Bendersky
Google Inc.
Mountain View, CA, USA
bemike@google.com

Marc Najork
Google Inc.
Mountain View, CA, USA
najork@google.com

ABSTRACT

While in a classification or a regression setting a label or a value is assigned to each individual document, in a ranking setting we determine the relevance ordering of the entire input document list. This difference leads to the notion of relative relevance between documents in ranking. The majority of the existing learning-to-rank algorithms model such relativity at the loss level using pairwise or listwise loss functions. However, they are restricted to *pointwise scoring functions*, i.e., the relevance score of a document is computed based on the document itself, regardless of the other documents in the list. In this paper, we overcome this limitation by proposing generalized *groupwise scoring functions* (GSFs), in which the relevance score of a document is determined jointly by groups of documents in the list. We learn GSFs with a deep neural network architecture, and demonstrate that several representative learning-to-rank algorithms can be modeled as special cases in our framework. We conduct evaluation using the public MSLR-WEB30K dataset, and our experiments show that GSFs lead to significant performance improvements both in a standalone deep learning architecture, or when combined with a state-of-the-art tree-based learning-to-rank algorithm.

CCS CONCEPTS

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

KEYWORDS

Groupwise scoring functions; deep neural networks; listwise loss

ACM Reference Format:
Qingyao Ai, Xuanhui Wang, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Scoring Functions Using Deep Neural Networks. In *Proceedings of 1st International Workshop on Deep Matching in Practical Applications (DAPA'19).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.475/123_4

DAPA'19, February 2019, Melbourne, Australia

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Ranking is at the heart of many information retrieval (IR) problems, including ad-hoc document retrieval, question answering, recommender systems, and many others. Unlike the traditional classification or regression setting, the main goal of a ranking problem is not to assign a label or a value to each individual document, but to determine the relative preference among a list of them. The relevance of the top ranked documents influences the perceived utility of the entire list more than others [10]. Thus, the notion of *relativity* is crucial in a ranking problem.

How to model relativity in ranking has been extensively studied, especially in the learning-to-rank setting [25]. In this setting, a scoring function that maps document feature vectors to real-valued scores is learned from training data. Documents are then ranked according to the predictions of the scoring function. To learn such a scoring function, the majority of the learning-to-rank algorithms use pairwise or listwise loss functions to capture the relative relevance between documents [5, 7, 8, 39]. Such a loss guides the learning of the scoring function to optimize preferences between documents or an IR metric such as NDCG [6, 21, 34].

Though effective, most existing learning-to-rank frameworks are restricted to the paradigm of *pointwise scoring functions*: the relevance score of a document to a query is computed independently of the other documents in the list. This setting could be less optimal for ranking problems for multiple reasons[1]. First, it has limited power to model cross-document comparison. Consider a search scenario where a user is searching for a name of a musical artist. If all the results returned by the query (e.g., *calvin harris*) are recent, the user may be interested in the latest news or tour information. If, on the other hand, most of the query results are older (e.g., *frank sinatra*), it is more likely that the user wants to learn about artist discography or biography. Thus, the relevance of each document depends on the distribution of the whole list. Second, user interaction with search results shows strong comparison patterns. Prior research suggests that preference judgments by comparing a pair of documents are faster to obtain, and are more consistent than the absolute ratings [3, 24, 40]. Also, better predictive capability is

[1] It is interesting to note the connection of pointwise scoring to Robertsons' probability ranking principle (PRP) [33]. PRP states that documents should be ranked by their probability of relevance to the query. While highly cited and influential, this principle was recognized to be problematic by Robertson himself: "The PRP works document-by-document, whereas the results should be evaluated request-by-request" [33].

achieved when user actions are modeled in a relative fashion (e.g., SkipAbove) [4, 10, 11, 14, 23, 41]. These indicate that users compare the clicked document to its surrounding documents prior to a click, and a ranking model that uses the direct comparison mechanism can be more effective as it mimics the user behavior more faithfully.

In this paper, we hypothesize that the relevance score of a document to a query should be computed by comparison with other documents at the feature level. Specifically, we explore a general setting of *groupwise scoring functions* (GSFs) for learning-to-rank. A GSF takes multiple documents as input and scores them together by leveraging a joint set of features of these documents. It outputs a *relative relevance score* of a document with respect to the other documents in a group, and the final score of each document is computed by aggregating all the relative relevance scores in a voting mechanism. The proposed GSF setting is general as we can define arbitrary number of input documents and any ranking loss in it. We show that several representative learning-to-rank models can be formulated as special cases in the GSF framework.

While it is easy to define a GSF, it is unclear how to learn such a function from training data efficiently, as well as how to use it to score a new list of documents during inference remains a challenge. To solve these challenges, we propose a novel deep neural network architecture in the learning-to-rank setting. We design a sampling method that allows us to train GSFs efficiently with back-propagation, and obtain a scalar value per document during inference using a Monte Carlo method.

We evaluate our proposed models in a standard learning-to-rank setting. Specifically, we evaluate GSFs using the LETOR data set MSLR-WEB30K [30], in which the relevance judgments are obtained from human ratings, and a pre-defined feature set is provided. We show that our GSFs perform reasonably well by themselves and can improve the state-of-the-art tree-based models in a *hybrid* setting where the output of GSFs is used as a feature for tree-based models.

2 RELATED WORK

Learning-to-rank refers to algorithms that try to solve ranking problems using machine learning techniques. There is a plethora of learning-to-rank work [5, 6, 8, 16, 22, 37], which mainly differs in its definitions of loss functions. Almost all of them use the setting of pointwise scoring functions. To the best of our knowledge, there are only a few exceptions.

The first ones are the score regularization technique [13] and the CRF-based model [31] that take document similarities to smoothe the initial ranking scores or as additional features for each query-document pair. When computing ranking scores, however, they take only one document at a time and do not consider comparison between features from different documents.

The second exception is a pairwise scoring function that takes a pair of documents together and predicts the preference between them [12]. We demonstrate that this pairwise scoring function can be instantiated in the proposed groupwise framework, and compare against it in our experiments.

The third exception is the neural click model [4] and the deep listwise context model [2] that builds an LSTM model on top of document lists. The neural click model summarizes other documents into a hidden state and eventually uses a pointwise loss (e.g., log loss) for each document. Such a pointwise loss is difficult to adapt to the unbiased learning-to-rank setting [35], while our GSF models can. The deep listwise context model is similar to our models in terms of loss functions, but our GSF is more flexible as it can take any document lists as its inputs, whether ordered or not.

Search result diversification is related to our GSF models since it also takes into account subsets of documents through maximizing objectives such as maximal marginal relevance [9] or subtopic relevance [1]. Recently, several deep learning algorithms were proposed, with losses corresponding to these objectives [20, 38]. In contrast to this work, the goal of our paper is to improve relevance modeling through groupwise comparisons, but not diversity.

Pseudo-relevance feedback [26] is a classic retrieval method that uses query expansion from the first-round top retrieved documents to improve the second-round retrieval. Our GSFs consider document relationship in the learning-to-rank setting, not in the retrieval setting, and do not require two rounds of retrieval. We also do not assume a pre-existing initial ordering of the document list.

Note that our work is complementary to the recently proposed neural IR techniques [12, 17, 27–29]. While these techniques focus on advanced representations of document and query text with the employment of standard pointwise or pairwise scoring and loss functions, our work focuses on the nature of the scoring functions and the combination of multiple feature types while employing a relatively simple matching model.

3 PROBLEM FORMULATION

In this section, we formulate our problem in the learning-to-rank framework. Let $\psi = (q, D, Y)$ represent a user query string q, its list of documents D, and their respective relevance labels Y. For each document $d \in D$, we have a relevance label $y_d \in Y$. Let Ψ be the training data set. The goal of learning-to-rank is to find a scoring function f that minimizes the average loss over the training data:

$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{\psi \in \Psi} l(\psi, f) \tag{1}$$

Without loss of generality, we assume that f takes both q and D as input, and produces a score s_d for each document $d \in D$:

$$[s_1, \ldots, s_{|D|}] = f(q, D).$$

The local loss function $l(\psi, f)$ is computed based on the relevance label list Y, and the scores produced by the scoring function f:

$$l(\psi, f) = l\big(Y, f(q, D)\big) \tag{2}$$

The main difference among the various learning-to-rank algorithms lies in how the scoring function f and the loss function l are defined. While there is much work in different types of loss functions, categorized as pointwise, pairwise or listwise loss [25], the majority of work assumes a *pointwise scoring function* (PSF) that takes a single document at a time as its input, and produces a score for every document separately:

$$f(q, D) = [f(q, d_1), ..., f(q, d_{|D|})]$$

We re-use f as the pointwise scoring function in this context for brevity.

In this paper, we explore a generic *groupwise scoring function* (GSF) to better compare documents at feature level. A GSF g takes

a group of m documents and scores them together:

$$[r_1, ..., r_m] = g(q, [d_1, ..., d_m])$$

PSF is a special case of GSF when $m = 1$. Intuitively, a requirement for GSF g is to be invariant to the input document order. However, it is not immediately clear on how to define $f(q, D)$ based on function g and how to learn g based on training data. This is what we are going to address in the next section.

4 METHODS

4.1 DNN-based Scoring Function

Because GSF takes multiple documents as an input, the feature vectors for GSF may be potentially exponentially larger than those for PSF. Therefore, the scoring function g in GSF should have good scalability and be easily applicable to different types of features with large dimensionality. In this paper, we use deep neural networks (DNNs) as the building block for constructing the scoring function g. Feed-forward DNN models have widely been used for learning-to-rank problems (e.g., [15, 18, 42]). Compared to tree-based models [16], they show better scalability in terms of input dimensionality.

Let $\boldsymbol{x}_d$ be the feature vector for a document d given a specific query q. For simplicity, we use concatenation of feature vectors of the m documents as the input layer in the DNN-based implementation of g, while leaving the exploration of more sophisticated input representations for future work. Specifically, let

$$\boldsymbol{h}_0 = \text{concat}(\boldsymbol{x}_{d_1}, ..., \boldsymbol{x}_{d_m})$$

and a multi-layer feed forward network with 3 hidden layers is defined as

$$\boldsymbol{h}_k = \sigma(\boldsymbol{w}_k^T \boldsymbol{h}_{k-1} + \boldsymbol{b}_k),\ k = 1, 2, 3 \tag{3}$$

where $\boldsymbol{w}_k$ and $\boldsymbol{b}_k$ denote the weight matrix and the bias vector in the k-th layer, σ is an activation function and we use the hyperbolic tangent function in our paper:

$$\sigma(t) = \frac{e^{2t} - 1}{e^{2t} + 1}$$

Our DNN-based function g is defined as

$$g(q, [d_1, ..., d_m]) = \boldsymbol{w}_o^T \boldsymbol{h}_3 + \boldsymbol{b}_o \tag{4}$$

where $\boldsymbol{w}_o$ and $\boldsymbol{b}_o$ are the weight vector and the bias in the output layer. The output dimension is set as m in the output layer.

4.2 Groupwise Scoring Architecture

Given a set of training data Ψ, a straightforward approach to learn a groupwise scoring function g is to concatenate all features of documents in a document list D together, and build a DNN with output size $|D|$ as the relevance prediction of the whole list. In this case, group size m is set to be $|D|$ and $f(q, D) = g(q, [d_1, ..., d_{|D|}])$. Such an approach, however, has two drawbacks: (1) The learned model is sensitive to document order; (2) The comparison is among all documents and this can make it difficult to learn the useful local comparison patterns. To overcome these drawbacks, we propose to limit the comparison within small groups and make the model invariant to the document order.

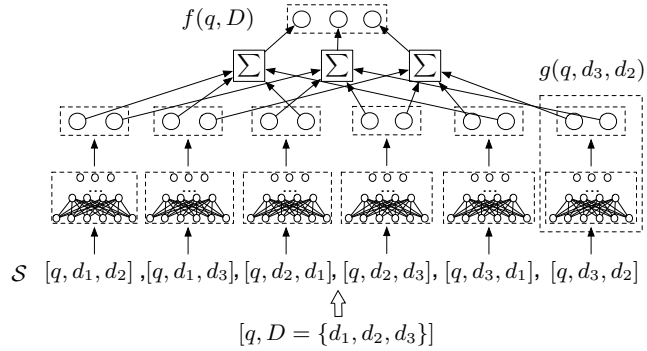

Figure 1: An example of Groupwise Scoring Functions (GSF) using permutations of size 2 in $\mathcal{S}$ for a D with 3 documents. Query q appears in all groups.

Let the input of a GSF model consist of a document list D with n documents. For simplicity, we assume that each D has exactly n documents and $n = |D|$. Notice that such an assumption trivially holds in the click data setting where the top-n documents per query is used, or can be ensured with padding or sampling strategy. For example, we can randomly sample n documents per query (without replacement) to form a new list. When $n = 2$, such a sampling strategy is similar to standard pairwise approaches where document pairs of a query are formed as training examples.

Let π_k be a list of m documents that forms a group. In each group, documents are compared and scored based on their feature vectors to predict which documents are more relevant.

$$[r_k^1, ..., r_k^m] = g(q, \pi_k) \tag{5}$$

where r_k^i is the ith dimension of $g(q, \pi_k)$, and also the intermediate relevance score for the i-th document in π_k. Note that we use the slightly different notation to make the following discussion easier.

The intermediate relevance scores from each individual groups are accumulated to generate final scores for each document. As we know, there are multiple (namely, $n!/(n-m)!$) permutations with size m in a list with size n, which means that there are as many possible inputs for a groupwise scoring function g in a list of documents D. Let $\mathcal{S} = \{\pi_1, \pi_2, ..., \pi_{|\mathcal{S}|}\}$ be all the permutations of the subset documents with size m from D, then we compute the final ranking score s_d for a document $d \in D$ as the accumulation of intermediate relevance scores from each individual group as

$$s_d = \sum_{k=1}^{|\mathcal{S}|} \sum_{i=1}^{m} \mathbf{1}_{\pi_k^i = d} \cdot r_k^i \tag{6}$$

where $\mathbf{1}_{\pi_k^i = d}$ is an indicator function that denotes whether d is the i-th document in π_k. And the final output of a GSF model would be

$$f(q, D) = [s_1, ..., s_{|D|}]$$

An example of such $f(q, D)$ with group size $m = 2$ and list size $n = 3$ is shown in Figure 1.

4.3 Loss Functions

Intuitively, we can define any loss functions between the scores and labels to train the GSF models. In this paper, we focus on a simple

loss function for graded relevance, leaving other loss functions to future studies.

Graded relevance is a multi-level relevance judgment and is commonly used in human-rated data sets. The loss for graded relevance is generally defined over document pairs. We extend the commonly used pairwise logistic loss for a list as

$$l(\psi, f) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}_{y_i > y_j} \cdot \log(1 + e^{-(s_i - s_j)}) \tag{7}$$

where $y_i > y_j$ means the i-th document is more relevant than the j-th document and the loss of a list is the sum of the loss of all pairs in the list. Please note that when $m = 1$, GSF is actually a pointwise scoring function. In this case, the loss in Equation (7) boils down to a pairwise loss function. However, it becomes a listwise loss for the general GSF models when $m \geq 2$.

4.4 Training and Inference

While GSF extends traditional pointwise scoring functions (PSF) by scoring each document according to their comparisons with others, it also loses the property of PSF that each model only produces one score for each document, which could cause multiple efficiency issues in practice. In this section, we propose multiple sampling strategies for the efficient training and inference of GSF models.

4.4.1 Speed Up Training. As shown in Equation (6), the final ranking score $f(q, D)$ of a GSF model is the aggregation of intermediate relevance scores from all possible permutations of a group with size m in a list with size n. Suppose that the computation complexity of a DNN with m input documents is $O(m)$, then such a scoring paradigm has a complexity of $O(m \cdot n!/(n-m)!)$, which is prohibitive in real systems. To speed up the training process, we conduct the following group sampling to reduce the permutation set $\mathcal{S}$. For each training instance with document list D, we randomly shuffle it before feeding into the model. Then, we only take the subsequences (instead of subsets) of size m from the shuffled input to form $\mathcal{S}$. In this way, we reduce the size $|\mathcal{S}|$ from $n!/(n-m)!$ to n subsequences: each starting at the i-th position and ending at $(i+m)$-th position in a circular manner. This leads to a reduced training complexity with $O(mn)$ time. Also, because each $d \in D$ appears in exactly m groups, all documents have equal probability to be compared to others and to be placed as the j-th document in each sublist π_k. With enough training examples, the GSF trained with our sampling strategy asymptotically approaches the GSF trained with all permutations, and it is insensitive to document order.

4.4.2 Efficient Inference. At inference time, the desired output is a ranked list. This can be done in a straightforward manner for pointwise scoring functions, but becomes non-trivial for GSF. To tackle this challenge, we propose our inference methods for the following two scenarios: fixed list size and varying list size.

Inference with Fixed List Size. Given a fixed list size n' at inference time, the most straightforward solution to do inference with GSF is to train the DNN model with the same list size $n = n'$. Thus, the score $f(q, d)$ for a document d can be directly computed as $f(q, d) = s_d$ with Equation (6).

Inference with Varying List Size. When it is impossible to make $n = n'$ or the list size at inference time cannot be determined beforehand, it is hard to use the GSF architecture in training directly. In this case, we extract the group comparison function g from GSF and use it to do inference. For each document d in the inference list D, we compute its score $f(q, d)$ by the expected marginal score as

$$f(q, d) = \mathbb{E}_{\pi \in \mathcal{S}, \pi^i = d}\, g(q, \pi)|_i \tag{8}$$

where the expectation is over all possible permutations of size m in $\mathcal{S}$ that contains document d, and $g(q, \pi)|_i$ is the i-th value of the function output (also referred to as r^i in Equation (5)). For example, when $m = 2$, $f(q, d)$ can be rewritten as

$$f(q, d) = \frac{1}{2|D|} \sum_{d' \in D} \left[g(q, d, d')|_1 + g(q, d', d)|_2 \right]$$

The expectation in Equation (8) can be approximated effectively by Monte Carlo methods [32]. For example, we can randomly sample a couple of documents and use the average over the sampled documents. The larger the sample size is, the better this approximation becomes. In our experiments, we found that sample size m for group size m is good enough. Thus the inference can be done efficiently in $O(mn) = O(n)$ since m is a predefined constant.

In fact, the case for fixed list size is a special case of the varying list size. When $\mathcal{S}$ contains all adjacent documents as groups, it is equivalent to sample m groups for each document and the document's position varies from 1 to m in these m groups. The score of the document is proportional to the sum of the corresponding values in these m groups.

5 RELATIONSHIP WITH EXISTING MODELS

In this section, we discuss the relationship between the existing learning-to-rank algorithms and our proposed models. There are two important hyper-parameters for GSF: the list size n and the group size m. By varying n and m, we will show that several representative algorithms can be formulated as special cases in our framework.

5.1 Pointwise Scoring and Pairwise Loss

Pairwise loss functions with pointwise scoring functions are popular in learning-to-rank algorithms. In these algorithms, the scoring function f takes a single document as input and outputs a single score. The loss function l takes a pair of scores from two documents and defines the loss based on the consistency between the scores and the preferences of the two documents. In our GSF model, let $n = 2$ and $m = 1$, d_1 and d_2 be the two documents in D, y_1 and y_2 be the graded relevance judgments, and s_1 and s_2 be the output scores. Then the logistic loss in our GSF model is

$$l(\psi, f) = \begin{cases} \log(1 + e^{-(s_1 - s_2)}) & y_1 > y_2 \\ \log(1 + e^{-(s_2 - s_1)}) & y_1 < y_2 \end{cases}$$

Such a loss is very similar to the one used in LambdaMART [6]. Other pairwise loss such as hinge loss used in RankingSVMs [21] can be defined similarly as Equation (7).

5.2 Pointwise Scoring and Listwise Loss

A traditional listwise model uses a pointwise scoring function with a listwise loss computed with all documents in the candidate list.

We show the connection between our method with a representative traditional listwise method – the ListNet method [8].

In our GSF model, let $m = 1$ and n be the length of the list. Then $\pi_k = [d_k]$ and s_i is computed based on document d_i only and we use $s_i = f(\boldsymbol{x}_i)$ to denote it. The softmax loss is

$$l(\psi, f) = -\sum_{i=1}^{n} y_i \log \frac{e^{f(\boldsymbol{x}_i)}}{\sum_{j=1}^{n} e^{f(\boldsymbol{x}_j)}}$$

In the ListNet approach, a distribution over all the permutations of the n documents is defined based on the scoring function, and another one is defined using the labels. The loss is the cross entropy between these two distributions. Because such a loss is not computationally expensive, a simplified version is to consider the marginal probability that a document d_i appears at the first position in the two distributions over the permutations. The resultant cross entropy loss is then:

$$l(\psi, f) = -\sum_{i=1}^{n} \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \log \frac{e^{f(\boldsymbol{x}_i)}}{\sum_{j=1}^{n} e^{f(\boldsymbol{x}_j)}}$$

5.3 Pairwise Scoring Functions

The pairwise scoring model is a recent model proposed by Dehghani et al. [12] that predicts the preference between two documents based on their features. As far as we know, this is the only published learning-to-rank model that takes more than one document as the input of its scoring function.

In the GSF model (see Section 4.1&4.2), let $n = 2$ and $m = 2$. Let $D = \{d_1, d_2\}$ be a pair of documents. Then

$$\mathcal{S} = [\pi_1, \pi_2], \pi_1 = [d_1, d_2], \pi_2 = [d_2, d_1].$$

If we define the aggregation step (Equation (6)) as

$$s_i = \sum_{k=1}^{2} \mathbf{1}_{\pi_k^1 = d_i} \cdot g(q, \pi_k) \tag{9}$$

where $\mathbf{1}_{\pi_k^1 = d_i}$ is an indicator function that denotes whether d_i is the first document in π_k, and use a sigmoid cross entropy loss as

$$l(\psi, f) = -\sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i),\ p_i = \frac{1}{1 + e^{-s_i}} \tag{10}$$

then it is equivalent to [12]. However, there is a slight difference between this pairwise input model and our GSF with $n = 2, m = 2$ (GSF(2, 2)). While [12] used only the first score from each group (pair) of documents, our GSF uses the scores of all documents from the two groups in Equation (6). In addition, our GSF model has the following two mathematically attractive properties that are not held by [12].

- **Reflexive.** When the input has two identical documents $d_1 = d_2$, then $s_1 = s_2$ always holds.
- **Antisymmetric.** If $[s_1, s_2]$ are scores for input $[d_1, d_2]$, then $[s_2, s_1]$ are scores for input $[d_2, d_1]$.

6 EXPERIMENTAL SETUP

To evaluate our proposed methods, we compare different types of learning-to-rank models on a standard learning-to-rank dataset. In this section, we describe our experimental design and setup.

Table 1: List of DNN models used in our experiments.

Pointwise Scoring Functions	
RankNet, GSF(2, 1)	GSF with $n = 2$ and $m = 1$. This model is comparable to a standard neural network model with pointwise scoring and pairwise loss, e.g., RankNet [5].
ListNet, GSF(n, 1)	GSF with n and $m = 1$. This is closely related to the existing models with pointwise scoring and listwise loss, e.g., ListNet [8].
Pairwise Scoring Functions	
GSF(2, 2)	GSF with $n = 2$ and $m = 2$. This is a DNN model with pairwise scoring and our proposed list loss functions.
Generic Groupwise Scoring Functions	
GSF(n, m)	The GSF model with list size n and group size m.

6.1 Learning-to-Rank Models

The learning-to-rank models we compared in our experiments include both DNN models and tree-based models.

6.1.1 DNN Models. Table 1 lists all the DNN models that we considered in our experiments. We group these models based on the input to their scoring functions as pointwise, pairwise, and groupwise. In this table, GSF(2, 1)and GSF(n, 1) represent the existing DNN models with pointwise scoring functions in the current literature. GSF(2, 2) is a GSF model that takes a pair of documents in their scoring function, and GSF(n, m) is the generic representation of the GSF model with a list size n and a group size m. We instantiate n and m in our experiments.

All DNN models in this paper were built on the 3-layer feed-forward network described in Section 3. The hidden layer sizes of DNN are set as 256, 128 and 64 for $\boldsymbol{h}_1$, $\boldsymbol{h}_2$ and $\boldsymbol{h}_3$. We used the TensorFlow toolkit for model implementations.

6.1.2 Tree-based Models. For tree-based models, we primarily use the state-of-the-art MART and LambdaMART [6] models. Furthermore, we also explore a hybrid approach in which predictions of DNN models are used as input features for tree-based models. We compare hybrid models with both standalone DNN and tree-based models in our experiments.

6.2 Data Sets

The data set used in our experiments is a public LETOR data set MSLR-WEB30K [30]. This is a large-scale learning-to-rank data set that contains 30K queries. On average there are 120 documents per query and each document has 136 numeric features. All the documents are labeled with graded relevance from 0 to 4. We report the results on Fold1 as a test set, while using the rest of the folds for training.

On the LETOR data set, there are more than a hundred documents per query. Instead of setting the list size n to be the largest possible number of documents per query, we conduct the following sampling to form training data for DNN models. Given an n (2 and 5 used in our paper), we randomly shuffle the list of documents for each query and then use a rolling window of size n over the shuffled document list to obtain a list of documents with size n. For robustness, we re-shuffle ten times, and thus obtain a collection of training data for DNN models.

Table 2: NDCG@5 (in percentage) on LETOR data using RankNet, MART, LambdaMART, GSF variants, and the hybrid approach LambdaMART+GSF. $*$ denotes statistically significant improvements over RankNet, and + denotes significant improvements over all other models in the table, both using t-test with $\alpha < 0.05$.

(a)		GSF			
RankNet		GSF(2, 1)	GSF(2, 2)	GSF(5, 1)	GSF(5, 2)
32.28		40.40*	41.10*	41.10*	**41.50***
(b)		LambdaMART+GSF			
MART	LambdaMART	GSF(2, 1)	GSF(2, 2)	GSF(5, 1)	GSF(5, 2)
43.51*	44.23*	44.51*	44.69*	44.60*	**44.90*+**

6.3 Evaluation Metrics

We train the models on the LETOR dataset with the logistic loss in Equation (7) given its graded relevance labels. The evaluation metric for this data set is the commonly used Normalized Discounted Cumulative Gain (NDCG) [19]. In this paper, we use NDCG@5 that truncates the cumulative sum to the top 5 positions. Significance test are conducted based on student's t-test.

7 RESULTS

In this section, we describe our experiment results in detail. We compare GSF with pointwise scoring models and pairwise scoring models with pairwise loss. And we also compare GSF with the state-of-the-art tree models in both standalone and hybrid way.

Table 2 shows the results of different models on the LETOR data set. For reproducibility, we use the learning-to-rank models, RankNet, MART and LambdaMART as implemented in the open-source Ranklib toolkit[2], and report the actual value of NDCG@5 for each model.

In Table 2(a), we observe that all GSF models outperform RankNet – another DNN-based learning-to-rank model – by a very large margin. For the different variants of GSF, GSF(2, 2) is better than GSF(2, 1), and GSF(5, 2) is better than GSF(5, 1), which indicates that having a group size m larger than 1 indeed improves the performance of a GSF model.

In Table 2(b), tree-based models are more competitive than the DNN models in standalone settings. However, we observe that the hybrid LambdaMART+GSF models outperform the state-of-the-art LambdaMART algorithm. For example, we achieve 1.5% improvement using GSF(5, 2) over LambdaMART, a statistically significant result. Also, in the hybrid mode, GSFs with $m > 1$ consistently outperform the GSFs with $m = 1$. This, again, confirms that, in general, a groupwise scoring function is better than a pointwise scoring function.

8 CONCLUSION

In this paper, we went beyond the traditional pointwise scoring functions and introduced a novel setting of groupwise scoring functions (GSFs) in the learning-to-rank framework. We implemented GSFs using a deep neural network (DNN) that can efficiently handle large input spaces. We showed that GSFs can include several existing learning-to-rank models as special cases. We compared both GSF models and tree-based models based on a standard learning-to-rank data set. Experimental results show that GSFs significantly benefit several state-of-the-art DNN and tree-based models, due to their ability to combine listwise loss and groupwise scoring functions.

Our work also opens up a few interesting future research directions: how to do inference with GSFs in a more principled way using techniques in [36], how to define GSFs using more sophisticated DNN like CNN, rather than simple concatenation, and how to leverage the more advanced DNN matching techniques proposed in [17, 27–29], in addition to the standard learning-to-rank features, in our GSFs.

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

REFERENCES

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying Search Results. In *Proc. of the 2nd ACM International Conference on Web Search and Data Mining (WSDM)*. 5–14.

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. 135–144.

[3] Paul N. Bennett, Ben Carterette, Olivier Chapelle, and Thorsten Joachims. 2008. Beyond Binary Relevance: Preferences, Diversity, and Set-level Judgments. *SIGIR Forum* 42, 2 (2008), 53–58.

[4] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A Neural Click Model for Web Search. In *Proc. of the 25th International Conference on World Wide Web (WWW)*. 531–541.

[5] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proc. of the 22nd International Conference on Machine Learning (ICML)*. 89–96.

[6] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report Technical Report MSR-TR-2010-82. Microsoft Research.

[7] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proc. of the 19th International Conference on Neural Information Processing Systems (NIPS)*. 193–200.

[8] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proc. of the 24th International Conference on Machine Learning (ICML)*. 129–136.

[9] Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 335–336.

[10] Olivier Chapelle, Donald Metzler, Ya Zhang, and Pierre Grinspan. 2009. Expected Reciprocal Rank for Graded Relevance. In *Proc. of the 18th ACM Conference on Information and Knowledge Management (CIKM)*. 621–630.

[11] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proc. of the 2008 International Conference on Web Search and Data Mining (WSDM)*. 87–94.

[12] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 65–74.

[13] Fernando Diaz. 2007. Regularizing query-based retrieval scores. *Information Retrieval* 10, 6 (2007), 531–562.

[14] Georges E. Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proc. of the 31st Annual International ACM SIGIR Conference on Reseach and Development in Information Retrieval (SIGIR)*. 331–338.

[15] Bora Edizel, Amin Mantrach, and Xiao Bai. 2017. Deep Character-Level Click-Through Rate Prediction for Sponsored Search. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 305–314.

[16] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* 29, 5 (2001), 1189–1232.

[17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proc. of the 25rd ACM International Conference on Information and Knowledge Management (CIKM)*. 55–64.

[18] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proc. of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*. 2333–2338.

[19] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[20] Zhengbao Jiang, Ji-Rong Wen, Zhicheng Dou, Wayne Xin Zhao, Jian-Yun Nie, and Ming Yue. 2017. Learning to Diversify Search Results via Subtopic Attention. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 545–554.
[21] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 133–142.
[22] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 217–226.
[23] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data As Implicit Feedback. In *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 154–161.
[24] Waldemar W. Koczkodaj. 1996. Statistically accurate evidence of improved error rate by pairwise comparisons. *Perceptual and Motor Skills* 82, 1 (1996), 43–48.
[25] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[26] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.
[27] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1291–1299.
[28] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. (2016). arXiv:1606.04648
[29] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *Proc. of the 2017 ACM Conference on Information and Knowledge Management (CIKM)*. 257–266.
[30] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. (2013). arXiv:1306.2597
[31] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. 2008. Global ranking using continuous conditional random fields. In *Proc. of the 21st International Conference on Neural Information Processing Systems (NIPS)*. 1281–1288.
[32] Christian P. Robert and George Casella. 2005. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag.
[33] Stephen E. Robertson. 1977. The probability ranking principle in IR. *Journal of Documentation* 33, 4 (1977), 294–304.
[34] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-smooth Rank Metrics. In *Proc. of the 2008 International Conference on Web Search and Data Mining (WSDM)*. 77–86.
[35] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proc. of the 11th International Conference on Web Search and Data Mining (WSDM)*. 610 –618.
[36] Fabian L. Wauthier, Michael I. Jordan, and Nebojsa Jojic. 2013. Efficient Ranking from Pairwise Comparisons. In *Proc. of the 30th International Conference on Machine Learning (ICML)*. 109–117.
[37] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proc. of the 25th International Conference on Machine Learning (ICML)*. 1192–1199.
[38] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2016. Modeling Document Novelty with Neural Tensor Network for Search Result Diversification. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 395–404.
[39] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 391–398.
[40] Peng Ye and David Doermann. 2013. Combining preference and absolute judgements in a crowd-sourced setting. In *ICML 2013 Workshop on Machine Learning Meets Crowdsourcing*.
[41] Emine Yilmaz, Manisha Verma, Nick Craswell, Filip Radlinski, and Peter Bailey. 2014. Relevance and effort: An analysis of document utility. In *Proc. of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*. 91–100.
[42] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1531–1540.