Information Processing and Management 59 (2022) 102962

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

ListMAP: Listwise learning to rank as maximum a posteriori estimation

Sanaz Keshvari [a], Faezeh Ensan [b,*], Hadi Sadoghi Yazdi [a]

[a] Ferdowsi University of Mashhad, Mashhad, Iran
[b] Ryerson University, Toronto, Canada

ARTICLE INFO

Keywords:
Learning to rank
Listwise loss function
Prior distribution

ABSTRACT

Listwise learning to rank models, which optimize the ranking of a document list, are among the most widely adopted algorithms for finding and ranking relevant documents to user information needs. In this paper, we propose ListMAP, a new listwise learning to rank model with prior distribution that encodes the informativeness of training data and assigns different weights to training instances. The main intuition behind ListMAP is that documents in the training dataset do not have the same impact on training a ranking function. ListMAP formalizes the listwise loss function as a maximum a posteriori estimation problem in which the scoring function must be estimated such that the log probability of the predicted ranked list is maximized given a prior distribution on the labeled data. We provide a model for approximating the prior distribution parameters from a set of observation data. We implement the proposed learning to rank model using neural networks. We theoretically discuss and analyze the characteristics of the introduced model and empirically illustrate its performance on a number of benchmark datasets; namely MQ2007 and MQ2008 of the Letor 4.0 benchmark, Set 1 and Set 2 of the Yahoo! learning to rank challenge data set, and Microsoft 30k and Microsoft 10K datasets. We show that the proposed models are effective across different datasets in terms of information retrieval evaluation metrics NDCG and MRR at positions 1, 3, 5, 10, and 20.

1. Introduction

Ranking algorithms, as the core of web search systems, are responsible for finding and ranking the most relevant documents to the user information needs from the crawled and indexed corpus. Learning to Rank (L2R) models, which learn a ranking model from the training data, are among the widely studied techniques in the field of ranking algorithms (Capannini et al., 2016; Liu, 2010; Tax, Bockting, & Hiemstra, 2015). Recently, L2R models have gained even more attention given the huge amount of labeled data extractable from explicit or implicit feedback generated by users on the relevance of documents to queries (Guo et al., 2020).

Based on the adopted approach for training ranking models, L2R algorithms are categorized into *pointwise*, *pairwise*, and *listwise* models. Listwise L2R algorithms (Cao, Qin, Liu, Tsai, & Li, 2007; Guo et al., 2020; Xia, Liu, Wang, Zhang, & Li, 2008) take ranked lists of documents as input and use them for estimating a loss function such as log-likelihood (Cao et al., 2007; Mollica & Tardella, 2021) or cross-entropy (Bruch, 2021; Li, Wang, Fleming, Thomas, & Cheung, 2019; Luo, Wang, Liu, & Pan, 2015; Xia et al., 2008) based on permutation probabilities. The dominant trend in existing listwise learning to rank models is to formalize loss without considering a prior distribution over training data, i.e., it is been assumed that all training data are coming from a uniform distribution and

* Corresponding author.
E-mail address: fensan@ryerson.ca (F. Ensan).

https://doi.org/10.1016/j.ipm.2022.102962
Received 9 November 2021; Received in revised form 28 February 2022; Accepted 24 April 2022
Available online 13 May 2022

hence, all instances of the training data have the same weight and impact on the training process, and consequently, the special characteristics of the training dataset are ignored in learning a ranking model.

Modeling characteristics of the training dataset has been a topic of interest in the learning to rank community in recent years for modeling position bias (Joachims, Swaminathan, & Schnabel, 2017; Wang, Bendersky, Metzler, & Najork, 2016; Wang, Golbandi, Bendersky, Metzler, & Najork, 2018; Yue, Patel, & Roehrig, 2010) and presentation bias (Li et al., 2020; Yue et al., 2010; Zhou et al., 2018; Zoghi et al., 2017) in click data. The work presented in Ai, Yang, Wang, and Mao (2021), Joachims et al. (2017), Oosterhuis and de Rijke (2018) and Wang et al. (2018) attempt at extracting unbiased document relevance from the biased implicit feedback extracted from user clicks by modeling user browsing behavior, formulating click models, and assigning different probabilities to training instances based on the presentation order. The work presented in Ai, Bi, Luo, Guo, and Croft (2018b) attempts at defining unbiased ranking loss by estimating a *propensity* model, i.e., the probability of observing a document for a query based on the relevance of the document and its position in the representation that has been offered to users.

Despite the existing rich body of work on unbiased learning to rank for click data, very few works explored the problem of modeling training data in general (Ding, Geng, & Zhang, 2015). Labeled data extracted from explicit feedback in learning to rank datasets can be noisy or in different levels of informativeness due to different labeling difficulties such as short and ambiguous queries (Cronen-Townsend, Croft, et al., 2002), problems in resolving disagreement between human judges, and subjective and non-clear boundaries between different labels (e.g., *completely relevant* and *somehow relevant*) (Ding et al., 2015). Similarly, machine-generated labeled data that is extracted from click logs can be out-of-date, noisy, and non-informative even after modeling position and selection bias, due to usual noises perceived from existing search engine data (Han, Hwang, Song, & Kim, 2020). Ignoring different levels of informativeness in training data will damage the performance of learning to rank algorithms.

In this paper, we attempt at approaching this issue by defining a new listwise loss function with prior distribution that encodes the informativeness of training data. Here, the main intuition is that documents in the training dataset do not have the same impact on training a ranking function. In our learning to rank model, the prior distributions of data labels and document scores are learnt from a set of observations derived from the training data. Likewise a number of prominent listwise models (Xia et al., 2008; Zhu & Klabjan, 2020a), we use the Plackett-Luce model (Plackett, 1975) for defining the permutation probability distribution. Based on the Bayes theorem, we incorporate a Gama distribution, as the prior distribution, into the loss function and define ListMAP, a new listwise learning to rank model on this basis. Based on how the prior distribution is learnt from the observations, three L2R models, ListMAP$_{SP}$, ListMAP$_{LP}$, and ListMAP$_{SiLP}$, are defined. We theoretically analyze the characteristics of the introduced loss functions and show that minimizing the ListMAP$_{SiLP}$ loss is consistent with the correct ranking order where a dataset is *coherent*.

We evaluate the introduced L2R models performance on the following six benchmark datasets, MQ2007, MQ2008, Yahoo!Set1, Yahoo!Set2, MSLR10K, and MSLR30K. These datasets are widely used as benchmark document collections for evaluating L2R models (Ai, Bi, Guo, & Croft, 2018a; Ai et al., 2018b; Bruch, 2021; Dai et al., 2020; Pang, Xu, Ai, Lan, Cheng, & Wen, 2020; Zhuang, Wang, Bendersky, & Najork, 2020). We show that our method outperforms the state-of-the-art L2R models based on different evaluation metrics.

The major contributions of this paper can be summarized as follows:

- We introduce a new listwise leaning to rank model that includes a prior distribution over training data, i.e., it assigns different weights to training instances. We also provide a model for approximating the prior distribution parameters from a set of observation data.
- We theoretically discuss and analyze the characteristics of the introduced L2R models with respect to coherency, i.e. whether minimizing a loss is consistent with the correct ranking order.
- We empirically illustrate the performance of the proposed learning to rank models on a number of benchmark datasets in terms of information retrieval evaluation metrics and compare them with prominent and state-of-the-art baselines. We also analyze the prior distributions learnt over different benchmark datasets, and report the number of queries whose performance are hurt or improved by incorporating the prior distribution into the list wise loss function.

This paper is organized as follows: The following section reviews related works on listwise learning to rank, modeling training click data and modeling training dataset characteristics. Section 3 introduces the proposed model, ListMAP, for learning a ranking model with encoding informativeness of training data. Section 4 reports the comprehensive experiments we conducted to evaluate the proposed learning model. Finally, Section 5 concludes the paper.

2. Related work

Learning to rank models learn a scoring or a ranking function from training data, which is represented as query-document feature vectors and their multi-level labels usually range from 0 to an integer number indicating how much related is a document to a given query (Tax et al., 2015). In traditional L2R methods, features are human-engineered (Liu, 2010), while in more recent neural IR methods these features are learnt from the training data along with the ranking model (Guo et al., 2020). Regardless of whether features are human-crafted or are automatically learnt, learning to rank methods can be categorized into three main groups of pointwise, pairwise, and listwise models based on their learning objective.

In pointwise models, the ranking problem is treated as a regression or a multi-class classification problem in which the relevance label of each document for a given query is predicted based on document-query features and without considering document ranks and preferences. The work presented in Cossock and Zhang (2006) and Friedman (2001) are samples of pointwise L2R with human-engineered features, and Severyn and Moschitti (2015) is a sample of neural pointwise L2R models. In pairwise models, the

learning objective is to predict the preferences between pairs of documents for a given query instead of predicting a label for each document. RankSVM (Chapelle & Keerthi, 2010), LamdaMart (Burges, 2010), RankBoost (Freund, Iyer, Schapire, & Singer, 2003), and RankNET (Cao et al., 2007) are samples of pairwise models with human-crafted features and Ai et al. (2019) and Dehghani, Zamani, Severyn, Kamps, and Croft (2017) are samples of neural pairwise methods. The learning objective in listwise models is to optimize the ranking of the document list retrieved for a query. Listwise algorithms have generally reported a better performance than pointwise and pairwise models (Guo et al., 2020; Qin, Liu, Xu, & Li, 2010). The work presented in Cao et al. (2007) and Xia et al. (2008) are samples of listwise learning to rank models with human-engineered features and the work in Chen and Zhou (2020) and Zhu and Klabjan (2020a) report neural L2R models with listwise loss functions.

The learning model introduced in this paper is a listwise one that is implemented using a neural network architecture. In the following, we thoroughly review two important related topics to our work: we first review listwise loss functions in L2R models. Second, we review existing works that model training datasets characteristics.

2.1. Listwise learning to rank

Listwise learning models range from works that define a listwise loss function (Burges, 2010; Liu, 2010) to those that directly optimize Information Retrieval (IR) performance measures such as MAP and NDCG (Ravikumar, Tewari, & Yang, 2011; Taylor, Guiver, Robertson, & Minka, 2008). Since our work is closely related to the former group of listwise methods, we provide a review of the prominent works based on listwise loss function. For a complete review of other listwise models please see Burges (2010) and Liu (2010).

Existing listwise loss functions mostly employ a permutation probability distribution model such as Plackett-Luce (Plackett, 1975) for defining $P(\pi|x;g)$, where x denotes a set of documents to be ranked in the input space, π denotes a permutation of documents in the output space, and g denotes the scoring function. ListNet (Cao et al., 2007), a prominent listwise algorithm, defines a top-one probability distribution over a set of documents based on permutation probabilities. Here, $P_s(j)$, a top one probability for the document j in a ranked list sorted by s, is defined as the sum of the probabilities of permutations in which j is ranked on the top of the list. ListNet minimizes the cross-entropy of top-one probabilities of predictions and labels. More specifically, assuming that y^i is a list of labels assigned to documents for query #i, z^i is a list of scores obtained by g for query #i, and n is the number of documents ranked for this query, the listwise loss function is defined as follows:

$$L(y^i, z^i) = -\sum_{j=1}^{n} P_{y^i}(j) log P_{z^i}(j) \tag{1}$$

The main challenge of ListNet is that Minimizing the defined loss function is not always result in a correct ranking (Zhu & Klabjan, 2020a). ListMLE (Xia et al., 2008) is a listwise model whose loss function is consistent with the correct ranking. It defines the loss function based on the likelihood of the permutation of a correct ranking. Given π_{y^i} is a list of documents sorted by the relevance labels (y^i) for query #i, x denotes the documents in the input space, and $g(x)$ is a scoring function defined over x, the listMLE loss function is defined as follows:

$$L(g; x, \pi_{y^i}) = -log\ P(\pi_{y^i}|x;g) \tag{2}$$

where $P(\pi_{y^i}|x;g)$ is defined based on the Plackett–Luce model as follows:

$$P(\pi_{y^i}|x;g) = \prod_{j=1}^{|\pi_{y^i}|} \frac{exp(g(x_j))}{\sum_{k=j}^{|\pi_{y^i}|} exp(g(x_k))} \tag{3}$$

where x_j denotes the jth document in the π_{y^i}.

During recent years, many alternative listwise loss functions have been proposed to improve the performance of ListMLE and ListNet. The listwise models presented in Li et al. (2019) and Luo et al. (2015) aim at improving the computation cost of ListNet by sampling a subset of training instances in model training. The listwise model presented in Jagerman, Kiseleva, and de Rijke (2017), focuses on label ambiguities, i.e., when the ground truth labels are the same for documents. This model attempts at learning no preference between documents that have the same label by employing a sampling method on permutations. The listwise loss function proposed in Zhu and Klabjan (2020b) addressees the same issue of documents that have the same label with respect to a query. It proposes a model that can be trained in order of the number of labels assigned to documents instead of the order of documents. WassRank (Yu et al., 2019) is another listwise learning to rank model that minimize the Wasserstein distance between the predicted list and the ground truth relevance scores. In this paper, we approach the problem of listwise learning to rank from a different viewpoint, i.e., modeling training data characteristics in the learning process. We compare our model with some of the prominent listwise rankings in Section 4 and show its effectiveness in ranking documents across different datasets.

Deep learning architectures for listwise learning have gained increasing attention in recent years (Rahimi, Montazeralghaem, & Allan, 2019). BanditRank (Gampa & Fujita, 2021) trains neural networks by directly maximizing IR measures by reformulating the retrieval problem as a reinforcement learning algorithm. DeepRank (Pang, Lan, Guo, Xu, Xu, & Cheng, 2017) is a deep architecture consists of three components for (1) detecting the query-centric context, (2) measuring the relevance of a query and each context, and (3) aggregating local signals and generate a final ranking score. SetRank (Pang et al., 2020) is another deep learning network that attempts at modeling cross-document interaction in its scoring method by a multi-head self attention architecture. Here, the main hypothesis is that the relevance of a document to a query is dependent to the relevance of documents seen before. The deep listwise

Table 1
Comparative analysis of existing works.

Model	Model type	Modeling data characteristics
Lawrence and Schölkopf (2001)	Not a learning to rank model	Assigning a probability of the label being flipped to each training data and incorporating that in a Log-likelihood Objective function.
Ding et al. (2015)	Extends several pointwise and pair-wise learning to rank models	Estimating the joint probability of the feature vector of a document and the relevance Label for a given query
Carvalho, Elsas, Cohen, and Carbonell (2008)	Pairwise	Sigmoid loss function
Pasumarthi et al. (2019)	Propose pointwise, pairwise, and listwise learning to rank models	Sigmoid loss function
ListMAP	Listwise	Prior Distribution on training data

context model proposed in Ai et al. (2018a) encodes the information in pseudo relevance feedback for the purpose of re-ranking. The AttentionRank loss function proposed in Ai et al. (2018a) aims at finding the relative importance of a document in a given ranked list retrieved for a query. Pobrotyn, Bartczak, Synowiec, Białobrzeski, and Bojar (2020) proposed a self-attentive neural network model that learns a document score based on the relevance of all other documents in the retrieved list of documents for a given query, and results in a significant performance improvement when incorporated with different listwise loss functions. Pobrotyn and Białobrzeski (2021) proposed a new ranking loss function, NeuralNDCG, which is based on minimizing a differentiable approximation to NDCG. A comprehensive review of neural IR models and architectures can be found in Guo et al. (2020). The loss function proposed in this paper can be employed in any neural framework that uses a listwise loss function (Ai et al., 2018a). In this paper, we use the self-attentive neural architecture proposed in Pobrotyn et al. (2020) for implementing the proposed listwise loss function.

2.2. Modeling training dataset characteristics

The approach adopted by our work is to estimate a prior distribution on document labels or scores and use it for defining a listwise loss function. Prior distributions on item labels have been used in contexts other than L2R before. In Lawrence and Schölkopf (2001), the parameters of a prior distribution on item labels are optimized through an expectation-maximization (EM) algorithm, when the prior is used for item classification. In Guiver and Snelson (2009) a Gamma prior distribution is defined on training data and is exploited on defining permutation probabilities.

The work presented in Ding et al. (2015) is among few works in the literature that addresses modeling training dataset in learning to rank. This work extends pairwise and pointwise learning to rank algorithms to encompass an estimated probability of labeling noise in their loss functions. For estimating labeling noises, Ding et al. (2015) propose a model to learn a query-specific joint probability of feature vectors and relevance labels. The parameters of this model are estimated by means of maximum likelihood estimation on the set of all documents retrieved for the given query. Carvalho et al. (2008), address the problem of robustness of ranking model with the presence of outliers in training data by using a non-linear sigmoid function in the RankSVM learning to rank algorithm instead of the hinge loss. They show that the new loss function does not give larger loss values to large negative scores and in this sense is robust to outliers. The same idea has been adopted in Svore, Volkovs, and Burges (2011) for extending the LambdaMART pairwise learning to rank algorithm. The application of the sigmoid function in different ranking models have been further analyzed in Pasumarthi et al. (2019) and is a baseline in our experiments.

Table 1 provides a comparative analysis of the learning to rank models described in this section, and their important features. This table also includes ListMAP, the learning to rank model which is introduced in this paper and is going to be illustrated in the next section.

3. Proposed method

In this section, we first motive the main intuition behind our proposed learning to rank algorithm through an example. Second, we introduce our proposed model, ListMAP, a listwise L2R model for learning a ranking model from training data. Third, we analyze the main characteristics of the ranking model through different examples.

3.1. Motivating example

The main intuition behind our learning to rank algorithm is to assign different weights to training instances. One of the expected results of this approach is to reduce the impact of documents with possible wrong labels on the training process. As a motivating example, assume that the training data includes the following queries, $\{q_1, q_2, q_3\}$, where the permutation of each query is as follows: $\pi(q_1) = \{4, 2, 1, 0\}$, $\pi(q_2) = \{4, 4, 4, 4\}$, $\pi(q_3) = \{0, 0, 0, 0\}$. Here, the list of $\{4, 2, 1, 0\}$ for q_1 means that the ground truth permutation for q_1 consists of four documents whose labels are equal to $4, 2, 1$ and 0, respectively. Without considering a prior distribution on training instances, all query-document pairs have the same impact in the learning process. For the sake of this example, assume that the dominant pattern in data is to have documents with the grater labels in the higher ranks of the queries' permutations. For example,

based on our assumption, the data inclines to have documents with the label of 4 in the first position of queries' permutations, and data with the label of 0 in the lower position of the permutations. Based on this pattern, our model assigns different prior probabilities to q_1, q_2, and q_3 and their associated documents, i.e., q_1 whose first document in the ranked list has the label of 4 and whose last document has the label of 0 has a greater weight in the learning process than q_2 whose last document has a label of 4, or q_3 whose first document has the label of 0. In other words, assuming that the relevance labels are reliable in general, our algorithm first learns a pattern on labels of documents in different ranks of queries permutations from a parameter setting data set and then apply that pattern to weight training data. In our example, we assumed that the learnt pattern tends to consider the forth document in the q_2 permutation and the first document in the q_3 permutation as probable labeling errors and assigns them a lower weight in training.

The next section thoroughly explain our learning to rank model.

3.2. ListMAP learning to rank model

Let $q \in Q = \{q_1, \dots, q_m\}$ denotes a query in the set of m queries in the training set, $D_q = \{d_1, \dots, d_n\}$ is the set of n documents associated with q. Further assume that $Y_q = \{y_1, \dots, y_n\}$ denotes the labels of the n documents in D_q, D is the set of all documents, and x_d represents the set of feature vector for the document $d \in D_q$. Also, assume that f is a ranker function that assigns a score to x_d for all $d \in D_q$. The task of the ranking model is to find a ranked list of documents sorted based on the score generated by f, such that the learning objective is optimized. The learning objective in our listwise model is defined based on the ranked list generated by f, and a permutation π_q of D_q in which documents are sorted based on Y_q. We call π_q a ground truth permutation, because it is sorted based on the labels. We use $\pi_q(d)$ for denoting the rank of d in the permutation π_q.

We define the ListMap loss function for a given query q as follows:

$$L_q(f; Y_q, D_q) = -log\ P(f, D_q, Y_q|\pi_q) \tag{4}$$

where π_q is a permutation sorted based on the labels Y_q.

In this learning to rank model, we formalize the loss function as a maximum a posteriori estimation (MAP) problem in which the scoring function f must be estimated such that the log probability of the predicted ranked list generated by f is maximized given the ground truth permutation.

Based on Bayesian rules, we can reformulate the loss function as Eq. (5).

$$L_q(f; Y_q, D_q) = -log\ \Big(P(f, D_q, Y_q)\ P(\pi_q|f, D_q, Y_q)\Big) \tag{5}$$

Note that the loss function defined in Section 3.2 leads to the same log likelihood approach adopted by ListMLE with an important difference that in ListMAP, $P(f, D_q, Y_q)$ does not come from a uniform distribution. Based on the ListMAP definition, $P(f, D_q, Y_q)$ is the prior probability on training documents associated with q, and $P(\pi_q|f, D_q, Y_q)$ is the likelihood.

Following the same approach adopted by a number of L2R models for using the Placket-Luce model for defining permutation probabilities (Cao et al., 2007; Zhu & Klabjan, 2020a), we define the likelihood function, $P(\pi_q|f, d, Y_q)$, as follows for $d \in D_q$:

$$P(\pi_q|f, d, Y_q) = \frac{\phi(f(x_d))}{\Sigma_{k=\pi_q(d)}^{k=|\pi_q|}\phi(f(x_d))} \tag{6}$$

where $\phi(.)$ is an increasing positive function and k iterates over all documents in the ground truth permutation of q from the rank of d in the permutation, denoted as $\pi_q(d)$, to the last document in the permutation, denoted as $|\pi_q|$. We use an exponential function for $\phi(.)$ in our model and experiments.

For estimating the prior probability, we employ the Gamma distribution. Gamma distribution has been used as a basis for defining prior probabilities on permutations before (Guiver & Snelson, 2009). In this paper, we adopted the same approach as (Guiver & Snelson, 2009) and use Gamma distribution over the permutation probability defined in Eq. (6).

In our learning to rank model, we provide two different definitions for the prior functions based on the Gamma distribution. In the first definition, Eq. (7), the prior probability is estimated over the scoring function. Here, the assumption is that the documents in the ground truth permutation are such that the scoring function f over their features follow a Gamma distribution. We name this prior distribution as the *score prior* and denote it as $P^S(f, D_q, Y_q)$. In the second definition, Eq. (8), the prior probability is estimated over the labels of documents. Here, the assumption is that the labels of training data in different ranks of the ground truth permutation comes from a Gamma distribution, i.e., some labels are more probable in specific ranks while the others are less probable in the same ranks. We name this prior distribution as the *label prior* and denote it as $P^L(f, D_q, Y_q)$.

Let $d \in D_q$ be a document and x_d be the feature vector of d, and $\phi(.)$ is an increasing positive function adopted in Eq. (6), we define $P^S(f, D_q, Y_q)$ based on the Gamma distribution as follows:

$$\begin{aligned} P^S(f, D_q, Y_q) &\approx \prod_{d \in D_q} Gamma(d; f|\alpha_d, \beta_d) \\ &\approx \prod_{d \in D_q} \frac{\beta_d^{\alpha_d}}{\Gamma(\alpha_d)} \phi(f(x_d))^{\alpha_d - 1} e^{-\beta_d \phi(f(x_d))} \end{aligned} \tag{7}$$

Furthermore, given Y_d is a positive number representing the label of d in training dataset, we define $P^L(f, D_q, Y_q)$ as follows:

$$\begin{aligned} P^L(f, D_q, Y_q) &\approx \prod_{d \in D_q} Gamma(d; Y_d | \alpha_d, \beta_d) \\ &\approx \prod_{d \in D_q} \frac{\beta_d^{\alpha_d}}{\Gamma(\alpha_d)} Y_d^{\alpha_d - 1} e^{-\beta_d Y_d} \end{aligned} \tag{8}$$

Based on two prior probabilities defined in these equations and given $\phi(.)$ is defines as an exponential function, we reformulate the ListMAP loss function defined in 3.2 as Eqs. (9) and (10).

$$L_q^S(f; Y_q, D_q) = -\log\left(\prod_{d \in D_q} \frac{exp(f(x_d))}{\Sigma_{k=\pi_q(d)}^{k=|\pi_q|} exp(f(x_d))} P^S(f, D_q, Y_q) \right) \tag{9}$$

$$L_q^S(f; Y_q, D_q) = -\log\left(\prod_{d \in D_q} \frac{exp(f(x_d))}{\Sigma_{k=\pi_q(d)}^{k=|\pi_q|} exp(f(x_d))} P^L(f, D_q, Y_q) \right) \tag{10}$$

In Eqs. (7) and (8), both the shape parameter α and the rate parameter, β, are estimated based on d. Estimating Gamma distribution parameters have been the subject of different research so far (Lawless, 1980; Song, 2008). In our learning to rank algorithm, we adopt the maximum likelihood estimation method proposed in Ye and Chen (2017) in the context of our problem. Here, we assume that there are m observations, $O = \{O_1, ..O_m\}$, derived from the parameter setting data (the data we use for parameter setting in our experiments and is completely exclusive from the training data), from which we can estimate α and β for each document in the training data. We assume that O is coming from the same distribution that our training data are supposed to come from. According to Ye and Chen (2017), $\hat{\alpha}$ and $\hat{\beta}$ are estimations for the Gamma distributions as follows:

$$\hat{\alpha} = \frac{m \Sigma O_i}{m \Sigma O_i \log O_i - \Sigma \log O_i \Sigma O_i} \tag{11}$$

$$\hat{\beta} = \frac{1}{m^2} \left(m \Sigma O_i \log O_i - \Sigma \log O_i \Sigma O_i \right) \tag{12}$$

In our learning to rank algorithm, we contextualize Eqs. (11) and (12) based on documents, queries, labels and the scoring function f. Given q be a query in the set of our observations, $\phi(.)$ is an increasing positive function adopted in Eq. (6), D_q be all documents retrieved for q, and π_q is a permutation on D_q derived from labels assigned to them, α and β for the score prior are estimated as follows:

$$\hat{\alpha}_d^S = \frac{|Q| \sum_{q \in Q} \phi(f(x_{sd_q}))}{|Q| \sum_{q \in Q} \phi(f(x_{sd_q})) \log \phi(f(x_{sd_q})) - \sum_{q \in Q} \log \phi(f(x_{sd_q})) \sum_{q \in Q} \phi(f(x_{sd_q}))} \tag{13}$$

where:

sd_q $\{d_q \in D_q$ such that $\pi_q(d_q) = \pi(d)\}$

In Eq. (13), sd_q means the document in D_q whose index in the permutation of query q is equal to the index of the given document d in its permutation in training data. Similarly, β is estimated based on Eq. (12):

$$\hat{\beta}_d^S = \frac{1}{|Q|^2} \left(|Q| \sum_{q \in Q} \phi(f(x_{sd_q})) \log \phi(f(x_{sd_q})) - \sum_{q \in Q} \log \phi(f(x_{sd_q})) \sum_{q \in Q} \phi(f(x_{sd_q})) \right) \tag{14}$$

where:

sd_q $\{d_q \in D_q$ such that $\pi_q(d_q) = \pi(d)\}$

Furthermore, α and β for the label prior are estimated as follows

$$\hat{\alpha}_d^L = \frac{|Q| \sum_{q \in Q} Y_{sd_q}}{|Q| \sum_{q \in Q} Y_{sd_q} \log Y_{sd_q} - \sum_{q \in Q} \log Y_{sd_q} \sum_{q \in Q} Y_{sd_q}} \tag{15}$$

$$\hat{\beta}_d^L = \frac{1}{|Q|^2} \left(|Q| \sum_{q \in Q} Y_{sd_q} \log Y_{sd_q} - \sum_{q \in Q} \log Y_{sd_q} \sum_{q \in Q} Y_{sd_q} \right) \tag{16}$$

Here, we adopted this assumption that the training data, if not affected by labeling errors, comes from a gamma distribution whose parameters are learnt by Eqs. (13) and (14) or (15) and (16). For example, the most related documents to the queries in our observations, i.e., the documents in the index 1th in the permutations of all observed queries, are used for estimating the Gamma distribution parameters for the documents in the index 1th of permutations in the training dataset. In our experiment, we also test a simplified method for estimating the parameters of the label prior probability by assuming that the difference between the scale of the Gamma distribution over labels are ignorable, i.e., although the shape of the Gamma distribution (stated by α) differs for each

index in the ground truth permutation, the scale (stated by β) can be assumed to be the same for all indexes. Eq. (17) formulates this Simplified Label (SiL) prior probability.

$$\begin{aligned} P^{\hat{S}iL}(f, D_q, Y_q) &\approx \prod_{d \in D_q} Gamma(d; Y_d | \alpha_d, \beta_d) \\ &\approx \prod_{d \in D_q} \frac{\beta^{\alpha_d}}{\Gamma(\alpha_d)} Y_d^{\alpha_d - 1} e^{-\beta Y_d} \end{aligned} \tag{17}$$

Based on the parameter estimation introduced for the label prior, α and β are estimated based on the labels of each index in the permutation derived from the observation dataset. Hence, instead of α_d and β_d, in Eqs. (13), (14), (15) and (16), and instead of α_d in Eq. (17), we can use α_i and β_i, where i is the index of d in the ground truth permutation. In other words, in the label prior probability, the parameters are only dependent to the index of documents in the permutation. On the other hand, the introduced formalization for the score prior defines α and β based on the $f(x)$ applied on documents in the ground truth permutation where $f(x)$ is jointly learnt with α and β on the parameter setting dataset. Contrary to the label prior probabilities, in both regular and simplified versions, where Gamma distribution parameters can be learnt from the observation data, for the score prior probability there are is no scoring function f that can be applied to the observation data and used for obtaining Gamma parameters (Because we have not trained the model on training data yet). Here, we jointly train the scoring function f and the Gamma distribution parameters on the observation data. This f is going to be trained using the exact same loss function, but because it has been trained over the observation data not training data and because it has been trained for the final purpose of setting Gamma distribution parameters, it will not be used in testing our model. Instead, when the Gamma distribution parameters have been set up, the scoring function will be trained again on the training data using the Gamma parameters.

In this paper, we name the introduced listwise learning to rank model as ListMAP and based on the exploited prior distribution in the listwise model, it is called ListMAP$_{SP}$, ListMAP$_{LP}$, and ListMAP$_{SiLP}$, when the loss function uses the score prior distribution, the label prior distribution, and the simplified label distribution; respectively.

3.3. Analysis of ListMAP

The probability density function for the Gamma distribution, which is used for defining $P(f, D_q, Y_q)$ gets different shapes based on the α parameter. Fig. 1 shows an example of two possible density functions and their corresponding $\hat{P^L}(f, D_q, Y_q)$ for documents d_i and d_j, given the query of q. Here, i, and j are the rank of d_i and d_j in the ground truth permutation, respectively. For estimating $\alpha^L_{d_i}$, the label of all documents placed in the rank of i of the ground truth permutation of all queries in the parameter setting dataset are used according to Eq. (15). In case we are estimating $P^S(f, D_q, Y_q)$, f has to be applied on all documents placed in the rank of i of the ground truth permutation of all queries in the parameter setting dataset according to Eqs. (13) and (14). In the example of Fig. 1, the blue plot shows the prior distribution based on the α estimated for the document placed in the ith position of the ground truth permutation for a sample dataset that has 10 queries, where the labels of documents for the ith rank in the observation data are as follows: $[8, 8, 8, 8, 8, 8, 8, 6, 6, 2]$. Also, in this figure, the red plot shows the prior distribution based on the α estimated for the document placed in the jth position of the ground truth permutation for the same dataset whose labels of documents for the jth rank are as follows: $[8, 4, 2, 2, 2, 2, 2, 1, 1]$. As we can see from this figure, the labels of documents in the training dataset are in line with the labels of the observations from which the prior probability parameters have been set. In this example, the label of d_i is 8, which is close to the dominant label pattern of observations in the ith position. Similarly, the label of d_j is 2, which is close to the dominant label pattern of observations in the jth position. Observably, these labels are considered as correct and got a high prior probability based on our model.

Fig. 2 shows another example, where the red plot represents the exact same prior probability as Fig. 1, but the label of d_j is 8. Here, the label of the document in the training data for the jth index got a different value than the dominant pattern in the observed data and hence it gets a relatively low prior probability, which means it can be a less informative instance or an outlier. In addition, in this figure, the green dashed line shows the prior probability based on the α estimated for the document placed in the kth position of the ground truth permutation for the same dataset of Example in Fig. 1, where the labels of documents in the kth position are: $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$. Here, the observation data for the kth index is not coherent, i.e. there is no dominant pattern for the labels in this position. That is in contrary with the label prior probabilities estimated for the ith and jth ranks in the permutation in the example of Fig. 1, which are coherent. Assuming β is identical for all labels, since the average of Y_{d_i} is greater than the average of Y_{d_j} in observations, the prior probability for d_i bends to the right to assign higher probability to the values closer to Y_{d_i}. Similarly, the prior probability for d_j bends to the left to assign higher probability to values closer to Y_{d_j}. In this case, $\alpha^L_j \leq \alpha^L_i$ (See Gamma Distribution properties (Thom, 1958)). Definition 3.1 formally defines *coherent* datasets. In Section 4 we analyze our experimental datasets with respect to this attribute.

Definition 3.1 (*Coherent Dataset*)**.** A dataset is n-coherent for the top n positions if for any $i \leq n$ and $j \leq n$, if $i \leq j$, $\alpha^L_j \leq \alpha^L_i$ for d_i and d_j retrieved for a given query.

Please note that Definition 3.1 provides a definition for coherency based on *label* prior probabilities. We use this definition when analyzing the label loss function.

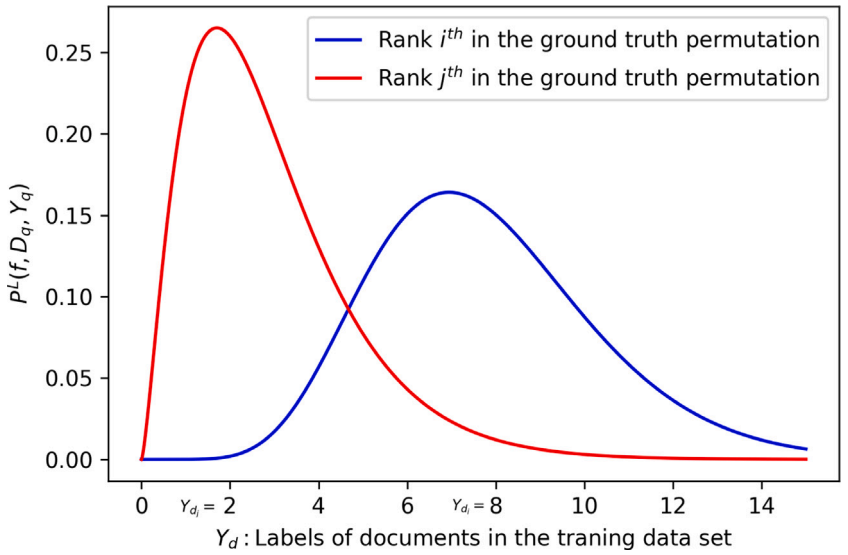


Fig. 1. An example of prior distribution function for different ranks in the ground truth permutation, when the training data is in line with the observed data.

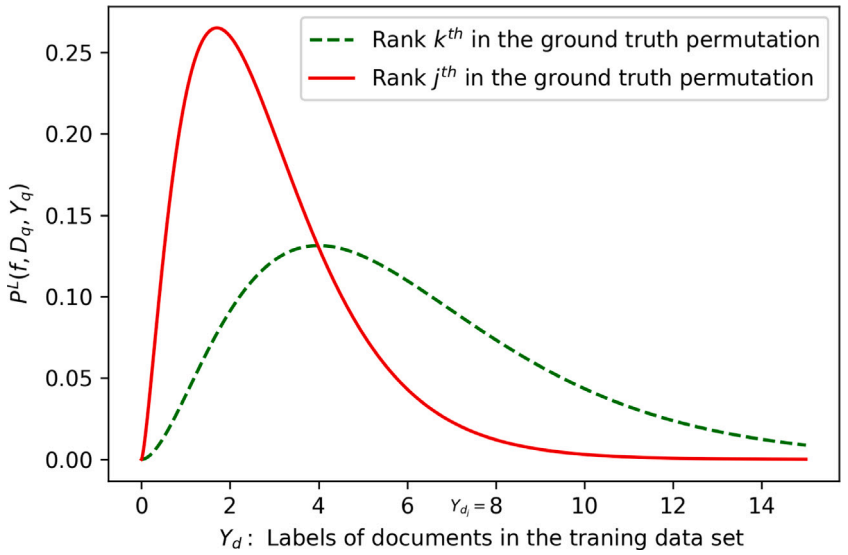


Fig. 2. An example of prior distribution function for noisy data and also for non-coherent data.

3.4. Order preserving

In this section we analyze the attribute of order-preserving for ListMAP, i.e., we analyze whether minimizing the ListMAP loss is consistent with the correct ranking order or not. It is been shown that some of the prominent listwise loss function such as ListNet lacks this attribute (Zhu & Klabjan, 2020a). We start our argument with the following Theorem that is based on the definition of order preserving provided in Xia et al. (2008) and has been proved in the same work.

Theorem 3.1. *Given q be a query, D_q be the set of documents retrieved for q, $d \in D_q$ and $p \in D_q$ two documents, and $f_{i,j}$ is a ranking function according to which d is placed in the ith position of the resulting ranking and p is placed in the jth position of this ranking, where $i \leq j$. Further, $f_{j,i}$ is a ranking function that results in the same ranking as $f_{i,j}$ except that d is placed in the jth position of the resulting ranking and p is placed in the ith position of this ranking. Further assume based on the ground truth labels, $Y_d \geq Y_p$. Then, $P(\pi_q|f_{i,j}, D_q, Y_q) \geq P(\pi_q|f_{j,i}, D_q, Y_q)$.*

Given that the ListMAP loss function is consist of the prior and the likelihood (Eq. (5)), Theorem 3.1 asserts that the likelihood part is preserving the order of documents based on the labels that are provided in the training data. In other words, minimizing

the likelihood part is consistent with the ranking order provided in the ground truth labels of training data. Now, let us analyze the prior probability based on this fact that it comes from a Gamma distribution whose parameters are learnt over a parameter setting dataset.

The following theory asserts that the simplified label prior probability, $P^{\hat{S}iL}(f, D_q, Y_q)$, is order preserving in coherent datasets.

Theorem 3.2. *Given q be a query, D_q be the set of documents retrieved for q, $d \in D_q$ and $p \in D_q$ two documents, and $f_{i,j}$ is a ranking function according to which d is placed in the ith place of the resulting ranking and p is placed in the jth place of this ranking, where $i \leq j$. Further, $f_{j,i}$ is a ranking function that results in the same ranking as $f_{i,j}$ except that d is placed in the jth place of the resulting ranking and p is placed in the ith place of this ranking. Further assume based on the ground truth labels, $Y_d \geq Y_p$, then in an $n-$coherent dataset, if $i, j \leq n$, $P^{\hat{S}iL}(f_{i,j}, D_q, Y_q) \geq P^{\hat{S}iL}(f_{j,i}, D_q, Y_q)$.*

Proof. Based on our assumption, for a given query q, $f_{j,i}$ results in the same ranking as $f_{i,j}$ for all documents except for documents d and p. Therefore, the comparison between $P^{\hat{S}iL}(f_{i,j}, D_q, Y_q)$ and $P^{\hat{S}iL}(f_{j,i}, D_q, Y_q)$ is reduced to comparing $P^{\hat{S}iL}(f_{i,j}, d, Y_q) \times P^{\hat{S}iL}(f_{i,j}, p, Y_q)$ with $P^{\hat{S}iL}(f_{j,i}, d, Y_q) \times P^{\hat{S}iL}(f_{j,i}, p, Y_q)$.

$$
\begin{aligned}
P^{\hat{S}iL}(f_{i,j}, d, Y_q) \times P^{\hat{S}iL}(f_{i,j}, p, Y_q) = & \\
& \frac{\beta^{\alpha_i}}{\Gamma(\alpha_i)} Y_d^{\alpha_i - 1} e^{-\beta Y_d} \times \frac{\beta^{\alpha_j}}{\Gamma(\alpha_j)} Y_p^{\alpha_j - 1} e^{-\beta Y_p} \\
& \text{(Definition (17))} \\
& \geq \frac{\beta^{\alpha_i}}{\Gamma(\alpha_i)} Y_p^{\alpha_i - 1} e^{-\beta Y_p} \times \frac{\beta^{\alpha_j}}{\Gamma(\alpha_j)} Y_d^{\alpha_j - 1} e^{-\beta Y_d} \\
& \text{(Since } Y_d \geq Y_p \text{, and } \alpha_i \geq \alpha_j \\
& \geq P^{\hat{S}iL}(f_{j,i}, d, Y_q) \times P^{\hat{S}iL}(f_{j,i}, p, Y_q) \quad \square
\end{aligned}
$$

Conclusively, the simplified label loss function is order preserving, based on the definition provided in Xia et al. (2008). The attribute of order-preserving cannot be generally proved for the other loss functions introduced in our learning to rank model. For example for $P^L(f, d, Y_q)$, different βs in positions i and j my result in a lower $P^L(f_{i,j}, d, Y_q)$ than $P^L(f_{j,i}, d, Y_q)$ for some datasets.

4. Experiments

4.1. Research objectives

In this section, we report the experiments we conducted for evaluating the performance of the ListMAP learning to rank model. Through our experiments, we attempt at analyzing the following three research objectives:

- **Research Objective 1: Performance across different datasets** We analyze whether the introduced listwise models are effective in retrieving and ranking relevant documents to a given query comparing to existing state-of-the-art methods. Furthermore, We study which of the proposed model would be the most effective across different datasets.
- **Research Objective 2: Prior Function Parameter Setting** We investigate the prior distribution parameters estimated for each dataset. We study which of the datasets are coherent and leads to an order-preserving property for the introduced models as we discussed in Section 3.4.
- **Research Objective 3: Success/Failure Analysis** Given that the main difference between the introduced ListMAP model and the prominent ListMLE model falls in incorporating a prior distribution in the loss function, we study the number of queries whose performance improved/hurt by ListMAP compared with ListMLE. We aim at analyzing the effectiveness of incorporating priors, i.e., given all other formalization and implementations be the same, how effective is a prior distribution in improving/hurting the performance of different queries.

In the following, we first report the datasets we used, our experimental setup, and the baselines. We then present our analysis on these research objectives by reporting experimental results.

4.2. Data sets

In this set of experiments, we use the following six benchmark datasets: MQ2007 and MQ2008 of the Letor 4.0 benchmark (Qin & Liu, 2013), Set 1 and Set 2 of the Yahoo! learning to rank challenge data set (Chapelle & Chang, 2011), denoted as Yahoo!Set1 and Yahoo!Set2 in our reported results, and Microsoft 30k and Microsoft 10K datasets (Qin & Liu, 2013) denoted as MSLR30K and MSLR10K, respectively. Table 2 shows the number of queries, documents, and features for each data set. The documents in MQ2007 and MQ2008 are retrieved from 25 million pages in the Gov2 web page collection (Qin, Liu, Xu et al., 2010) for queries in the Million Query track of TREC 2007 and TREC 2008, respectively. Documents are labeled with relevance judgment ranging from 0 (not related) to 2 (highly related). Yahoo! data sets consists of top documents retrieved for randomly sampled queries from the query logs of the Yahoo! search engine. Documents are labeled with relevance judgment ranging from 0 (not related) to 5 (highly

Table 2
Data set statistics.

Data set	#Documents	#Queries	#Features
MQ2007	65323	1 692	46
MQ2008	14384	784	46
Yahoo!Set1	~709 K	29 921	700
Yahoo!Set2	~172 K	6 330	700
MSLR10K	~1200 K	10 000	136
MSLR30K	~3771 K	31 531	136

related). MSLR30K is created from a retired labeling set of the Bing search engine, and contains 125 documents for each query by average. MSLR10K is created by a random sub-sampling from MSLR30K with 10,000 queries. Similar to Yahoo!Sets, the relevance judgments in MSLR datasets range from 0 (irrelevant) to 5 (perfectly relevant). We used 50 percent of the training data for setting the α and β parameters of the prior distribution models and use the rest for the training ListMAP learning to rank models.

MSLR datasets and MQ2007, and MQ2008 comes with five folds, where each fold has a test, train and a validation set. We used the train set of each fold for training the models, and report the average results across test sets of all folds. Yahoo!Sets comes with one set of test, train, and validation data and the reported results are those obtained from running model on the test set.

4.3. Experimental setup

For implementing the proposed listwise model and the baselines, we used the self attention neural network architecture introduced in Pobrotyn et al. (2020). We made use of the open-source Pytorch implementation of the network[1] and extend it with the implementation of ListMAP loss functions and other baseline losses. In these experiments, all documents are represented as sets of features that are provided by the experimental datasets, so we have not applied any embedding on words or characters. All codes are available in our GitHub repository.[2]

For tuning network parameters, we used the validation data set and maximize NDCG@5. For ListMAP losses, the number of encoders in the self-attentive network is set to 4 for MSLR collections and is set to 2 for other collections. The number of self attention heads is set to 2 for MSLR collections and set to 1 for other datasets. The only exception is ListMAP$_{LP}$ that is set to have 4 encoders and 4 attention heads in MQ2008 dataset. The size of the input embedding layer for MSLR datasets is set to 144, and for Yahoo!Sets it is set to 96. For MQ2008, the input embedding layer are 96, 144, and 144 and for MQ2007 they are 96, 256, and 128 for ListMAP$_{SP}$, ListMAP$_{LP}$, and ListMAP^{SiP}; respectively. The training batch size is set to 32 queries. The learning rate is set to 0.001 and the drop rate is 0.3. We repeated all experimental runs 20 times over the test dataset folds and reported the averaged results. We applied the Adam batch normalization (Kingma & Ba, 2014) between consecutive layers. The details of the neural network setting is reported in our Github repository.

4.4. Baselines

We compared our learning to rank models with a number of prominent and state-of-the-art baselines namely *ListNet* (Cao et al., 2007), *LitMLE* (Xia et al., 2008), *Approximate-NDCG* learning models (Qin, Liu, & Li, 2010), *Sigmoid-Loss* (Carvalho et al., 2008). We also compare our models with two learning to rank models, *OrdinalRank* (Pobrotyn et al., 2020) and *NeuralNDCG* (Pobrotyn & Białobrzeski, 2021), that have been shown to outperform other baselines in the self-attentive neural architecture that we employed for implementing ListMAP loss functions. The network setting for the neural architecture of each model across datasets are reported in our Github repository.

4.5. Results

4.5.1. Research objective 1: Retrieval performance

In this section, we report the performance of the learning to rank models introduced in this paper, namely ListMAP$_{SP}$, ListMAP$_{LP}$, and ListMAP$_{SiLP}$, and compare them with the baseline models; performance over MQ2007, MQ2008, Yahoo!Set1, YahooSet2, MSKR10K, and MSLR30K datasets. In line with recent related work in the literature (Pasumarthi, Zhuang, Wang, Bendersky, & Najork, 2020; Zhu, Cao, Lu, & Gu, 2021), we report the models performance using NDCG and MRR evaluation metrics at positions 1, 3, 5, 10, and 20.

Tables 3 and 4 show the performance of ListMap models and the baselines in terms of NDCG and MRR. Bold numbers are the highest in each column. In these tables, ⋆, †, and ‡ indicate a statistical significant improvement over ListNet, ListMLE, and OrdinalRank methods based on a Paired t-test with a confidence level of 0.05, respectively. A complete report of t-test P-values for the improvement over all baselines can be found in our github repository.

[1] https://github.com/allegro/allRank
[2] https://github.com/sanazkeshvari/allrank

Table 3
NDCG and MRR performance on the MQ2007, MQ2008, Yahoo!Set1, and Yahoo!Set2 datasets. Statistically significant improvements over ListNet, ListMLE and AllRank are shown by ⋆, †, and ‡, respectively.

(a) MQ2007

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	0.495	0.465	0.500	0.545	0.509	0.564	0.535	0.577	0.598	0.584
ListMLE	0.477	0.448	0.482	0.529	0.489	0.547	0.517	0.562	0.580	0.569
ApproxNDCG	0.514	0.482	0.517	0.564	0.526	0.582	0.556	0.597	0.618	0.602
Sigmoid-Loss	0.473	0.445	0.476	0.526	0.484	0.544	0.512	0.559	0.574	0.566
Allrank	0.526	0.492	0.529	0.573	0.538	0.590	0.569	0.604	0.629	0.610
NeuralNDCG	0.518	0.484	0.530	0.574	0.538	0.590	0.571	0.604	0.629	0.610
ListMAP$_{SP}$	**0.547***†	**0.515***†	**0.542***†	**0.589***†	**0.551***†‡	**0.608***†	**0.580***†‡	**0.621***†	**0.640***†	**0.626***†
ListMAP$_{LP}$	0.522*†	0.491*	0.521*†	0.567*†	0.528*†	0.586*†	0.555*†	0.599*†	0.618*†	0.605*†
ListMAP$_{SiLP}$	0.521	0.491	0.525*†	0.575*†	0.530*†	0.593*†	0.556*†	0.605*†	0.617*†	0.611*†

(b) MQ2008

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	0.597	0.546	0.628	0.634	0.666	0.653	0.708	0.665	0.733	0.667
ListMLE	0.507	0.482	0.553	0.569	0.603	0.595	0.674	0.612	0.708	0.616
ApproxNDCG	0.520	0.492	0.548	0.575	0.595	0.598	0.672	0.617	0.708	0.620
Sigmoid-Loss	0.591	0.563	0.638	0.651	0.677	0.672	0.733	0.683	0.760	0.686
AllRank	0.633	0.597	0.681	0.692	0.727	0.712	0.775	0.721	0.796	0.723
NeuralNDCG	0.618	0.587	0.662	0.674	0.710	0.696	0.760	0.706	0.784	0.708
ListMAP$_{SP}$	**0.652**	**0.617**	**0.682**‡	**0.701**	**0.731**	**0.721**	**0.775**	**0.730**	**0.797**	**0.732**
ListMAP$_{LP}$	0.593	0.564	0.632	0.648	0.678	0.670	0.733*	0.682	0.759	0.685
ListMAP$_{SiLP}$	0.632*†	0.599*†‡	0.668*†	0.683*†	0.713*†	0.703*†	0.764*†	0.715*†	0.788*†	0.716*†

(c) Yahoo!Set1

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	0.705	0.610	0.709	0.691	0.729	0.707	0.774	0.718	0.818	0.721
ListMLE	0.685	0.576	0.691	0.661	0.714	0.678	0.763	0.689	0.808	0.693
ApproxNDCG	0.667	0.563	0.670	0.656	0.691	0.674	0.742	0.684	0.790	0.688
Sigmoid-Loss	0.553	0.434	0.593	0.546	0.630	0.571	0.693	0.586	0.748	0.590
AllRank	0.707	**0.611**	0.714	0.693	0.737	0.711	0.782	0.719	0.822	0.722
NeuralNDCG	**0.707**	0.610	0.715	**0.694**	0.735	**0.711**	0.779	**0.720**	0.821	**0.723**
ListMAP$_{SP}$	0.626	0.539	0.737*†‡	0.670*†	0.789*†‡	0.693†	0.832*†	0.701†	0.843*†‡	0.702†
ListMAP$_{LP}$	0.645	0.559	**0.751***†‡	0.688†	0.799*†	0.708†	**0.841***†	0.716†	**0.851***†‡	0.717†
ListMAP$_{SiLP}$	0.640	0.556	0.749*†‡	0.684†	**0.800***†	0.705	0.841*†‡	0.713†	0.851*†‡	0.713†

(d) Yahoo!Set2

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	**0.705**	**0.611**	0.699	0.698	0.714	0.712	0.752	0.721	0.813	0.724
ListMLE	0.703	0.599	0.694	0.682	0.706	0.696	0.746	0.705	0.808	0.709
ApproxNDCG	0.682	0.585	0.676	0.675	0.688	0.690	0.730	0.700	0.794	0.703
Sigmoid-Loss	0.691	0.595	0.689	0.687	0.703	0.701	0.742	0.710	0.804	0.713
AllRank	0.701	0.600	0.702	0.686	0.716	0.699	0.755	0.709	0.813	0.712
NeuralNDCG	0.696	0.599	0.696	0.683	0.708	0.698	0.747	0.707	0.808	0.711
ListMAP$_{SP}$	0.665	0.574	**0.762**‡	**0.703***†‡	**0.817***†‡	**0.722***†‡	**0.859***†‡	**0.729***†‡	**0.865***†‡	**0.730***†‡
ListMAP$_{LP}$	0.552	0.445	0.677	0.584	0.749*†‡	0.615*†‡	0.809*†	0.627	0.818*†‡	0.629
ListMAP$_{SiLP}$	0.579	0.476	0.699	0.614	0.766*†‡	0.643	0.821*†‡	0.653	0.830*†‡	0.654

It can be seen in these tables that ListMap$_{SP}$ is the best performing L2R model on both MQ2007 and MQ2008 datasets in terms of both NDCG and MRR in all positions. The NDCG obtained by all listMap models outperform all baselines from the position of 3 to 20 on both Yahoo!Set1 and Yahoo!Set2 datasets. Furthermore, ListMap$_{SP}$ is the best performing one in terms of both NDCG and MRR metrics on Yahoo!Set2 dataset from the position of 3 to 20. All ListMap models perform better than the baselines in terms of both NDCG and MRR from the positions of 5 to 20 across MSLR10K and MSLR30k datasets. On the MSLR10K dataset, ListMAP$_{LP}$ performs better than the other ListMap models.

4.5.2. Research objective 2: Prior function parameter setting

In this section, we analyze setting parameters of the prior function introduced in Section 3. Figs. 3(a), (c), (e) and Figs. 4(a), (c) and (e) show the probability density functions approximated for different datasets by observing 50% of the training data for

Table 4
NDCG and MRR performance on the MSLR10k and MSLR30k datasets. Statistically significant improvements over ListNet, ListMLE and AllRank are shown by ⋆, †, and ‡, respectively.

(c) MSLR10K

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	0.390	0.274	0.383	0.351	0.393	0.372	0.416	0.388	0.454	0.397
ListMLE	0.383	0.249	0.381	0.320	0.389	0.338	0.410	0.353	0.446	0.361
ApproxNDCG	0.403	0.280	0.391	0.352	0.396	0.371	0.414	0.386	0.448	0.393
Sigmoid-Loss	0.173	0.096	0.188	0.137	0.202	0.153	0.231	0.171	0.277	0.182
AllRank	**0.413**	0.278	0.408	0.352	0.414	0.371	0.434	0.386	0.468	0.393
NeuralNDCG	0.395	0.265	0.401	0.348	0.409	0.368	0.429	0.384	0.462	0.391
ListMAP$_{SP}$	0.357	0.293*†‡	0.427*†‡	0.401*†‡	0.491*†‡	0.436*†‡	0.599‡	0.463*†‡	0.662*†‡	0.470*†‡
ListMAP$_{LP}$	0.363	**0.301**‡	**0.441**	**0.415**†	**0.505***†‡	**0.449***†	**0.610***†‡	**0.474***†‡	**0.670***†‡	**0.480***†‡s
ListMAP$_{SiLP}$	0.361	**0.301**†	0.437*†	0.412*†	0.502*†‡	0.448*†	0.609*†‡	0.473*†	0.669*†‡	0.479*†‡

(c) MSLR30K

	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
	@1		@3		@5		@10		@20	
ListNet	0.423	**0.310**	0.411	0.388	0.417	0.409	0.440	0.425	0.479	0.432
ListMLE	0.426	0.288	0.420	0.364	0.426	0.384	0.446	0.398	0.479	0.406
ApproxNDCG	0.421	0.296	0.407	0.370	0.413	0.390	0.431	0.404	0.464	0.412
Sigmoid-Loss	0.164	0.089	0.182	0.125	0.196	0.141	0.225	0.157	0.270	0.168
AllRank	**0.454**	0.321	**0.446**	**0.401**	0.451	0.420	0.472	0.435	0.505	0.442
NeuralNDCG	0.434	0.301	0.432	0.385	0.440	0.406	0.460	0.421	0.492	0.428
ListMAP$_{SP}$	0.335	0.275	0.411	0.384†	0.478*†‡	0.420*†	0.587*†	0.448*†‡	0.652*†‡	0.455*†‡
ListMAP$_{LP}$	0.342	0.281	0.416	0.389†	0.482*‡	0.426*‡	0.591*†‡	0.453*‡	0.655*†‡	0.460*‡
ListMAP$_{SiLP}$	0.356	0.292	0.426*†	**0.401***†	**0.491***†‡	**0.437***‡	**0.597***†‡	**0.463***‡	**0.661***†‡	**0.470***‡

setting the parameters of the Gamma distribution for documents that are placed in ranks 1th, 5th, 10th and 15th of a permutation. The probability density function is used for approximating the prior probability defined in ListMAP loss functions. Furthermore, Figs. 3(b), (d), (f) and Figs. 4 (b), (d) and (f) show the probability density functions approximated for these datasets by using 80% of the training data as observations for prior function parameter setting. As you can see in these figures, the Gamma distribution is defined over labels of the documents, i.e., it assigns different prior probability to each label when a document is placed in different ranks in a given permutation. For example, according to Fig. 3, a document that is placed in the first rank of a permutation for a query in the MQ2007 dataset is more probable to have a label of 2 (close to 0.4) rather than a label equal to 0 (around 0). On the other hand, a document that is placed in the 15th rank in a given permutation for a query in the MQ2007 dataset has an almost 0 probability to have a label equal to 2. In other words, a pair of query-document, where the document is the 15th in the ground truth permutation and has a label of 2, is recognized a noise and has no impact in the learning process.

The Gamma distributions depicted in Figs. 3 and 4 are calculated based on the observations randomly selected from the training data in the first fold of datasets MQ2007, MQ2008, MSLR10K and MSLR30K. For Yahoo!Set datasets that have no predefined folds, parameter setting is performed based on the randomly selected observations from the whole training data. As we can see in Fig. 3, MQ2007, MQ2008, MSLR30K and Yahoo!Set1 are coherent for the top 15 positions when 80% of the training data used as observations for parameter setting. MQ2008, the smallest dataset in our evaluation, struggles with prior function approximation when 50% of the training data used as observations. Here, the probability of observing other values than 0 is small for all ranking positions. For MSLR10K in both cases, when 50% and 80% used as observations, and for Yahoo!Set2 with 80% of training data as observations, the Gamma distributions for the ranking positions look closely similar, i.e., different ranks in the permutation have no clearly distinguishable pattern in their labels. Fig. 3 (e and d) and Fig. 4(f) show that the most relevant documents in these datasets (i.e., the documents that have been placed in the Ranks of #1 and #5 in the ground truth permutations of dataset queries) are more probable to be assigned a label in the range of 0 to 2 than a label in other ranges such as 4. The intuition behind this observation is that documents in this datasets do not usually get high relevant scores in the ground truth data. In other words, given a query posed to this datasets, the label of the first document in the ground truth permutation is approximately close to the labels of the documents in the rank of 5, 10, and 15, and all are close to a value in the range of 0 to 2, by average.

4.5.3. Research objective 3: Success/failure analysis

Fig. 5 illustrates queries whose effectiveness are improved/hurt by ListMAP. In these figures, the difference between NDCG@20 for ListMAP$_{SP}$ over ListMLE is reported across different datasets. We used the test folder of the first fold of MQ2007 and MQ2008 and MSLR10K and MSLR30K datasets and the whole test folder of Yahoo!Set datasets for this analysis. Here a positive value indicates an improvement achieved by ListMAP. In this figures, we chose ListMAP$_{SP}$ due to its relatively better performance than ListMAP$_{LP}$ and ListMAP$_{SiLP}$ across MQ and Yahoo!Set datasets. Since the loss function defined in ListMAP differs from the one in ListMLE in exploiting prior probabilities, in Fig. 5, we selected ListMLE from the existing baselines to highlight the role of the prior function in the learning process.

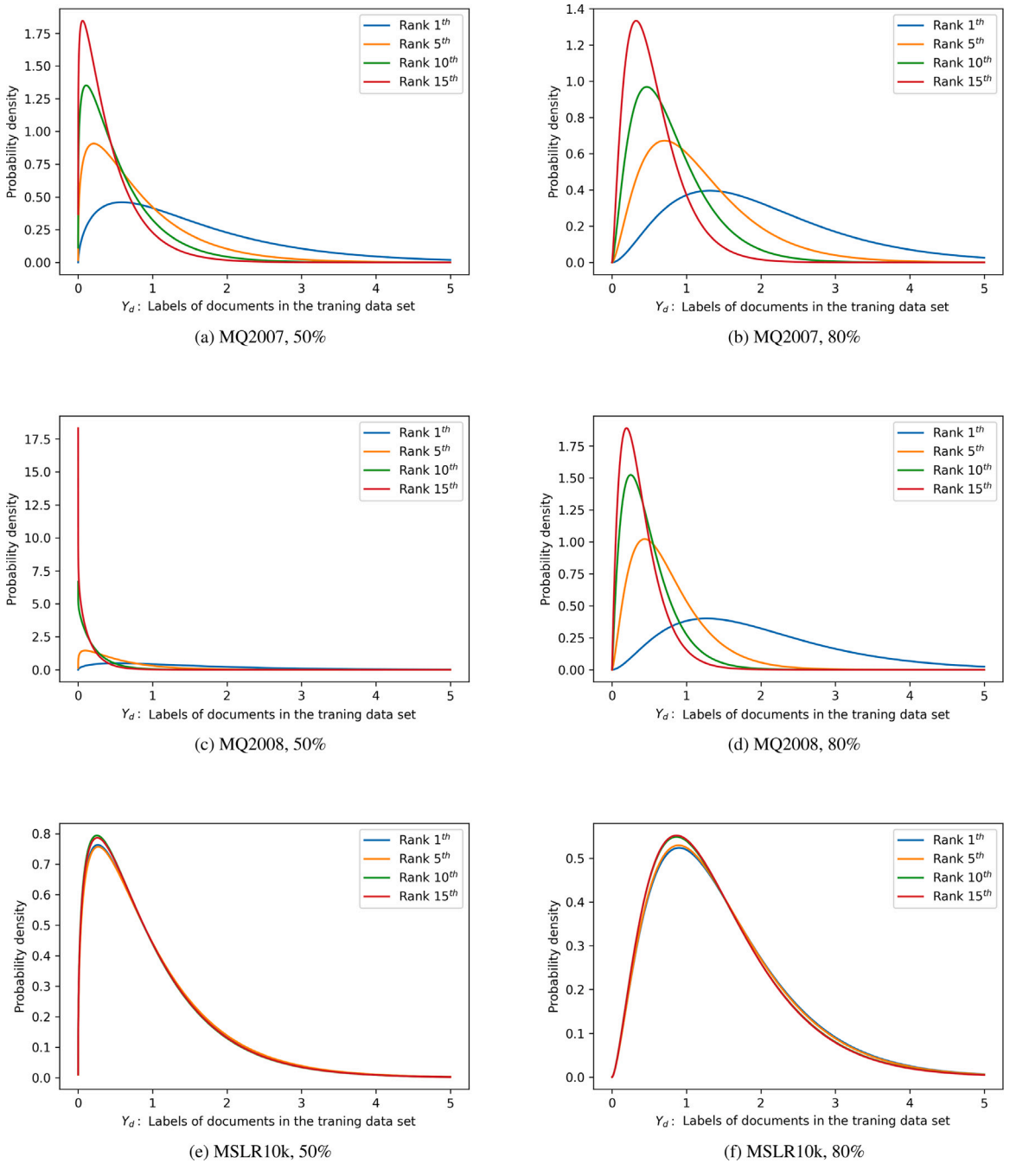
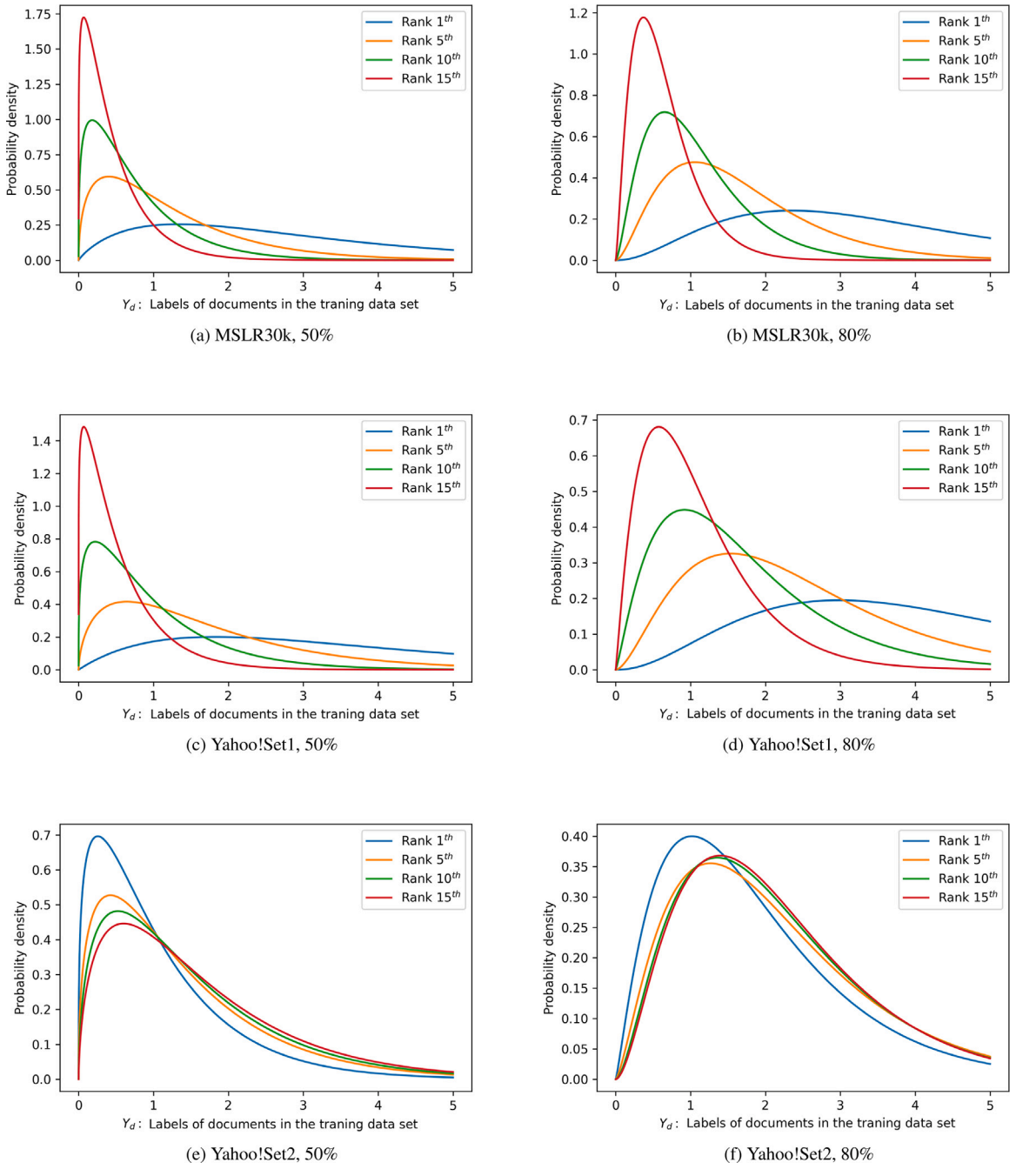


Fig. 3. Probability density functions for documents placed in ranks 1, 5, 10, and 15 of ground truth permutations for queries over MQ2007, MQ2008 and MSLR10K data sets.

As it can be observed from Fig. 5, in MQ2007, ListMAP achieves a better or the same performance in terms of NDCG@20 for 256 queries out of 336 one compared with ListMLE. In MSLR10K, out of 1938 queries, ListMAP helps 1459 queries. From 6178 queries in the MSLR30K dataset, 4350 queries have a better performance by ListMAP than the baseline, while 42 queries gets exactly the same value for their NDCG metric. For Yahoo!Set1, ListMAP improves the NDCG@20 of 3497 queries out of 6303, while it has the same performance as ListMLE for 277 queries. Out of the 3645 queries for Yahoo!Set2, 2418 queries have a better or the same value for the NDCG@20 metric compared with the baseline. The only dataset that hurts from using prior functions is MQ2008, where ListMAP helps 34 out of 156 queries, while it achieves exactly the same performance as ListML for 75 queries. The reason can be



(a) MSLR30k, 50%

(b) MSLR30k, 80%

(c) Yahoo!Set1, 50%

(d) Yahoo!Set1, 80%

(e) Yahoo!Set2, 50%

(f) Yahoo!Set2, 80%

Fig. 4. Probability density functions for documents placed in ranks 1, 5, 10, and 15 of ground truth permutations for queries over MSLR30K, Yahoo!Set1, and Yahoo!Set2 data sets.

due to the fact that MQ2008 is a relatively small dataset where the prior function is distributed around 0 for the top 15 ranks (See Fig. 3(c)).

4.6. Discussion

In Section 4.1, we defined three research objectives to analyze whether the proposed learning to rank models are effective in ranking, to analyze the prior parameters derived from the observation data of each dataset, and to study the impact of the prior distribution on improving/hurting queries' retrieval performance.

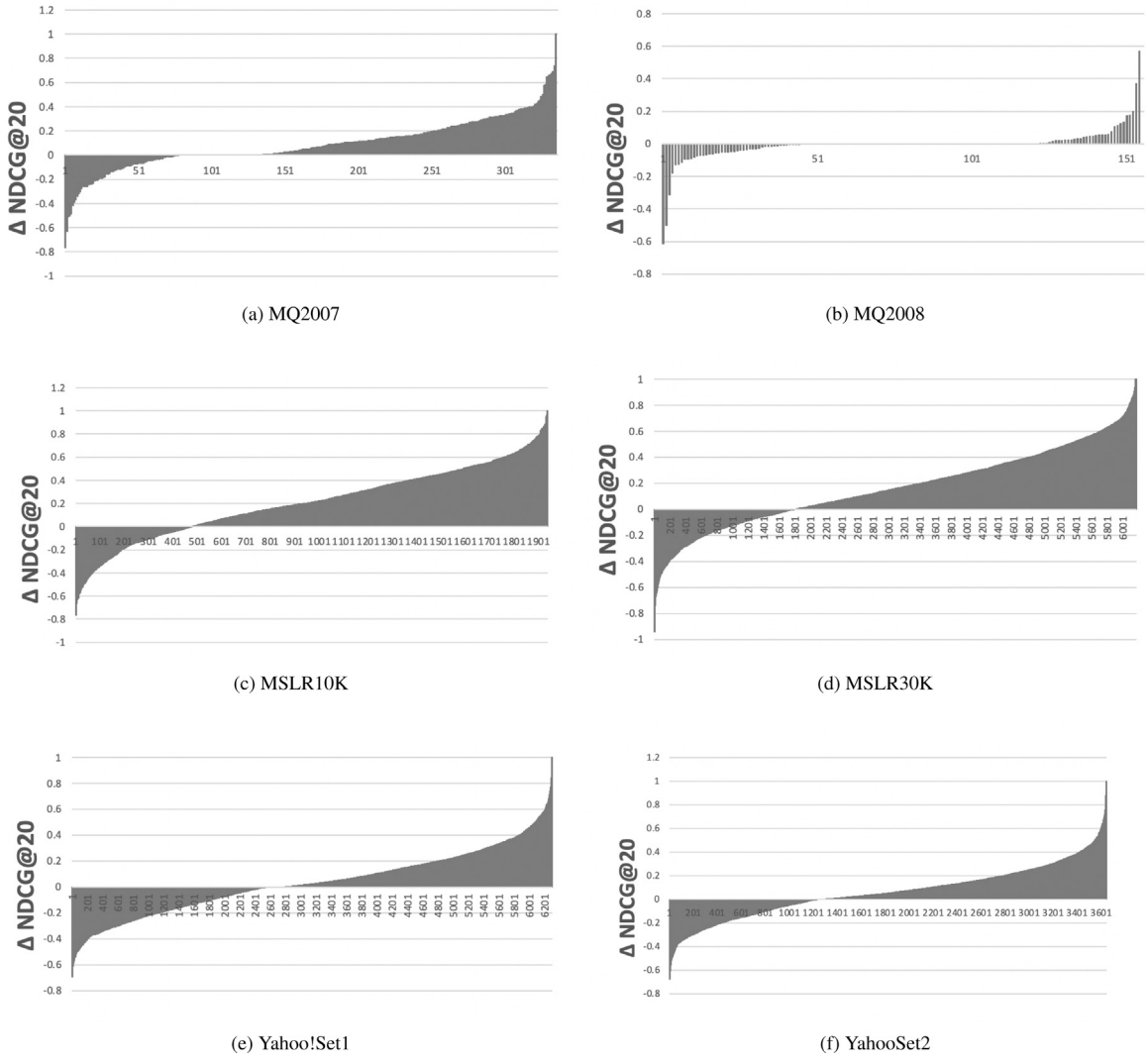


Fig. 5. The difference between the NDCG@20 achieved by ListMap$_{SP}$ and the one achieved by ListMLE (denoted as Δ NDCG@20) for each query over different datasets. In these plots, the X axis represents queries sorted by Δ NDCG@20 and Y axis represents Δ NDCG@20.

Regarding the first research objective, the following conclusions can be derived from the experiments:

- All three proposed learning to rank models; namely ListMAP$_{SP}$, ListMAP$_{LP}$, and ListMAP$_{SiLP}$; are more effective than the four prominent models; ListNet, LISTMLE, ApproxNDCG, and Sigmoid-Loss; in terms of the nDCG metric at positions 3, 5, 10, and 20 in almost all datasets. Furthermore, the proposed models outperform these baselines in terms of MRR in most cases. Exceptions include the reported MRR for ListMAP$_{LP}$, and ListMAP$_{SiLP}$ over the Yahoo!Set2. This conclusion is significant in light of this fact that all baselines and the proposed models are implemented using the same neural network architecture.
- ListMAP$_{SP}$ is the model that outperform all baselines in terms of nDCG metric at positions 3 to 20. ListMAP$_{SP}$ is also the best performing model among the proposed models across MQ2007, MQ2008, and Yahoo!Set2 datasets and outperform all baselines in terms of nDCG and MRR at positions 3 to 20 across MSLR10K and MSLR30K.

Regarding to the second research objective, our experiments show that the prior distribution gets different shapes across different datasets and by observing different percentage of the training data. The findings for the second research objective can be summarized as follows:

- MQ2007, MSLR30K, and Yahoo!Set1 are 15-coherent datasets when 50% of the training data is used for setting the prior distribution parameters. Furthermore, MQ2008 is a 15-coherent dataset, when 80% of the training data is used for prior parameter setting.

- In some datasets, such as MSLR10K, the prior distributions for the ranking positions look closely similar. In other words, different ranks in the ground truth permutation of these datasets have no clearly observable patterns in their labels, and each document can get different labels with nearly the same probability in different positions.
- Our experiments show that ListMAP is effective in improving the retrieval performance across both types of datasets: those with clearly observable pattern in their labels (e.g MSLR30K) and those with no clear pattern in their labels (e.g MSLR10K).

Finally, in terms of the third research objective, our experiments show that incorporating the introduced prior into the learning to rank model improves the performance of a larger number of queries compared with those whose performance are hurt. The following are the main findings regards to the third research objective:

- Incorporating the prior distribution helps the retrieval performance of the majority of queries measured by nDCG@20 across almost all datasets.
- Incorporating the prior distribution is not helpful in datasets whose prior function is distributed around 0 for the top ranks. For example, MQ2008 is the only dataset in our experiments where more queries hurt by incorporating priors. MQ2008 is a dataset whose prior function is distributed around 0 for the top 15 ranks when 50% of the training data is used as prior parameter setting data.

5. Conclusion

In this paper, we propose a new listwise learning to rank loss function for modeling training datasets. The loss function is defined based on a prior distribution over labels and scores of documents in the ground-truth permutation. The prior function controls the impact of each training instance in the learning process. We use a part of the training data for learning a pattern on reliable scores and labels of documents in different ranks of a permutation in each dataset and then apply this pattern to weight the remaining training instances in listwise learning to rank.

We empirically evaluated the proposed learning to rank models through a complete set of experiments on MQ2007, MQ2008, MSLR10K, MSLR30K, Yahoo!Set1 and Yahoo!Set2 datasets. We show that the proposed model outperform prominent and novel learning to rank models in terms of NDCG and MRR evaluation metrics in different positions. We also illustrate that the proposed loss function helps to improve the performance of the majority of queries in most datasets.

CRediT authorship contribution statement

Sanaz Keshvari: Conceptualization, Methodology, Validation, Data curation, Writing – original draft, Software, Data curation. **Faezeh Ensan:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Hadi Sadoghi Yazdi:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ai, Q., Bi, K., Guo, J., & Croft, W. B. (2018a). Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 135–144).

Ai, Q., Bi, K., Luo, C., Guo, J., & Croft, W. B. (2018b). Unbiased learning to rank with unbiased propensity estimation. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 385–394).

Ai, Q., Wang, X., Bruch, S., Golbandi, N., Bendersky, M., & Najork, M. (2019). Learning groupwise multivariate scoring functions using deep neural networks. In *ICTIR '19* (pp. 85–92). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3341981.3344218.

Ai, Q., Yang, T., Wang, H., & Mao, J. (2021). Unbiased learning to rank: Online or offline? *ACM Transactions on Information Systems (TOIS)*, *39*(2), 1–29.

Bruch, S. (2021). An alternative cross entropy loss for learning-to-rank. In *Proceedings of the web conference 2021* (pp. 118–126).

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. *Learning*, *11*(23–581), 81.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on machine learning* (pp. 129–136).

Capannini, G., Lucchese, C., Nardini, F. M., Orlando, S., Perego, R., & Tonellotto, N. (2016). Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing & Management*, *52*(6), 1161–1177.

Carvalho, V. R., Elsas, J. L., Cohen, W. W., & Carbonell, J. G. (2008). A meta-learning approach for robust rank learning. In *SIGIR 2008 workshop on learning to rank for information retrieval, Vol. 1*.

Chapelle, O., & Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge* (pp. 1–24). PMLR.

Chapelle, O., & Keerthi, S. S. (2010). Efficient algorithms for ranking with SVMs. *Information Retrieval*, *13*(3), 201–215.

Chen, M., & Zhou, X. (2020). DeepRank: LEarning to rank with neural networks for recommendation. *Knowledge-Based Systems*, *209*, Article 106478.

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. In *International conference on computational learning theory* (pp. 605–619). Springer.

Cronen-Townsend, S., Croft, W. B., et al. (2002). Quantifying query ambiguity. In *Proceedings of HLT, Vol. 2* (pp. 94–98). Citeseer.

Dai, X., Hou, J., Liu, Q., Xi, Y., Tang, R., Zhang, W., et al. (2020). U-rank: Utility-oriented learning to rank with implicit feedback. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 2373–2380).

Dehghani, M., Zamani, H., Severyn, A., Kamps, J., & Croft, W. B. (2017). Neural ranking models with weak supervision. In *SIGIR '17* (pp. 65–74). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3077136.3080832.

Ding, W., Geng, X., & Zhang, X.-D. (2015). Learning to rank from noisy data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *7*(1), 1–21.
Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*(Nov), 933–969.
Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 1189–1232.
Gampa, P., & Fujita, S. (2021). BanditRank: LEarning to rank using contextual bandits. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 259–271). Springer.
Guiver, J., & Snelson, E. (2009). Bayesian inference for plackett-luce ranking models. In *Proceedings of the 26th annual international conference on machine learning* (pp. 377–384).
Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., et al. (2020). A deep look into neural ranking models for information retrieval. *Information Processing & Management*, *57*(6), Article 102067.
Han, H., Hwang, S.-w., Song, Y.-I., & Kim, S. (2020). Training data optimization for pairwise learning to rank. In *Proceedings of the 2020 ACM SIGIR on international conference on theory of information retrieval* (pp. 13–20).
Jagerman, R., Kiseleva, J., & de Rijke, M. (2017). Modeling label ambiguity for neural list-wise learning to rank. arXiv preprint arXiv:1707.07493.
Joachims, T., Swaminathan, A., & Schnabel, T. (2017). Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 781–789).
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Lawless, J. F. (1980). Inference in the generalized gamma and log gamma distributions. *Technometrics*, *22*(3), 409–419.
Lawrence, N., & Schölkopf, B. (2001). Estimating a kernel fisher discriminant in the presence of label noise. In *18th international conference on machine learning (ICML 2001)* (p. 306). Morgan Kaufmann.
Li, C., Kveton, B., Lattimore, T., Markov, I., de Rijke, M., Szepesvári, C., et al. (2020). BubbleRank: SAfe online learning to re-rank via implicit click feedback. In *Uncertainty in artificial intelligence* (pp. 196–206). PMLR.
Li, Q., Wang, E., Fleming, S. T., Thomas, D. B., & Cheung, P. Y. (2019). Accelerating position-aware top-k ListNet for ranking under custom precision regimes. In *2019 29th international conference on field programmable logic and applications (FPL)* (pp. 81–87). IEEE.
Liu, T.-Y. (2010). *Learning to rank for information retrieval*. New York, NY, USA: Association for Computing Machinery.
Luo, T., Wang, D., Liu, R., & Pan, Y. (2015). Stochastic top-k ListNet. In *EMNLP*.
Mollica, C., & Tardella, L. (2021). Bayesian analysis of ranking data with the Extended Plackett–Luce model. *Statistical Methods & Applications*, *30*(1), 175–194.
Oosterhuis, H., & de Rijke, M. (2018). Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 1293–1302).
Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017). Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 257–266).
Pang, L., Xu, J., Ai, Q., Lan, Y., Cheng, X., & Wen, J. (2020). Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 499–508).
Pasumarthi, R. K., Bruch, S., Wang, X., Li, C., Bendersky, M., Najork, M., et al. (2019). TF-Ranking: Scalable TensorFlow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* KDD '19, (pp. 2970–2978). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3292500.3330677.
Pasumarthi, R. K., Zhuang, H., Wang, X., Bendersky, M., & Najork, M. (2020). Permutation equivariant document interaction network for neural learning to rank. In *Proceedings of the 2020 ACM SIGIR on international conference on theory of information retrieval* ICTIR '20, (pp. 145–148). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3409256.3409819.
Plackett, R. L. (1975). The analysis of permutations. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, *24*(2), 193–202.
Pobrotyn, P., Bartczak, T., Synowiec, M., Białobrzeski, R., & Bojar, J. (2020). Context-aware learning to rank with self-attention. arXiv preprint arXiv:2005.10084.
Pobrotyn, P., & Białobrzeski, R. (2021). NeuralNDCG: DIrect optimisation of a ranking metric via differentiable relaxation of sorting. arXiv preprint arXiv:2102.07831.
Qin, T., & Liu, T.-Y. (2013). Introducing LETOR 4.0 datasets. arXiv preprint arXiv:1306.2597.
Qin, T., Liu, T.-Y., & Li, H. (2010). A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, *13*(4), 375–397.
Qin, T., Liu, T.-Y., Xu, J., & Li, H. (2010). LETOR: A Benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, *13*(4), 346–374.
Rahimi, R., Montazeralghaem, A., & Allan, J. (2019). Listwise neural ranking models. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval* (pp. 101–104).
Ravikumar, P., Tewari, A., & Yang, E. (2011). On NDCG consistency of listwise ranking methods. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 618–626). JMLR Workshop and Conference Proceedings.
Severyn, A., & Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 373–382).
Song, K.-S. (2008). Globally convergent algorithms for estimating generalized gamma distributions in fast signal and image processing. *IEEE Transactions on Image Processing*, *17*(8), 1233–1250.
Svore, K. M., Volkovs, M. N., & Burges, C. J. (2011). Learning to rank with multiple objective functions. WWW '11, (pp. 367–376). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/1963405.1963459.
Tax, N., Bockting, S., & Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information Processing & Management*, *51*(6), 757–772.
Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 international conference on web search and data mining* (pp. 77–86).
Thom, H. C. (1958). A note on the gamma distribution. *Monthly Weather Review*, *86*(4), 117–122.
Wang, X., Bendersky, M., Metzler, D., & Najork, M. (2016). Learning to rank with selection bias in personal search. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 115–124).
Wang, X., Golbandi, N., Bendersky, M., Metzler, D., & Najork, M. (2018). Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 610–618).
Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on machine learning* (pp. 1192–1199).
Ye, Z.-S., & Chen, N. (2017). Closed-form estimators for the gamma distribution derived from likelihood equations. *The American Statistician*, *71*(2), 177–181.
Yu, H.-T., Jatowt, A., Joho, H., Jose, J. M., Yang, X., & Chen, L. (2019). WassRank: LIstwise document ranking using optimal transport theory. In *Proceedings of the twelfth ACM international conference on web search and data mining* (pp. 24–32).
Yue, Y., Patel, R., & Roehrig, H. (2010). Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th international conference on world wide web* (pp. 1011–1018).
Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., et al. (2018). Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1059–1068).
Zhu, N., Cao, J., Lu, X., & Gu, Q. (2021). Leveraging pointwise prediction with learning to rank for top-N recommendation. *World Wide Web*, *24*(1), 375–396.

Zhu, X., & Klabjan, D. (2020a). Listwise learning to rank by exploring unique ratings. In *Proceedings of the 13th international conference on web search and data mining* (pp. 798–806).

Zhu, X., & Klabjan, D. (2020b). Listwise learning to rank by exploring unique ratings. In *Proceedings of the 13th international conference on web search and data mining* WSDM '20, (pp. 798–806). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3336191.3371814.

Zhuang, H., Wang, X., Bendersky, M., & Najork, M. (2020). Feature transformation for neural ranking models. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1649–1652).

Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., & Wen, Z. (2017). Online learning to rank in stochastic click models. In *International conference on machine learning* (pp. 4199–4208). PMLR.