Learning to Rank from Relevance Judgments Distributions

Discussion Paper

Alberto Purpura[1,*], Gianmaria Silvello[2] and Gian Antonio Susto[2]

[1]*IBM Research Europe, Dublin, Ireland*
[2]*University of Padua, Padova, Italy*

Abstract

LEarning TO Rank (LETOR) algorithms are usually trained on annotated corpora where a single relevance label is assigned to each available document-topic pair. Within the Cranfield framework, relevance labels result from merging either multiple expertly curated or crowdsourced human assessments. In this paper, we explore how to train LETOR models with relevance judgments distributions (either real or synthetically generated) assigned to document-topic pairs instead of single-valued relevance labels. We propose five new probabilistic loss functions to deal with the higher expressive power provided by relevance judgments distributions and show how they can be applied both to neural and gradient boosting machine (GBM) architectures. Overall, we observe that relying on relevance judgments distributions to train different LETOR models can boost their performance and even outperform strong baselines such as LambdaMART on several test collections.

Keywords

Learning to Rank, Machine Learining, Optimization Functions, Information Retrieval

1. Introduction

Ranking is a problem that we encounter in a number of tasks we perform every day: from searching on the Web to online shopping. Given an unordered set of items, this problem consists of ordering the items according to a certain notion of relevance. Generally, in Information Retrieval (IR) we rely on a notion of relevance that depends on the information need of a user, expressed through a keyword query. When creating a new experimental collection, the corresponding relevance judgments are obtained by asking different judges to assign a relevance score to each document-topic pair. Multiple judges – either trained experts or participants of a crowdsourcing experiment – usually assess the same document-topic pair, and the final relevance label for the pair is obtained by aggregating these scores [1]. This process is a cornerstone for system training and evaluation and has contributed to the continuous development of IR, especially in the context of international evaluation campaigns. Nonetheless, the opinion of different judges on the same document-topic pair might be very different or even diverge

to the opposite ends of the spectrum – either because of random human errors or due to a different interpretation of a topic. Inevitably, the aggregation process conflates the multiple assessors viewpoints on document-topic pairs onto a single one, thus losing some information – even though it also reduces annotation errors and outliers. Our research hypothesis is that Machine Learning (ML) models – i.e., LEarning TO Rank (LETOR) [2] and Neural Information Retrieval (NIR) [3] models – could use all the labels collected in the annotation process to improve the quality of their rankings. Indeed, judges' disagreement on a certain document-topic pair can be due to an inherent difficulty of the topic or to the existence of multiple interpretations of it. We argue that designing ML models able to learn from the whole distributions of relevance judgments could improve the models' representation of relevance and their performance through the usage of this additional information. Following this idea, we propose to interpret the output of a LETOR model as a probability value or distribution – according to the experimental hypotheses – and define different Kullback–Leibler (KL) divergence-based loss functions to train a model using a distribution of relevance judgments associated to the current training item. Such a training strategy allows us to leverage all the available information from human judges without additional computational costs compared to traditional LETOR training paradigms.

The loss functions we propose [4] can be used to train any ranking model that relies on gradient-based learning, including popular NIR models or LETOR ones. In our experiments [4] we focus on one transformer-based neural LETOR model and on one decision tree-based Gradient Boosting Machine (GBM) model – the model at the base of the popular LambdaMART [5] ranker and used as a strong baseline in many recent LETOR research papers such as [6, 7, 8, 9, 10]. We assess the quality of the proposed training strategies on four standard LETOR collections (MQ2007, MQ2008, MSLR-WEB30K [11] and OHSUMED [12]). The paper is organized as follows: in Section 2 we present the details of the training strategies we propose; in Section 3 we present the most significant evaluation results we achieve and in Section 4 we report our conclusions.

2. Proposed Approach

We propose five different loss functions formulations that allow ranking models to take advantage of relevance judgments distributions prior to their aggregation. They are all based on two intuitions that allow us to model relevant judgments as probability distributions and to use KL divergence-based measures to compare them. We first propose to interpret the relevance label (or a set of relevance labels) assigned to the same document as if it was generated by a Binomial (or Multinomial) random variable modeling the judges' annotation process. For example, we assume that n assessors provided one binary relevance label for each document-topic pair, i.e., to state whether the pair was a relevant or a not-relevant one. This process can be modeled as a Binomial random variable $P \sim \text{Bin}(n, p)$ where the success probability p for each sample is the average of the binary responses submitted in n trials. We can follow the same process to represent a set of relevance labels associated to the same document as a sample from a Multinomial random variable. We then apply the same reasoning for the interpretation of our model output probability score as another Binomial (or Multinomial) distribution $\hat{P} \sim \text{Bin}(n, \hat{p})$ with the same parameter n – empirically tuned for the numerical stability of the gradients during training – and probability $\hat{p}$ equal to the output of the model. The second option we

propose is to consider relevance labels associated to each document (or batch of documents) as samples from Gaussian (or multivariate Gaussian) random variables $P \sim \mathcal{N}(\mu_p, \sigma)$ with the same standard deviation σ but centered on a different point depending on the relevance label associated with the document. Depending on the modeling strategy, the proposed loss functions take the following formulations typical of pointwise, pairwise hinge [13] or listwise losses that are frequently employed in NIR [14] and LETOR approaches [15, 16, 17, 6].

- Pointwise$_{KL(Bin)} = \left(D_{KL}(P_i||\hat{P}_i) + D_{KL}(\hat{P}_i||P_i) \right) * w_i$, where we rescale each term in a training batch by a factor w_i, inversely proportional to the number of times an item of the same class (relevant or not relevant) appeared in it;

- Pointwise$_{KL(Mul)} = \left(D_{KL(Mul)}(P_i||\hat{P}_i) + D_{KL(Mul)}(\hat{P}_i||P_i) \right) * w_i$, where we employ a Multinomial random variable instead of a Binomial one to represent a set of relevance labels associated to a certain topic-document pair by an annotator;

- Pairwise$_{KL(Bin)} = \max(0, m - \text{sign}(p^+ - p^-)D_{KL(Bin)}(P_{Bin}^+, P_{Bin}^-)$, where m is a slack parameter to adjust the distance between the two distributions, p^+ and p^- are the outputs of the LETOR model associated to two documents – the former with a higher relevance label than the latter – $P_{Bin}^+ \sim \text{Bin}(n, p^+)$ and $P_{Bin}^- \sim \text{Bin}(n, p^-)$ are two Binomial distributions corresponding to a relevant and to a not-relevant document-topic pair, respectively;

- Pairwise$_{KL(\mathcal{N})} = \max(0, m - \text{sign}(p^+ - p^-)D_{KL(\mathcal{N})}(P_{\mathcal{N}}^+, P_{\mathcal{N}}^-)$, where we adapt the previous hinge-style loss function to represent labels distributions with a Gaussian random variable instead of a Binomial one;

- Listwise$_{KL(\mathcal{N}_{mult})} = D_{KL(\mathcal{N}_{mult})}(\hat{P}||P) * \boldsymbol{w}$, where we consider the relevance labels associated to multiple documents to rerank for the same query at the same time, modeling the list of relevance scores and their respective labels as multinomial Gaussian distributions.

We evaluate the proposed loss functions on different LETOR models, i.e. the LightGBM implementation of LambdaMART [5], and a simpler transformer-based neural model [18] with one self attention layer followed by a feed forward one. The experimental collections that we consider in our experiments are: MQ2007, MQ2008, MSLR-WEB30K [11] and OHSUMED [12], which are the experimental collections of reference in the LETOR domain. All collections are already organized in five different folds with the respective training, test and validation subsets. We report the performance of our model averaged over these folds with the exception of the MSLR-WEB30K collection where we only consider Fold 1 as in other popular research works [19, 20, 21, 10]. Our code and implementation of the proposed loss functions are available at: https://github.com/albpurpura/PLTR.

3. Evaluation

In Table 1, we report the most significant results of our performance evaluation. For each experimental collection, we report the performance of the best variants of a LambdaMART and a Transformer Neural Network (NN) model when trained with the different loss functions we

	Loss Function	ERR	P@1	P@3	P@5	nDCG@1	nDCG@3	nDCG@5	AP
MQ2007	GBM – LambdaMART	0.3211	0.4669	0.4397	0.4167	0.4217	0.4243	0.4285	0.4646
	GBM – Pointwise KL (Binomial)	**0.3233**	0.4752	0.4354	0.4167	0.4303	0.4207	0.4275	0.4591↓
	GBM – Listwise KL (Gaussian)	0.3219	0.4592	**0.4399**	0.4164	0.4152	0.4240	0.4267	0.4595
	NM – Pairwise KL (Gaussian)	0.3218	**0.4817**	0.4381	**0.4201**	**0.4350**	**0.4249**	**0.4318**	**0.4665**
	NM – Listwise KL (Gaussian)	0.3177	0.4657	0.4332	0.4145	0.4173	0.4192	0.4255	0.4634
MQ2008	GBM – LambdaMART	0.3045	0.4413	0.3869	0.3446	0.3858	0.4260	0.4664	0.4746
	GBM – Pointwise KL (Binomial)	**0.3072**	**0.4439**	**0.3941**	**0.3464**	**0.3935**	**0.4325**	**0.4690**	0.4771
	GBM – Listwise KL (Gaussian)	0.2991	0.4362	0.3852	0.3444	0.3801	0.4214	0.4633	**0.4775**
	NM – Pairwise KL (Gaussian)	0.3019	0.4375	0.3852	0.3398	0.3871	0.4222	0.4603	0.4697
	NM – Listwise KL (Gaussian)	0.3008	0.4349	0.3814	0.3457	0.3827	0.4171	0.4630	0.4729
WEB30K	**GBM – LambdaMART**	**0.3955**	**0.7918**	0.7541	0.7288	**0.5925**	**0.5711**	**0.5670**	0.6299
	GBM – Pointwise KL (Binomial)	0.3550↓	0.7789↓	0.7503	0.7261	0.5435↓	0.5344↓	0.5343↓	0.6326↑
	GBM – Listwise KL (Gaussian)	0.3861↓	**0.7918**	**0.7565**	**0.7323↑**	0.5825↓	0.5647↓	0.5615↓	**0.6347↑**
	NM – Pairwise KL (Gaussian)	0.3454↓	0.7753↓	0.7423↓	0.7166↓	0.5315↓	0.5209↓	0.5214↓	0.5971↓
	NM – Listwise KL (Gaussian)	0.3523↓	0.7612↓	0.7258↓	0.7016↓	0.5322↓	0.5152↓	0.5141↓	0.5871↓
OHSUMED	GBM – LambdaMART	0.4704	0.5283	0.4874	0.4906	0.4387	0.3980	0.4037	0.4175
	GBM – Pointwise KL (Binomial)	0.5036	0.5755	0.5220	0.5000	0.4953	0.4474	0.4330	0.4210
	GBM – Listwise KL (Gaussian)	0.5139	0.5755	0.5314	**0.5151**	0.5000	0.4643↑	**0.4525↑**	**0.4243**
	NM – Pairwise KL (Gaussian)	0.4520	0.5377	0.4403	0.4000↓	0.4481	0.3707	0.3481↓	0.2903↓
	NM – Listwise KL (Gaussian)	**0.5248**	**0.6038**	**0.5692↑**	0.5057	**0.5189**	**0.4796↑**	0.4456	0.3861↓

Table 1

Performance of different LETOR models (decision tree-based Gradient Boosted Machine (GBM) model or a simpler Transformer-Based Neural Model (NM)) trained with the best-performing proposed loss functions averaged over all topics. ↑ or ↓ indicate a statistically significant ($\alpha < 0.05$) difference with the LambdaMART model trained on the original relevance judgments. Best performance measures per collection are in bold as the loss function with the most best measures per collection.

described above. The performance measures we consider are the Precision@k, nDCG@k – with $k \in \{1, 3, 5\}$ – mean Average Precision (AP) and ERR [22]. In most of the cases, our simple Tranformer-based neural model trained with the proposed loss functions is able to outperform a LambdaMART model – one of the best performing state-of-the-art models often used as baseline in the LETOR literature. We also observe how a GBM-based model is able to benefit from the proposed loss functions. In fact, when evaluated on the MQ2007, MQ2008 and OHSUMED collections, the proposed variant of the GBM model trained with the Pointwise KL (Binomial) loss function outperforms the GBM - LambdaMART model according to different performance measures.

4. Conclusions

We presented different strategies to train a LETOR model relying on relevance judgments distributions. We introduced five different loss functions relying on the KL divergence between distributions, opening new possibilities for the training of LETOR models. The proposed loss functions were evaluated on a transformer-based neural model and on a decision tree-based GBM model – the same model employed by the popular LambdaMART algorithm [5] – over a number of experimental collections of different sizes. In our experiments, the proposed loss functions outperformed the aforementioned baselines in several cases and gave a significant performance boost to LETOR approaches – especially the ones based on neural models – allowing them to also outperform other strong baselines in the LETOR domain such as the LightGBM implementation of LambdaMART [5, 10].

References

[1] M. Hosseini, I. Cox, N. Milić-Frayling, G. Kazai, V. Vinay, On aggregating labels from multiple crowd workers to infer relevance of documents, in: Proc. of ECIR, 2012.

[2] N. Tax, S. Bockting, D. Hiemstra, A cross-benchmark comparison of 87 learning to rank methods, IP&M 51 (2015).

[3] K. Onal, Y. Zhang, I. Altingovde, M. Rahman, P. Karagoz, A. Braylan, B. Dang, H. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, T. Mcdonnell, A. Nguyen, D. Xu, B. Wallace, M. Rijke, M. Lease, Neural information retrieval: At the end of the early years, Information Retrieval 21 (2018).

[4] A. Purpura, G. Silvello, G. Susto, Learning to rank from relevance judgments distributions, Journal of the Association for Information Science and Technology (2022).

[5] C. Burges, From ranknet to lambdarank to lambdamart: An overview, in: MSR-TR-2010-82, 2010.

[6] S. Bruch, M. Zoghi, M. Bendersky, M. Najork, Revisiting approximate metric optimization in the age of deep neural networks, in: Proc. of SIGIR, 2019.

[7] S. Bruch, An alternative cross entropy loss for learning-to-rank, arXiv:1911.09798 (2019).

[8] R. Pasumarthi, H. Zhuang, X. Wang, M. Bendersky, M. Najork, Permutation equivariant document interaction network for neural learning to rank, in: Proc. of ICTIR, 2020.

[9] S. Bruch, S. Han, M. Bendersky, M. Najork, A stochastic treatment of learning to rank scoring functions, in: Proc. of WSDM, 2020.

[10] Q. Zhen, Y. Le, Z. Honglei, T. Yi, K. Rama, W. Xuanhui, B. Michael, N. Marc, Neural rankers are hitherto outperformed by gradient boosted decision trees, in: Proc. of ICLR, 2021.

[11] T. Qin, T. Liu, Introducing letor 4.0 datasets, arXiv:1306.2597 (2013).

[12] T. Qin, T. Liu, J. Xu, H. Li, Letor: A benchmark collection for research on learning to rank for information retrieval, Information Retrieval 13 (2010).

[13] W. Chen, T. Liu, Y. Lan, Z. Ma, H. Li, Ranking measures and loss functions in learning to rank, Proc. of NIPS (2009).

[14] S. Marchesin, A. Purpura, G. Silvello, Focal elements of neural information retrieval models. an outlook through a reproducibility study, Information Processing & Management 57 (2020) 102109.

[15] T. Qin, T. Liu, H. Li, A general approximation framework for direct optimization of information retrieval measures, IR Journal 4 (2010).

[16] A. Purpura, M. Maggipinto, G. Silvello, G. Susto, Probabilistic word embeddings in neural ir: A promising model that does not work as expected (for now), in: Proc. of ICTIR 2019, 2019, pp. 3–10.

[17] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, Cedr: Contextualized embeddings for document ranking, in: Proc. of SIGIR, 2019.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. of NIPS, 2017.

[19] H. Zhuang, X. Wang, M. Bendersky, M. Najork, Feature transformation for neural ranking models, in: Proc. of SIGIR, 2020.

[20] H. Zhuang, X. Wang, M. Bendersky, A. Grushetsky, Y. Wu, P. Mitrichev, E. Sterling, N. Bell, W. Ravina, H. Qian, Interpretable learning-to-rank with generalized additive models,

arXiv:2005.02553 (2020).

[21] M. Ibrahim, M. Carman, Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank, ACM TOIS 34 (2016).

[22] O. Chapelle, D. Metlzer, Y. Zhang, P. Grinspan, Expected reciprocal rank for graded relevance, in: Proc. of CIKM, 2009.