Deep Neural Networks for Learning to Rank

Aymen CHERIF AYMEN.CHERIF@EURANOVA.EU
EURA NOVA

Salim JOUILI SALIM.JOUILI@EURANOVA.EU
EURA NOVA

Goeric HUYBRECHTS GOERIC.HUYBRECHTS@GMAIL.COM
Universite Catholique de Louvain

Keywords: fa

Abstract

In this work, we focus on the issue of learning to rank in the document retrieval area. We propose a deep neural network architecture to address this problem. Our approach is based on the comparative network proposed in (Rigutini et al., 2011). We show that deeper architecture can perform better by applying training strategies that are particularly suited for deep learning. Our experimental results on the TD2003 dataset of the LETOR benchmark (Qin et al., 2010) show that these well-trained deep neural networks outperform the state-of-the-art algorithms.

1. Introduction

Learning to rank refers to the application of supervised machine learning techniques to construct ranking models for information retrieval systems. In such a task, the objective is to sort as accurately as possible a list of items with respect to a query. According to (Liu, 2009), the existing algorithms can be categorized into three main groups:

- **Pointwize approach**(Liu, 2009; Li et al., 2007) Each query-document pair has a given score. In this case, the learning to rank problem can be seen as a regression or classification problem. Although many existent algorithms can solve easily such a problem, the relative order between documents isn't taken into account.

- **Pairwize approach**(Rigutini et al., 2011; Burges et al., 2007; Freund et al., 2003; Burges et al., 2005) The model is given a pair of document with respect to a query and learns a preference function that predicts the most relevant one. The problem becomes a classical binary classification while keep partially relative order information.

- **Listwize approach**(Xu & Li, 2007; Cao et al., 2007) The models takes lists as instances in the learning process. This produces a more natural way to tackle the problem. However, it becomes more challenging due to the non-continuous and non-differentiable aspect of the loss function.

Deep leaning has emerged recently as a promising field of machine learning. Recently many successful applications in various fields such as computer vision and speech recognition has been proposed. For more information about deep learning the reader may refer to (Bengio, 2009). In this contribution we will focus on pairwize approaches.

2. Proposed algorithm

Our neural networks are inspired by the comparative networks(CmpNN) proposed in (Rigutini et al., 2011). It is a feed-forward neural network consisting of one hidden layer and two output nodes. The input layer takes the concatenation ($[x, y]$) of two documents, x and y, as input. The values of the two output nodes estimate respectively the evidence of $x < y$ and the evidence of $x > y$. The particularity about the CmpNN architecture is its weight sharing property that leads to a certain symmetry between hidden neurons. This is especially suited to model a preference function since

Preliminary work. Under review for Benelearn 2016. Do not distribute.

in allows the network to respect the reflexivity, equivalence and anti-symmetry properties (Rigutini et al., 2011).

It's useful to stress the importance of weight sharing over using simple dense layers, especially for deep architecture. Since adding symmetric layers do not compromise the initial properties and help very significantly the network to learn. During training, we adopted a learning rate decay with a classical gradient descent method with the Squared Error loss.

To train efficiently the network, we adopted relu activations, dropout for regularization and a pre-training for network initialization through a symmetric auto-encoders (Bengio, 2009). These choice results of several experiments.

Finally, we also adopted a rank agregation techniques which improved significantly the results. It consists of using an ensemble of ranker. Similar to a bagging, the idea is to combine several ranking results to obtain the final decision. the a document is ranked on the more it's final ranking will be on the top.

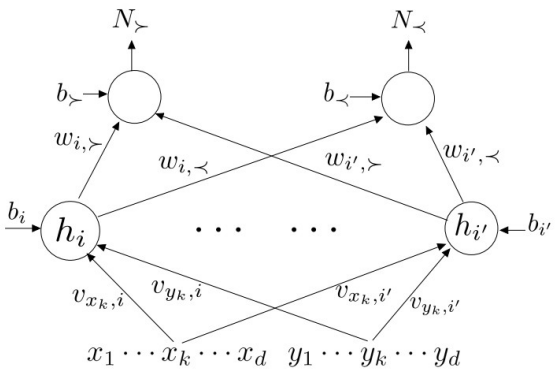


Figure 1. example of deep comparative neural network architecture from (Rigutini et al., 2011). We extend this architecture with more hidden symmetric layers

3. Results

The table 1 summarizes the results on TD2003 dataset. Our models are trained on a small version of the dataset with respectively $10K$ and $5K$ training and validation pairs. We referenced it as DeepNet in table 1. Except for the ranked aggregation model which is train on big dataset ($100K$ and $50K$) referenced as DeepNet(B) in table 1. The results are compared to state of the art algorithms. Including two version of

Table 1. Summary of performed experiences on TD2003 dataset.

	MAP	P@10	NDCG@10
RankBoost	0.212	0.18	0.29
RankSVM	0.256	0.21	0.34
FRank	0.245	0.19	0.34
ListNet	0.273	0.22	0.37
AdaRank	0.185	0.16	0.27
SortNet	0.307	0.24	0.39
SortNet (B)	0.259	0.22	0.36
DeepNet	0.308	0.24	0.41
DeepNet (B)	**0.320**	**0.25**	**0.42**

SortNET: the paper version with active learning procedure trained on the big dataset (SortNet) and our implementation without the active learning procedure and trained on the small set (SortNet (B)). In both cases we used a three layered neural network 24-12-6 respectively in hidden layer 1, 2 and 3, relu activation function for the hidden layers and dropout as for regularization. Note that the networks initialization through symmetric auto-encoder in the pre-training phase has significantly contributed in improving the results.

We used in table 1 three evaluation measures to compare all these algorithms. the Mean Absolute Precision (MAP), the Precision at Position n (P@n) and Normalized Discounted Cumulative Gain (NDCG@n). For detailed information about these evaluation see (Liu, 2009). Our experiments clearly show that using well-trained deep neural networks significantly improves the results of shallow neural networks. An improvement of 7% and 5% can be observed compared to the two shallow neural networks. DeepNet outperforms all the considered state-of-the-art algorithms. It even outperforms SortNet for almost all evaluation measures, although it doesnt make use of the incremental learning procedure and is only considering a small fraction of the learning set.

4. Conclusion

In this work we addressed the pairwize learning to rank problem. We focused on neural network approaches and we investigated deeper architecture. The experiments show that starting from the comparative network architecture and with better training strategies deep architectures are able improve significantly the results. However, as future direction we may think to validate our results on other datasets. We may think taking into account datasets with more than one level of relevance. We can also address some dataset with low level features such as images.

References

Bengio, Y. (2009). Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, *2*, 1–127.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to Rank Using Gradient Descent. *Proceedings of the 22Nd International Conference on Machine Learning* (pp. 89–96). New York, NY, USA: ACM.

Burges, C. J., Ragno, R., & Le, Q. V. (2007). Learning to Rank with Nonsmooth Cost Functions. In B. Schölkopf, J. C. Platt and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19*, 193–200. MIT Press.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to Rank: From Pairwise Approach to Listwise Approach. *Proceedings of the 24th International Conference on Machine Learning* (pp. 129–136). New York, NY, USA: ACM.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An Efficient Boosting Algorithm for Combining Preferences. *J. Mach. Learn. Res.*, *4*, 933–969.

Li, P., Wu, Q., & Burges, C. J. (2007). McRank: Learning to Rank Using Multiple Classification and Gradient Boosting (pp. 897–904.).

Liu, T.-Y. (2009). Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.*, *3*, 225–331.

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*, 346–374.

Rigutini, L., Papini, T., Maggini, M., & Scarselli, F. (2011). SortNet: Learning to Rank by a Neural Preference Function. *IEEE Transactions on Neural Networks*, *22*, 1368–1380.

Xu, J., & Li, H. (2007). AdaRank: A Boosting Algorithm for Information Retrieval. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 391–398). New York, NY, USA: ACM.