Learning to Rank with Query-level Semi-supervised Autoencoders

Bo Xu
IR Lab., School of Computer Science and Technology,
Dalian University of Technology, Dalian, China.
xubo2011@mail.dlut.edu.cn

Hongfei Lin
IR Lab., School of Computer Science and Technology,
Dalian University of Technology, Dalian, China.
hflin@dlut.edu.cn

Yuan Lin
Wise Lab., School of Public Administration and Law,
Dalian University of Technology, Dalian, China.
zhlin@dlut.edu.cn

Kan Xu
IR Lab., School of Computer Science and Technology,
Dalian University of Technology, Dalian, China.
xukan@dlut.edu.cn

ABSTRACT

Learning to rank utilizes machine learning methods to solve ranking problems by constructing ranking models in a supervised way, which needs fixed-length feature vectors of documents as inputs, and outputs the ranking models learned by iteratively reducing the pre-defined ranking loss. The document features are always extracted based on classic textual statistics, and different features contribute differently to ranking performance. Given that well-defined features would contribute more to the retrieval performance, we investigate the usage of autoencoders to enrich the feature representations of documents. Autoencoders, as basic building blocks of deep neural networks, have been successfully used in many text mining tasks for generating effective features. To enrich the feature space for learning to rank, we introduce supervision into the loss functions of autoencoders. Specifically, we first train a linear ranking model on the training data, and then incorporate the learned weights into the reconstruction costs of an autoencoder. Meanwhile, we accumulate the costs of documents for a given query with query-level constraints for producing more useful features. We evaluate the effectiveness of our model on three LETOR datasets, and show that our model can generate effective document features to improve the retrieval performance.

CCS CONCEPTS

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

KEYWORDS

Learning to rank; autoencoders; semi-supervised learning

1 INTRODUCTION

Ranking is one of the central issues in information retrieval. Given a query, information retrieval systems seek to rank the candidate documents based on their relevance to the given query. To solve the problem, learning to rank has been proposed and proved effective to learn powerful ranking models using supervised machine learning methods [7, 9].

Like traditional machine learning methods, learning to rank entails fix-length features vectors of documents as input, and outputs ranking models by iteratively reducing the ranking loss. One of the key issues for learning to rank is how to select the features for different documents, which should not only reflect the characteristics of the documents, but also model the relationships between documents and the corresponding query. In most cases, classic unsupervised ranking models or textual statistics can be used as document features for learning to rank to produce relatively good performance. In this sense, learning to rank can also be taken as a way to combine different ranking models or textual statistics to form a strong one. However, since the number of existed ranking models and textual statistics is limited, it becomes more and more difficult for enriching the feature space to enhance the learned ranking models.

In recent years, deep learning methods exhibit powerful capabilities in generating highly useful and compact features from structured information of data for many natural language processing tasks [6, 10, 11]. Autoencoder [2], as one of the basic building blocks for deep neural network-based framework, has been proven effective in learning high-quality representations of data by reconstructing the input and meantime retaining the inherent structure of the input. The reconstruction is achieved by optimizing the pre-defined loss function for measuring the differences between the input and the output. The loss function in reconstruction restricts the learned feature keeping the data structure as much as possible. Therefore, the reliability of an autoencoder can be estimated based on its reconstruction capability.

Some previous studies have focused on learning effective features for different tasks using autoencoders. For example, Zhai et al. [13] proposed to model textual data using autoencoders by introducing supervision via the loss function, and successfully improved the performance of sentiment analysis. Motivated by their work, we propose to employ autoencoders to enrich the feature space for learning to rank. Specially, we first train a linear ranking model to learn the importance degrees of the original features, and then optimize the reconstruction loss based on the weights from the linear ranking model to learn an autoencoder. Meanwhile,

CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3133049

we accumulate the costs of documents for different queries using defined query-level constraints to produce more useful features. Based on the enriched features, we train ranking models and examine their usefulness. Experiments on LETOR datasets demonstrate the effectiveness of our methods in generating effective document features for improving the retrieval performance. Since our method is general, it can be applied for different tasks to solve ranking problems.

2 QUERY-LEVEL SEMI-SUPERVISED AUTOENCODERS

In this section, we will introduce our method in detail. We first give a brief introduction on denoising autoencoder, and then introduce the loss function of semi-supervised autoencoders for learning to rank. Finally, we define and incorporate the query-level constraints into the loss function to enhance the reconstruction capability of autoencoders for query-level performance.

2.1 Denoising Autoencoders

Autoencoders, as building blocks for deep neural networks, comprise one input layer, one output layer, and at least one hidden layer. A well-performed autoencoder can encode its inputs as low-dimensional representations in its hidden layer, and decode the representations of the hidden layer as outputs for reconstructing the original inputs. In other words, the aim of an autoencoder is to learn a hidden code of original inputs, which reconstructs the inputs with least deviations. Formally, the representations of the hidden layer and the output layer can be formalized as follows.

$$\mathbf{y} = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{1}$$

$$\hat{\mathbf{x}} = f(\mathbf{W}_2\mathbf{y} + \mathbf{b}_2) \tag{2}$$

where $\mathbf{W}_1$ is the weight matrix between the input layer and the hidden layer, $\mathbf{W}_2$ is the weight matrix between the hidden layer and the output layer, and $\mathbf{b}_1$ and $\mathbf{b}_2$ are the biases for the hidden layer and the output layer. Tied weights are usually used to speed up the training and avoid overfitting when applying autoencoders to different tasks, which means to set $\mathbf{W}_1 = \mathbf{W}_2$. We also adopt tied weights in our experiments.

As one variant of traditional autoencoders, denoising autoencoders have been demonstrated effective and robust in many tasks, which reconstructs a noised version of the inputs from the outputs to learn more powerful representations of the original inputs. To improve the performance of learning to rank, we adopt the denoising autoencoder to learn robust and compact features of documents, and extend the input space of learning to rank method with feature representations $\mathbf{y}$ in the hidden layer of the denoising autoencoders.

2.2 Loss Function as Bregman Divergence

Loss function measures the differences between the inputs and the outputs of an autoencoder, which is defined to target the learning and generate effective feature representations. To learn a better autoencoder for learning to rank, we attempt to adopt an effective loss function to measure the reconstruction errors in the training process, and optimize the autoencoder iteratively to produce high-quality hidden representations. A general form of loss function for an autoencoder can be defined as follows.

$$loss = \sum_{i=1}^{n} ||x_i - \hat{x_i}||_2^2 \tag{3}$$

where n is the total number of instances in the training data. Euclidean distance is often used to measure the differences between the inputs and the outputs for an autoencoder. The learning target is to minimize the loss and find the optimal parameters. Since the loss function in Eq.(3) takes all the features of the inputs equally to accumulate the losses, the performance may be limited because different features contribute unevenly to characterize the inputs. Therefore, it is necessary to take feature importance into consideration to accumulate the losses in the reconstruction of an autoencoder. To this end, we introduce a modified version of loss function of autoencoders based on Bregman divergence [1], which has been demonstrated effective in a sentiment analysis task [13]. The modified loss function can be formalized as follows.

$$loss = \sum_{i=1}^{n} \theta^T (x_i - \hat{x_i})^2 \tag{4}$$

where θ is a vector of pre-trained weights on different features. Based on this setting, the modified autoencoder can encode the representations of the original data as more powerful representations in the hidden layers, where θ is learned based on a pre-trained model. To reduce the loss, parameters including $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, $\mathbf{b}_2$ are optimized for learning effective hidden representations.

Motivated by this idea, we pretrain an initial linear ranker to obtain the weights on different features in building autoencoders to produce effective features for learning to rank. In the proposed method, we use ListNet [4] to train the initial ranker. ListNet is a listwise learning to rank method based on neural networks, which is the listwise version of the pairwise method RankNet [3]. To pre-train a linear ranking function, we employ a linear version of ListNet as its scoring function, and train a ranking model to obtain the weights on different features. Since we use the ground truth labels to pretrain a linear ranker for supervised feature weighting, and meanwhile train the denoising autoencoder in an unsupervised way, our method can be considered as learning semi-supervised denoising autoencoders.

2.3 Query-level Constraints

Unlike traditional machine learning tasks, the training data for learning to rank consists of multiple subsets of documents, each of which corresponds to one query. The learning target is to learn a ranking model to improve the average performance for ranking lists of documents to these queries. Although the loss function defined in Eq. (4) takes the importance of different features into consideration while learning the autoencoders, it ignore the query-level constraints, which may be very important for learning to rank, because documents with respect to different queries are incomparable with each other. To deal with the problem, we attempt to incorporate another item into the loss function of autoencoders in consideration of the query-level constraints to produce more effective document features. Since an autoencoder is designed to reconstruct the inputs by measuring the distance between its inputs and its outputs, we attempt to model the query-level constraints by measuring the differences of the query-level retrieval performance between the inputs and the outputs.

Specifically, given a query, we use the pre-trained ranker based on ListNet to measure the retrieval performance in terms of a certain evaluation measure. We record the retrieval performance both for that based on the original features and for that based on the extended feautres. Intuitively, the differences between the performance based on the two feature sets can indicate the reconstruction capability from the query level. Namely, if the performance varies a lot, the reconstruction loss would be increased because the deviation may be large. We formalize the idea as follows in the loss function.

$$loss = \sum_{q \in Q} \eta(q)(\sum_{i=1}^{n(q)} \theta^T (\hat{x_i} - x_i)^2) \tag{5}$$

where $\eta(q)$ measures the differences of the changes on retrieval performance, and $n(q)$ is the number of documents corresponding to the query q. We accumulate the losses across all the queries in the training set by iterations. To reduce the loss, parameters including $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are optimized for learning effective hidden representations. $\eta(q)$ can be defined as follows.

$$\eta(q) = \frac{|Eval_{ori} - Eval_{rec}|}{Eval_{ori}} \tag{6}$$

where $Eval_{ori}$ and $Eval_{rec}$ are the retrieval performance of the original inputs and the reconstructed outputs of an autoencoder evaluated by any given IR evaluation measure denoted as $Eval$. We estimate the performance by applying the ListNet-based ranker on the corresponding features. Any evaluation measures can be used to measure the differences. Based on the equation, we incorporate query-level constraints of learning to rank into the loss function of the modified autoencoders for learning more effective document features.

3 EXPERIMENTS AND ANALYSIS

3.1 Experimental Settings

We examine the effectiveness of our method on three LETOR[1] datasets released by Microsoft Research Asia, i.e. the OHSUMED collection, the MQ2007 collection and the MQ2008 collection. We adopt P@k, NDCG@k (short for N@k) and Mean Average Precision (MAP) as evaluation measures to evaluate the ranking performance. Five-fold cross validations are performed based on the standard divisions of these datasets. For the query-level constraint, we take NDCG@10 as the evaluation measure $Eval$ to compute the query-level loss during model training. We take autoencoders with different constraints as baseline models for learning the features, including naive autoencoders, sparse autoencoders [8] and denoising autoencoders [12]. We extend the feature spaces of original datasets to train different ranking models for comparisons.

[1]http://research.microsoft.com/enus/um/people/letor/

Table 1: Retrieval performance of retrieval models based on different extended features on three datasets. Significant improvement of the proposed models with respect to the baseline models (*original*) (two-tailed paired t test, $p \leq 0.05$) is indicated with a dagger† .

OHSUMED	P@3	P@10	N@3	N@10	MAP
original	0.6016	0.4975	0.4732	0.4410	0.4457
naive	0.4969	0.4811	0.4019	0.4056	0.4298
sparse	0.5252	0.4925	0.4310	0.4234	0.4364
denoising	0.5063	0.4925	0.4312	0.4294	0.4381
SA	0.5881	0.5094†	0.4784†	0.4535†	0.4505†
QSA	**0.6132**†	**0.5142**†	**0.4961**†	**0.4574**†	**0.4537**†
MQ2007	P@3	P@10	N@3	N@10	MAP
original	0.4334	0.3798	0.4091	0.4440	0.4652
naive	0.3974	0.3512	0.3686	0.4047	0.4362
sparse	0.4375	0.3788	0.4136	0.4457	0.4677
denoising	0.4352	0.3788	0.4131	0.4460	0.4683
SA	0.4379	0.3785	0.4130†	0.4456	0.4697†
QSA	**0.4450**†	**0.3819**†	**0.4195**†	**0.4496**†	**0.4720**†
MQ2008	P@3	P@10	N@3	N@10	MAP
original	0.3835	0.2476	0.4324	0.2303	0.4775
naive	0.3618	0.2394	0.4009	0.2112	0.4528
sparse	0.3941	**0.2490**	0.4401	0.2330	0.4867
denoising	0.3946	0.2473	0.4385	0.2291	0.4850
SA	0.3958†	0.2481	0.4428†	0.2322†	0.4881†
QSA	**0.3984**†	0.2487†	**0.4433**†	**0.2333**†	**0.4929**†

3.2 Overall Retrieval Performance of Different Autoencoders

We first report the overall retrieval performance of different autoencoders on three datasets in Table 1, where ListNet is used both for learning the pre-trained rankers and for learning the final ranking models. In the table, *original* stands for the ranking models solely based on the original features without extension. *naive*, *sparse* and *denoising* stand for the ranking models based on the extended features by naive autoencoders, sparse autoencoders and denoising autoencoders, respectively. *SA* and *QSA* stand for the ranking models based on the extended features by semi-supervised autoencoders with and without query-level constraints, respectively.

From the table, we can see that ranking model based on original feature set outperforms those based on either naive autoencoders or autoencoders with sparse or denoising constraints, which indicates that directly applying autoencoders to learn extra document features is of little use to improve the performance. The performance can be enhanced when incorporating semi-supervised feature weighting and further enhanced with semi-supervised query-level autoencoders. One possible explanation for this can be that our method learns useful and implicit information of documents, which cannot easily been captured with only original document features, thus enhancing the retrieval performance.

Table 2: Retrieval performance of ranking models using different learning to rank methods on OHSUMED dataset. The values in parentheses are the relative rates of improvement of the extended feature set over the original feature set. Significant improvements are marked with a dagger† .

Method	P@3	p@10	N@3	N@10	MAP
RF_{ori}	0.5000	0.4755	0.3966	0.3976	0.4269
RF_{ext}	0.5818†	0.4953†	0.4645†	0.4389†	0.4350†
	(16.35%)	(4.17%)	(17.13%)	(10.38%)	(1.91%)
RB_{ori}	0.5609	0.4966	0.4555	0.4302	0.4411
RB_{ext}	0.6195†	0.5038†	0.4897†	0.4484†	0.4478†
	(10.45%)	(1.44%)	(7.51%)	(4.24%)	(1.52%)
LN_{ori}	0.6016	0.4975	0.4732	0.4410	0.4457
LN_{ext}	0.6132†	0.5142†	0.4961†	0.4574†	0.4537†
	(1.93%)	(3.35%)	(4.85%)	(3.72%)	(1.81%)

3.3 Retrieval Performance of Different Learning to Rank Methods

We then examine the performance of our method using different learning to rank methods, including Random Forests (RF), RankBoost [5] (RB), ListNet [4](LN). These methods are used to construct the final ranking models in the experiment, and due to the limited space we only give the results on OHSUMED dataset in Table 2, where *ori* stands for the ranking models learned with the original features, and *ext* stands for the ranking models learned with extended features using ListNet-based pretraining.

Based on the experimental results, we find that ranking models can achieve better performance when learned with the extended features across different learning to rank methods, which indicates that the learned document features by our method are effective to improve the retrieval performance.

3.4 Impact of Dimensionality

We also examine the impact of dimensionality of learned features on the retrieval performance, shown in Fig.1. In the figure, horizontal axis stands for the percentage of new features compared to the original features, and the vertical axis stands for the retrieval performance evaluated by MAP. From the results, we find that when the number of new learned features amounts to sixty or seventy percent of the number of original features, we can achieve the best performance on the datasets.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel method to construct query-level semi-supervised autoencoders for learning to rank to learn effective document features. In the method, we pre-train a ListNet-based ranker to give weights on original features, and incorporate query-level constraints into the loss function to improve the effectiveness of the learned features. Experimental results show that our method outperforms different autoencoder-based models for learning document features, and enhances the ranking performance for different learning to rank methods over the original feature set. We will carry out our future work by improving our method with further optimizations and investigating the effectiveness of the learned features in some domain-specific IR tasks.

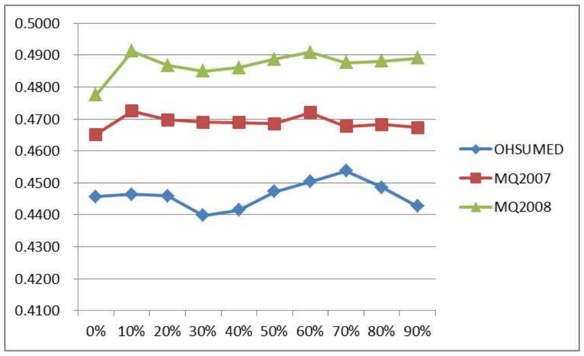

Figure 1: Impact of dimensionality of the learned features

ACKNOWLEDGMENTS

This work is partially supported by grant from the Natural Science Foundation of China (No. 61632011, 61572102, 61402075, 61602078, 61572098), the Ministry of Education Humanities and Social Science Project (No. 16YJCZH12), the Fundamental Research Funds for the Central Universities, the National Key Research Development Program of China (No. 2016YFB1001103).

REFERENCES

[1] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. 2004. Clustering with Bregman Divergences. *Journal of Machine Learning Research* 6, 4 (2004), 1705–1749.
[2] Yoshua Bengio. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127.
[3] Chris J C Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. *International conference on machine learning (ICML)* (2005), 89–96.
[4] Zhe Cao, Tao Qin, Tie Yan Liu, Ming Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. *International conference on machine learning (ICML)*, 129–136.
[5] Yoav Freund, Raj D Iyer, Robert E Schapire, and Yoram Singer. 1998. An Efficient Boosting Algorithm for Combining Preferences. *International conference on machine learning (ICML)*, 170–178.
[6] Posen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. *ACM international conference on information and knowledge management (CIKM)* (2013), 2333–2338.
[7] Tieyan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[8] Andrew Ng. 2011. Sparse autoencoder. *CS294A Lecture notes* 72, 2011 (2011), 1–19.
[9] Tao Qin, Tie Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval Journal* 13, 4 (2010), 346–374.
[10] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. *International ACM SIGIR conference on research and development in information retrieval(SIGIR)* (2015), 373–382.
[11] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. *International world wide web conferences (WWW)* (2014), 373–374.
[12] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierreantoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. (2008), 1096–1103.
[13] Shuangfei Zhai and Zhongfei Zhang. 2016. Semisupervised Autoencoder for Sentiment Analysis. *The association for the advancement of artificial intelligence (AAAI)* (2016), 1394–1400.