Improving Learning to Rank By Leveraging User Dynamics and Continuation Methods

Discussion Paper*

Nicola Ferro[1], Claudio Lucchese[2], Maria Maistro[3], and Raffaele Perego[4]

[1] University of Padua, ferro@dei.unipd.it
[2] Ca' Foscari University of Venice, Italy, claudio.lucchese@unive.it
[3] University of Copenhagen, Denmark, mm@di.ku.dk
[4] ISTI CNR Pisa, Italy, raffaele.perego@isti.cnr.it

Abstract. *Learning to Rank (LtR)* techniques leverage assessed samples of query-document relevance to learn ranking functions able to exploit the noisy signals hidden in the features used to represent queries and documents. In this paper, we explore how to enhance the state-of-the-art LambdaMart algorithm by integrating in the training process an explicit knowledge of the underlying user-interaction model and the possibility of targeting different objective functions, that can effectively drive the algorithm towards promising areas of the search space. We enrich the learning algorithm in two ways: (i) by considering complex query-based user dynamics instead than simply discounting the gain by the rank position; (ii) by designing a learning path across different loss functions that can capture different signals in the training data. Our extensive experiments, conducted on publicly available datasets, show that the proposed solution permits to improve various ranking quality measures by statistically significant margins.

Keywords: Learning to Rank · User Dynamics · Continuation Methods

1 Introduction

In [8], we explored whether the explicit knowledge of the underlying user interaction model [6] and the possibility of targeting different objective functions, by means of *Continuation Methods (CM)* [1, 5, 7], make it possible to improve the ranking models learned by *Learning to Rank (LtR)* algorithms. In particular, our investigation focused on improving the state-of-the-art LtR algorithm LambdaMart [3]. This extended abstract summarizes the main contributions discussed in [8] by relying on a extensive assessment conducted on publicly available datasets:

* Extended abstract of the paper originally published in [8]

– *modelling user dynamics into* LambdaMart: instead of proposing new features to account for user behavior and then training the LtR model on this extended set of features, we: (1) explicitly model the user dynamics in scanning a ranked result list with Markov chains trained on query log data; (2) define a novel measure which embeds this trained Markov chain (*Normalized Markov Cumulated Gain (nMCG)*), and modify the LambdaMart loss function to rely on this measure instead of the usual *Normalized Discounted Cumulated Gain (nDCG)* [9];
– *user interaction-based curriculum learning for* LambdaMart: designing a curriculum of objective functions of increasing complexity has shown to be a promising research direction. Therefore we explore if our nMCG measure provides further gain when building a curriculum based on CM techniques.

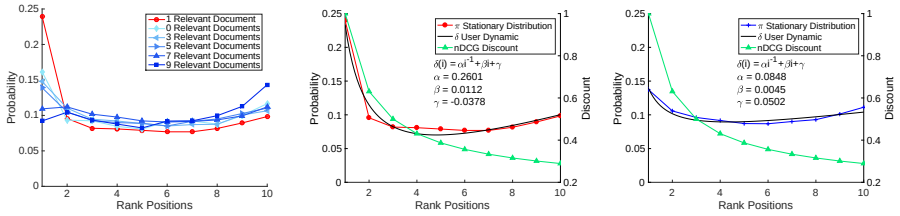
2 Methodology

An LtR algorithm exploits a *ground-truth* set of training examples in order to learn a document scoring function σ [10]. Such training set is composed of a large collection of queries $\mathcal{Q}$, where each query $q \in \mathcal{Q}$ is associated with a set of assessed documents $D = \{d_0, d_1, \ldots\}$. Each document d_i is labeled by a *relevance judgment* l_i according to its relevance to the query q. These labels induce a partial ordering over the assessed documents, thus defining an *ideal ranking* which the LtR algorithm aims at approximating. Each query-document pair (q, d_i) is represented by a vector of features x, able to describe the query (e.g., its length), the document (e.g., the in-link count) and their relationship (e.g., the number of occurrences of each query term in the document).

LambdaRank can be summarized as follows. Given a query q and two candidate documents d_i and d_j in the training set with relevance labels l_i and l_j respectively, s_i and s_j are the scores currently predicted for the documents. The lambda gradient of any given *Information Retrieval (IR)* quality function Q, as defined in [3], is:

$$\lambda_{ij} = \frac{\partial Q_{ij}}{\partial (s_i - s_j)} = \operatorname{sgn}(y_i - y_j) \left| \Delta Q_{ij} \cdot \frac{1}{1 + e^{s_i - s_j}} \right|$$

where, the sign is determined by the document labels only, the first factor ΔQ is the quality variation when swapping scores s_i and s_j, and the second factor is the derivative of the RankNet cost [2], which minimizes the number of disordered pairs. When $l_i \geq l_j$, the quality Q increases with the score of document d_i. The larger the quality variation ΔQ, the higher the document d_i should be scored. Note that the RankNet multiplier fades ΔQ if documents are scored correctly, i.e. $s_i \geq s_j$, and boosts ΔQ otherwise. The lambda gradient for a document d_i is computed by marginalizing over all possible pairs in the result list: $\lambda_i = \sum_j \lambda_{ij}$. LambdaRank uses nDCG as Q and so ΔQ is the variation in nDCG caused by the swap of two documents.



(a) Different Query Types (b) Navigational Queries (c) Informational Queries

Fig. 1. Stationary distributions for queries retrieving different numbers of relevant documents (a); and stationary distribution with its fitted curve and DCG discount for navigational (b), and informational queries (c).

2.1 User Dynamics for LtR

We summarize here our methodology [6] for including the user dynamics into the above discussed LambdaRank algorithm. We model the user dynamics with a Markovian process [11], where the user scans the ranked documents in the *Search Engine Result Page (SERP)* according to possibly complex paths, moving both forward and backward. Let us denote by $X_1, X_2, \ldots$ the sequence of random variables representing the rank positions in $\mathcal{R} = \{1, 2, \ldots, R\}$ visited by the user, where $X_n = j$ means that the n^{th} document visited by the user is at rank j. Moreover, we assume that the probability to move from the document at rank i to the document at rank j depends on the document at rank i only and is independent of all the previously visited documents. Finally, we denote by P the transition matrix whose entries represent the transition probabilities $P = (p_{ij} : i, j \in \mathcal{R})$, where $p_{ij} = \mathbb{P}[X_{n+1} = j | X_n = i]$. The sequence of random variables $(X_n)_{n>0}$ defines a discrete-time homogeneous Markov chain.

Figure 1(a) plots the stationary distributions obtained from the Yandex query log described in Section **??**. We focus on two distinct macroscopic behavior types, and, for the sake of simplicity, we call *navigational* the queries where users concentrated on just the first item, and we consider all the other queries as *informational* since users tend to visit more documents. On the basis of the above experimental observations, we claim that the user dynamics can be described as a mixture of the navigational and informational behavior. The navigational component is represented by the inverse of the rank position $\frac{1}{i}$, while the informational component is linear with respect to the rank position i. Therefore, we model the *user dynamics* as

$$\delta(i) = \alpha i^{-1} + \beta i + \gamma$$

where the parameters α, β and γ are calibrated in order to fit the estimated stationary distributions computed on the Yandex dataset.

Figures 1(b) and 1(c) show the stationary distributions together with the fitted curves for the navigational and informational cases, respectively. In Figure 1(b) the stationary distribution is the same reported in the red line of Fig-

ure 1(a), i.e. queries with just 1 relevant document, while to compute the stationary distribution reported in Figure 1(c) we aggregate all the user dynamics corresponding to the other queries, i.e. queries without relevant documents or with more than one relevant document.

We enhance the existing LambdaMart algorithm by replacing the above Q with a new quality measure which integrates the proposed user dynamics δ. This new measure is called *Normalized Markov Cumulated Gain (nMCG)* and it is defined as follows:

$$\text{nMCG@k} = \frac{\sum_{i \leq k} \left(2^{l_i} - 1\right) \cdot \delta^c(i)}{\sum_{h \leq k, \text{ sorted by } l_h} (2^{l_h} - 1) \cdot \delta^c(h)}$$

where l_i is the relevance label of the i-th ranked document and $\delta^c(i)$ is the user dynamics function at rank i relative to the query class c, either navigational or informational. Basically, nMCG can be seen as an extension of nDCG, where the discount function is defined by the user dynamic and depends on the query class. Moreover, since δ^c depends on the query class, i.e. depends on the query q, we are optimizing two different variants of the same quality measure nMCG across the training dataset. Finally, $\Delta\text{nMCG@k}_{ij}$ can be computed efficiently as follows:

$$\Delta\text{nMCG@k}_{ij} = \frac{-\left(2^{l_i} - 2^{l_j}\right)\left(\delta^c(i) - \delta^c(j)\right)}{\sum_{h \leq k, \text{ sorted by } l_h} (2^{l_h} - 1) \cdot \delta^c(h)}.$$

Hereinafter, we use nMCG-MART to refer to the described variant of LambdaMart aimed at maximizing nMCG.

2.2 Continuation Methods for LtR

Applying a *Continuation Methods (CM)* to a cost function C means to define a sequence of cost functions C_γ with $\gamma \in [0, 1]$, such that by increasing γ, the complexity of the function increases. Therefore, C_0 represents a highly smoothed version of the original cost function corresponding to C_1.

We implement CMs in the contest of forests of decision trees as a two steps learning process, where the initial trees are trained by minimizing a cost function C_0 and the remaining trees are trained by optimizing C_1. We did not explore in this work more complex multi-stage scenarios, which can trivially extend this work. We denote continuation methods as $C_0T_0\_C_1T_1$, meaning that the first T_0 trees of the ensemble minimize C_0, while the remaining T_1 trees minimize C_1, additively refining the prediction of the first T_0 trees.

In order to apply a CM to LambdaMart we need to smooth the LambdaMart loss function. Smoother variants of nDCG have been previously proposed, e.g. SoftNDCG [15]; however, their performance did not prove to be significantly better than the original LambdaMart loss function. Therefore, we look at two different smoother replacements of the nDCG measure.

The first option we consider is to use as C_0 the *Mean Square Error (MSE)* as a smooth variant of the target nDCG measure. Even if MSE is easily differentiable

and an MSE equal to 0 corresponds to the maximum nDCG value, the MSE cost cannot be considered a smooth approximation of nDCG. Nevertheless, Gradient Boosted Regression Trees that minimize MSE are known to perform well, even at optimizing nDCG. The rationale of using C_0 equal to MSE is that of using a smooth function that alone exhibits good performance as a good seed for the refinement that the optimization of nDCG or nMCG may provide.

The second option we consider is to use Recall@k as C_0. Recall is not a differentiable function either, as it suffers the same issues as nDCG. Still, it is an *easier* function to be optimized because it considers binary relevance instead of graded, and it discriminates between documents above or below the cut-off k without further discounting according to document ranks. The rationale is to train the first portion of the forest in order to place the most relevant documents above the cut-off, and then to complete the training with the target metric nDCG or nMCG to adjust their relative ranking.

Since Recall is not differentiable, we devised a LambdaMart-based approach similar to nMCG-MART. We define ΔnRecall@k$_{ij}$ as the normalized change in *Recall*@k when swapping the i-th and j-th documents in the current ranking. Given a relevance binarization thresholds ρ, ΔnRecall@k$_{ij}$ is defined as follows:

$$\Delta\text{nRecall@k}_{ij} = \frac{-\left(\mathbb{1}_{l_i \geq \rho} - \mathbb{1}_{l_j \geq \rho}\right)\left(\mathbb{1}_{i \geq k} - \mathbb{1}_{j \geq k}\right)}{\sum_h \mathbb{1}_{l_h \geq \rho}},$$

where $\mathbb{1}_p$ is the indicator function evaluating to 1 if predicate p is true and 0 otherwise.

Finally, in addition to C_1 being equal to nDCG, we also consider the case where nMCG-MART is used as C_1. Our goal is to evaluate the added value of exploiting the user dynamics in producing the final ranking, even when the first trees of the forest where built by optimizing a different metric.

In conclusion, we compare the effectiveness of LambdaMart and nMCG-MART and we asses the benefit they can receive by a pre-training on a different metric, which allows the two algorithms to initiate their search process from a different point in the solution space, possibly leading to a better final solution.

3 Experimental Evaluation

We remark that since there is no publicly available dataset providing, at the same time, user session data, document relevance and query-document pairs features, we have to use two different datasets in our analysis. Therefore, we calibrate the user model on the click log dataset provided by Yandex [13], while the training, validation and test for the LtR model are performed on MSLR-WEB30K and MSLR-WEB10K, provided by Microsoft [12].

We used LambdaMart as the reference LtR algorithm. Specifically, we used (and modified) the open source implementation of LambdaMart provided in the LightGBM gradient boosting framework and swiped the hyper parameters

Available at https://github.com/Microsoft/LightGBM.

with cross-validation so as to maximize the average performance over the validation folds.

As significance test, we computed Fisher's randomization test with 100 000 random permutations and $\alpha = 0.05$, which is the most appropriate statistical test to evaluate whether two approaches differ significantly [14].

As name convention, all the CM approaches are referred with the name of the first objective function C_0, the corresponding number of trees used during the first training phase T_0, followed by the name of the second objective function C_1 with the corresponding number of trees T_1. Therefore, `recallT300_ndcgT200` means that the model is trained with Recall@10 for the first 300 trees and then with nDCG@10 for the following 200 trees. Moreover, we exploit as reference models for CM approaches nMCG-MART and `mseT200_ndcgT300`, which is the best performing model proposed in [7].

3.1 Experimental Results

In this section we summarize the main findings without reporting any figure or table, the complete results can be found in the original paper [8].

First, we evaluate the performance of the proposed models against the baselines on MSLR-WEB10K and MSLR-WEB30K:

- With nDCG@10, CM approaches combining Recall and nMCG, namely `recallT300_nmcgT200` and `recallT400_nmcgT100`, achieve the best performance on each fold. They are significantly different from both the baseline and the corresponding CM approaches which combine Recall followed by nDCG. This supports our hypothesis that the combination of CM and the user dynamics integrated by nMCG can boost LambdaMart performance.
- With nDCG@100, CM approaches using nMCG are less effective and `mseT200_ndcgT300` is the best performing model, significantly improving over the baseline both on each fold and on the average across folds. Moreover, CM approaches using nMCG are not significantly different from those using nDCG. However, they are still significantly improving over the baseline.
- With Recall@10, the results are somehow in between nDCG@10 and nDCG@100. `mseT200_ndcgT300` is the best performing model when the average across each fold is considered. However, if we break down the results on each fold, `recallT400_nmcgT100` is the best performing model on three folds out of five for MSLR-WEB30K. Moreover, `recallT400_nmcgT100` is significantly better than the baseline and than `recallT400_ndcgT100`.
- With Recall@100, CM approaches using nDCG, i.e. `mseT200_ndcgT300`, `recallT300_ndcgT200` and `recallT400_ndcgT100` perform better than those using nMCG. When considering the score across all the folds, `mseT200_ndcgT300` is still the best performing model. Thus, when using Recall@100 as evaluation measure, we reach conclusions similar to those inferred with nDCG@100.
- With *Expected Reciprocal Rank (ERR)*@10, `recallT300_nmcgT200` is the best performing model and it is the only approach which is significantly

better than the baseline. Moreover, CM approaches using nMCG are also significantly better than those using nDCG, with respect to both each separate fold and the average across folds.

To summarize, CM approaches exploiting the user dynamics perform better at lower rank positions, while CM approaches using nDCG, as `mseT200_ndcgT300`, are somehow better in retrieving a higher number of relevant documents in the first 100 rank positions.

Second, we repeated the previous analysis by considering the query type, i.e. we compared the proposed models on navigational and informational queries separately. On a high level the experimental results show that:

- With navigational queries, there are just a few models which are significantly better than the baselines, independently of the measure or the fold. Furthermore, none of the model performs markedly better than the others, there is instead a lot of variability depending on the measure and the fold under consideration. We argue that with navigational queries the number of relevant documents is limited and systems have little room for manoeuvre: they need to place the most relevant document at the top of the ranked list and, when doing this properly, they all perform equally well in achieving this task.
- The experimental comparison of models on informational queries provides more insights on the performance of CM approaches combined with the user dynamics. The exploitation of the user dynamics together with CM boosts the ranking of queries for which several relevant documents are available. Indeed, it helps in identifying those documents that are more relevant than others and in placing them at the beginning of the ranked list.

Finally, we analyse the tree-wise performance of the proposed models: we compare the performance of different models with a varying number of trees. All CM approaches leveraging the user dynamics results to be always better than the baselines as the number of trees increases. Moreover, each boosting of the performance corresponds exactly to the switching point T_0, showing that replacing the objective function from Recall to nMCG is beneficial for the learning algorithm at any switching point.

4 Conclusion and Future Work

This paper investigated whether integrating in LtR the knowledge of the user-interaction model and the possibility of targeting different objective functions is profitable and allows us to train more effective ranking functions. The results of the experiments, measured with different metrics and cut-offs, gave us a more clear understanding of the problem addressed and confirmed our initial intuitions. We showed, by also breaking down the analysis to different classes of queries, that the proposed continuation methods exploiting our user-aware nMCG measure consistently outperform the baselines by statistically significant margins.

As future work we will study different methods for applying click models to LtR datasets. A possibility suggested in [4] could be to exploit editorial relevance labels as a link between different datasets and use a trained click model to generate new click-based features for datasets not providing them.

Acknowledgements: This paper is partially supported by the BIGDATAGRAPES (EU H2020 RIA, grant agreement N. 780751) and the OK-INSAID (MIUR-PON 2018, grant agreement N.ARS01_009 17) projects. The work is also partially funded by the "DAta BenchmarK for Keyword-based Access and Retrieval" (DAKKAR) Starting Grants project sponsored by University of Padua and Fondazione Cassa di Risparmio di Padova e di Rovigo, and by AMAOS (Advanced Machine Learning for Automatic Omni-Channel Support), funded by Innovationsfonden, Denmark.

References

[1] Allgower, E.L., Georg, K.: Numerical Continuation Methods. An Introduction. Springer-Verlag, Heidelberg, Germany (1980)
[2] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to Rank using Gradient Descent. In: ICML 2005. pp. 89–96. ACM Press, New York, USA (2005)
[3] Burges, C.J.C.: From RankNet to LambdaRank to LambdaMART: An Overview. Tech. rep., Microsoft Research, MSR-TR-2010-82 (2010)
[4] Chuklin, A., Markov, I., de Rijke, M.: Click Models for Web Search. Morgan & Claypool Publishers, USA (2015)
[5] Coleman, T.F., Wu, Z.: Parallel Continuation-based Global Optimization for Molecular Conformation and Protein Folding. Journal of Global Optimization **8**(1), 49–65 (1996)
[6] Ferro, N., Lucchese, C., Maistro, M., Perego, R.: On Including the User Dynamic in Learning to Rank. In: SIGIR 2017. pp. 1041–1044. ACM Press, New York, USA (2017)
[7] Ferro, N., Lucchese, C., Maistro, M., Perego, R.: Continuation Methods and Curriculum Learning for Learning to Rank. In: CIKM 2018. pp. 1523–1526. ACM Press, New York, USA (2018)
[8] Ferro, N., Lucchese, C., Maistro, M., Perego, R.: Boosting Learning to Rank with User Dynamics and Continuation Methods. Information Retrieval Journal (2019)
[9] Järvelin, K., Kekäläinen, J.: Cumulated Gain-Based Evaluation of IR Techniques. TOIS **20**(4), 422–446 (2002)
[10] Liu, T.Y.: Learning to Rank for Information Retrieval. (FnTIR **3**(3), 225–331 (2009)
[11] Norris, J.R.: Markov chains. Cambridge University Press, UK (1998)
[12] Qin, T., Liu, T.Y.: Introducing LETOR 4.0 Datasets. arXiv.org, Information Retrieval (cs.IR) **arXiv:1306.2597** (2013)
[13] Serdyukov, P., Craswell, N., Dupret, G.: WSCD2012: Workshop on Web Search Click Data 2012. In: WSDM 2012. pp. 771–772. ACM Press, New York, USA (2012)
[14] Smucker, M.D., Allan, J., Carterette, B.A.: A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In: CIKM 2007. pp. 623–632. ACM Press, New York, USA (2007)
[15] Taylor, M., Guiver, J., Robertson, S., Minka, T.: SoftRank: Optimizing Non-Smooth Rank Metrics. In: WSDM 2008. pp. 77–86. ACM Press, New York, USA (2008)