# WassRank: Listwise Document Ranking Using Optimal Transport Theory

Hai-Tao Yu*
yuhaitao@slis.tsukuba.ac.jp
University of Tsukuba, Japan

Adam Jatowt
adam@dl.kuis.kyoto-u.ac.jp
Kyoto University, Japan

Hideo Joho
hideo@slis.tsukuba.ac.jp
University of Tsukuba, Japan

Joemon M. Jose
Joemon.Jose@glasgow.ac.uk
University of Glasgow, UK

Xiao Yang
xiao.yang@ebi.ac.uk
European Bioinformatics Institute, UK

Long Chen
Long.Chen@glasgow.ac.uk
University of Glasgow, UK

## ABSTRACT

Learning to rank has been intensively studied and has shown great value in many fields, such as web search, question answering and recommender systems. This paper focuses on *listwise document ranking*, where all documents associated with the same query in the training data are used as the input. We propose a novel ranking method, referred to as *WassRank*, under which the problem of listwise document ranking boils down to the task of learning the optimal ranking function that achieves the minimum Wasserstein distance. Specifically, given the query level predictions and the ground truth labels, we first map them into two probability vectors. Analogous to the *optimal transport problem*, we view each probability vector as a pile of *relevance mass* with peaks indicating higher relevance. The listwise ranking loss is formulated as the minimum cost (the Wasserstein distance) of transporting (or reshaping) the pile of predicted relevance mass so that it matches the pile of ground-truth relevance mass. The smaller the Wasserstein distance is, the closer the prediction gets to the ground-truth. To better capture the inherent *relevance-based order* information among documents with different relevance labels and lower *the variance of predictions* for documents with the same relevance label, *ranking-specific cost matrix* is imposed. To validate the effectiveness of WassRank, we conduct a series of experiments on two benchmark collections. The experimental results demonstrate that: compared with four non-trivial listwise ranking methods (i.e., LambdaRank, ListNet, ListMLE and ApxNDCG), WassRank can achieve substantially improved performance in terms of nDCG and ERR across different rank positions. Specifically, the maximum improvements of WassRank over LambdaRank, ListNet, ListMLE and ApxNDCG in terms of nDCG@1 are 15%, 5%, 7%, 5%, respectively.

## KEYWORDS

Wasserstein distance, optimal transport, learning to rank

*Hai-Tao Yu is the corresponding author.

**ACM Reference Format:**
Hai-Tao Yu, Adam Jatowt, Hideo Joho, Joemon M. Jose, Xiao Yang, and Long Chen. 2019. WassRank: Listwise Document Ranking Using Optimal Transport Theory. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3291006

## 1 INTRODUCTION

Nowadays, enormous volume of search requests are submitted daily. For example, Google processes over 3.5 billion searches per day[1]. In order to accurately and efficiently provide desired information to users, there are many problems that are far from being resolved, and continue to constitute open challenges in both academic and industrial communities. A key problem is *ranking*, which has attracted significantly increasing attention in recent years across many fields, such as document retrieval and recommender systems. In this paper, we focus on the field of document retrieval. In particular, given a query and a set of documents to be ranked, the desired ranking model (or function) assigns a score to each document, then a ranked list of documents can be obtained by sorting the documents in descending order of scores. In general, the document with the highest score is assigned a rank of 1. In other words, the rank position of a document represents its relevance with respect to the query.

The modern approach is to use machine learning technologies to train the ranking model, and a new research area called *learning-to-rank* [33] has emerged and become popular. In general, in *training* phase, a number of queries are provided. Each query is associated with a set of documents to be ranked, of which the standard relevance labels are also included. Moreover, each query-document pair is represented through a feature vector. The target of learning is to create a ranking model, which produces rankings of documents that best agree with the rankings derived from the relevance labels. In the phase of *testing*, given a new query, the learned ranking model is used to rank documents associated with this query. The advantages of learning-to-rank are straightforward. On one hand, compared with the traditional score-based models (such as TF-IDF[48] and BM25[43]), it is a fully automatic learning process based on training data and can easily incorporate a large number of features. For example, a total of 700 different features are used in the datasets released for the Yahoo! Learning to Rank Challenge[8]. On the other hand, the rich learning frameworks (such as *support vector machines* and *deep neural networks*) enables the flexible development of powerful ranking models. Three major categories

[1]http://www.internetlivestats.com/google-search-statistics/

of learning-to-rank methods are *pointwise*, *pairwise* and *listwise*, respectively. The pointwise methods [11–13, 35] transform the ranking problem into a task of (ordinal) regression or classification on individual documents. The idea is natural and many existing mature learning techniques on classification and regression can be directly deployed. A major problem is that the pointwise methods are agnostic to the relevance-based order information among documents that are associated with the same query. To make a step forward, the pairwise methods [17, 28, 45] were then proposed, which transform the ranking problem into a task of pairwise classification. However, the loss functions merely consider the relative order between two documents rather than the total order relationship among all documents associated with the same query. In this regard, the adopted loss functions are not in accordance with the evaluation measures. Moreover, the number of document pairs per query may differ from query to query, thus the result can be biased in favor of queries with more documents in the training data [41]. To overcome the shortcomings of the aforementioned two categories of ranking methods, the listwise methods [6, 7, 9, 20, 40, 41, 49, 50, 53, 54, 56, 58] learn the ranking function by taking all documents associated with the same query as the input. The loss functions are commonly designed by minimizing the discrepancy between the predicted ranking and that generated based on the ground truth. The previous studies [7, 40, 41, 54] have demonstrated that the listwise paradigm commonly show superior performance over the other two paradigms of pointwise and pairwise.

Recently, due to the breakthrough success of deep learning in computer vision[31], neural networks (NNs) have been widely applied in the field of information retrieval. Compared with other learning frameworks of which designing features used to be a crucial step, the key advantage of NNs is the ability of working with raw queries and documents. Moreover, latent representations of the query and document can be learned in situ. In this work, we also implement the rank function using neural networks. In particular, we focus on investigating the listwise approach for learning-to-rank from the viewpoint of loss functions. A lot of efforts have been made to find meaningful loss functions to capture the discrepancy between the predicted ranking list and the ground-truth ranking list. Despite the success achieved by the state-of-the-art methods, there are still many open issues. For instance, ListNet [7] and ListMLE [54] may fail to distinguish a large ratio of high-quality rankings from low-quality rankings due to the limitation of adopted loss functions (cf. §6.1). For the type of methods that perform listwise ranking by associating a specific metric, it is difficult to know whether the failures are caused by the optimization or by the adopted metric. Because the approximated objective based on a metric is commonly not convex, there can be many local optima when optimizing the objective. Moreover, owing to the inherent differences among different metrics, training with an approximated metric (e.g., nDCG) may not lead to the optimal performance in terms of another metric (e.g., ERR) (Table 3 shows some example cases).

The aforementioned drawbacks motivate us to approach listwise ranking in a novel way. In this paper, we quantify the listwise ranking loss leveraging on the theory of *optimal transport* (also known as the *Wasserstein distance*). In the discrete case, optimal transport measures the discrepancy between two probability distributions over a metric space. In a nutshell, if we view each distribution as a pile of sand, the Wasserstein distance can be interpreted as the minimum cost of transporting (or reshaping) the pile of one distribution into the pile of the other distribution. Analogously, given the two probability vectors derived from the predicted relevance scores and the ground-truth relevance labels, we view each probability vector as a pile of *relevance mass* with peaks indicating higher relevance. The listwise ranking loss is formulated as the minimum cost of transporting (or reshaping) the pile of predicted relevance mass so that it best matches the pile of ground-truth relevance mass, namely the Wasserstein distance between these two probability vectors. The smaller the Wasserstein distance is, the closer the prediction gets to the ground-truth. The goal of learning is thus to find the optimal parameters of the ranking function, which achieves the minimum Wasserstein distance. The main contributions of this paper are summarized as follows:

**(1)** Analogous to the classic optimal transport problem, we formulate the task of listwise document ranking as a task of learning the optimal ranking function that achieves the minimum Wasserstein distance. In order to capture well the relevance-based order among documents associated with the same query, ranking-specific cost matrix is imposed when quantifying the discrepancy between the predicted ranking and the ranking derived from ground truth labels based on the Wasserstein distance.

**(2)** Based on two benchmark datasets, we show the superior performance of WassRank by comparing with four listwise ranking methods (i.e., LambdaRank, ListNet, ListMLE and ApxNDCG). The maximum improvements of WassRank over LambdaRank, ListNet, ListMLE and ApxNDCG in terms of nDCG@1 are 29%, 5%, 7%, 5%, respectively. By discussing the pros and cons of each method, we shed new light on the nature of listwise document ranking.

**(3)** We also thoroughly investigate the effects of a series of factors on the performance of WassRank. Our main finding is that the factors, such as the number of hidden layers of the ranking function, the regularization parameter, and the variance penalty greatly affect the effectiveness of WassRank, and careful examinations of these factors are highly recommended.

The remainder of the paper is structured as follows. In the next section, we briefly survey the prior studies on learning-to-rank and the work based on optimal transport. In Section 2, we give the mathematical formulation of the listwise ranking framework and provide some backgrounds on Wasserstein distance. In Section 3, we present how to perform listwise document ranking based on tailored Wasserstein distance, where the ranking-specific cost matrix is developed. A series of experiments that we conducted are discussed in Section 4. Finally, we conclude the paper in Section 5.

## 2 RELATED WORK

In this section, we discuss related work on learning-to-rank methods, neural ranking models, and the optimal transport based studies.

**Learning-to-rank** refers to a broad range of approaches that aim to tackle ranking problems using machine learning techniques. It is a broad topic of great value in multiple fields, such as document retrieval, question answering and recommender systems, where ranking lies in the core. Here we limit our discussion to document retrieval. We refer the reader to the work [32, 33] for a detailed review. As mentioned above, major learning-to-rank approaches can be classified into three categories: pointwise, pairwise, and listwise. The key distinctions are the underlying hypotheses, loss functions, the input and output spaces. The typical pointwise approaches include regression-based [13], classification-based [35], and ordinal regression-based algorithms [11, 12]. The loss functions of these algorithms is defined on the basis of each individual document. The pairwise approaches care about the relative order between two documents. The goal of learning is to maximize the number of correctly ordered document pairs. The assumption is that the optimal ranking of documents can be achieved if all the document pairs are correctly ordered. Towards this end, many representative methods have been proposed [5, 17, 27, 45, 57]. The listwise approaches take all the documents associated with the same query in the training data as the input. In particular, there are two types of loss functions when performing listwise learning. For the first type, the loss function is related to a specific evaluation metric (e.g., nDCG [26] and ERR [10]). Due to the non-differentiability and non-decomposability of the commonly used metrics, the methods of this type either try to optimize the upper bounds as surrogate objective functions [9, 56, 58] or approximate the target metric using some smooth functions [20, 40, 49]. However, there are still some open issues regarding the first type methods. On one hand, some adopted surrogate functions or approximated metrics are not convex, which makes it hard to optimize. On the other hand, the relationship between the surrogate function and the adopted metric has not been sufficiently investigated, which makes it unclear whether optimizing the surrogate functions can indeed optimize the target metric. For the second type, the loss function is not explicitly related to a specific evaluation metric. The loss function reflects the discrepancy between the predicted ranking and the ground-truth ranking. Example algorithms include [6, 7, 54]. Although no particular evaluation metrics are directly involved and optimized here, it is possible that the learned ranking function can achieve good performance in terms of evaluation metrics. Our work belongs to the second type. Different from prior studies, we investigate the problem of listwise document ranking from a unique perspective of optimal transport. The query-level ranking loss is quantified based on the smoothed Wasserstein distance between the predicted ranking and the ranking derived from ground truth labels, where ranking-specific cost matrix is developed.

**Neural ranking models** refer to the recent ranking methods [21, 22, 24, 37, 46, 51] building upon neural networks. For example, the ranking models, such as DSSM [24] and CDSSM [46], map both queries and documents into the same semantic space based on deep neural networks. The relevance score between a query and a document is assumed to be proportional to the cosine similarity between their corresponding vectors in the semantic space. The follow-up studies [21, 22, 37, 51] look into the inherent characteristics of information retrieval. The DRMM model by Guo et al. [21] take into account more factors, such as query term importance, exact matching signals, and diverse matching requirement. The methods like [22, 37, 51] first look at the local interactions between two texts, then design different network architectures to learn more complicated interaction patterns for relevance matching. We refer the reader to [36] for an overview of neural ranking models.

**Optimal transport** research has a rich and varied history [38]. The traditional approaches for getting the optimal transport plan involves solving of a *linear program*, which makes it prohibitive in machine learning due to the large memory requirement and long run time. Thanks to the work by Cuturi [14], the theory of optimal transport has resurfaced as a popular paradigm in many machine learning applications. For instance, the recent applications [2, 34] in learning generative models have shown the potential of optimal transport. Montavon et al. [34] explored how to train Restricted Bolzmann Machines with the Wasserstein distance. Arjovsky et al. [2] view the learning of generative adversarial networks (GANs) as a transportation problem, where the Wasserstein distance is used as the training metric. In natural language processing (NLP), Rolet et al. [44] and Huang et al. [23] explored the usage of Wasserstein distance in dictionary learning. In the field of computer vision, the Wasserstein distance guided studies include, but not limited to, image segmentation [39], artistic image indexation [25], texture synthesis [16] and video restoration [15]. Yet, to the best of our knowledge, we are the first to adapt the theory of optimal transport for document ranking.

## 3 PRELIMINARIES

In this section, we begin by describing a general ranking framework, hereby referred to as the *listwise ranking framework*. We note that this framework is the same as or generalizes the ones employed in prior studies [29, 42]. Then we provide a brief description on *optimal transport theory* and the entropy-regularized *Wasserstein distance*, upon which our proposed ranking method builds. Due to space constraints, for a detailed description of optimal transport theory, we refer the reader to the work [38].

### 3.1 The Listwise Ranking Framework

Let $\mathcal{Q}$ and $\mathcal{D}$ be the query space and the document space, respectively, we use $\Phi : \mathcal{Q} \times \mathcal{D} \rightarrow \mathcal{Z} := \mathbb{R}^d$ to denote the mapping function for generating a feature vector for a query-document pair, where $\mathcal{Z}$ represents the $d$-dimensional feature space. We use $\mathcal{R} := \mathbb{R}_+$ to denote the space of the ground-truth relevance scores each document receives. Thus for each query, we have a list of document feature vectors $X = (X_1, ..., X_m) \in \mathcal{X} := \mathcal{Z}^m$ and a corresponding list $Y = (Y_1, ..., Y_m) \in \mathcal{Y} := \mathcal{R}^m$ of ground-truth relevance scores. In practice, we get independently and identically distributed (i.i.d) samples $\mathcal{S} = \{(X_i, Y_i)\}_{i=1}^n$ from an unknown joint distribution $P(\cdot, \cdot)$ over $\mathcal{X} \times \mathcal{Y}$. We use $f_\theta : X \rightarrow \mathbb{R}_+^m$ parameterized by $\theta \in \Theta$ to denote the real-valued ranking function, which assigns each document a score. The scores of the documents associated with the same query are used to rank the documents. We measure the loss of ranking documents for a query using $f_\theta$ with the loss function $\ell(f_\theta(X), Y)$. The goal is to learn the optimal ranking function over a hypothesis space $\mathcal{F}$ of ranking functions that can *minimize the expected risk* as defined below:

$$\min_{f_\theta \in \mathcal{F}} \mathfrak{R}(f_\theta) = \min_{f_\theta \in \mathcal{F}} \int_{\mathcal{X} \times \mathcal{Y}} \ell(f_\theta(X), Y) dP(X, Y) \tag{1}$$

Typically, $\mathfrak{R}(f_\theta)$ is intractable to optimize directly and the joint distribution is unknown, we appeal to the *empirical risk minimization* to approximate the expected risk, which is defined as follows:

$$\min_{f_\theta \in \mathcal{F}} \hat{\mathfrak{R}}(f_\theta; \mathcal{S}) = \min_{f_\theta \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X), Y) \tag{2}$$

Given the above listwise ranking framework, one can design various listwise ranking methods by deploying different loss functions to learn the parameters $\theta$ based on the training documents. In the testing phase, the predicted ranking can be obtained efficiently by sorting the testing documents in descending order of their individual relevance scores $f_\theta(X_i)$.

### 3.2 Optimal Transport and The Entropy-Regularized Wasserstein Distance

In this section, we state the formulation of optimal transport tailored for discrete measures. For a more general mathematical exposition, we refer the reader to the work [38]. In the following, $\mathbf{1}_n$ denotes the $n$-dimensional vector with all ones. We use $\Delta_n := \{\mathbf{o} \in \mathbb{R}_+^n : \mathbf{1}_n^\top \mathbf{o} = 1\}$ to denote the probability simplex, whose elements are called probability vectors, or equivalently histograms. Given two probability vectors $\mathbf{p} \in \Delta_n$ and $\mathbf{q} \in \Delta_n$, we use $\Pi(\mathbf{p}, \mathbf{q}) := \{\pi \in \mathbb{R}_+^{n \times n} : \pi^\top \mathbf{1}_n = \mathbf{p}, \pi \mathbf{1}_n = \mathbf{q}\}$ to denote the set of coupling matrices. In the context of optimal transport, $\mathbf{p}$ and $\mathbf{q}$ are viewed as mass distributions, an element $\pi \in \Pi(\mathbf{p}, \mathbf{q})$ is referred to as a transport plan. An entry $\pi_{ij}$ indicates how much of the mass $\mathbf{p}_j$ is transported to $\mathbf{q}_i$. Given the cost matrix $C \in \mathbb{R}_+^{n \times n}$, where $C_{ij}$ indicates the corresponding transport cost per unit mass from $\mathbf{p}_j$ to $\mathbf{q}_i$, the optimal transport problem is defined as

$$\mathcal{W}(\mathbf{p}, \mathbf{q}) = \min_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} < C, \pi >_F \tag{3}$$

where $< \cdot, \cdot >_F$ denotes the Frobenius inner product of two matrices. $\mathcal{W}(\mathbf{p}, \mathbf{q})$ is called the Wasserstein distance (also known as the earth mover's distance), which indicates the minimum transportation cost. Computationally, it is quite expensive to obtain the optimal solution of Eq-3. For a general cost matrix $C$, the worst-case complexity of achieving the optimum scales in $O(n^3 \log n)$ [14]. Moreover, optimizing Eq-3 by gradient descent can be prohibitive due to the fact that it entails solving a linear program with $O(n^2)$ constraints [18].

Recently, Cuturi [14] has proposed to regularize the optimal transport problem by means of an entropy term. Specifically, let $\lambda > 0$ be the regularization parameter, the $\lambda$-smoothed Wasserstein distance is given as:

$$\mathcal{W}_\lambda(\mathbf{p}, \mathbf{q}) = \min_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} < C, \pi >_F - \lambda \mathcal{H}(\pi) \tag{4}$$

where $\mathcal{H}(\pi) = -\sum_{i,j} \pi_{ij} \log \pi_{ij}$ is the entropy of $\pi$. Thanks to the entropic regularization, the problem by Eq-4 is strictly convex and admits a unique solution which has an almost explicit form. Specifically, the optimal solution $\pi^*$ of Eq-4 is a diagonal scaling of the matrix $M = \exp(\frac{-C}{\lambda})$,

$$\pi^* = diag(\mathbf{v}) M diag(\mathbf{u}) \tag{5}$$

for $\mathbf{v}_i = \exp(-\frac{\mathbf{b}_i}{\lambda} - \frac{1}{2})$ and $\mathbf{u}_j = \exp(-\frac{\mathbf{a}_j}{\lambda} - \frac{1}{2})$, where $\mathbf{a}$ and $\mathbf{b}$ are the Lagrange dual variables for Eq-4. According to the study by Altschuler et al. [1], applying the Sinkhorn's algorithm [47] allows to achieve a near-$O(n^2)$ complexity.

## 4 PROPOSED METHOD

In this section, we present the ingredients of our proposed method step by step. The key idea is to use the tailor-made Wasserstein distance to quantify the discrepancy between the ranked list given by the ranking model and that generated based on the ground truth. Specifically, we first describe the proposed ranking framework. Then we formulate the Wasserstein ranking loss. Finally, we describe the necessity of imposing a ranking-specific cost matrix.

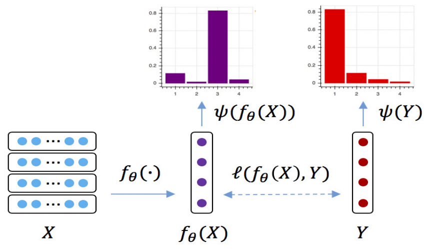


**Figure 1: The proposed listwise ranking framework.**

### 4.1 Wasserstein Ranking Loss

As an illustration, Fig. 1 shows the skeleton of the proposed listwise ranking framework. Before feeding the training sample $X$ and the ground-truth relevance vector $Y$, the elements of $Y$ are sorted in descending order of their individual relevance scores, and the document feature vectors in $X$ are also sorted correspondingly. Given the ranking function $f_\theta(\cdot)$ , the predicted relevance scores are obtained as $f_\theta(X)$. To measure the discrepancy between the vector of predicted relevance $f_\theta(X)$ and the vector of ground-truth relevance $Y$, we first perform a uniform mapping $\psi(\cdot)$ on both $f_\theta(X)$ and $Y$, which guarantees $\psi(f_\theta(X)) \in \Delta_n$ and $\psi(Y) \in \Delta_n$. In this work, $\psi(\cdot)$ is configured with the *softmax* function, namely $\psi(Y)_i = \frac{\exp(Y_i)}{\sum_j \exp(Y_j)}$ and $\psi(f_\theta(X))_i = \frac{\exp(\epsilon \cdot f_\theta(X)_i)}{\sum_j \exp(\epsilon \cdot f_\theta(X)_j)}$. When computing $\psi(f_\theta(X))$, the constant $\epsilon$ (i.e., the maximum level of relevance label) is added so that the predicted relevance scores lie in the same interval as the ground truth scores. Eventually, we quantify the loss of ranking documents for a query using $f_\theta(\cdot)$ as follows:

$$\ell_{\mathcal{W}}(f_\theta(X), Y) = \mathcal{W}(\psi(f_\theta(X)), \psi(Y)) \tag{6}$$

Hereby we call $\ell_{\mathcal{W}}(f_\theta(X), Y)$ the exact Wasserstein ranking loss. Owing to the difficulty in optimizing $\ell_{\mathcal{W}}(f_\theta(X), Y)$ using simple gradient based strategies, we appeal to its smoothed version,

$$\ell_{\mathcal{W}_\lambda}(f_\theta(X), Y) = \mathcal{W}_\lambda(\psi(f_\theta(X)), \psi(Y)) \tag{7}$$

We call $\ell_{\mathcal{W}_\lambda}(f_\theta(X), Y)$ the smoothed Wasserstein ranking loss. Analogous to the traditional optimal transport problem, given the two probability vectors $\psi(f_\theta(X))$ and $\psi(Y)$, we view each probability vector as a pile of relevance mass, and the total mass is one. The height indicates the relevance degree of the corresponding document. The listwise ranking loss is then defined as the minimum cost of transporting the pile of predicted relevance mass into the pile of ground-truth relevance mass. As shown in the top of Fig. 1, it is easy to observe that the closer the probability vector $\psi(f_\theta(X))$ derived from the predicted relevance scores approximates to $\psi(Y)$ derived from the ground-truth, the smaller the Wasserstein ranking loss is. The goal of learning is to find the optimal parameters of the ranking function, that achieves the minimum Wasserstein ranking loss. In particular, the gradient for optimizing the ranking function $f_\theta(\cdot)$

per query is summarized in Algorithm 1. $S$ denotes the pre-defined threshold of Sinkhorn iteration, and $\oslash$ denotes the element-wise division.

**Algorithm 1** The gradient of the smoothed Wasserstein ranking loss

```
1: Given $X$, $Y$, $f_\theta(\cdot)$, $\psi(\cdot)$, $\lambda$, $M$ and $s$.
2: $\mathbf{u} \leftarrow \frac{\mathbf{1}_n}{m}$
3: while $s \leq S$ do
4:     $\mathbf{v}^{(s)} = Y \oslash (M\mathbf{u}^{(s-1)})$
5:     $\mathbf{u}^{(s)} = \psi(f_\theta(X)) \oslash (M^\top \mathbf{v}^{(s)})$
6:     $s$++
7: $\nabla_{\psi(f_\theta(X))}^{\ell_{W_\lambda}(\cdot)} = \lambda(\log(\mathbf{u}^{(s)}) + \frac{1}{2})$
8: Compute $\nabla_\theta^{\ell_{W_\lambda}(\cdot)}$ according to the chain rule:
9: $\nabla_\theta^{\ell_{W_\lambda}(\cdot)} = [\nabla_{\psi(f_\theta(X))}^{\ell_{W_\lambda}(\cdot)}]^\top \nabla_\theta^{\psi(f_\theta(X))}$
```

## 4.2 Ranking-specific Cost Matrix

So far, our description of Wasserstein ranking loss has been generic to any cost matrix $C$ (cf. Section 3.2), which determines the direction in which the ranking loss is higher or lower. To well capture the relevance-based order information among documents that are associated with the same query, we impose the ranking-specific cost matrix on the computation of Wasserstein ranking loss. Algorithm 2 shows the detailed steps when initializing the cost matrix per query. Given the ground truth relevance vector $Y$ of which the elements are sorted in descending order, the corresponding rank positions $1, ..., m$ can be grouped into a number of bins $\{B_j\}$, where the ground truth judgments corresponding to the rank positions within the same bin are of the same level. When quantifying $C_{ij}$, we take into account the information across bins and that within the same bin. For the case that $i$ and $j$ are in the same bin (i.e., step-6), the cost of transportation is given as $\alpha$. We interpret $\alpha$ as the variance penalty, by which we try to achieve lower variance of predictions for documents with the same relevance label. For the case that $i$ and $j$ belong to different bins, the transportation cost is given as the difference between the gain values of the corresponding documents. Analogous to the evaluation metrics, such as nDCG [26] and ERR [10], the gain value is computed as $g(Y_i) = b^{Y_i} - 1$, where $b$ is the base. In particular, an additional cost $\beta$ is added when transporting between the position of a relevant document and the position of a non-relevant document (i.e., step-10). We interpret $\beta$ as the gap between a relevant document as an irrelevant document.

## 5 EXPERIMENTAL SETUP

In this section, we describe the experimental setup. We first introduce the test collections and the evaluation metrics. We then describe the configuration of each method to be evaluated, including the parameter setting and the way of training.

### 5.1 Datasets and Metrics

In our experiments, we used two widely used benchmark datasets, MQ2007 and MQ2008. Each query-document pair is represented with a feature vector. The ground truth is a multiple-level relevance judgment, which takes three values from 0 (irrelevant) to 2 (perfectly relevant). The basic statistics of each dataset are listed in Table 1, which includes the number of queries, the number of documents, the relevance level of ground truth, the number of features and the average number of relevant documents. For more detailed information, such as the source documents and the feature description, we refer readers to the official website of LETOR[2]. We note that there are some other learning-to-rank datasets, such as Microsoft 10k, Microsoft 30k and Yahoo! Learn to Rank Challenge v2.0. We believe that the experiments based on MQ2007 and MQ2008 are reasonable enough to justify the pros and cons of each method from the viewpoint of loss functions, since we do not use deeper neural networks that rely heavily on larger datasets to tune parameters. We leave the detailed comparison based on larger datasets as a future work.

**Algorithm 2** The ranking-specific cost matrix.

```
1: Order the elements of $Y$ in descending order.
2: for $i$ in $[0, m]$ do
3:     for $j$ in $[0, m]$ do
4:         if $i \neq j$ then
5:             if $Y_i == Y_j$ then
6:                 $C[i, j] = \alpha$
7:             else
8:                 $C[i, j] = |g(Y_i) - g(Y_j)|$
9:                 if $0 == Y_i$ or $0 == Y_j$ then
10:                    $C[i, j] += \beta$
11:        else
12:            $C[i, j] = 0$
13: return $C$
```

**Table 1: The characteristics of MQ2007 and MQ2008.**

| Dataset | Queries | Doc. | Rel. | Feat. | Avg.Rel. |
|---|---|---|---|---|---|
| MQ2007 | 1,692 | 69,623 | $\{0, 1, 2\}$ | 64 | 11 |
| MQ2008 | 784 | 15,211 | $\{0, 1, 2\}$ | 64 | 4 |

The metrics we adopted are nDCG [26] and ERR [10]. Both nDCG and ERR take into account the rank position and the relevance level. We report the results with different cutoff values 1, 3, 5 and 10 to show the performance of each method at different positions.

### 5.2 Baselines and Model Configuration

In our experiments, four typical listwise ranking methods are used as our baselines: ApproxNDCG [40], LambdaRank [6], ListNet [7] and ListMLE [54]. ApproxNDCG represents the first type listwise ranking method (cf. Section 2), where the goal is to optimize the smoothed nDCG. LambdaRank, ListNet and ListMLE represent the second type listwise ranking method, where the loss function is not related to a specific evaluation metric. Following prior work [7, 54], a simple 1-layer feed-forward neural network (a dropout rate of 0.01) with the Sigmoid activation function is used as the ranking

[2]https://www.microsoft.com/en-us/research/project/mslr/

**Table 2: Performance of different models on MQ-2007. The best result of each setting is indicated in bold.**

| Models | MQ2007 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | nDCG@1 | ERR@1 | nDCG@3 | ERR@3 | nDCG@5 | ERR@5 | nDCG@10 | ERR@10 |
| LambdaRank | 0.4196* | 0.1932* | 0.4313* | 0.2843* | 0.4413* | 0.3111* | 0.4790* | 0.3324* |
| ListNet | 0.4713 | 0.2218* | 0.4727 | 0.3145 | 0.4806 | 0.3405 | 0.5123 | 0.3599 |
| ListMLE | 0.4610* | 0.2124* | 0.4597* | 0.3048* | 0.4687* | 0.3309* | 0.5060* | 0.3514* |
| ApxNDCG | 0.4590* | 0.2191* | 0.4602* | 0.3097* | 0.4676* | 0.3348* | 0.5015* | 0.3553* |
| WassRank | **0.4807** | **0.2272** | **0.4731** | **0.3167** | **0.4837** | **0.3431** | **0.5135** | **0.3622** |

**Table 3: Performance of different models on MQ-2008. The best result of each setting is indicated in bold.**

| Models | MQ2008 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | nDCG@1 | ERR@1 | nDCG@3 | ERR@3 | nDCG@5 | ERR@5 | nDCG@10 | ERR@10 |
| LambdaRank | 0.4136* | 0.2515* | 0.4351* | 0.3465* | 0.4775* | 0.3739* | 0.5549* | 0.3916* |
| ListNet | 0.4173 | 0.2526 | 0.4476 | 0.3508 | 0.4878* | 0.3769* | 0.5752* | 0.3936* |
| ListMLE | 0.4097 | 0.2429* | 0.4456* | 0.3461* | 0.4831* | 0.3718* | 0.5666* | 0.3894* |
| ApxNDCG | 0.4224 | 0.2507 | 0.4457 | 0.3481* | 0.4939 | 0.3782* | 0.5740 | 0.3941* |
| WassRank | **0.4384** | **0.2644** | **0.4594** | **0.3635** | **0.4987** | **0.3912** | **0.5833** | **0.4069** |

function. We chose such settings as it helps to reflect the effectiveness of the adopted loss function without the possible impacts due to tuning a more complex network. For both the proposed method and the baseline approaches, we apply the same framework (e.g., the ranking function and the tuning strategy) except the component of loss function. This enables us to conduct a fair comparison when investigating the impact of a specific component on the performance. For other methods (e.g., the listwise model AdaRank-NDCG [56], the pairwise models RankBoost [17] and RankSVM-Struct [28]), we refer the reader to the LETOR official website for detailed comparisons.

We implemented and trained all the above models using PyTorch v0.4.0[3]. We used the L2 regularization with a decaying rate of $1 \times 10^{-3}$ and the Adam optimizer with a learning rate of $1 \times 10^{-3}$. The 5-fold cross validation strategy has been adopted. In particular, each dataset is randomly partitioned into five equal sized subsets. In each fold, three subsets are used as the training data, the remaining two subsets are used as the validation data and the testing data, respectively. We use the training data to learn the ranking model, use the validation data to select the hyper parameters based on nDCG@10, and use the testing data for evaluation. Finally, we report the ranking performance based on the averaged evaluation scores across five folds with 100 epochs. For ApproxNDCG, the parameter $\alpha$ is tuned with a set of values $\{50, 100, 150, 200\}$. For ListNet, the ranking loss is computed based on the top-1 approximation as in the original paper [7], namely each element of the probability vector represents the probability of the corresponding document being ranked at the top-1 position. For WassRank, the parameters of $\lambda$, $\beta$, $\alpha$ and $b$ are configured as 0.1, 100, $e$ (i.e., the base of the natural logarithm) and 4, respectively. We further explore to what extent these parameters affect the performance of WassRank in Section 6.2.

[3]Source code: https://ptl2r.github.io

## 6 RESULTS AND ANALYSIS

In this section, we report the experimental results and conduct detailed analysis. Particularly, we want to show how effective is WassRank compared to the baseline methods and shed some light on why it is able to achieve improved performance. In the following, we first describe the overall performance, and then explore how sensitive is the performance of WassRank to the hyper-parameters.

### 6.1 Overall Performance

The overall performance of the baseline approaches and the proposed method are shown in Table 2 and Table 3. The superscript $*$ indicates statistically significant difference when compared to the best result based on the Wilcoxon signed-rank test with $p < 0.05$.

First of all, we can observe that LambdaRank shows poor performance on MQ2007 compared with other methods. ListNet and ListMLE show relatively better performance on MQ2007 than that shown on MQ2008. The results are also consistent with the study by Lan et al. [30], where ListNet outperforms ListMLE. ApxNDCG demonstrates relatively stable performance on both datasets. WassRank outperforms all the baseline methods across two datasets in terms of different evaluation metrics. A closer look at Table 2 and Table 3 reveals that WassRank achieves statistically significantly better performance in most cases. The improvements over LambdaRank, ListNet, ListMLE and ApxNDCG in terms of nDCG@1 on MQ2007 are 15%, 2%, 4%, 5%, respectively. On MQ2008, the improvements over LambdaRank, ListNet, ListMLE and ApxNDCG in terms of nDCG@1 are 6%, 5%, 7%, 4%, respectively.

Now we investigate the possible reasons for the above findings. In a nutshell, LambdaRank is a general framework that merely requires the gradient to be defined, rather than the objective function. Essentially, for a given sorted list of documents, the loss function of LambdaRank boils down to a weighted version of the RankNet objective [5]. LambdaRank modifies the original gradient of RankNet defined on pairwise loss by multiplying the absolute change in the evaluation metric (e.g., nDCG) due to swapping two documents. However, the number of documents associated with the same query in the training data commonly vary from query to query. This is to say, the number of document pairs can be different from query to query. The result can be biased in favor of queries with more document pairs [41]. For instance, the minimum and maximum number of documents associated with the same query for MQ2007 are 6 and 147, respectively. For MQ2008, they are 5 and 121, respectively. This is one possible explanation for why LambdaRank does not work so well as other listwise ranking methods. ApxNDCG is a typical method that performs listwise ranking by optimizing the smoothed nDCG metric. It is notable that the approximated nDCG objective is not convex. In result, there may be many local optima when tuning the parameters, which makes it hard to find a good solution. On the contrary, the loss function of WassRank is convex, which circumvents this issue. Interestingly, from Table 3, we can observe that ApxNDCG outperforms ListNet in terms of nDCG@1, but underperforms ListNet in terms of ERR@1. This, however, is not surprising considering the inherent differences among the evaluation metrics, such as the underlying evaluation mechanisms and the discounting manners of rank positions. Training with an approximated metric (e.g., nDCG) may not lead to the optimal performance in terms of another metric (e.g., ERR). Due to these shortcomings, ApxNDCG underperforms WassRank on both datasets, and achieves a little bit worse performance than both ListNet and ListMLE on MQ2007.

We next turn towards investigating why WassRank shows superior performance than ListNet and ListMLE. To demonstrate the ability of WassRank in differentiating high-quality ranking from low-quality ranking, we conduct a case study with a small toy data. Due to space limitation, we omit the computation details. Suppose we are given a small set of documents $\{d_1, d_2, d_3, d_4, d_5\}$, and the corresponding ground truth labels are $[4, 3, 2, 1, 0]$. This is to say, the optimal ranking is just $\{d_1, d_2, d_3, d_4, d_5\}$. Furthermore, suppose there are two ranking functions $f_1$ and $f_2$, and their predicted relevance scores are $[\ln 3, \ln 4, \ln 2.5, \ln 2, \ln 0.1]$ and $[\ln 4, \ln 3, \ln 0.1, \ln 2, \ln 2.5]$, respectively. Under ListNet [7] (with the top-1 approximation), the ranking losses for $f_1$ and $f_2$ are 1.3532 and 1.4772, respectively. Under ListMLE [54], the ranking losses for $f_1$ and $f_2$ are 2.7764 and 6.6338, respectively. Eventually, according to both ListNet and ListMLE, the ranking $\{d_2, d_1, d_3, d_4, d_5\}$ will be regarded as a better ranking rather than $\{d_1, d_2, d_5, d_4, d_3\}$. However, according to nDCG and ERR, the evaluation results achieved by $f_1$ and $f_2$ are 0.8616 and 0.9841 in terms of nDCG@5, 0.7038 and 0.9530 in terms of ERR@5, respectively. This indicates that both ListNet and ListMLE fail to differentiate high-quality ranking from low-quality ranking due to the limitation of adopted loss functions. In other words, if the adopted loss function is not consistent with the widely used metrics, we can hardly achieve any expect performance. This echoes the findings in prior studies [30, 42]. On the contrary, based on WassRank, the ranking losses for $f_1$ and $f_2$ are 101.0895 and 87.6230. Here the exact Wasserstein ranking loss with the same parameter setting in Section 5.2 (such as $\alpha$, $\beta$ and $b$) is computed. It is easy to find that the results of WassRank are more consistent with the evaluation metrics nDCG and ERR. This reflects the effectiveness of WassRank in capturing position importance, which can not be easily captured indeed. In the practical sense, we want to rank highly relevant documents on the top positions rather than to have the whole list slightly more relevant. The eye-tracking based studies [19, 52] also demonstrate that users are more concerned on results in top positions of a result list (i.e., the *position bias*). It is important to keep in mind that the input space of listwise ranking consists of permutations, which is of the size $O(m!)$. In the practical sense, every permutation is possible. As a consequence, ListNet and ListMLE would fail to distinguish a large ratio of high-quality rankings from low-quality rankings. The experimental results on both MQ2007 and MQ2008 show that WassRank achieves better performance than both ListNet and ListMLE, especially at higher positions, which also echoes the above analysis based on a toy example data.

To summarize, through imposing a ranking-specific cost matrix, the proposed smoothed Wasserstein ranking loss inherits the excellent properties (such as convexity and completeness) of optimal transport while capturing the relevance-based order information among documents. The experimental results also demonstrate the superior performance of WassRank compared with the four nontrivial baseline methods.

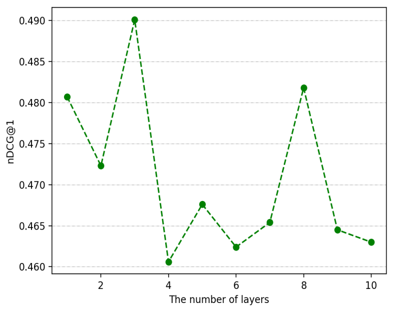
## 6.2 Parameter Sensitivity

To evaluate the parameter sensitivity of WassRank, we test WassRank with different hyper-parameters on MQ2007. Specifically, Fig. 2 shows how the hyper-parameters affect the performance of WassRank in terms of nDCG@1, such as increasing number of hidden layers of the ranking function, the regularization parameter $\lambda$ and the variance penalty $\alpha$.

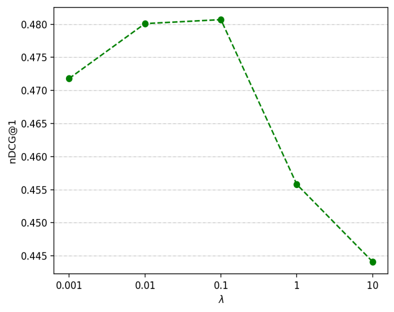
In Fig. 2(a), we plot the performance of WassRank in terms of nDCG@1 with respect to the number of hidden layers of the ranking function from 1 to 10. It is noticeable that WassRank has gained additional improvement (the nDCG1 value is 0.4901) with three hidden layers over the results shown in Table 2. But the performance of WassRank fluctuates with different number of hidden layers rather than a proportional improvement. One possible explanation is that: as the number of hidden layers increase, the ability of approximating more complex ranking functions (i.e., the model capacity) also increases. However, too many hidden layers may result in overfitting. Overall, Fig. 2(a) implies that better performance can be expected when incorporating more complex ranking functions into WassRank.

In Fig. 2(b), we plot the performance of WassRank in terms of nDCG@1 with respect to the regularization parameter $\lambda$ which has assigned values as $[0.001, 0.01, 0.1, 1, 10]$. As shown in Eq-4, there is a trade-off between enforcing the entropic regularizer to improve computational properties and maintaining convergence to the true optimal transport value. The higher value we assign to $\lambda$, the farther the approximated value deviates from the true optimal transport value. However, if $\lambda$ is too small, there will be issues with the gradient computation. This is also why we observe a ridge-shaped curve in Fig. 2(b), and WassRank achieves the best performance with $\lambda = 0.1$.

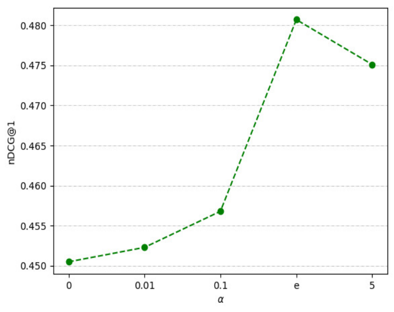
In Fig. 2(c), we plot the performance of WassRank in terms of nDCG@1 with respect to the change of the variance penalty $\alpha$ with values of $[0, 0.01, 0.1, e, 5]$. For the case of $\alpha = 0$, it means that the transport cost between two positions whose corresponding two documents are of the same relevance level is zero. Given a positive



(a) The effect of increasing number of hidden layers.



(b) The effect of the regularization parameter $\lambda$.



(c) The effect of the variance panelty $\alpha$.

**Figure 2: How the hyper-parameters affect the performance of WassRank, where the x-axis represents the change of a specific parameter, and the y-axis represents the performance in terms of nDCG@1.**

value for $\alpha$, the ranking loss will increase if the ranking function assigns highly varied scores for documents with the same relevance label. From Fig. 2(c), we can observe the necessity of imposing a relatively small variance penalty, which helps to improve the ranking performance. However, if we assign a relatively large value, e.g., 5, the ranking performance will be impacted. One possible explanation is that the total variance penalty dominates the overall ranking loss. In practice, many documents associated with a query in the training data are irrelevant documents. Given a large variance penalty, the total variance penalty derived from the predictions for irrelevant documents may dominate the overall ranking loss. In result, the model's probability in differentiating highly relevant documents from less relevant documents will be impacted.

In Fig. 3, we plot the performance of WassRank in terms of nDCG@10 with respect to the number of training epochs on MQ2007. Using the 5-fold cross validation strategy, the ranking performance is averaged across five folds per epoch with a step of 2. From Fig. 3, we can observe that the performance of WassRank in terms of nDCG@10 converges after about 100 epochs. This also indicates that using a pre-defined threshold of epochs can be used as an alternative way for training and evaluating WassRank in practice.

To summarize, the hyper-parameters, such as number of hidden layers, the regularization parameter and the variance penalty, significantly affect the performance of WassRank. Careful examinations of these factors are highly recommended when developing or extending models based on Wasserstein ranking loss.

## 7 CONCLUSIONS

In this paper, we propose a novel model called WassRank to solve the problem of listwise document ranking. The key idea is to utilize the tailored Wasserstein distance to measure the discrepancy between the ranking predicted by the ranking model and the ranking derived from the ground truth labels. Inspired by the entropic regularization technique by Cuturi [14], we are able to optimize the formulated Wassertein ranking loss by gradient descent. Moreover, by imposing the ranking-specific cost matrix, we take into account not only the ranking loss with respect to documents with different labels, but also the ranking loss with respect to documents of

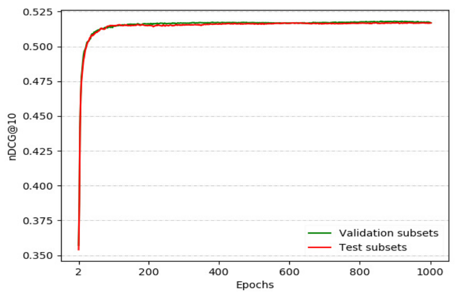


**Figure 3: Averaged nDCG@10 performance on validation subsets and test subsets.**

the same relevance level. We have shown that WassRank leads to substantially improved performance when compared to four representative listwise ranking methods. Also, our analysis indicates that WassRank is more capable of differentiating high-quality ranking from low-quality ranking than the baseline methods, such as ListNet and ListMLE. Since problems analogous to document ranking arise in a variety of applications, such as recommender systems and question answering, we believe that our method provides a new perspective for addressing problems of this kind.

For future work, the following practical issues are worthy to be investigated. First, in terms of scalability, we did not conduct an in-depth investigation on how to accelerate training due to the variable number of documents per query. However we do note that a number of strategies, such as parallelized stochastic gradient descent [59] and using MapReduce [55], can be applied. Second, instead of using the mapping function $\psi(\cdot)$ for deriving vectors within the probability simplex, it is interesting to extend the smoothed Wasserstein ranking loss to unnormalized relevance vectors. Then a more general ranking framework can be expected, which avoids the effort in finding the optimal mapping function. Third, based on the Rademacher Average technique [3, 4], we also plan to study the tightness of the generalization bound of WassRank, which can provide guidelines on framework design and parameter tuning.

## 8 ACKNOWLEDGMENTS

This research has been supported by JSPS KAKENHI Grant Number JP17K12784.

## REFERENCES

[1] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Proceedings of NIPS conference.* 1964–1974.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th ICML.* 214–223.

[3] Peter L. Bartlett and Shahar Mendelson. 2003. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research* 3 (2003), 463–482.

[4] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. 2004. Introduction to Statistical Learning Theory. *Advanced Lectures on Machine Learning* (2004), 169–207.

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

[6] Christopher J.C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of NIPS conference.* 193–200.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th ICML.* 129–136.

[8] Olivier Chapelle and Yi Chang. 2010. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the 2010 International Conference on YLRC.* 1–24.

[9] Olivier Chapelle, Quoc Le, and Alex Smola. 2007. Large margin optimization of ranking measures. In *NIPS workshop on Machine Learning for Web Search.*

[10] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th CIKM.* 621–630.

[11] Wei Chu and Zoubin Ghahramani. 2005. Gaussian Processes for Ordinal Regression. *Journal of Machine Learning Research* 6 (2005), 1019–1041.

[12] Wei Chu and S. Sathiya Keerthi. 2005. New Approaches to Support Vector Ordinal Regression. In *Proceedings of the 22nd ICML.* 145–152.

[13] David Cossock and Tong Zhang. 2006. Subset Ranking Using Regression. In *Proceedings of the 19th Annual Conference on Learning Theory.* 605–619.

[14] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Proceedings of NIPS 26.* 2292–2300.

[15] Julie Delon. 2006. Movie and video scale-time equalization application to flicker reduction. *IEEE Transactions on Image Processing* 15, 1 (2006), 241–248.

[16] Sira Ferradans, Gui-Song Xia, Gabriel Peyré, and Jean-François Aujol. 2013. Static and Dynamic Texture Mixing Using Optimal Transport. In *Scale Space and Variational Methods in Computer Vision.* 137–148.

[17] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 2003. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research* 4 (2003), 933–969.

[18] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso Poggio. 2015. Learning with a Wasserstein Loss. In *Proceedings of NIPS 28.* 2053–2061.

[19] Laura A. Granka, Thorsten Joachims, and Geri Gay. 2004. Eye-tracking Analysis of User Behavior in WWW Search. In *Proceedings of the 27th SIGIR.* 478–479.

[20] John Guiver and Edward Snelson. 2008. Learning to Rank with SoftRank and Gaussian Processes. In *Proceedings of the 31st SIGIR.* 259–266.

[21] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th CIKM.* 55–64.

[22] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Proceedings of NIPS 27.* 2042–2050.

[23] Gao Huang, Chuan Quo, Matt J. Kusner, Yu Sun, Kilian Q. Weinberger, and Fei Sha. 2016. Supervised Word Mover's Distance. In *Proceedings of NIPS conference.* 4869–4877.

[24] 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 *CIKM2013.* 2333–2338.

[25] Thomas Hurtut, Yann Gousseau, and Francis Schmitt. 2008. Adaptive image retrieval based on the spatial organization of colors. *Computer Vision and Image Understanding* 112, 2 (2008), 101–113.

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

[27] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th KDD.* 133–142.

[28] Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th KDD.* 217–226.

[29] Yanyan Lan, Tie-Yan Liu, Zhiming Ma, and Hang Li. 2009. Generalization Analysis of Listwise Learning-to-rank Algorithms. In *Proceedings of the 26th ICML.* 577–584.

[30] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. 2014. Position-aware ListMLE: A Sequential Learning Process for Ranking. In *Proceedings of the 30th Conference on UAI.* 449–458.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (2015), 436–444. https://doi.org/10.1038/nature14539

[32] Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing.* Vol. 4. Synthesis Lectures on Human Language Technologies.

[33] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval.* Springer.

[34] Grégoire Montavon, Klaus-Robert Müller, and Marco Cuturi. 2016. Wasserstein Training of Restricted Boltzmann Machines. In *Proceedings of NIPS conference.* 3718–3726.

[35] Ramesh Nallapati. 2004. Discriminative Models for Information Retrieval. In *Proceedings of the 27th SIGIR.* 64–71.

[36] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, et al. 2018. Neural Information Retrieval: At the End of the Early Years. *Journal of Information Retrieval* 21, 2-3 (2018), 111–182.

[37] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching As Image Recognition. In *Proceedings of AAAI Conference on Artificial Intelligence.* 2793–2799.

[38] Gabriel Peyré and Marco Cuturi. 2018. *Computational Optimal Transport.*

[39] Gabriel Peyré, Jalal Fadili, and Julien Rabin. 2012. Wasserstein active contours. In *19th IEEE International Conference on Image Processing.* 2541–2544.

[40] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Journal of Information Retrieval* 13, 4 (2010), 375–397.

[41] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. 2008. Query-level loss functions for information retrieval. *Information Processing and Management* 44, 2 (2008), 838–855.

[42] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. 2011. On NDCG Consistency of Listwise Ranking Methods. In *Proceedings of Machine Learning Research.* 618–626.

[43] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of TREC.*

[44] Antoine Rolet, Marco Cuturi, and Gabriel Peyré. 2016. Fast Dictionary Learning with a Smoothed Wasserstein Loss. In *Proceedings of the 19th International Conference on AIS.* 630–638.

[45] Libin Shen and Aravind K. Joshi. 2005. Ranking and Reranking with Perceptron. *Machine Learning* 60, 1-3 (2005), 73–96.

[46] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd WWW.* 373–374.

[47] Richard Sinkhorn. 1967. Diagonal Equivalence to Matrices with Prescribed Row and Column Sums. *The American Mathematical Monthly* 74, 4 (1967), 402–405.

[48] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28, 1 (1972), 11–21.

[49] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-smooth Rank Metrics. In *Proceedings of the 1st WSDM.* 77–86.

[50] Maksims N. Volkovs and Richard S. Zemel. 2009. BoltzRank: Learning to Maximize Expected Ranking Gain. In *Proceedings of ICML conference.* 1089–1096.

[51] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN. In *Proceedings of IJCAI conference.* 2922–2928.

[52] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating Non-sequential Behavior into Click Models. In *Proceedings of the 38th SIGIR.* 283–292.

[53] Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting Boosting for Information Retrieval Measures. *Journal of Information Retrieval* 13, 3 (2010), 254–270.

[54] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th ICML.* 1192–1199.

[55] Jia Xu, Bin Lei, Yu Gu, Marianne Winslett, Ge Yu, and Zhenjie Zhang. 2015. Efficient Similarity Join Based on Earth MoverâĂŹs Distance Using MapReduce. *IEEE Transactions on Knowledge and Data Engineering* 27, 8 (2015), 2148–2162.

[56] Jun Xu and Hang Li. 2007. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the 30th SIGIR.* 391–398.

[57] Fajie Yuan, Guibing Guo, Joemon Jose, Long Chen, Hai-Tao Yu, and Weinan Zhang. 2016. LambdaFM: Learning Optimal Ranking with Factorization Machines Using Lambda Surrogates. In *Proceedings of the 25th CIKM.* 227–236.

[58] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A Support Vector Method for Optimizing Average Precision. In *Proceedings of the 30th SIGIR.* 271–278.

[59] Martin A. Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. 2010. Parallelized Stochastic Gradient Descent. In *Proceedings of NIPS conference.* 2595–2603.