Listwise Neural Ranking Models

Razieh Rahimi, Ali Montazeralghaem, and James Allan
Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
{rahimi,montazer,allan}@cs.umass.edu

ABSTRACT

Several neural networks have been developed for end-to-end training of information retrieval models. These networks differ in many aspects including architecture, training data, data representations, and loss functions. However, only pointwise and pairwise loss functions are employed in training of end-to-end neural ranking models without human-engineered features. These loss functions do not consider the ranks of documents in the estimation of loss over training data. Because of this limitation, conventional learning-to-rank models using pointwise or pairwise loss functions have generally shown lower performance compared to those using listwise loss functions. Following this observation, we propose to employ listwise loss functions for the training of neural ranking models. We empirically demonstrate that a listwise neural ranker outperforms a pairwise neural ranking model. In addition, we achieve further improvements in the performance of the listwise neural ranking models by query-based sampling of training data.

KEYWORDS

Neural network, document ranking, listwise loss, query-based sampling

ACM Reference Format:
Razieh Rahimi, Ali Montazeralghaem, and James Allan. 2019. Listwise Neural Ranking Models. In *The 2019 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '19), October 2–5, 2019, Santa Clara, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3341981.3344245

1 INTRODUCTION

Neural network models have been developed and successfully applied to many tasks including information retrieval. The key advantage of (deep) neural networks relies in automatic learning of features or representations at multiple levels of abstraction. Therefore, human-engineered features used in conventional learning-to-rank models are not required in neural networks. Herein, we refer to feature-based representation of queries and documents and learning algorithms on this type of representations, such as RankNet [4] and ListNet [5], as *conventional representations* and *conventional learning algorithms*, respectively. This is to differentiate them from learning-to-rank models that also learn representations from raw input data, such as the duet model [15], which we refer to as *neural ranking models*. More specifically, the division between *conventional* versus *neural* ranking models is based on the input of models. For example, although the ListNet algorithm uses a neural network for learning a ranking function, it is considered as a conventional ranking algorithm since its inputs are human-engineered feature representations of queries and documents.

The parameters of learning-to-rank models are optimized according to the employed loss function. Learning-to-rank models are then generally categorized into pointwise, pairwise, and listwise approaches, according to their loss functions [12]. While pointwise and pairwise learning-to-rank models cast the ranking problem as classification, the listwise learning-to-rank approach learns a ranking model in a more natural way. As a result, conventional learning-to-rank algorithms using listwise loss functions have shown better performance than pointwise and pairwise algorithms on most datasets, especially on top-ranked documents [17]. This is mainly because loss functions of pointwise and pairwise algorithms are not dependent on the position of documents in the final ranked lists, and thus are not in accordance with typical IR evaluation metrics.

Although conventional learning-to-rank models using listwise loss functions have shown promising results [17], to the best of our knowledge, no existing neural ranking model that learns entirely from input data without using human-engineered features, employs a listwise loss function for training. We thus explore the potential of using listwise loss functions for training of discriminative neural ranking models. More specifically, we examine the *deep relevance matching model* [8] which uses a pairwise hinge loss, with a listwise loss function, to investigate how the learned ranking models may differ.

Conventional learning-to-rank models using pointwise or pairwise loss functions cannot distinguish if two training instances are associated with the same query. Therefore, the trained models can be biased towards queries with more judged or retrieved documents. Using pointwise or pairwise loss functions for training of neural ranking models, the research question is then if these models may also be biased although they are trained on a large amount of data. Our evaluation shows that some neural ranking models that operate on query-document interaction signals also suffer from this limitation. These neural ranking models are referred to as early interaction models for document ranking [7].

Although listwise loss functions comply with the nature of ranking problems, training neural ranking models using listwise loss functions is more challenging. Given a set of labeled data, pointwise or pairwise learning algorithms have more training instances than listwise algorithms, while neural networks need to be trained on

ICTIR '19, October 2–5, 2019, Santa Clara, CA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6881-0/19/10…$15.00
https://doi.org/10.1145/3341981.3344245

adequately large data. To compensate for this problem, we propose random sampling of documents associated with each query before each epoch of training. Our evaluation demonstrates that reshuffling and sampling improves the performance of the listwise neural ranker for two reasons: (1) generating a slightly different set of training instances for each epoch of training which helps the network not to memorize training data [3], and (2) getting a flat minimizer of the loss function which helps the trained model to better generalize to test data [11].

2 RELATED WORK

Neural networks have been used in different ways to improve ranking tasks including ad-hoc information retrieval. We focus here on discriminative training models using (deep) neural networks for ad-hoc information retrieval, and do not discuss the models that improve document rankings by using the output of generally trained neural networks, such as distributed representations of words, to weight or expand queries [10, 20].

Following the convention of learning-to-rank models, we categorize end-to-end neural ranking models into pointwise, pairwise, or listwise models based on their loss functions. In addition, these networks can be categorized into early or late combination models [7] based on the input space of the network. Input of late interaction models are such that the network can independently learn representations for queries and documents. The representations are then compared against each other for document scoring. On the other hand, input of early interaction models consists of matching signals between a query and document.

Deep structured semantic model (DSSM) is a late-interaction model that learns semantic representations for query and documents, given their raw text features such as term frequency, using a feed forward neural network [9]. Shen et al. [18] improved the DSSM model by adding a convolutional layer to derive contextual feature vectors for words. These networks are trained using click-through data, where each training instance includes one clicked and four randomly selected non-clicked documents. The log-likelihood loss function is used to learn the model parameters, which is not a listwise loss [13]. Mitra et al. [15] proposed the *duet* model which is a combination of two neural models jointly trained for document ranking based on lexical matching and distributional similarity of a query and document. Therefore, their network combines early and late interaction models. The duet model is trained on data sampled from real query logs and judged by human. The model parameters are learned by maximizing the likelihood of relevant documents where negative documents are selected from documents with lower relevance degrees than the positive document, instead of random selection.

Guo et al. [8] proposed the *Deep Relevance Matching Model* (DRMM) for document ranking. First, local interactions between each query term and document terms are mapped to a fixed-length matching histogram, encoding exact matching and distributional similarity of terms. Therefore, their model belongs to early interaction category of models. Then, a feed forward neural network is employed to learn hierarchical matching patterns between queries and documents, and score documents. They used hinge loss defined on a pair of documents to optimize the model parameters in training. McDonald et al. [14] proposed a ranking model on top of DRMM model also using a pairwise loss function. Dehghani et al. [7] proposed weakly supervised training of neural ranking models. They trained different network architectures and accordingly different loss functions using weak data. Employed loss functions are mean squared error which is a pointwise loss, and two pairwise loss functions, hinge loss and cross-entropy.

While most of the existing neural ranking models are pointwise scoring, Ai et al. [2] recently proposed a neural network that scores a list of documents and the model parameters are updated based on a listwise loss function. The loss for a list of documents is calculated by summing pairwise logistic loss for all document pairs where the first document has a higher relevance degree than the second one. However, they tried the network on conventional feature-based representations of queries and documents. There are other neural ranking models considering additional ranking constraints or additional information than document texts [1, 19] that are not within the scope of this study.

3 NEURAL RANKING MODEL

3.1 Network Architecture

We use the neural network proposed by Guo et al. [8] to estimate retrieval scores of documents. This model, known as *deep relevance matching model* (DRMM), exploits exact and semantic matching of query and document terms by deriving fixed-size matching histograms as input of the network. A feed-forward is then used for non-linear matching between a query term and document terms. Finally, a gating network indicating the importance of each query term is applied to compute the score of each document. The network is trained using a pairwise hinge loss function as follows:

$$\mathcal{L}(q, d^+, d^-; \theta) = \max(0, 1 - s(q, d^+) + s(q, d^-)), \qquad (1)$$

where d^+ and d^- are documents relevant and non-relevant to the query q, respectively, and θ indicates model parameters.

3.2 Listwise Loss Function

We use the loss function of the ListNet algorithm [5] to train a neural ranking model. This loss function is based on estimating a probability distribution for a list of scored documents, indicating the probability of different rankings of documents. The probability distribution can be estimated using permutation or top-1 probabilities. Because of the computational complexity of permutation probabilities, we use top-1 probabilities following the original model and its subsequent use, as follows:

$$p(d_j) = \frac{\exp(s(q, d_j))}{\sum_{k=1}^{n} \exp(s(q, d_k))}. \qquad (2)$$

The true probability distribution, $y^{(i)}$, is estimated using human relevance judgments. Cross-entropy is then utilized to measure the distance between the two probability distributions, estimated based on predicted scores for documents ($z^{(i)}$) and estimated based on relevance judgments as:

$$\mathcal{L}(y^{(i)}, z^{(i)}; \theta) = -\sum_{j=1}^{n} p_{y^{(i)}}(d_j) \log p_{z^{(i)}}(d_j). \qquad (3)$$

3.3 Training samples

With the same amount of labeled data, learning-to-rank algorithms with pointwise or pairwise loss functions have more training instances than algorithms with listwise loss functions. This is because each retrieved document or each pair of retrieved documents can be a training instance in the former, while each query constitutes only one training instance in the latter. However, neural networks need to be trained on adequately large amounts of data. To compensate for the low amount of training data for a listwise neural ranking model, we propose random sampling of documents retrieved for each query before a new epoch of training. By sampling documents before each training epoch, we actually do not increase the number of training instances, but we feed the network with slightly different data. This data alteration enables better training of the ranker model by avoiding memorization of data by the network. As Arpit et al. [3] have shown, deep neural networks can memorize training data very fast. Thus, avoiding memorization of data leads to improvements in the learned model.

4 EXPERIMENTS

Datasets. We conduct our experiments on two datasets. The first one is Robust04, consisting of more than 500k news articles from news agencies, used in the TREC Robust Track 2004, with 250 topics. The second dataset is ClueWeb09-CategoryB, including over 50 million English Webpages, used in the TRECWeb Track for several years. We filtered out the spam documents detected using the Waterloo spam ranker [6] with the default threshold 60%. This dataset includes 200 topics. The title of topics are used as queries. For each query, we retrieve the top 2,000 documents as training data or as candidate documents to be re-ranked by the trained models at inference time. In our experiments, we use the Lemur implementation of the query likelihood model with the default parameters to retrieve initial documents. Words are stemmed using the Porter stemmer and stopwords are removed.

Available labeled data is divided into train, validation, and test sets by random selection of 40%, 30%, and 30% of data, respectively. The results on test folds are then evaluated using Mean Average Precision (MAP), precision at top k documents (P@k), and normalized Discounted Cumulative Gain calculated for the top k documents (nDCG@k), where k is 1, 3, 5, and 10. Statistical significant tests are performed using the two-tailed paired t-test at the 0.05 level.

Experimental setup. We use the publicly available implementation of the DRMM model. The input of the network is built by using pre-trained word embeddings of dimension 300 from Glove [16]. In our experiments, the model parameters are optimized employing the AdaDelta optimizer and back-propagation algorithm to compute gradients. All model hyperparameters are tuned on the validation set. For each model, the number of hidden layers and their sizes are selected from $\{2, 3\}$ and $\{50, 100, 150, 200\}$, respectively. The initial learning rate is selected from $\{0.2, 0.4, 0.6, 0.8, 1\}$.

For training with the listwise loss function, we re-shuffle the queries in the training data and documents associated with each query, before each epoch of training. We then randomly sample x out of 2,000 documents available for each query to derive training data, where x is selected from $\{500, 1000, 1500\}$. For test queries, we use all 2,000 documents to provide a ranking, thus no sampling is done at inference time.

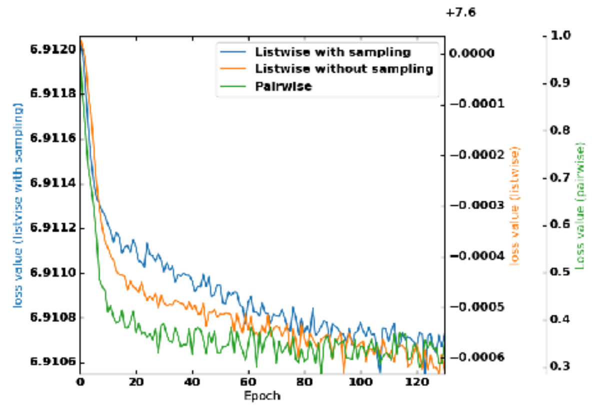

Figure 1: Training loss curves.

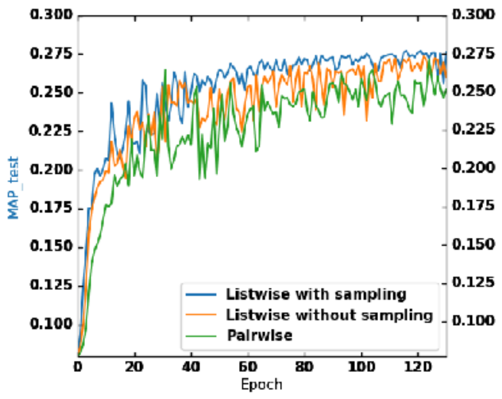

Figure 2: MAP performance on test set calculated during training of neural ranking models with different loss functions (not used for optimization of model parameters).

We compare three neural ranking models: (1) DRMM$_{pl}$ which is the original DRMM model using its pairwise loss function [8], (2) DRMM$_{ll}$ which is the DRMM model trained using the listwise loss function in Eq. (3), and (3) DRMM$_{ll-ws}$ which is the DRMM model when it is trained using the listwise loss function but without random sampling of documents. As we do not aim to obtain the best possible performance using listwise loss functions, we leave training the DRMM model with loss functions of conventional LambdaMart or LambdaRank algorithms for future.

4.1 Results and discussion.

Tables 1 and 2 show the performance of the DRMM model trained with different loss functions. As the results show, training the neural ranking model with the listwise loss function improves almost all evaluation metrics, and improvements on top-ranked documents are higher. This conforms with the comparative results of conventional listwise to pairwise learning algorithms [17]. The other interesting point about the results is that training using listwise loss function shows higher improvement percentage on the ClueWeb dataset compared to the Robust04 dataset. Since ClueWeb is a dataset built from Webpages, the obtained result proves the importance of using a listwise loss function in practical settings where queries can have unbalanced relevance distributions.

Table 1: Precision and MAP performance of the neural rankers on different datasets. Better performance between $\mathrm{DRMM}_{\mathrm{pl}}$ and $\mathrm{DRMM}_{\mathrm{ll}}$ models is indicated in bold format. Symbol $^{\bullet}$ indicates that improvements of $\mathrm{DRMM}_{\mathrm{ll}}$ over $\mathrm{DRMM}_{\mathrm{pl}}$ are statistically significant.

Method	Robust04					ClueWeb				
	P@1	P@3	P@5	P@10	MAP	P@1	P@3	P@5	P@10	MAP
$\mathrm{DRMM}_{\mathrm{pl}}$	0.52	0.4844	0.4613	**0.424**	0.2923	0.3833	0.3278	0.3367	0.3267	0.2634
$\mathrm{DRMM}_{\mathrm{ll}}$	**0.56**$^{\bullet}$	**0.5244**$^{\bullet}$	**0.4773**	**0.424**	**0.2989**	**0.4167**	**0.3722**$^{\bullet}$	**0.37**	**0.355**$^{\bullet}$	**0.2725**$^{\bullet}$
$\mathrm{DRMM}_{\mathrm{ll-ws}}$	0.4667	0.5289	0.4747	0.4227	0.2962	0.4	0.3722	0.3667	0.36	0.2634

Table 2: nDCG performance of the neural rankers on different datasets. Better performance between $\mathrm{DRMM}_{\mathrm{pl}}$ and $\mathrm{DRMM}_{\mathrm{ll}}$ models is indicated in bold format. Symbol $^{\bullet}$ indicates that improvements of $\mathrm{DRMM}_{\mathrm{ll}}$ over $\mathrm{DRMM}_{\mathrm{pl}}$ are statistically significant.

Method	Robust04				ClueWeb			
	nDCG@1	nDCG@3	nDCG@5	nDCG@10	nDCG@1	nDCG@3	nDCG@5	nDCG@10
$\mathrm{DRMM}_{\mathrm{pl}}$	0.4666	0.4422	0.4385	0.4252	0.3484	0.3118	0.32	0.3277
$\mathrm{DRMM}_{\mathrm{ll}}$	**0.4711**	**0.4742**$^{\bullet}$	**0.4528**	**0.4355**	**0.3732**	**0.359**$^{\bullet}$	**0.3595**$^{\bullet}$	**0.3666**$^{\bullet}$
$\mathrm{DRMM}_{\mathrm{ll-ws}}$	0.4133	0.4589	0.442	0.4283	0.3565	0.3539	0.3519	0.3638

Figure 1 shows learning curves of the DRMM model trained with pairwise and listwise loss functions. As the diagram indicates, optimization of listwise loss converges with less fluctuation compared to that of pairwise loss. This observation demonstrates that more reliable estimates of the gradient of the listwise loss function are obtained compared to those of the pairwise loss function. Figure 2 shows MAP performance on test data after each step of training. In the final steps shown in the diagram, all networks using pairwise or listwise loss functions are sufficiently trained according to the loss of validation set. However, performance of the pairwise trained model still shows large fluctuations, which is not the case for the listwise trained model. This observation shows that the learned ranking model using a listwise loss function is more robust than that using a pairwise loss function.

4.2 Impacts of Data Sampling

Tables 1 and 2 also include the performance of DRMM model when it is trained using the listwise loss function but without random sampling of documents. Although training on the entire set of documents associated with each query has lower performance than that with random sampling of documents, it improves the performance of the pairwise model for all metrics except P@1, P@10, and nDCG@1 over Robust04. We believe the lower performance is because of fewer training samples in the listwise setting compared to the pairwise setting given the same amount of labeled data. The learning curves for the cases of training with and without random sampling of documents before each epoch of training are also depicted in Figure 1. The learning curves demonstrate that without sampling, we get a sharper decrease in training loss. Keskar et al. [11] have shown that a flat minimizer of loss better generalizes to test data. This fact justifies the performance improvements obtained by random sampling of documents. Figure 2 shows that with sampling, not only does the performance of the trained model increase on test data, but the performance also has less fluctuations in the final steps of training, which is desirable.

5 CONCLUSION AND FUTURE WORK

We demonstrate how listwise loss functions can improve the retrieval performance of neural ranking models. More specifically, we examine how training the *deep relevance matching model*, which is a pairwise model, with a listwise loss function impacts the performance of retrieval. We also show that reshuffling and random sampling of documents associated with each query before each epoch of training improves the performance of retrieval. There are several possible directions for future work. A promising line is to examine more smart sampling approaches as we demonstrated that sampling can improve the retrieval performance. We also would like to investigate how other listwise loss functions impact the performance of neural ranking models.

Acknowledgements. This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *SIGIR '18*. 135–144.

[2] Qingyao Ai, Xuanhui Wang, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Scoring Functions Using Deep Neural Networks. In *DAPA: The WSDM 2019 Workshop on Deep Matching in Practical Applications*.

[3] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A Closer Look at Memorization in Deep Networks. In *ICML'17*. 233–242.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *ICML*.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *ICML '07*. 129–136.

[6] Gordon V. Cormack, Mark D. Smucker, and Charles L. Clarke. 2011. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Inf. Retr.* (2011).

[7] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR '17*. 65–74.

[8] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM'16*. 55–64.

[9] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM '13*. 2333–2338.

[10] Ayyoob Imani, Amir Vakili, Ali Montazer, and Azadeh Shakery. 2019. Deep neural networks for query expansion using word embeddings. In *European Conference on Information Retrieval*. Springer, 203–210.

[11] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *CoRR* abs/1609.04836 (2016).

[12] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (March 2009), 225–331.

[13] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *HLT | NAACL*. 912–921.

[14] Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep Relevance Ranking Using Enhanced Document-Query Interactions. In *EMNLP*. 1849–1860.

[15] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *WWW '17*.

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP'14*. 1532–1543.

[17] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Inf. Retr.* 13, 4 (2010).

[18] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *WWW '14 Companion*. 373–374.

[19] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. 2018. Neural Ranking Models with Multiple Document Fields. In *WSDM '18*.

[20] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *SIGIR '15*. 575–584.