Listwise Learning to Rank Based on Approximate Rank Indicators

Thibaut Thonet[1*], **Yagmur Gizem Cinar**[2†], **Eric Gaussier**[3], **Minghan Li**[3], **Jean-Michel Renders**[1]

[1] NAVER LABS Europe [2] Amazon UK [3] Univ. Grenoble Alpes, CNRS

thibaut.thonet@naverlabs.com, cinary@amazon.com, eric.gaussier@univ-grenoble-alpes.fr,
minghan.li@univ-grenoble-alpes.fr, jean-michel.renders@naverlabs.com

Abstract

We study here a way to approximate information retrieval metrics through a softmax-based approximation of the rank indicator function. Indeed, this latter function is a key component in the design of information retrieval metrics, as well as in the design of the ranking and sorting functions. Obtaining a good approximation for it thus opens the door to differentiable approximations of many evaluation measures that can in turn be used in neural end-to-end approaches. We first prove theoretically that the approximations proposed are of good quality, prior to validate them experimentally on both learning to rank and text-based information retrieval tasks.

Introduction

Learning to rank (Liu 2011) is a sub-field of Machine Learning and Information Retrieval (IR) that aims at learning, from some training data, functions able to rank a set of objects – typically a set of documents for a given query. Learning to rank is currently one of the privileged approaches to build IR systems. This said, one important problem faced with learning to rank is that the metrics considered to evaluate the quality of a system, and the losses they underlie, are usually not differentiable. This is typically the case in IR: popular IR metrics such as precision at K, mean average precision or normalized discounted cumulative gain, are neither continuous nor differentiable. As such, state-of-the-art optimization techniques, such as stochastic gradient descent, cannot be used to learn systems that optimize their values.

To address this problem, researchers have followed two main paths. The first one consists in replacing the loss associated with a given metric by a *surrogate loss* which is easier to optimize. A surrogate loss typically upper bounds the true loss and, if consistent, asymptotically (usually when the number of samples tends to infinity) behaves like it. One of the main advantages of surrogate losses lies in the fact that it is sometimes possible to rely on an optimization problem that is convex and thus relatively simple to solve. However, Calauzènes et al. (Calauzènes, Usunier, and Gallinari 2012; Calauzènes and Usunier 2020) have shown that convex and consistent surrogate ranking losses do not always exist, as for example for the mean average precision or the expected reciprocal rank. The second solution is to identify *differentiable approximations* of the metrics considered. Typically, such approximations converge towards the true metrics when an hyperparameter that controls the quality of the approximation tends to a given value. One of the main advantages in using a differentiable approximation of a metric is the fact that one directly approximates the true loss, the quality of the approximation being controlled by an hyperparameter and not the number of samples considered. One of the main disadvantages of differentiable approximations is that the optimization problem obtained is in general non-convex. That said, the recent success of deep learning shows that solving non-convex optimization problems can nonetheless lead to state-of-the-art systems.

We follow here this latter path and study differentiable approximations of standard IR metrics. We focus on one ingredient at the core of ranking metrics: the rank indicator function. We show how one can define high-quality, differentiable approximations of the rank indicator and how these lead to good approximations of the losses associated with standard IR metrics. Our contributions are thus three-fold:

- We introduce SmoothI ("smoothie"), a novel differentiable approximation of the rank indicator function that can be used in various ranking metrics and losses.
- We furthermore show that this approximation, as well as the differentiable IR metrics and losses derived from it, converge towards their true counterpart with theoretical
- Lastly, we empirically illustrate the behavior of our proposal on both learning to rank features and standard, text-based features, and show that it is, in both cases, very competitive compared to previous approaches.

Related Work

Listwise approaches are widely used in IR as they directly address the ranking problem (Cao et al. 2007; Xia et al. 2008). A first category of methods developed for listwise learning to rank aimed at building surrogates for non-differentiable loss functions based on a ranking of the objects. In this line, RankCosine (Qin et al. 2008) used a loss

[*] Most of the work was done while the author was working at Univ. Grenoble Alpes.

[†] The work was done while the author was working at Univ. Grenoble Alpes.

function based on the cosine of two rank vectors while ListNet (Cao et al. 2007) adopted a cross-entropy loss. ListMLE and its extensions (Lan et al. 2014; Xia et al. 2008) introduced a likelihood loss and a theoretical framework for statistical consistency (extended in (Lan et al. 2009, 2012; Xia, Liu, and Li 2009)), while (Kar, Narasimhan, and Jain 2015) and (Bruch 2019; Ravikumar, Tewari, and Yang 2011; Valizadegan et al. 2009) studied surrogate loss functions for P@K and NDCG, respectively. Lastly, LambdaRank (Burges, Ragno, and Le 2007) used a logistic loss weighted by the cost, according to the targeted evaluation metric, of swapping two documents. This approach has then been extended to tree-based ensemble methods in LambdaMART (Burges et al. 2011), and finally generalized in LambdaLoss (Wang et al. 2018), the best performing method according to (Wang et al. 2018) in this family.

If surrogate losses are interesting as they can lead to simpler optimization problems, they are sometimes only loosely related to the target loss, as pointed out in Bruch et al. (2019). A typical example is the Top-K loss proposed in (Berrada, Zisserman, and Kumar 2018) (see also (Chen et al. 2009; Xu et al. 2008) for a study of the relations between evaluation metrics and surrogate losses). Furthermore, using a notion of consistency based on the concept of calibration developed in (Steinwart 2007), Calauzènes et al. (Calauzènes, Usunier, and Gallinari 2012; Calauzènes and Usunier 2020) have shown that convex and consistent surrogate ranking losses do not always exist, as for example for the mean average precision or the expected reciprocal rank. Researchers have thus directly studied differentiable approximations of loss functions and evaluation metrics, from SoftRank (Taylor et al. 2008), which proposed a smooth approximation of NDCG, to the recent differentiable approximation of MAP, called ListAP, in the context of image retrieval (Revaud et al. 2019). Some of the proposed approaches are based on a soft approximation of the position function (Wu et al. 2009) or of the rank indicator (Chapelle and Wu 2010), from which one can derive differentiable approximations of most standard IR metrics. However, (Wu et al. 2009) is specific to DCG whereas (Chapelle and Wu 2010) assumes that the inverse of the rank function is known. Qin, Liu, and Li (2010) proposed differentiable approximations of P@K, MAP, P@K and NDCG@K, recently used in (Bruch et al. 2019), based of the composition of two approximation functions, namely the position and the truncation functions. In contrast, our approach makes use of a single approximation, that of the rank indicator, for all losses and metrics considered, and thus reduces the risk of composing errors of different approximations.

More recently, different studies, mostly in the machine learning community, have been dedicated to differentiable approximations of the sorting and ranking functions. A fundamental relation between optimal transport and generalized sorting is for example provided in (Cuturi, Teboul, and Vert 2019), with an approximation based on Sinkhorn quantiles (note that (Yu et al. 2019) also exploits optimal transport for listwise document ranking, without however proving that the approximation used is correct). (Blondel et al. 2020) have focused on devising fast approximations of the sorting and ranking functions by casting differentiable sorting and ranking as projections onto the convex hull of all permutations. In the context of K-NN classification, Plötz and Roth (2018) proposed a recursive formulation of an approximation of the ranking function. However, no theoretical guarantees are provided, neither for this approximation nor for the K-NN loss it is used in. A more general framework, based on unimodal row-stochastic matrices, is introduced in (Grover et al. 2019) with an approximation of the sorting operator which is used in (Pobrotyn and Bialobrzeski 2021) to derive a differentiable approximation to NDCG. It can be shown that the approximate rank indicator matrix our approach leads to is a unimodal row-stochastic matrix, so that our proposal can be used in their framework as well. (Prillo and Eisenschlos 2020) further improved the above proposal by simplifying it, an approach referred to as SoftSort. Lastly, we want to mention the approach developed by Kong et al. (2020) who propose an adaptive projection method, called Rankmax, that projects, in a differentiable manner, a score vector onto the (n, k)-simplex. This method is particularly well adapted to multi-class classification. Its application to IR metrics remains however to be studied.

Differentiable IR Metrics

For a given query, an IR system returns a list of ranked documents. The ranking is based on scores provided by the IR system, scores that we assume here to be *strictly positive and distinct*[1] and that will be denoted by $\mathcal{S} = \{S_1, \ldots, S_N\}$ for a list of N documents. To assess the validity of an IR system, one uses gold standard collections in which the true relevance scores of documents are known, and IR metrics that assess to which extent the IR system is able to place documents with higher relevance scores at the top of the ranked list it returns. The most popular metrics are certainly the precision at K (denoted by P@K) which measures the precision in the list of top-K documents, its extension Mean Average Precision (MAP), as well the Normalized Discounted Cumulative Gain at K (NDCG@K) which can take into account graded relevance judgements.

P@K is the average over queries of P@K_q, defined for a given query q by:

$$\text{P@}K_q = \frac{1}{K} \sum_{r=1}^{K} rel_q(j_r), \tag{1}$$

where j_r is the r^{th} highest document in the list of scores $\mathcal{S}$ (*i.e.*, the document with the r^{th} largest score in $\mathcal{S}$), $rel_q(j)$ is a binary relevance score that is 1 if document j is relevant to q and 0 otherwise. MAP is the average over queries of AP_q defined by:

$$\text{AP}_q = \frac{1}{\sum_{j=1}^{N} rel_q(j)} \sum_{K=1}^{N} rel_q(j_K) P@K_q, \tag{2}$$

[1]This is not a restriction *per se* as one can add an arbitrary large value to the scores without changing their ranking, and ties can be broken randomly.

The normalized discounted cumulative gain at rank K, NDCG@K, is the average over queries of NDCG@K_q, defined for a given query q by:

$$\text{NDCG@}K_q = \frac{1}{N_K^q} \sum_{r=1}^{K} \frac{2^{rel_q(j_r)} - 1}{\log_2(k+1)}, \tag{3}$$

where $rel_q(j)$ is now a (not necessarily binary) positive, bounded relevance score for document j with respect to query q (higher values correspond to higher relevance) and N_K^q a query-dependent normalizing constant. The standard NDCG metric corresponds to NDCG@N (Järvelin and Kekäläinen 2002).

As the reader may have noticed, the common building block and main ingredient of the above IR metrics (Eqs. 1, 2, 3) is the relevance score of the document at any rank r, namely $rel_q(j_r)$. If one can define a "good" differentiable approximation of $rel_q(j_r)$, then one obtains a "good" differentiable approximation of IR metrics. The goal of this paper is to introduce such a differentiable approximation, while giving "good" a precise meaning.

SmoothI: Smooth Rank Indicators

The relevance score of the document at any rank r in a list of N documents can be rewritten as:

$$rel_q(j_r) = \sum_{j=1}^{N} rel_q(j) I_j^r,$$

where I_j^r is the rank indicator function at rank r, which is equal to 1 if j is the r^{th} highest document in the list and 0 otherwise. Thus, the rank indicator function at rank r can be defined by:

$$I_j^r = \begin{cases} 1 & \text{if } j = \underset{\substack{j' \in \{1,\dots,N\} \\ \forall l < r, I_{j'}^l = 0}}{\text{argmax}} \; S_{j'}, \\ 0 & \text{otherwise.} \end{cases}$$

Given the strict positivity assumption on the scores, the argmax above can be equivalently expressed as:

$$\underset{\substack{j' \in \{1,\dots,N\} \\ \forall l < r, I_{j'}^l = 0}}{\text{argmax}} \; S_{j'} = \underset{j' \in \{1,\dots,N\}}{\text{argmax}} \; S_{j'} \prod_{l=1}^{r-1} (1 - I_{j'}^l),$$

as the product $\prod_{l=1}^{r-1}(1 - I_{j'}^l)$ is 0 for the $(r-1)$ highest documents.

A widespread smooth approximation of the argmax is the parameterized softmax. It has been employed in, *e.g.*, (Moradi Fard, Thonet, and Gaussier 2018) in the context of deep k-means clustering, in (Plötz and Roth 2018) in the context of neural nearest neighbor networks, as well as in (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2017) within a Gumbel-softmax distribution employed to approximate categorical samples. The parameterized softmax defines, for any rank $r \in \{1, \dots, N\}$ and document $j \in \{1, \dots, N\}$, a smooth rank indicator $I_j^{r,\alpha}$ of the form:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j \prod_{l=1}^{r-1}(1 - I_j^{l,\alpha})}}{\sum_{j'} e^{\alpha S_{j'} \prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha})}},$$

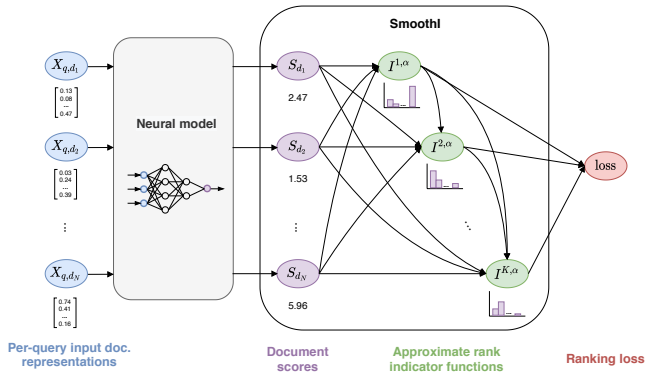

Figure 1: Illustration of SmoothI and its positioning in a neural retrieval system. Given a query q, the document representations $\{X_{q,d_i}\}_{i=1}^N$ are first passed through a neural model which outputs a set of scores $\{S_{d_i}\}_{i=1}^N$. The scores are then processed by the SmoothI module, yielding smooth rank indicators $\{I^{r,\alpha}\}_{r=1}^K$ up to rank K, which are ultimately used to calculate the ranking loss.

where α is an hyperparameter that plays the role of an inverse temperature guaranteeing that $I_j^{r,\alpha}$ converges to the true rank indicator function I_j^r (for any document j and rank r) when $\alpha \to +\infty$.

Numerical approximations in the above formulation may however lead in practice to choosing a document at a given rank r that was already selected at a lower rank. Indeed, it may be that for a document j of rank $l' < r$, $I_j^{l',\alpha}$ gets a value slightly below 1, with the risk that $S_{j'} \prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha})$ takes the highest value for $j' = j$ and, in turn, that j is selected for both ranks l' and r. We thus slightly modify the above formulation by introducing an additional hyperparameter, leading, for any rank $r \in \{1, \dots, N\}$ and document $j \in \{1, \dots, N\}$, to:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j \prod_{l=1}^{r-1}(1 - I_j^{l,\alpha} - \delta)}}{\sum_{j'} e^{\alpha S_{j'} \prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha} - \delta)}}. \tag{4}$$

The hyperparameter $\delta \in (0, 0.5)$ controls the mass of the distribution that is allocated to the $(r-1)$ highest documents: a larger δ leads to further reducing the contribution of the $(r-1)$ highest documents in the distribution at rank r. We refer to the above approximation of the rank indicator function as SmoothI.

The following theorem, the proof of which is given in the Supplementary Material, states that $I_j^{r,\alpha}$ given in Eq. 4 plays the role of a smooth, differentiable approximation of the true rank indicator I_j^r:

Theorem 1. *For any $r \in \{1, \dots, N\}$, and $j \in \{1, \dots, N\}$:*

(i) $I_j^{r,\alpha}$ is differentiable wrt any score in $\mathcal{S}$,

(ii) $\lim_{\alpha \to +\infty} I_j^{r,\alpha} = I_j^r$.

Figure 1 further shows how SmoothI can be integrated in a neural retrieval system. It simply consists of the last element

that is used to compute the overall loss corresponding to the desired IR metrics.

Quality of the Approximation

Interestingly, $I_j^{r,\alpha}$ is a *good* approximation to I_j^r as the error decreases exponentially with α, as stated in the following theorem, the proof of which is given in the Supplementary Material[2].

Theorem 2. *Let S_{min} be the smallest score in $\mathcal{S}$ and β the minimal ratio between scores S_j and $S_{j'}$ when $S_j > S_{j'}$: $\beta = \min_{(j,j'),\, S_j > S_{j'}} \frac{S_j}{S_{j'}}$. Furthermore, let $c = (\frac{\beta+1}{2})^{\frac{1}{K-1}}$, where K is the rank at which the IR metric is considered, and let $\gamma = \min\left\{\delta,\, 0.5-\delta,\, (1-\delta)\frac{c-1}{c+1}\right\}$. If:*

$$\text{(C1)} \quad \alpha > \frac{2^{K-1}\left[\log(K-1) - \log\gamma\right]}{S_{min}\min\left\{1, \frac{\beta-1}{2}\right\}},$$

then:

$$\forall r \in \{1,\dots,K\}, \forall j \in \{1,\dots,K\},\ |I_j^r - I_j^{r,\alpha}| \leq \epsilon_{\alpha,K},$$

with $\epsilon_{\alpha,K} = (K-1)e^{-\alpha\frac{S_{min}}{2^{K-1}}\min\left\{1,\frac{\beta-1}{2}\right\}}$.

Note that, due to the exponential function, both the right-hand side of Condition (C1) and $\epsilon_{\alpha,K}$ can be made as small as one wants by increasing α or, equivalently, rescaling the scores of the documents without changing their ranking.

Lastly, the following corollary shows that the same approximation quality and exponential speed of convergence holds for compositions of linear combinations and Lipschitz functions of the rank indicators, which are widely used in different ranking and sorting functions.

Corollary 1. *For $K \in \{1,\dots,N\}$, let $\mathbf{I} = \{I_j^r\}_{1\leq r\leq K, 1\leq j\leq N}$ and $\mathbf{I}^\alpha = \{I_j^{r,\alpha}\}_{1\leq r\leq K, 1\leq j\leq N}$. Consider the function h such that $h(\mathbf{I};\mathbf{a},\mathbf{b}) = \sum_{r=1}^K a_r g(\sum_{j=1}^N b_j I_j^r)$, where g is a Lipschitz function with Lipschitz constant ℓ, and $\mathbf{a} = \{a_r\}_{r=1}^K$ and $\mathbf{b} = \{b_j\}_{j=1}^N$ are real-valued constants. Then:*

$$|h(\mathbf{I};\mathbf{a},\mathbf{b}) - h(\mathbf{I}^\alpha;\mathbf{a},\mathbf{b})| \leq \left(\sum_{r=1}^K |a_r|\right)\left(\sum_{j=1}^N |b_j|\right)\ell\epsilon_{\alpha,K}.$$

Gradient Stabilization in Neural Architectures

In pilot experiments, we found that the recursive computation in $I_j^{r,\alpha}$ (Eq. 4) could sometimes lead to numerical instability when computing its gradient with respect to the scores $\mathcal{S}$. To alleviate this issue, we adopted a simple solution which consists in applying the stop-gradient operator to $\prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha} - \delta)$ in the definition of $I_j^{r,\alpha}$ to "prune" the computation graph in the backward pass. This operator, which was used in previous works such as (van den Oord, Vinyals, and Kavukcuoglu 2017), acts as the identity function in the forward pass and sets the partial derivatives of its argument to zero in the backward pass, leading to the following slightly modified definition of $I_j^{r,\alpha}$ which we use in practice:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j \text{sg}\left[\prod_{l=1}^{r-1}(1-I_j^{l,\alpha}-\delta)\right]}}{\sum_{j'} e^{\alpha S_{j'} \text{sg}\left[\prod_{l=1}^{r-1}(1-I_{j'}^{l,\alpha}-\delta)\right]}},$$

where sg$[\cdot]$ is the stop-gradient operator. In other words, we consider that the lower-rank smooth indicators $I_{j'}^{l,\alpha}$ $(l < r)$ in $I_j^{r,\alpha}$ are constant with respect to $\mathcal{S}$.

Application to IR Metrics

Based on the proposed smooth rank indicators, one can obtain simple approximations of IR metrics by replacing $rel_q(j_r)$ with $\sum_{j=1}^N rel_q(j) I_j^{r,\alpha}$, leading to the following approximation for P@K_q:

$$\text{P@}K_q^\alpha = \frac{1}{K}\sum_{r=1}^K\sum_{j=1}^N rel_q(j) I_j^{r,\alpha},$$

from which one obtains the following approximation of AP$_q$:

$$\text{AP}_q^\alpha = \frac{1}{\sum_{j=1}^N rel_q(j)}\sum_{K=1}^N\left(\sum_{j=1}^N rel_q(j) I_j^{K,\alpha}\right)\text{P@}K_q^\alpha.$$

Similarly, the approximation for NDCG@K_q is given by:

$$\text{NDCG@}K_q^\alpha = \frac{1}{N_K^q}\sum_{r=1}^K \frac{2^{\sum_{j=1}^N rel_q(j) I_j^{r,\alpha}} - 1}{\log_2(k+1)}.$$

A direct application of Corollary 1 leads to:

$$|\text{P@}K - \text{P@}K^\alpha| \leq m\epsilon_{\alpha,K},$$

where m is the average number of relevant documents per query ($m = \frac{1}{Q}\sum_{q=1}^Q\sum_{j=1}^N rel_q(j)$). This leads for MAP to:

$$|\text{MAP} - \text{MAP}^\alpha| \leq N(m+1)\epsilon_{\alpha,N}.$$

For NDCG@K, using a Taylor expansion of the function 2^x around $x = 0$, one gets:

$$|\text{NDCG@}K - \text{NDCG@}K^\alpha| \leq N\epsilon_{\alpha,K}.$$

This shows that the approximations obtained for P@K, MAP and NDCG@K (and *a fortiori* NDCG) are of exponential quality.

Experiments

We conducted both feature-based learning to rank and text-based IR experiments to validate SmoothI's ability to define high-quality differentiable approximations of IR metrics, and hence meaningful listwise losses. In particular, our evaluation seeks to address the following two questions: On learning to rank collections, how does SmoothI compare to state-of-the-art listwise approaches? Do neural models for text-based IR (*e.g.*, models based on BERT) benefit from SmoothI's listwise loss?

In the remainder of this section, we first describe the experimental setup of the learning to rank experiments, then we discuss the learning to rank results. Finally, we detail our experiments with BERT on text-based IR.

[2]Note that the proof of this theorem requires $0 < \delta < 0.5$, hence the condition $\delta \in (0, 0.5)$ mentioned before.

Learning to Rank Experimental Setup

Datasets. To evaluate our approach, we conducted learning to rank experiments on standard, publicly available datasets, namely LETOR 4.0 MQ2007, MQ2008 and MSLR-Web30K (Qin and Liu 2013), respectively containing 1,692/69,623, 784/15,211 and 31,531/3,771,125 queries/documents, and the Yahoo learning to rank Set-1 dataset (Chapelle and Chang 2010), containing 29,921/709,877 queries/documents. In these datasets, each query-document pair is associated with a feature vector. We rely on the standard 5-fold train/validation/test split for the LETOR collections and the standard train/validation/test split for YLTR. In the remainder, MSLR-Web30K and Yahoo learning to rank Set-1 will respectively be referred to as Web30K and YLTR.

The statistics of the different datasets for their respective folds are further detailed in Section 3 of the Supplementary Material.

Baseline methods. Differentiable approximations of IR metrics (and their associated losses) can be classified under two categories: surrogate losses and direct approximations. Among approaches based on surrogate losses, we have retained the three state-of-the-art approaches **ListNET** (Cao et al. 2007), **ListMLE** (Xia et al. 2008), and **LambdaLoss** (Wang et al. 2018). The latter is considered the best performing method in the Lambda* family (LambdaRank, LambdaMART, LambdaLoss) (Wang et al. 2018), and we thus omit the comparison against LambdaMART and LambdaRank. Among approaches based on direct approximations, we have retained the recently proposed **ListAP** (Revaud et al. 2019) and the state-of-the-art method **Approx** (Qin, Liu, and Li 2010), which was also recently used in (Bruch et al. 2019). In addition, we also considered losses derived from recent approaches for differentiable sorting and ranking (Blondel et al. 2020; Cuturi, Teboul, and Vert 2019; Prillo and Eisenschlos 2020). We used here as baselines the most recent representatives of these approaches, namely **OT** (Cuturi, Teboul, and Vert 2019), which frames differentiable sorting as an optimal transport problem, **FastSort** (Blondel et al. 2020), which devises an efficient differentiable approximation based on projections onto the convex hull of permutations, and **SoftSort** (Prillo and Eisenschlos 2020), which proposes a continuous relaxation of the sorting operator based on unimodal row-stochastic matrices and is comparable, both in terms of method and results, to the NeuralSort method introduced in (Grover et al. 2019).

As LambdaLoss, Approx and SmoothI can be used to approximate different IR metrics, we defined several variants for each approach, respectively optimizing P@$\{1, 5, 10\}$, NDCG@$\{1, 5, 10, N\}$ and MAP. The losses defined by ListNET, ListAP[3], Approx and SmoothI were implemented in PyTorch (Paszke et al. 2019), using our own implementation for ListNET, Approx and SmoothI[4]. For ListMLE and LambdaLoss, we relied on the TF-Ranking library (Pasumarthi et al. 2019). OT[5], FastSort[6] and SoftSort[7] all propose differentiable approximations of the position function. This is the same position function as the one used by Approx for computing an approximation of NDCG@N. We thus directly used this latter approximation from the outputs of OT, FastSort and SoftSort (note that Approx uses, in addition to the approximation of the position function, an approximation of the truncation function for computing approximations of truncated IR-metrics P@K and NDCG@K (Qin, Liu, and Li 2010)). For evaluating the performance of the different approaches, we used the fast Python implementation of the TREC evaluation tool (Van Gysel and de Rijke 2018), which calls the trec_eval evaluation metrics[8] from Python.

Lastly, in order to have a fair comparison of the losses defined by the different methods in the context of modern neural end-to-end approaches, we used the same fully-connected feedforward neural network for all methods. It is composed of an input layer followed by batch normalization, a 1024-dimensional hidden layer with ReLU activation again followed by batch normalization, and a fully-connected output layer that provides a score for each document in the list. The different hyperparameters and additional details on the learning to rank experimental setup are summarized in Section 4 of the Supplementary Material.

Learning to Rank Results

In this section, we study the retrieval performance of the different learning to rank losses derived from SmoothI and baseline approaches. Table 1 presents the learning to rank results, averaged over 5 folds for MQ2007, MQ2008 and Web30K, each fold using a different random initialization, and averaged over five random initializations for YLTR. We reported the significance using a paired Student t-test with Bonferroni correction at 5% significance level. For space reasons, we only show in Table 1 the mean results according to P@$\{1, 5\}$, NDCG@$\{1, 5, N\}$, and let the reader refer to Section 5 of the Supplementary Material for the P@10, MAP and NDCG@10 metrics as well as the standard errors around the mean. For LambdaLoss, Approx and SmoothI, Table 1 contains the best results obtained across the variants – optimizing different metrics – of each approach.

As one can notice, SmoothI is the best performing method on MQ2007, MQ2008 and Web30K. On MQ2007 and MQ2008, SmoothI outperforms all other methods for P@$\{1, 5\}$ and NDCG@$\{1, 5, N\}$. Approx is, on these collections, the second best method. On Web30K, SmoothI significantly outperforms all methods on P@1, NDCG@1, NDCG@5, and all methods but Approx on P@5 and NDCG. The results are more contrasted on YLTR. On the one hand, SmoothI and ListMLE are on par according to the NDCG-based metrics as they respectively obtained the best performance in terms of NDCG@$\{1, 5\}$ and NDCG@N, with sig-

[3]https://github.com/almazan/deep-image-retrieval

[4]https://github.com/ygcinar/SmoothI

[5]https://github.com/google-research/google-research/tree/master/soft_sort

[6]https://github.com/google-research/fast-soft-sort

[7]https://github.com/sprillo/softsort

[8]https://github.com/usnistgov/trec_eval

Table 1: Learning to rank retrieval results. Mean test performance is calculated over 5 folds for MQ2007, MQ2008 and Web30K, and 5 random initializations for YLTR as no predefined folds are available. Standard errors as well as additional results are provided in the Supplementary Material. Best results are in bold and "†" indicates a model significantly worse than the best one according to a paired t-test with Bonferroni correction at 5%.

	P@1	P@5	NDCG@1	NDCG@5	NDCG	P@1	P@5	NDCG@1	NDCG@5	NDCG
			MQ2007					MQ2008		
ListNet	0.463	0.412†	0.420	0.422†	0.603†	0.392†	0.318†	0.339†	0.422†	0.514†
ListMLE	0.442†	0.397†	0.395†	0.405†	0.594†	0.415†	0.337†	0.365†	0.445†	0.526†
LambdaLoss	0.452†	0.403†	0.407†	0.415†	0.601†	0.441	0.337†	0.385	0.457†	0.540
ListAP	0.457†	0.405†	0.405†	0.414†	0.600†	0.420	0.330†	0.371	0.442†	0.532†
Approx	0.479	0.419	0.430	0.430	0.611	0.457	0.349	0.401	0.471	0.549
OT	0.451†	0.405†	0.406†	0.414†	0.602†	0.431	0.342†	0.382	0.461†	0.542
FastSort	0.461	0.405†	0.413†	0.417†	0.599†	0.430	0.332†	0.371	0.450†	0.537†
SoftSort	0.469	0.413†	0.425	0.426†	0.608	0.411†	0.335†	0.360†	0.449†	0.534†
SmoothI (ours)	**0.488**	**0.424**	**0.441**	**0.439**	**0.612**	**0.459**	**0.353**	**0.402**	**0.477**	**0.550**
			Web30K					YLTR		
ListNet	0.694†	0.649†	0.496†	0.483†	0.741†	0.858†	0.814†	0.726†	0.741†	0.857†
ListMLE	0.620†	0.544†	0.404†	0.383†	0.646†	**0.874**	**0.829**	0.724†	0.746	**0.859**
LambdaLoss	0.697†	0.617†	0.497†	0.466†	0.691†	0.868†	0.822†	0.731†	0.743†	0.854†
ListAP	0.715†	0.658†	0.503†	0.483†	0.733†	0.820†	0.768†	0.685†	0.686†	0.820†
Approx	0.767†	0.716	0.544†	0.523†	0.754	0.870	0.828	0.731†	0.745†	0.858
OT	0.682†	0.637†	0.459†	0.456†	0.729†	0.846†	0.793†	0.710†	0.719†	0.842†
FastSort	0.722†	0.660†	0.525†	0.494†	0.738†	0.857†	0.812†	0.724†	0.729†	0.851†
SoftSort	0.724†	0.669†	0.521†	0.500†	0.747†	0.861†	0.814†	0.729†	0.739†	0.854†
SmoothI (ours)	**0.776**	**0.717**	**0.552**	**0.530**	**0.754**	0.869†	0.826†	**0.735**	**0.748**	0.858

nificant differences only at cutoff 1. On the other hand, in terms of precision-based metrics, ListMLE outperformed all other approaches except Approx.

Turning to the listwise losses obtained from the differentiable sorting approaches (OT, FastSort, and SoftSort), we observe that these methods demonstrate competitive performance on the learning to rank task. SmoothI nonetheless outperformed all of these approaches, in particular on Web30K on which the differences are significant for all metrics. In summary, over all the collections, we conclude that SmoothI proves to be very competitive on learning to rank with respect to traditional listwise losses and differentiable sorting approaches.

As a complement to this experiment, we investigate in Section 6 of the Supplementary Material how the choice of the optimized metric influences SmoothI's performance. This study highlights that optimizing NDCG@N leads to the best performance according to any IR metric and thus constitutes an overall safe choice. We also discuss the efficiency of the different approaches in Section 7 of the Supplementary Material. Overall, all approaches but ListMLE, LambdaLoss and OT – which are significantly slower than the other approaches on different datasets – are comparable and scale reasonably well.

Experiments on Text-based IR

To further validate the efficacy of SmoothI, we conducted experiments on text-based information retrieval, *i.e.*, with raw texts as input. In particular, the task consists here in optimizing a given neural model to appropriately rank the documents for each query, where the documents and queries are raw texts. This differs from the previous sections which focus on feature-based learning to rank, *i.e.*, where each query-document pair is represented by a feature vector.

Experimental setup. The standard TREC Robust04 collection, which consists of 250 queries and 0.5M documents, is used here as the text-based IR collection. For queries, we used the keyword version which corresponds to the title fields of the TREC topics (Dai and Callan 2019; McDonald, Brokos, and Androutsopoulos 2018). We experimented with **vanilla BERT** as the neural ranking model, using the pretrained uncased BERT-base version. This model is at the core of recent state-of-the-art IR models (Devlin et al. 2019; Dai and Callan 2019; MacAvaney et al. 2019; Li et al. 2020). We make use here of the version proposed by (MacAvaney et al. 2019), which is slightly better than the other ones.

Most text-based IR neural models based on BERT are trained with a pointwise or pairwise loss, and not a listwise loss (Li et al. 2020; MacAvaney et al. 2019). This is not really surprising as the calculation of the loss requires that the representations of all the documents to be ranked for a query hold together in memory, which can lead to a prohibitive memory cost for BERT if the list of documents associated to a query is large. To overcome this potential problem when using a listwise loss such as SmoothI, we computed the loss only on the documents of the training batch, where each batch contains two pairs of (relevant, non-relevant) documents associated to one query. For each query, one thus has a list of four documents, which are all fed to the vanilla BERT model as a list of query-document pairs. The input of the BERT models for each query-document pair is obtained by

Table 2: Text-based retrieval results on Robust04: mean test performance $\pm$ standard error calculated over 5 folds. The best results are in bold and "$\dagger$" indicates a model significantly worse than the best one according to a paired t-test at 5%.

	P@1	P@5	P@10	P@20	MAP
vanilla-BERT (MacAvaney et al. 2019)	0.631±0.026	0.544±0.028	0.474±0.028†	0.396±0.019	0.236±0.006†
vanilla-BERT (Approx-NDCG@N loss)	**0.651±0.023**	0.529±0.020†	0.465±0.025†	0.392±0.021†	0.237±0.008†
vanilla-BERT (SmoothI-NDCG@N loss)	0.635±0.014	**0.562±0.025**	**0.494±0.024**	**0.407±0.019**	**0.245±0.007**
	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
vanilla-BERT (MacAvaney et al. 2019)	0.592±0.022	0.528±0.024	0.493±0.023†	0.464±0.020†	0.434±0.010
vanilla-BERT (Approx-NDCG@N loss)	**0.602±0.017**	0.521±0.017†	0.490±0.020†	0.462±0.018†	0.436±0.010
vanilla-BERT (SmoothI-NDCG@N loss)	0.601±0.010	**0.548±0.017**	**0.515±0.019**	**0.480±0.017**	**0.441±0.007**

concatenating the [CLS] token, query tokens, the [SEP] token and document tokens. From BERT's output [CLS] vector, a dense layer generates the relevance score for the corresponding query-document pair. Following MacAvaney et al. (2019), documents are truncated at 800 tokens in order to handle documents longer than the capacity of BERT. In this case, a document is split into two inputs and the [CLS] vectors from each split are averaged to get BERT's output [CLS] vector.

We use as baselines the standard vanilla BERT model (MacAvaney et al. 2019) as well as its version with Approx-NDCG@N, which is the second best performing method in our previous comparison. We compare both approaches to the vanilla BERT with SmoothI-NDCG@N, the best method overall in our previous comparison. All models are trained for 100 epochs using Adam optimizer with a learning rate of $2 \cdot 10^{-5}$ for BERT, as suggested in MacAvaney et al. (2019), and 10^{-3} for the top dense layer, which is a common default value. As mentioned before, the batch size is set to four and gradient accumulation is used every eight steps (MacAvaney et al. 2019). We furthermore followed a five-fold cross validation protocol: the models are trained on the training set (corresponding to three folds), tuned on the validation set (one fold) with early stopping, and evaluated on the test set (the remaining fold). We use the standard re-ranking setting and re-rank the top-150 documents returned by BM25 (Robertson and Walker 1994). The hyperparameters α and δ for SmoothI are set to 1.0 and 0.1 respectively. The hyperparameter α for Approx-NDCG@N is set to 1.0 (note that only one hyperparameter is needed for approximating NDCG@N with Approx as the second hyperparameter relates to the truncation function used for the truncated IR-metrics P@K and NDCG@K (Qin, Liu, and Li 2010)). The random seed integer was set to 66 and we ran our experiments on an Intel Xeon server with a Nvidia GTX 1080 Ti GPU.

Results. Table 2 reports the text-based retrieval performance, averaged over 5 folds, of the standard vanilla BERT model and the two vanilla BERT models with the listwise losses Approx-NDCG@N and SmoothI-NDCG@N. The best results are in bold and "$\dagger$" indicates a model significantly worse than the best one according to a paired t-test at 5%. Note that we observed no significant differences between Approx-NDCG@N and the pairwise hinge loss. In contrast, one can observe that vanilla BERT performs better when it is trained with SmoothI-NDCG@N and achieves the highest scores on all metrics but P@1 and NDCG@1 for which it is on par with the other approaches. The improvement over the original vanilla BERT model with the pairwise hinge loss is in particular significant on P@10, MAP, NDCG@10 and NDCG@20. The improvement over Approx-NDCG@N is significant on P@5, P@10, P@20, MAP, NDCG@5, NDCG@10 and NDCG@20. Furthermore, the vanilla BERT model with SmoothI-NDCG@N achieves 0.480 on NDCG@20 on the TREC Robust04 collection, which is the best result this model has achieved on this collection to our knowledge (Devlin et al. 2019; MacAvaney et al. 2019; Ma et al. 2020).

Conclusion

We presented in this study a unified approach to build differentiable approximations of IR metrics (P@K, MAP and NDCG@K) on the basis of an approximation of the rank indicator function. We further showed that the errors associated with these approximations decrease exponentially with an inverse temperature-like hyperparameter that controls the quality of the approximations. We also illustrated the efficacy and efficiency of our approach on four standard collections based on learning to rank features, as well as on the popular TREC Robust04 text-based collection. All in all, our proposal, referred to as *SmoothI*, constitutes a tool for differentiable ranking that proved very competitive compared with previous approaches on several collections, either based on learning to rank or textual features.

We also want to stress that the approach we proposed is more general and can directly be applied to other losses, such as the K-NN loss studied in Grover et al. (2019), and functions that are directly based on the rank indicator. Among such functions, we are particularly interested in the ranking function, which aims at ordering the documents in decreasing order of their scores, the sorting function, which aims at ordering the scores, and the position function, which aims at providing, for each document, its rank in the ordered list of scores. We plan to study, on the basis of the development given in this paper, differentiable approximations of these functions in a near future.

Acknowledgements This research was partly supported by MIAI@Grenoble Alpes (ANR-19-P3IA-0003).

References

Berrada, L.; Zisserman, A.; and Kumar, M. P. 2018. Smooth Loss Functions for Deep Top-k Classification. In *Proceedings of the 6th International Conference on Learning Representations*.

Blondel, M.; Teboul, O.; Berthet, Q.; and Djolonga, J. 2020. Fast Differentiable Sorting and Ranking. In *Proceedings of the 37th International Conference on Machine Learning*, 950–959.

Bruch, S. 2019. An Alternative Cross Entropy Loss for Learning-to-Rank. *arXiv:1911.09798*.

Bruch, S.; Zoghi, M.; Bendersky, M.; and Najork, M. 2019. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Burges, C. J. C.; Ragno, R.; and Le, Q. V. 2007. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*.

Burges, C. J. C.; Svore, K. M.; Bennett, P. N.; Pastusiak, A.; and Wu, Q. 2011. Learning to Rank Using an Ensemble of Lambda-Gradient Models. *Journal of Machine Learning Research*.

Calauzènes, C.; and Usunier, N. 2020. On ranking via sorting by estimated expected utility. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*.

Calauzènes, C.; Usunier, N.; and Gallinari, P. 2012. On the (Non-)existence of Convex, Calibrated Surrogate Losses for Ranking. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 197–205.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning*, 129–136.

Chapelle, O.; and Chang, Y. 2010. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge*, 1–24.

Chapelle, O.; and Wu, M. 2010. Gradient Descent Optimization of Smoothed Information Retrieval Metrics. *Information Retrieval*, 13(3): 216–235.

Chen, W.; Liu, T.; Lan, Y.; Ma, Z.; and Li, H. 2009. Ranking Measures and Loss Functions in Learning to Rank. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 315–323.

Cuturi, M.; Teboul, O.; and Vert, J.-P. 2019. Differentiable Ranking and Sorting using Optimal Transport. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*.

Dai, Z.; and Callan, J. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 985–988.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics)*, 4171–4186.

Grover, A.; Wang, E.; Zweig, A.; and Ermon, S. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *Proceedings of the 7th International Conference on Learning Representations*.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the 5th International Conference on Learning Representations*.

Järvelin, K.; and Kekäläinen, J. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4).

Kar, P.; Narasimhan, H.; and Jain, P. 2015. Surrogate Functions for Maximizing Precision at the Top. In *Proceedings of the 32nd International Conference on Machine Learning*, 189–198.

Kong, W.; Krichene, W.; Mayoraz, N.; Rendle, S.; and Zhang, L. 2020. Rankmax: An Adaptive Projection Alternative to the Softmax Function. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*.

Lan, Y.; Guo, J.; Cheng, X.; and Liu, T. 2012. Statistical Consistency of Ranking Methods in A Rank-Differentiable Probability Space. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 1241–1249.

Lan, Y.; Liu, T.; Ma, Z.; and Li, H. 2009. Generalization analysis of listwise learning-to-rank algorithms. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 577–584.

Lan, Y.; Zhu, Y.; Guo, J.; Niu, S.; and Cheng, X. 2014. Position-Aware ListMLE: A Sequential Learning Process for Ranking. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 449–458.

Li, C.; Yates, A.; MacAvaney, S.; He, B.; and Sun, Y. 2020. PARADE: Passage representation aggregation for document reranking. *arXiv:2008.09093*.

Liu, T.-Y. 2011. *Learning to rank for information retrieval*. Springer. ISBN 978-3-642-14266-6.

Ma, X.; Guo, J.; Zhang, R.; Fan, Y.; Ji, X.; and Cheng, X. 2020. PROP: Pre-training with Representative Words Prediction for Ad-hoc Retrieval. *arXiv:2010.10137*.

MacAvaney, S.; Yates, A.; Cohan, A.; and Goharian, N. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1101–1104.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of the 5th International Conference on Learning Representations*.

McDonald, R.; Brokos, G.; and Androutsopoulos, I. 2018. Deep Relevance Ranking Using Enhanced Document-Query

Interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1849–1860.

Moradi Fard, M.; Thonet, T.; and Gaussier, E. 2018. Deep k-Means: Jointly Clustering with k-Means and Learning Representations. *arXiv:1806.10069*.

Pasumarthi, R. K.; Bruch, S.; Wang, X.; Li, C.; Bendersky, M.; Najork, M.; Pfeifer, J.; Golbandi, N.; Anil, R.; and Wolf, S. 2019. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2970–2978.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 8024–8035.

Plötz, T.; and Roth, S. 2018. Neural Nearest Neighbors Networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*.

Pobrotyn, P.; and Bialobrzeski, R. 2021. NeuralNDCG: Direct Optimisation of a Ranking Metric via Differentiable Relaxation of Sorting. *CoRR*, abs/2102.07831.

Prillo, S.; and Eisenschlos, J. 2020. SoftSort: A Continuous Relaxation for the argsort Operator. In *Proceedings of the 37th International Conference on Machine Learning*, 7793–7802.

Qin, T.; and Liu, T. 2013. Introducing LETOR 4.0 Datasets. *arXiv:1306.2597*.

Qin, T.; Liu, T.-Y.; and Li, H. 2010. A General Approximation Framework for Direct Optimization of Information Retrieval Measures. *Information Retrieval, 13(4)*.

Qin, T.; Zhang, X.; Tsai, M.; Wang, D.; Liu, T.; and Li, H. 2008. Query-level loss functions for information retrieval. *Information Processing & Management*, 44(2): 838–855.

Ravikumar, P.; Tewari, A.; and Yang, E. 2011. On NDCG Consistency of Listwise Ranking Methods. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 618–626.

Revaud, J.; Almazán, J.; Rezende, R. S.; and de Souza, C. R. 2019. Learning With Average Precision: Training Image Retrieval With a Listwise Loss. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision*, 5106–5115.

Robertson, S. E.; and Walker, S. 1994. Some Simple Effective Approximations to the 2-poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 232–241.

Steinwart, I. 2007. How to compare Different Loss Functions and Their Risks. *Constructive Approximation*, 26(2): 225–287.

Taylor, M.; Guiver, J.; Robertson, S.; and Minka, T. 2008. SoftRank: Optimizing Non-Smooth Rank Metrics. In *Proceedings of the 1st International Conference on Web Search and Data Mining*, 77–86.

Valizadegan, H.; Jin, R.; Zhang, R.; and Mao, J. 2009. Learning to Rank by Optimizing NDCG Measure. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*.

van den Oord, A.; Vinyals, O.; and Kavukcuoglu, K. 2017. Neural Discrete Representation Learning. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, 6309–6318.

Van Gysel, C.; and de Rijke, M. 2018. Pytrec_eval: An Extremely Fast Python Interface to trec_eval. In *Proceedings of the 41st International ACM SIGIR conference on Research and Development in Information Retrieval*.

Wang, X.; Li, C.; Golbandi, N.; Bendersky, M.; and Najork, M. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.

Wu, M.; Chang, Y.; Zheng, Z.; and Zha, H. 2009. Smoothing DCG for Learning to Rank: A Novel Approach Using Smoothed Hinge Functions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 1923–1926.

Xia, F.; Liu, T.; and Li, H. 2009. Statistical Consistency of Top-k Ranking. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 2098–2106.

Xia, F.; Liu, T.; Wang, J.; Zhang, W.; and Li, H. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*.

Xu, J.; Liu, T.; Lu, M.; Li, H.; and Ma, W. 2008. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 107–114.

Yu, H.; Jatowt, A.; Joho, H.; Jose, J. M.; Yang, X.; and Chen, L. 2019. WassRank: Listwise Document Ranking Using Optimal Transport Theory. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 24–32.