Information Retrieval Journal
https://doi.org/10.1007/s10791-019-09366-9

LEARNING FROM USER INTERACTIONS

# Boosting learning to rank with user dynamics and continuation methods

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

Received: 6 March 2019 / Accepted: 13 October 2019


**Abstract**
*Learning to rank* (*LtR*) techniques leverage assessed samples of query-document relevance to learn effective 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 LtR 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 iterative process followed by the learning algorithm in two ways: (1) by considering complex query-based user dynamics instead than simply discounting the gain by the rank position; (2) 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

✉ Maria Maistro
mm@di.ku.dk; maistro@dei.unipd.it

Nicola Ferro
ferro@dei.unipd.it

Claudio Lucchese
claudio.lucchese@unive.it

Raffaele Perego
raffaele.perego@isti.cnr.it

[1] Department of Information Engineering, University of Padua, Via Gradenigo 6/b, 35131 Padua, PD, Italy

[2] Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Via Torino 155, 30172 Mestre, VE, Italy

[3] Department of Computer Science, University of Copenhagen, Universitetsparken 1, 2100 Copenhagen, Denmark

[4] Istituto di Scienza e Tecnologie dell'Informazione A. Faedo (ISTI), National Research Council (CNR), Area di Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, PI, Italy

Published online: 05 November 2019

## 1 Introduction

*Information retrieval* (*IR*) systems are nowadays challenged with increasingly complex search tasks where information about how users interact with the system plays a central role to adapt them to the needs and interests of users. A lot of research efforts focused on enhancing user engagement and retrieval effectiveness by exploiting information about user-system interactions available, for example, from the query logs of Web search engines (Silvestri 2009; Lucchese et al. 2013; Mehrotra et al. 2018). The ability to interpret and learn from the complex and noisy traces of user-system interactions is fundamental for IR advance. The number of clicks on a given query-result pair, the *click-through rate* (*CTR*), and the dwell time, are examples of actionable information to improve various aspects of IR systems (Chuklin et al. 2015). In the context of *learning to rank* (*LtR*) (Liu 2009), user actions recorded in query logs are used to extract several important features (Agichtein et al. 2006) with proper adjustments in order to remove the position bias (Joachims and Radlinski 2007; Joachims et al. 2017); the same information can be used to model and predict user interests (White et al. 2009).

In this paper, we build on our previous work which explored how to exploit user interaction for LtR (Ferro et al. 2017) and how to apply *curriculum learning* (*CL*) and *continuation methods* (*CM*) to LtR (Ferro et al. 2018). We investigate here whether exploiting in conjunction these two techniques—i.e., the explicit knowledge of the underlying user-interaction model and the possibility of targeting different objective functions—makes it possible to further improve the quality of the ranking model learned as measured by different metrics. In particular, our investigation focuses on improving LambdaMart, a state-of-the-art LtR algorithm (Burges 2010; Wu et al. 2010), and tries to answer the following research question: "*how can user interaction and continuation methods be jointly exploited to improve LtR effectiveness?*".

The user interaction model is integrated directly in the discount component of the loss function used by LambdaMart at training time. Specifically, we model the user dynamics in scanning a ranked result list with Markov chains trained on query log data and we modify the loss function to embed this trained Markov chain. Moreover, since different loss functions can capture different relevance signals hidden in the features modeling queries and documents, we instantiate the training process as a continuous learning path across different loss functions, starting from easier functions considering retrieval quality, e.g., *recall* or *mean square error* (*MSE*), up to the most complex one embedding our Markovian user model. Learning through a *curriculum* is an interesting idea, borrowed from cognitive science, according to which a complex learning process is designed as a multi-step *training path* (Bengio et al. 2009). Initially, the learning algorithm is trained over simple training examples and smooth loss functions, and then it is progressively fine-tuned so as to deal with examples and loss functions of increasing complexity. *continuation methods* (*CM*) (Coleman and Wu 1996; Allgower and Georg 1980) specifically refer to approaches that instead of optimizing difficult objective functions, i.e., objective functions with many local minima, transform the original function into a class of smoother or easier to minimize functions. The intuition behind this approach is that a smooth version of the target objective function can quickly and effectively drive the learning algorithm to a promising area of the search space that possibly includes the global optimum.

The contributions of the paper are the following:

- *modelling user dynamics into* LambdaMart: instead of proposing new features to account for this particular aspect of user behavior and then training the LtR model on this extended set of features, we model the user dynamics in scanning a ranked result list with Markov chains trained on query log data. Moreover, we define the *normalized Markov cumulated gain* (*nMCG*) measure which embeds this trained Markov chain, and we modify the LambdaMart objective function to rely on nMCG instead of the usual *normalized discounted cumulated gain* (*nDCG*) (Järvelin and Kekäläinen 2002);
- *user interaction-based curriculum learning for* LambdaMart: designing a curriculum of objective functions of increasing complexity has shown to be a promising research direction and, therefore, we extend it by exploring whether nMCG, i.e. a user interaction-based objective function, can provide further gain when building a curriculum based on CM;
- *extensive experimentation*: we rely on the publicly available MSLR-WEB10K and MSLR-WEB30K datasets (Qin and Liu 2013) to conduct an extensive validation of the proposed approaches and derive key insights about them.

The paper is organized as follows: Sect. 2 introduces the related works; Sect. 3 presents our methodology; Sect. 4 describes the experimental setup and discusses the experimental findings; finally, Sect. 5 draws some conclusions and outlooks for future work.

## 2 Related work

### 2.1 User dynamics for LtR

LtR datasets already contain some user-related features. For example, the MSLR-WEB30K and MSLR-WEB10K datasets provided by Microsoft (Qin and Liu 2013) contain the following user-related features: `query-url click count`, i.e. the click count of a query-url pair at a search engine in a period; `url click count`, i.e. the click count of a URL aggregated from user browsing data in a period; and, `url dwell time`, i.e. the average dwell time of a URL aggregated from user browsing data in a period. As an empirical evidence of the importance of the user interaction features, we trained a LambdaMart (Burges 2010; Wu et al. 2010) model on the MSLR-WEB10K LtR dataset with and without these user-interaction features: the nDCG measured on the test set without such features drops from 0.4636 to 0.4410.

Jiang et al. (2013) provide a good overview of how search logs and user click data can be exploited to improve search in many respects, among which rankings of documents. Joachims et al. (2005) explored the use of clickthrough data as user preferences among documents and implicit feedback, while Joachims (2002) trained a *support vector machine* (*SVM*) ranker using such data. Dou et al. (2008) followed up by comparing the performance of RankNet (Burges et al. 2005) trained on aggregated clickthrough data with respect to using human relevance judgments.

Click models (Craswell et al. 2008; Chuklin et al. 2015) are another active area of research aimed at training models able to predict the probability of clicking on a document for a given query. Click models have not been directly exploited within LtR algorithms, but Chapelle and Zhang (2009) proposed to use them to predict relevance labels. Then, Chapelle and Zhang have shown that, when these predicted relevance labels are used to train the boosting algorithm QBRank (Zheng et al. 2007), they give just a 4% performance

loss, while when used together with editorial judgements they produce a 2% performance gain. Recent works, e.g. by Zoghi et al. (2017), explore how to best exploit click models in online LtR.

Online LtR exploits click logs and users interactions data to infer preferences between rankers in order to make online LtR faster (Hofmann et al. 2013; Schuth et al. 2016; Wang et al. 2018). As another example, Lerot (Schuth et al. 2013) proposes an online LtR algorithm which uses clicks as feedback for interleaving methods.

To the best of our knowledge, the integration of the user dynamics directly within the objective function of an offline LtR algorithm is novel and our previous work (Ferro et al. 2017) was the first to explore it.

### 2.2 Curriculum learning for LtR

*Continuation methods* (*CM*) have shown to be effective in the optimization of complex objective functions (Coleman and Wu 1996; Allgower and Georg 1980). When the target objective function has many local minima, its direct optimization may lead to a "bad" sub-optimal result. In Continuation Methods, a multi-step optimization process is thus followed. At each step a different smooth function is optimized, i.e. a function easy to minimize that approximates the desired target function. The complexity and difficulty of the optimization function chosen is increased until the original target function is finally used. The intuition behind this approach is that the smooth versions of the original target function can provide a global representation of the search space, which highlights the regions where the best local optima and the global optima are located.

*Curriculum learning* (*CL*) can be seen as a particular instantiation of continuation methods (Bengio et al. 2009). The basic idea is to organize the training examples in such a way that the easiest training examples are presented first and the complexity of the following ones is gradually increased. This strategy guides the training and allows the learner to exploit previously seen concepts to ease the acquisition of subsequent more difficult concepts. Thus, Curriculum Learning can be seen as a process exploiting a sequence of training criteria. Each training criterion corresponds to a different set of examples that can be differently weighted based on their complexity. At the subsequent steps, slightly more difficult examples are assigned with new weights. This is different from boosting approaches commonly adopted also in LtR, as in boosting the instance weights are determined during the training according to the mis-classification risk, while in Curriculum Learning weights are predetermined according to a training schedule.

Both approaches have been used successfully in several fields. Continuation Methods are applied when the function to optimize is not convex, as for example non linear optimization problems (Chen and Xiu 1999) in computational chemistry (Coleman and Wu 1996; Moré and Wu 1997), computational physics (Rabani et al. 2002) and automatic control (Nagamune 2003). In *machine learning* (*ML*), continuation methods are used with semi-supervised SVM, showing that this approach leads to lower test errors (Chapelle et al. 2006). CL is often applied with *neural networks* (*NN*)*s* (*NN*)*s* and deep NNs, as for example in Collobert et al. (2011), Hu et al. (2014), where CL is used for *natural language processing* (*NLP*), or in Chen and Gupta (2015), where CL is exploited for representations of images. In Bengio et al. (2009), curriculum learning is exploited to train a language model (not used for ranking) with a deep neural network. In Qu et al. (2018) a curriculum learning approach is applied to sort the training data to ease the learning of nodes representations in a heterogeneous star network.

To the best of our knowledge, our previous work (Ferro et al. 2018) has been the first to explore how CM and CL methods can be exploited in combination with LtR algorithms. We found that CM is more effective than CL on this task and this is why in this paper we rely only on CM.

## 3 Methodology

A LtR algorithm exploits a *ground-truth* set of training examples in order to learn a document scoring function $\sigma$ (Liu 2009). 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 $D = \{d_0, d_1, \dots\}$. 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).

According to the above notation, a LtR algorithm produces a function $\sigma(x)$ predicting a relevance score for the input feature vector $x$. Such function $\sigma(x)$ is finally used online to compute a score for documents matching a query and ranking them accordingly.

Since IR measures are not differentiable, their optimization is very challenging. To address this issue, the state-of-the-art solution is the LambdaRank gradient approximation (Burges et al. 2006), which is based on the idea of measuring the cost variation after swapping any two documents in a given result list. As discussed in Donmez et al. (2009), this approach can be applied to several IR measures and it is capable of accurately discovering local optima.

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 IR quality function $Q$, as defined in Burges (2010), 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 $\Delta Q$ is the quality variation when swapping scores $s_i$ and $s_j$, and the second factor is the derivative of the RankNet cost (Burges et al. 2005), 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 $\Delta Q$, the higher the document $d_i$ should be scored. Note that the RankNet multiplier fades $\Delta Q$ if documents are scored correctly, i.e. $s_i \geq s_j$, and boosts $\Delta Q$ 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 $\Delta Q$ is the variation in nDCG caused by the swap of two documents.

### 3.1 User dynamics for LtR

We summarize here our methodology (Ferro et al. 2017) for including the user dynamics into the above discussed LambdaRank algorithm.

Yilmaz et al. (2010) have shown that effectiveness is often measured as the inner product of a relevance vector $\mathcal{J}$ and a discounting vector $\mathcal{D}$. The elements $\mathcal{J}_i$ account for the benefit of ranking an high-quality document at the $i$-th position of the *search engine result page* (*SERP*), while $\mathcal{D}$ denotes such contribution for low-ranked documents. For instance, according to Discounted Cumulated Gain (DCG) (Järvelin and Kekäläinen 2002) the $i$-th element of $\mathcal{J}$ is defined as $\mathcal{J}_i = 2^{l_i} - 1$, where $l_i$ is the relevance label of the $i$-th ranked document, and $\mathcal{D}_i = \log(i+1)$. The underlying assumption is that low-ranked documents receive less attention by the user and therefore they contribute less to the user-perceived quality of the SERP. Defining a proper quality metric is crucial both for evaluating retrieval systems and for learning effective ranking models as such metrics are used to drive the training process.

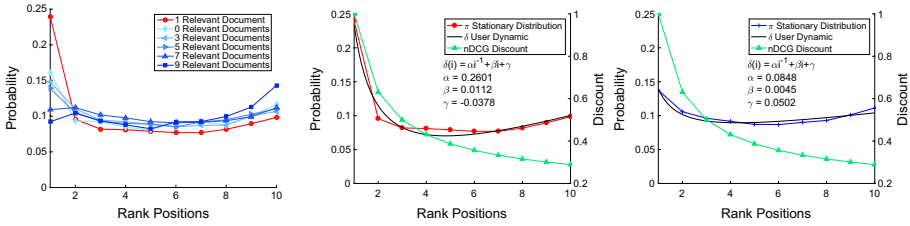
Most metrics assume that the user analyzes a SERP from top to bottom, and therefore define a decreasing discount vector, however some user studies suggest that the probability of observing a result depends on the quality of the documents ranked higher: if the user finds a relevant document at position $i$ it is less likely that he will inspects the document at position $i+1$ (Zhang et al. 2010). However, the user behavior is more complex as he/she can move forward and backward, can jump from one document to any other and visit already visited documents, as suggested by Ferrante et al. (2014), Sakai and Dou (2013).

Our work stems from the simple observation that the user behavior in visiting a SERP differs depending on the query type and the number and position of the relevant results. For example, it is likely that on a SERP with a single highly relevant result in the first position the user assumes a *navigational* behavior, while a SERP with several relevant results may likely correspond to an *informational* query, where a more complex SERP visiting behavior can be observed (Broder 2002). Since at training time a list-wise LtR algorithm such as LambdaMart is aware of the number and distribution of relevance labels associated with the training samples for each query, we suppose that it can profit from the knowledge of the user dynamics associated with the specific kind of query.

We model the user dynamics with a Markovian process (Norris 1998), where the user scans the ranked documents in the 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.

If we consider the transition matrix and compute $P^n$, we obtain the probability to see the Markov chain in a given state after $n$ steps. By assuming that $P$ is irreducible, which means that the probability of passing in a finite number of steps from a given document in the ranked list to any other document is positive, it can be proven that $P$ admits a unique stationary distribution $\pi P = \pi$ which, under the hypothesis of $P$ being aperiodic (Norris 1998), is the limit of the $n$-step transition probabilities, $p_{ij}^{(n)} \to \pi_j$ as $n \to \infty$ for all $i, j$.

When extending this analysis to a long-term query log, we can consider the behavior recorded for each user as a different observation of the same stochastic process, and the resulting stationary distribution can be considered as an aggregated representation of user dynamics. Notice that the convergence is typically very fast, since already after $n \approx 10$ iterations of the power method we obtain an accurate estimation of $\pi$.




**(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)

Since we observe that the behavior of users change depending on the number of relevant documents in the SERP, we can classify queries on the basis of the number of relevant documents returned and estimate different transition matrices $\hat{P}$ for different classes of queries. Specifically, we first aggregate the dynamics of different users on the basis of the typology of query, then we adopt the maximum likelihood estimator approach (Teodorescu 2009) on the aggregated data:

1. for each $i \in \mathcal{R}$ let $v_i$ be the number of times that the users visited the document at rank $i$ given the query;
2. if $v_i = 0$, then $\hat{p}_{ij} = 0$ for all $j \neq i$ and $\hat{p}_{ii} = 1$;
3. if $v_i > 0$, let $v_{ij}$ be the number of transitions from the document at rank $i$ to the document at rank $j$, then $\hat{p}_{ij} = \frac{v_{ij}}{v_i}$.

Figure 1a plots the stationary distributions obtained from the Yandex query log described in Sect. 4.1. We split queries in the log on the basis of the number of relevant documents retrieved and we computed the stationary distribution by aggregating the sessions of all the users sharing the same typology of query. When considering queries with just one relevant retrieved document, i.e. the red line with circle markers in Fig. 1a, the user dynamics exhibit a spike with respect to the first rank position, while for queries without any relevant documents or with more than one relevant document, i.e. the blue lines, the probability tends to be distributed more uniformly, meaning that the user is exploring the whole SERP.

We focus on these 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. To embed user dynamics in the LambdaMart cost function, as detailed in the next section, we abstract these two observed behavioral types (navigational and informational) by fitting a curve to the corresponding stationary distributions.

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 $i$, $\frac{1}{i}$, 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 $\alpha$, $\beta$ and $\gamma$ are calibrated in order to fit the estimated stationary distributions computed on the Yandex dataset.

Figure 1b, c show the stationary distributions together with the fitted curves for the navigational and informational cases, respectively. In Fig. 1b the stationary distribution is the same reported in the red line of Fig. 1a, while to compute the stationary distribution reported in Fig. 1c we aggregate all the user dynamics corresponding to the other queries, i.e. queries without relevant documents or with more than one relevant document.

Since it is clearly visible that users behave differently depending on the query at hand, if we consider the notation introduced at the beginning of Sect. 3, we might require different scoring functions $\sigma_i$ depending on the query type, and a classifier $c$ able to discriminate between queries types. However, at query time, the classifier $c$ can exploit only query-based features $x_q$ as, in general, documents have not yet been retrieved, and neither their relevance nor their features are available. There is a large literature of query classification for different tasks, and query topic categorization is well addressed within Web companies to increase effectiveness, efficiency, and revenue potential in general-purpose Web search engines (Beitzel et al. 2005). Therefore, building such a classifier $c(x_q)$ is feasible, but beyond the purpose of this paper.

However, the desired scoring model should be a function of both the query classification $c(x_q)$ and the class-dependent document scorer $\sigma_{c(x_q)}$. Since both functions depend on $x$, they can be replaced trivially with a single scoring function $\sigma(x)$. This allows us to avoid the design of a query classifier, and let the LtR algorithm to learn the query classification and document scoring at the same time.

In the following we will illustrate the approach adopted in order to include the user dynamics in LambdaMart. Roughly speaking, the idea is to consider the user dynamics as a discount function and to optimize a new evaluation metric based on the stationary distribution and query class. A similar approach is presented in Yilmaz et al. (2010), where the expected browsing utility is estimated on log data and used as discount function.

The *user dynamics* defined above can actually be considered as a discounting vector to be exploited in any given quality metric. Differently from other approaches, the *user dynamics* is defined on the basis of two different query classes which exhibit a different user behavior. Figure 1b, c show how different is the derived user dynamics w.r.t. the DCG discounting component. Below we discuss how $\delta$ can be exploited in a state-of-the-art LtR algorithm.

We enhance the existing LambdaMart algorithm by replacing the above $Q$ with a new quality measure which integrates the proposed user dynamics $\delta$. This new measure is called *normalized Markov cumulated gain* (*nMCG*) and it is defined as follows:

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

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 $\delta^c$ 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, $\Delta\mathrm{nMCG@k}_{ij}$ can be computed efficiently as follows:

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

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

Note that the query class is known at training time, and therefore the algorithm can optimize the proper user dynamics $\delta^c$. Nor the document relevance, neither the query class information are available at test time, therefore the algorithm should, at the same time, classify queries and rank documents according to the different class-based dynamics $\delta^c$.

### 3.2 Continuation methods for LtR

Applying a Continuation Method to a cost function $C$ means to define a sequence of cost functions $C_\gamma$ with $\gamma \in [0, 1]$, such that by increasing $\gamma$, 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 Continuation Methods in the contest of forests of decision trees as a two step 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 (Taylor et al. 2008); 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 $\Delta\text{nRecall@k}_{ij}$ 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 $\rho$, $\Delta\text{nRecall@k}_{ij}$ 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, by evaluating different combinations of $C_0$ and $C_1$, we are comparing the effectiveness of LambdaMart and nMCG-MART, and we are assessing the benefit they can receive by a pre-training on a different metric, which actually allows the two algorithms to initiate their search process from a different point in the solution space, possibly leading to a better final solution.

## 4 Experimental evaluation

### 4.1 Experimental setup

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. A first dataset providing user session data is used for the user dynamics derivation, while standard LtR datasets are used for learning the ranking models.

We calibrate the proposed user model on the basis of the click log dataset provided by Yandex (Serdyukov et al. 2012) (http://imat-relpred.yandex.ru/en/). The dataset is composed of 340,796,067 records with 30,717,251 unique queries, retrieving 10 URLs each. We used the training set, which consists of 5,191 assessed queries with binary judgments, corresponding to 30,741,907 records. Notice that 9% of the sessions corresponds to navigational queries, while the remaining 91% corresponds to informational ones.

The accuracy of the proposed algorithms is instead evaluated on two public LtR datasets, MSLR-WEB30K and MSLR-WEB10K, provided by Microsoft (Qin and Liu 2013). Dataset MSLR-WEB30K encompasses 31,531 queries from the Microsoft Bing search engine for a total of 3,771,125 query-document pairs represented by 136 features. The dataset is provided as a 5-fold split. The MSLR-WEB10K dataset contains instead 10,000 queries samples at random from the previous dataset.

The query-document pairs in both the datasets are assessed with integer relevance labels in the range [0, 4].

In order to classify queries as navigational or informational we adopt the following criterion: a query is considered as navigational if it contains only one result with relevance label $\geq 3$. Approximatively 15% of the queries in the Microsoft datasets are classified according to this heuristic as navigational queries, which is a fraction quite similar to the one measured on the Yandex dataset. The exact number of navigational and informational queries for each fold is reported in Table 1 for MSLR-WEB10K and in Table 2 for MSLR-WEB30K.

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[1] and swiped the hyper parameters with cross-validation so as to maximize the average performance over the validation folds. Learning rate $\nu$ was varied in the

[1] Available at https://github.com/Microsoft/LightGBM.

**Table 1** Number of navigational and informational queries included in MSLR-WEB10K

| Dataset | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Total |
|---|---|---|---|---|---|---|
| test_nav | 293 (15%) | 323 (16%) | 264 (13%) | 292 (15%) | 284 (14%) | 1456 (15%) |
| test_inf | 1707 (85%) | 1677 (84%) | 1736 (87%) | 1708 (85%) | 1716 (86%) | 8544 (85%) |

**Table 2** Number of navigational and informational queries included in MSLR-WEB30K

| Dataset | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Total |
|---|---|---|---|---|---|---|
| test_nav | 871 (14%) | 927 (15%) | 895 (14%) | 957 (15%) | 864 (14%) | 4514 (14%) |
| test_inf | 5435 (86%) | 5380 (85%) | 5411 (86%) | 5349 (85%) | 5442 (86%) | 27017 (86%) |

set $\{0.01, 0.05, 0.1, 0.5\}$, while the maximum number of leaves $L$ in the set $\{8, 16, 32, 64\}$, with best results observed with the setting $\nu = 0.05$ and $L = 64$. Notice that in the following sections, whenever we refer to the baseline we mean a LambdaMart model trained using the above settings and with nDCG@10 as effectiveness measure. We keep the same setting for all the other proposed models, which are trained with the same values of learning rate and number of leaves, and with a total of $T = 500$ trees. We implemented the proposed nMCG and CM algorithms as LightGBM plug-ins providing the required derivative and hessian matrices for the gradient boosting optimization. The source code of our implementation will be made available upon publication of the paper.[2]

For continuation methods approaches, we report the results achieved when training a first set of $T_0$ trees with the first objective function $C_0$ and the remaining $T_1 = T - T_0$ by exploiting a different objective function $C_1$. We explored different switching points for the cost functions at $T_0$ equals to 100, 200, 300, 400 trees, while the total number of trees $T$ is always equal to 500. As function $C_0$ we use Recall@10 or MSE, while as second objective function we use either nDCG@10 or nMCG@10.

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 report as reference models for CM approaches nMCG-MART and the best performing model proposed in Ferro et al. (2018), i.e. `mseT200_ndcgT300`, which is the continuation method approach where the first 200 trees where trained with MSE as objective function, and the following 300 trees where trained with nDCG@10.

In the following sections we describe the experimental evaluation conducted to investigate the performance of each model. More in detail, we address the following research questions:

**RQ1—what are the best performing models?** Section 4.2 presents the evaluation of each model with respect to different measures;

[2] Available at https://bitbucket.org/mmaistro/irj_2019_boosting_ltr/src/master/.

**RQ2—what is the impact of navigational and informational queries?** Section 4.3 addresses the evaluation of each model focusing on navigational and informational queries;
**RQ3—what is the impact of the number of used trees?** Section 4.4 studies the tree-wise performance of each model and compares the ranking models on the basis of the number of trees and the CM switching points.

### 4.2 RQ1: Models evaluation

In this section we evaluate the proposed models with respect to different measures. Besides the baseline, nMCG-MART and `recallT300_ndcgT200`, we report the evaluation performance of `recallT300_ndcgT200` and `recallT400_ndcgT100`, and `recallT300_nmcgT200` and `recallT300_nmcgT200`. We tested all the continuation method approaches with combinations of recall followed by nDCG or nMCG and switching point $T_0 \in \{100, 200, 300, 400\}$, however, due to space constraints, we report only those combinations that lead to better performance.

Each model is evaluated separately on each fold and then the evaluation scores are averaged across the folds. As evaluation measures we use nDCG[3] with two different cut-off thresholds, 10 and 100, Recall with cut-off at 10 and 100, and *expected reciprocal rank* (*ERR*) (Chapelle et al. 2009) with cut-off at 10. Tables 3 and 4 present the evaluation scores on MSLR-WEB10K and MSLR-WEB30K respectively.

In each table the best score achieved with respect to each fold and measure is highlighted in bold. We remark that nDCG evaluation of different rankers are commonly very close, and that, in web search rankers, changes of over half a percentage point of nDCG are considered major revisions (Chapelle et al. 2012). Moreover we computed Fisher's randomization test with 100,000 random permutations, which accordingly to Smucker et al. (2007) is the most appropriate statistical test to evaluate whether two approaches differ significantly. Thus, when an approach is marked with $^\bullet$, it means that it is significantly different from the baseline with the two-sided $p$ value lower than $\alpha = 0.05$. Similarly we compare each continuation method combining recall followed by nDCG, with the corresponding continuation method combining recall and nMCG, i.e. we compare `recallT300_ndcgT200` against `recallT300_nmcgT200` and `recallT400_ndcgT100` against `recallT400_nmcgT100`. Significantly different approaches with the two-sided $p$ value lower than $\alpha$ are marked with $^+$.

Table 3 shows that the CM approaches combining Recall and nMCG, namely `recallT300_nmcgT200` and `recallT400_nmcgT100`, achieve the best performance on each fold in terms of nDCG@10. 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.

The third best performing model is `recallT300_ndcgT200`, which is significantly different from the baseline, on three out of five folds and also when the average

[3] We exploited LightGBM implementation of nDCG, which assigns nDCG score equal to 1 to queries with no relevant documents.

**Table 3** Models evaluation on MSLR-WEB10K test set with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.5037 | 0.4984 | 0.4933 | 0.5047 | 0.5083 | 0.5017 |
| | nMCG-MART | 0.5049 | 0.4996 | 0.4951 | 0.5024 | 0.5085 | 0.5021 |
| | mseT200_ndcgT300 | 0.508$^{\bullet}$ | 0.5059$^{\bullet}$ | 0.4972$^{\bullet}$ | 0.508 | 0.512 | 0.5062$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.5032 | 0.501 | 0.4945 | 0.5051 | 0.5093 | 0.5026 |
| | recallT400_ndcgT100 | 0.502 | 0.5015 | 0.4952 | 0.5036 | 0.5105 | 0.5025 |
| | recallT300_nmcgT200 | 0.5072$^{\bullet,+}$ | **0.5062**$^{\bullet,+}$ | **0.4986**$^{\bullet,+}$ | 0.5091$^{\bullet,+}$ | 0.5136$^{\bullet,+}$ | 0.507$^{\bullet,+}$ |
| | recallT400_nmcgT100 | **0.5083**$^{\bullet,+}$ | 0.5045$^{\bullet,+}$ | 0.4983$^{\bullet,+}$ | **0.5094**$^{\bullet}$ | **0.5146**$^{\bullet}$ | **0.507**$^{\bullet,+}$ |
| nDCG@100 | LambdaMart | 0.6834 | 0.6819 | 0.6761 | 0.6818 | 0.6872 | 0.6821 |
| | nMCG-MART | 0.6841 | 0.6814 | 0.6775 | 0.6811 | 0.6872 | 0.6823 |
| | mseT200_ndcgT300 | **0.6867**$^{\bullet}$ | **0.6857**$^{\bullet}$ | **0.6813**$^{\bullet}$ | 0.6844$^{\bullet}$ | **0.6908**$^{\bullet}$ | **0.6858**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6847 | 0.6832 | 0.6791$^{\bullet}$ | 0.6824 | 0.6884 | 0.6836$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.6838 | 0.684 | 0.679$^{\bullet}$ | 0.6816 | 0.6891 | 0.6835$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.6852 | 0.6855$^{\bullet,+}$ | 0.6803$^{\bullet}$ | **0.6848**$^{\bullet,+}$ | 0.6899$^{\bullet}$ | 0.6851$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.6857$^{\bullet,+}$ | 0.6844$^{\bullet}$ | 0.6797$^{\bullet}$ | 0.6847$^{\bullet,+}$ | 0.6904$^{\bullet}$ | 0.685$^{\bullet,+}$ |
| Recall@10 | LambdaMart | 0.3573 | 0.3526 | 0.3656 | 0.3537 | 0.3584 | 0.3575 |
| | nMCG-MART | 0.3587 | 0.3556 | 0.3648 | 0.3525 | 0.3595 | 0.3582 |
| | mseT200_ndcgT300 | 0.362$^{\bullet}$ | **0.3579**$^{\bullet}$ | 0.3664 | **0.3593**$^{\bullet}$ | 0.3629$^{\bullet}$ | **0.3617**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.3579 | 0.3574$^{\bullet}$ | 0.3646 | 0.3549 | 0.3623 | 0.3594 |
| | recallT400_ndcgT100 | 0.3601 | 0.3565 | 0.3657 | 0.3534 | **0.365**$^{\bullet}$ | 0.3601$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.3609 | 0.3568$^{\bullet}$ | **0.3679** | 0.3567 | 0.3626 | 0.361$^{\bullet}$ |
| | recallT400_nmcgT100 | **0.3624**$^{\bullet}$ | 0.3573$^{\bullet}$ | 0.3666 | 0.3557 | 0.3647$^{\bullet}$ | 0.3614$^{\bullet}$ |
| Recall@100 | LambdaMart | 0.9015 | 0.9016 | 0.9038 | 0.902 | 0.9064 | 0.9031 |
| | nMCG-MART | 0.902 | 0.9014 | 0.9033 | 0.9025 | 0.906 | 0.9031 |
| | mseT200_ndcgT300 | **0.9051**$^{\bullet}$ | 0.9029 | 0.9071$^{\bullet}$ | 0.9052$^{\bullet}$ | 0.9095$^{\bullet}$ | 0.906$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.9043$^{\bullet}$ | 0.9034 | **0.9078**$^{\bullet}$ | **0.9054**$^{\bullet}$ | 0.9094$^{\bullet}$ | **0.9061**$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.905$^{\bullet}$ | **0.9034** | 0.907$^{\bullet}$ | 0.9046$^{\bullet}$ | **0.9098**$^{\bullet}$ | 0.906$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.9019 | 0.9018 | 0.9063$^{\bullet}$ | 0.9034 | 0.9079 | 0.9043$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.9021 | 0.9018 | 0.9068$^{\bullet}$ | 0.9034 | 0.9076 | 0.9043$^{\bullet}$ |
| ERR@10 | LambdaMart | 0.5389 | 0.5379 | 0.533 | 0.5467 | 0.5509 | 0.5415 |
| | nMCG-MART | **0.5406** | 0.536 | 0.5364 | 0.5444 | 0.5509 | 0.5417 |
| | mseT200_ndcgT300 | 0.5373 | 0.5433 | **0.5381** | 0.5436 | 0.5525 | 0.543 |
| | recallT300_ndcgT200 | 0.5315 | 0.5342 | 0.5293 | 0.5389 | 0.5448 | 0.5357 |
| | recallT400_ndcgT100 | 0.5298 | 0.5349 | 0.5279 | 0.5354 | 0.5438 | 0.5344 |
| | recallT300_nmcgT200 | 0.5382$^{+}$ | **0.5452**$^{\bullet,+}$ | 0.5378$^{+}$ | 0.5466$^{+}$ | **0.5541**$^{+}$ | **0.5444**$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.5383$^{+}$ | 0.5407$^{+}$ | 0.5345$^{+}$ | **0.5481**$^{+}$ | 0.554$^{+}$ | 0.5431$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is highligthed in bold

is considered. On the other hand, nMCG-MART[4] and CM approaches recallT300_ndcgT200 and recallT400_ndcgT100 perform better than the baseline, but they are not significantly different.

When we consider nDCG with a greater cut-off, i.e. nDCG@100, mseT300_ndcgT200 becomes the best performing model on four out of five folds, being significantly different from the baseline on four folds and also on the average results. CM approaches involving the user dynamics represent the second and third best performing models, again significantly improving over the baseline. Even if overall they are significantly different from the corresponding CM approaches exploiting nDCG, this does not always hold when each fold is considered separately. Indeed, only on two out of five folds recallT300_ndcgT200 and recallT400_nmcgT100 are significantly different from recallT300_ndcgT200 and recallT400_ndcgT100. However, the CM approaches which exploit nMCG are significantly better than the baseline, while this is not true for those CM approaches exploiting nDCG.

[4] Note that nDCG@10 scores are slightly different from those reported in Ferro et al. (2017) for nMCG-MART due to the different implementation of nDCG@10 metric used in this paper. See Footnote 3.

**Table 4** Models evaluation on MSLR-WEB30K test set with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.5152 | 0.5175 | 0.5201 | 0.5153 | 0.5137 | 0.5164 |
| | nMCG-MART | 0.517$^{\bullet}$ | 0.5169 | 0.5202 | 0.5154 | 0.5135 | 0.5166 |
| | mseT200_ndcgT300 | 0.5194$^{\bullet}$ | 0.519 | 0.5221$^{\bullet}$ | 0.5182$^{\bullet}$ | 0.5147 | 0.5187$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.5185$^{\bullet}$ | 0.5189 | 0.5204 | 0.518$^{\bullet}$ | 0.5166$^{\bullet}$ | 0.5185$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.5185$^{\bullet}$ | 0.5178 | 0.5206 | 0.517 | 0.5159 | 0.518$^{\bullet}$ |
| | recallT300_nmcgT200 | **0.5201**$^{\bullet}$ | **0.5206**$^{\bullet,+}$ | 0.5228$^{\bullet,+}$ | **0.519**$^{\bullet}$ | **0.5174**$^{\bullet}$ | **0.52**$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.5199$^{\bullet}$ | 0.5198$^{\bullet,+}$ | **0.5233**$^{\bullet,+}$ | 0.5184$^{\bullet}$ | 0.517$^{\bullet}$ | 0.5197$^{\bullet,+}$ |
| nDCG@100 | LambdaMart | 0.6917 | 0.6925 | 0.6942 | 0.6912 | 0.6914 | 0.6922 |
| | nMCG-MART | 0.6931$^{\bullet}$ | 0.692 | 0.6945 | 0.6911 | 0.6916 | 0.6925 |
| | mseT200_ndcgT300 | **0.6959**$^{\bullet}$ | **0.6946**$^{\bullet}$ | **0.6965**$^{\bullet}$ | **0.6936**$^{\bullet}$ | 0.6941$^{\bullet}$ | **0.6949**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6953$^{\bullet}$ | 0.6937 | 0.6952 | 0.6933$^{\bullet}$ | **0.6947**$^{\bullet}$ | 0.6944$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.6952$^{\bullet}$ | 0.6932 | 0.6949 | 0.6928$^{\bullet}$ | 0.6943$^{\bullet}$ | 0.6941$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.695$^{\bullet}$ | 0.6941$^{\bullet}$ | 0.6957$^{\bullet}$ | 0.6929$^{\bullet}$ | 0.6941$^{\bullet}$ | 0.6944$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.6952$^{\bullet}$ | 0.6935 | 0.6959$^{\bullet,+}$ | 0.6924$^{\bullet}$ | 0.6941$^{\bullet}$ | 0.6942$^{\bullet}$ |
| Recall@10 | LambdaMart | 0.3579 | 0.366 | 0.3631 | 0.364 | 0.3648 | 0.3632 |
| | nMCG-MART | 0.3584 | 0.3665 | 0.3641 | 0.365 | 0.3647 | 0.3637 |
| | mseT200_ndcgT300 | 0.3624$^{\bullet}$ | 0.3682 | 0.3658$^{\bullet}$ | **0.3696**$^{\bullet}$ | **0.3666** | **0.3665**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.3599 | 0.3679 | 0.3639 | 0.3673$^{\bullet}$ | 0.3656 | 0.3649$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.3617$^{\bullet}$ | 0.3662 | 0.3644 | 0.3663$^{\bullet}$ | 0.3648 | 0.3647$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.3616$^{\bullet}$ | 0.3681 | 0.365 | 0.3663$^{\bullet}$ | 0.3656 | 0.3653$^{\bullet}$ |
| | recallT400_nmcgT100 | **0.3618**$^{\bullet}$ | **0.3691**$^{\bullet,+}$ | **0.366**$^{\bullet}$ | 0.3669$^{\bullet}$ | 0.3646 | 0.3657$^{\bullet,+}$ |
| Recall@100 | LambdaMart | 0.9045 | 0.9051 | 0.9068 | 0.9038 | 0.9072 | 0.9055 |
| | nMCG-MART | 0.9045 | 0.9049 | 0.9069 | 0.9043 | 0.9075 | 0.9056 |
| | mseT200_ndcgT300 | 0.9076$^{\bullet}$ | 0.9079$^{\bullet}$ | **0.9094**$^{\bullet}$ | **0.907**$^{\bullet}$ | 0.9101$^{\bullet}$ | **0.9084**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.9074$^{\bullet}$ | 0.9079$^{\bullet}$ | 0.9092$^{\bullet}$ | 0.9068$^{\bullet}$ | **0.9104**$^{\bullet}$ | 0.9083$^{\bullet}$ |
| | recallT400_ndcgT100 | **0.9077**$^{\bullet}$ | **0.908**$^{\bullet}$ | 0.909$^{\bullet}$ | 0.9066$^{\bullet}$ | 0.9103$^{\bullet}$ | 0.9083$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.9062$^{\bullet}$ | 0.9063$^{\bullet}$ | 0.9074 | 0.904 | 0.9092$^{\bullet}$ | 0.9066$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.9064$^{\bullet}$ | 0.9064$^{\bullet}$ | 0.9077 | 0.9046 | 0.9092$^{\bullet}$ | 0.9069$^{\bullet}$ |
| ERR@10 | LambdaMart | 0.5598 | 0.5581 | 0.5637 | 0.5554 | 0.5544 | 0.5583 |
| | nMCG-MART | 0.5611 | 0.5568 | **0.5649** | 0.5551 | 0.5543 | 0.5584 |
| | mseT200_ndcgT300 | 0.5616 | 0.5564 | 0.5619 | 0.5534 | 0.5545 | 0.5575 |
| | recallT300_ndcgT200 | 0.5578 | 0.5523 | 0.5561 | 0.5517 | 0.5533 | 0.5542 |
| | recallT400_ndcgT100 | 0.5558 | 0.5515 | 0.5551 | 0.5493 | 0.5516 | 0.5527 |
| | recallT300_nmcgT200 | **0.5622**$^{+}$ | **0.5584**$^{+}$ | 0.5634$^{+}$ | **0.5576**$^{+}$ | **0.5573**$^{+}$ | **0.5598**$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.5614$^{+}$ | 0.5566$^{+}$ | 0.5629$^{+}$ | 0.5543$^{+}$ | 0.556$^{+}$ | 0.5582$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is higlighted in bold

When considering Recall@10, the evaluation outcomes are similar to those using nDCG@100. Thus, mseT300_ndcgT200 is the best performing model followed by CM which exploit the user dynamics. Overall, all these approaches achieve statistically significant improvements over the baseline, but when each fold is considered separately, this behaviour can not be generalized, being these approaches significantly different just on some folds.

Moreover, CM approaches perform close to the mseT300_ndcgT200 baseline, being recallT300_ndcgT200 the best performing model for nDCG on fold 5. This is further highlighted when the models are evaluated with respect to Recall@100. In this case, CM approaches that exploit nDCG instead of nMCG perform better and recallT300_ndcgT200 represent the best performing model, significantly improving over the baseline. CM approaches using nDCG are followed by mseT300_ndcgT200 and then by CM approaches exploiting nMCG.

As shown by the comparison between nDCG@10 against nDCG@100, and Recall@10 against Recall@100, CM approaches using nMCG perform better with lower cut-off threshold. Therefore, they better place relevant documents at the beginning of the ranked list. This is further confirmed when ERR@10 is taken into account. Table 3 shows that when ERR@10 is considered, CM approaches using nMCG are the best performing one.

Moreover, recallT200_nmcgT300 is the only approach which is significantly better than the baseline with respect to the average over all the folds. CM approaches using nMCG are also significantly better than the corresponding approaches using nDCG both overall and with respect to each separate fold.

It is well known that ERR is an heavily top-biased measure, i.e. it assigns very high weights to relevant documents at top rank positions and assigns a great discount factor to the following rank positions. Therefore, the evaluation conducted with ERR@10 confirms our hypothesis that CM approaches leveraging the user dynamics improve the effectiveness at the top rank positions. Moreover, since the user dynamics exploited by nMCG is defined just for the first 10 rank positions, by extending the user dynamics further down in the ranking, we might be able to improve the score of our models at lower rank positions. However this is out of the scope of the present work and will be considered for future investigations.

Table 4 reports the evaluation scores of each model on MSLR-WEB30K instead of MSLR-WEB10K. The conclusions that we can infer from Table 4 are aligned to those inferred from Table 3.

Therefore, when we consider nDCG@10 as evaluation measure, CM approaches using nMCG, namely recallT300_nmcgT200 and recallT400_nmcgT100, are the best performing ones both on each separate folds and on the average across folds. Furthermore, they are significantly better than the baseline both overall and on each fold. Overall, they are also significantly different from the corresponding CM approaches adopting nDCG instead of nMCG. Again, this corroborates our intuition that combining CM and the user dynamics represents a successful strategy, which leads the models to perform better than the baseline and the CM approaches without user dynamics.

As it happens for MSLR-WEB10K, when nDCG@100 is considered, 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.

When we consider Recall@10, the outcomes of the evaluation are similar to those using nDCG@10. mseT200_ndcgT300 is still 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. Moreover, recallT400_nmcgT100 is significantly better than the baseline and than recallT400_ndcgT100.

When we consider Recall@100, CM approaches using nDCG perform better than those using nMCG. Both recallT300_ndcgT200 and recallT400_ndcgT100 are the best performing models on three folds out of five and they are significantly better than the baseline. However, 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 that can be inferred from nDCG@100.

Summing up, CM approaches exploiting the user dynamics perform better at lower rank positions, while CM approaches using nDCG and mseT200_ndcgT300 are somehow better in retrieving a higher number of relevant documents in the first 100 rank positions. This is further supported by the results when using ERR@10. recallT300_nmcgT200 is the best performing model in terms of ERR@10 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.

### 4.3 RQ2: Models comparison on navigational and informational queries

In this section we focus the comparison among models on navigational and informational queries separately. Tables 5 and 6 report the evaluation scores on MSLR-WEB10K for navigational and informational queries, respectively. We adopt the same notation used for the tables presented in Sect. 4.2: the best score is reported in bold, $^{\bullet}$ denotes statistically significant improvements with respect to LambdaMart, and $^{+}$ denotes statistically significant improvements of CM approaches trained with nMCG with respect to those trained with nDCG.

From Table 5 we can note that there are just a few models which are significantly better than the baseline for navigational queries. This trend is independent of the evaluation measure adopted, with a few exceptions represented by `mseT200_ndcgT300` with respect to nDCG@10 and nDCG@100, and `recallT300_nmcgT200` with respect to nDCG@10. This suggests that all the models achieve somehow a comparable performance when evaluated with respect to navigational queries. 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.

Furthermore, even if the results are not statistically significant, we can observe from Table 5 that no model is performing markedly better than the others, but there is instead a lot of variability depending on the measure and the fold under consideration. For example, when using nDCG@10, `mseT200_ndcgT300` is the best performing model on average but it is the best model just on three out of five folds; when using nDCG@100 instead, `mseT200_ndcgT300` is the best performing model on four folds. The scenario is different when we consider Recall@10, Recall@100 or ERR@10, where there isn't any regular trend across the folds and the best performing model changes on each fold.

Thus, we conclude that, when using navigational queries, all the models achieve a comparable performance and none of them should be actually preferred. Indeed, except for a couple of cases, none of the proposed models is significantly better than the baseline.

However, this is not the case when we consider informational queries, as shown in Table 6. When using nDCG@10, CM approaches embedding the user dynamics are the best performing models on each fold. Moreover, they are significantly better than the baseline on each fold, while all the other models are not significantly different, except for `mseT200_ndcgT300` in two cases: fold 2 and on the average. CM approaches using nMCG are also significantly better than those using nDCG, both on each fold and on the overall average.

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 lists.

As observed on the whole dataset in Table 3, when the nDCG cut-off is increased to 100, CM approaches using nMCG are less effective than `mseT200_ndcgT300`. `mseT200_ndcgT300` is the best performing model on the overall average and on three folds, while CM approaches using nMCG are the best performing models just on two folds. Both `mseT200_ndcgT300` and CM approaches using nMCG are significantly better than the baseline. Furthermore, when the average across folds is considered, CM approaches using nMCG are better than their corresponding models using nDCG.

**Table 5** Models evaluation on MSLR-WEB10K navigational queries with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.4833 | 0.5033 | 0.4803 | 0.5149 | 0.5142 | 0.4992 |
| | nMCG-MART | 0.4837 | **0.5108** | 0.4804 | 0.5175 | 0.5068 | 0.4998 |
| | mseT200_ndcgT300 | **0.4924** | 0.5106 | **0.4914** | **0.5238** | 0.5161 | **0.5069**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.4767 | 0.5021 | 0.4819 | 0.5171 | 0.5107 | 0.4977 |
| | recallT400_ndcgT100 | 0.4723 | 0.5002 | 0.4824 | 0.52 | 0.5134 | 0.4976 |
| | recallT300_nmcgT200 | 0.4836 | 0.5105$^{+}$ | 0.485 | 0.5234 | 0.5156 | 0.5036$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.4847$^{+}$ | 0.506 | 0.4826 | 0.5232 | **0.5181** | 0.5029$^{+}$ |
| nDCG@100 | LambdaMart | 0.6645 | 0.6833 | 0.6675 | 0.6879 | 0.6882 | 0.6783 |
| | nMCG-MART | 0.6661 | 0.6852 | 0.669 | 0.6911 | 0.6829 | 0.6788 |
| | mseT200_ndcgT300 | **0.6677** | **0.687** | **0.677**$^{\bullet}$ | 0.6931 | **0.6913** | **0.6832**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6623 | 0.6827 | 0.6696 | 0.6876 | 0.6835 | 0.6772 |
| | recallT400_ndcgT100 | 0.6582 | 0.6826 | 0.6705 | 0.6891 | 0.685 | 0.6771 |
| | recallT300_nmcgT200 | 0.6648 | 0.6864 | 0.6723 | 0.6934 | 0.688 | 0.681$^{+}$ |
| | recallT400_nmcgT100 | 0.6649$^{+}$ | 0.6843 | 0.6715 | **0.6942**$^{\bullet,+}$ | 0.6876 | 0.6805$^{+}$ |
| Recall@10 | LambdaMart | 0.3079 | 0.326 | **0.3214** | 0.3609 | 0.3639 | 0.336 |
| | nMCG-MART | 0.307 | 0.3316 | 0.3134 | 0.3609 | 0.3614 | 0.3349 |
| | mseT200_ndcgT300 | **0.3126** | 0.3274 | 0.3152 | **0.3745**$^{\bullet}$ | 0.3663 | **0.3392** |
| | recallT300_ndcgT200 | 0.3021 | 0.3266 | 0.3126 | 0.3698 | 0.3716 | 0.3365 |
| | recallT400_ndcgT100 | 0.304 | 0.3252 | 0.3138 | 0.3715 | **0.3726** | 0.3374 |
| | recallT300_nmcgT200 | 0.3113$^{+}$ | **0.3318** | 0.3179 | 0.3708 | 0.3695 | 0.3403 |
| | recallT400_nmcgT100 | 0.3074 | 0.3294 | 0.3164 | 0.3685 | 0.3692 | 0.3382 |
| Recall@100 | LambdaMart | 0.9016 | 0.9114 | 0.918 | 0.9249 | 0.9288 | 0.9169 |
| | nMCG-MART | 0.9037 | 0.9113 | 0.9164 | 0.9282 | 0.9242 | 0.9168 |
| | mseT200_ndcgT300 | 0.9034 | 0.9106 | 0.9189 | 0.9269 | **0.9316** | **0.9183** |
| | recallT300_ndcgT200 | 0.9029 | **0.9121** | **0.9194** | 0.9258 | 0.9277 | 0.9176 |
| | recallT400_ndcgT100 | **0.904** | 0.911 | 0.9171 | 0.9264 | 0.928 | 0.9173 |
| | recallT300_nmcgT200 | 0.8971 | 0.9101 | 0.9174 | 0.9289 | 0.9288 | 0.9164 |
| | recallT400_nmcgT100 | 0.8967 | 0.9093 | 0.9176 | **0.9289** | 0.9252 | 0.9156 |
| ERR@10 | LambdaMart | 0.4729 | 0.4945 | 0.4669 | 0.5028 | **0.5021** | 0.4878 |
| | nMCG-MART | **0.4783** | **0.501** | 0.4687 | 0.5042 | 0.4951 | 0.4894 |
| | mseT200_ndcgT300 | 0.4776 | 0.4994 | **0.4813** | 0.5006 | 0.5006 | **0.4919** |
| | recallT300_ndcgT200 | 0.4678 | 0.4893 | 0.4684 | 0.4956 | 0.485 | 0.4812 |
| | recallT400_ndcgT100 | 0.4592 | 0.4864 | 0.4712 | 0.4983 | 0.488 | 0.4806 |
| | recallT300_nmcgT200 | 0.476 | 0.4954 | 0.4745 | **0.5112**$^{+}$ | 0.4971 | 0.4908$^{+}$ |
| | recallT400_nmcgT100 | 0.4767$^{+}$ | 0.491 | 0.4723 | 0.5111$^{+}$ | 0.4984 | 0.4899$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is highlighted in bold

The evaluation using Recall@10 leads to conclusions comparable to those for nDCG@10. Even if the best performing model is mseT200_ndcgT300 instead of recallT400_nmcgT100, CM approaches using nMCG are the best performing models on three folds and they are significantly improving over the baseline. Again, when the Recall cut-off is extended to 100, CM approaches using nMCG are less effective than those using nDCG, which significantly improve over the baseline.

Therefore, we come to a conclusion similar to the one we claimed for the whole dataset: CM approaches using nMCG perform better at the top rank positions, while they are less effective when all the rank positions up to 100 are considered. Even if there is no statistical difference with respect to LambdaMart, this is further supported by ERR@10, which considers CM approaches using nMCG as the best performing models also on informational queries. Furthermore, CM approaches using nMCG are significantly better than those using nDCG over each fold with respect to ERR@10.

We conducted the same analysis for navigational and informational queries on MSLR-WEB30K, as reported on Tables 7 and 8 respectively. The experimental results are aligned with those reported for MSLR-WEB10K.

**Table 6** Models evaluation on MSLR-WEB10K Informational queries with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.5072 | 0.4975 | 0.4952 | 0.503 | 0.5074 | 0.5021 |
| | nMCG-MART | 0.5086 | 0.4975 | 0.4973 | 0.4998 | 0.5088 | 0.5024 |
| | mseT200_ndcgT300 | 0.5107 | 0.505$^{\bullet}$ | 0.4981 | 0.5053 | 0.5113 | 0.5061$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.5078 | 0.5008 | 0.4964 | 0.5031 | 0.509 | 0.5034 |
| | recallT400_ndcgT100 | 0.507 | 0.5017 | 0.4971 | 0.5008 | 0.5101 | 0.5033 |
| | recallT300_nmcgT200 | 0.5113$^{\bullet,+}$ | **0.5054**$^{\bullet,+}$ | **0.5007**$^{\bullet,+}$ | 0.5067$^{\bullet,+}$ | 0.5133$^{\bullet,+}$ | 0.5075$^{\bullet,+}$ |
| | recallT400_nmcgT100 | **0.5124**$^{\bullet,+}$ | 0.5043$^{\bullet}$ | 0.5006$^{\bullet,+}$ | **0.5071**$^{\bullet,+}$ | **0.514**$^{\bullet,+}$ | **0.5077**$^{\bullet,+}$ |
| nDCG@100 | LambdaMart | 0.6866 | 0.6816 | 0.6774 | 0.6807 | 0.687 | 0.6827 |
| | nMCG-MART | 0.6872 | 0.6807 | 0.6788 | 0.6794 | 0.6879 | 0.6828 |
| | mseT200_ndcgT300 | **0.6899**$^{\bullet}$ | **0.6855**$^{\bullet}$ | **0.6819**$^{\bullet}$ | 0.6829 | 0.6907$^{\bullet}$ | **0.6862**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6886 | 0.6833 | 0.6805$^{\bullet}$ | 0.6816 | 0.6893 | 0.6846$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.6881 | 0.6843 | 0.6803$^{\bullet}$ | 0.6803 | 0.6897$^{\bullet}$ | 0.6846$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.6887$^{\bullet}$ | 0.6853$^{\bullet,+}$ | 0.6815$^{\bullet}$ | **0.6833**$^{\bullet}$ | 0.6902$^{\bullet}$ | 0.6858$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.6893$^{\bullet}$ | 0.6845$^{\bullet}$ | 0.681$^{\bullet}$ | 0.683$^{\bullet,+}$ | **0.6908**$^{\bullet}$ | 0.6857$^{\bullet,+}$ |
| Recall@10 | LambdaMart | 0.3658 | 0.3577 | 0.3724 | 0.3524 | 0.3575 | 0.3612 |
| | nMCG-MART | 0.3676 | 0.3603 | 0.3726 | 0.351 | 0.3592 | 0.3621 |
| | mseT200_ndcgT300 | 0.3705$^{\bullet}$ | **0.3638**$^{\bullet}$ | 0.3742 | **0.3566** | 0.3624$^{\bullet}$ | **0.3655**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.3675 | 0.3634$^{\bullet}$ | 0.3725 | 0.3524 | 0.3607 | 0.3633 |
| | recallT400_ndcgT100 | 0.3697 | 0.3625 | 0.3736 | 0.3503 | 0.3637$^{\bullet}$ | 0.364$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.3694 | 0.3616 | **0.3755** | 0.3543 | 0.3614 | 0.3644$^{\bullet}$ |
| | recallT400_nmcgT100 | **0.3718**$^{\bullet}$ | 0.3627$^{\bullet}$ | 0.3743 | 0.3535 | **0.364**$^{\bullet}$ | 0.3653$^{\bullet}$ |
| Recall@100 | LambdaMart | 0.9015 | 0.8997 | 0.9017 | 0.8981 | 0.9028 | 0.9007 |
| | nMCG-MART | 0.9017 | 0.8995 | 0.9013 | 0.8981 | 0.903 | 0.9007 |
| | mseT200_ndcgT300 | **0.9053**$^{\bullet}$ | 0.9014 | 0.9053$^{\bullet}$ | 0.9015$^{\bullet}$ | 0.9059$^{\bullet}$ | 0.9039$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.9045$^{\bullet}$ | 0.9018 | **0.906**$^{\bullet}$ | **0.9019**$^{\bullet}$ | 0.9064$^{\bullet}$ | 0.9041$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.9052$^{\bullet}$ | **0.902**$^{\bullet}$ | 0.9055$^{\bullet}$ | 0.9009$^{\bullet}$ | **0.9067**$^{\bullet}$ | **0.9041**$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.9028 | 0.9002 | 0.9047$^{\bullet}$ | 0.899 | 0.9045 | 0.9022$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.903 | 0.9004 | 0.9051$^{\bullet}$ | 0.8991 | 0.9047 | 0.9024$^{\bullet}$ |
| ERR@10 | LambdaMart | 0.5502 | 0.5463 | 0.5431 | 0.5542 | 0.559 | 0.5505 |
| | nMCG-MART | **0.5513** | 0.5427 | 0.5467 | 0.5513 | 0.5602 | 0.5504 |
| | mseT200_ndcgT300 | 0.5476 | 0.5517 | 0.5467 | 0.5509 | 0.5611 | 0.5516 |
| | recallT300_ndcgT200 | 0.5424 | 0.5429 | 0.5386 | 0.5463 | 0.5547 | 0.545 |
| | recallT400_ndcgT100 | 0.542 | 0.5443 | 0.5366 | 0.5418 | 0.553 | 0.5435 |
| | recallT300_nmcgT200 | 0.5489$^{+}$ | **0.5547**$^{\bullet,+}$ | **0.5475**$^{+}$ | 0.5527$^{+}$ | **0.5636**$^{+}$ | **0.5535**$^{+}$ |
| | recallT400_nmcgT100 | 0.5489$^{+}$ | 0.5502$^{+}$ | 0.544$^{+}$ | **0.5544**$^{+}$ | 0.5632$^{+}$ | 0.5521$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is highlighted in bold

Firstly, except for a few cases, there is no statistical improvement over the baseline for any measure or models when navigational queries are considered. This is the same trend observed in Table 5. However, CM approaches using nMCG are significantly better than those using nDCG when evaluated with nDCG@10, nDCG@100 and ERR@10. Thus, even if none of the models is performing better than the baseline, CM approaches using nMCG can be preferred to those using nDCG, especially when the interest is at the top rank positions.

Secondly, on MSLR-WEB30K the high variability of the best performing model with respect to the measure and the fold is even more visible than on MSLR-WEB10K. Given an evaluation measure, none of the proposed models is constantly the best performer across every fold. For instance, the best performing model is always different for each fold when nDCG@10 or ERR@10 are considered. The only exception is represented by Recall@10, where mseT200_ndcgT300 is significantly better than the other models on three folds and on the average across folds.

Therefore, also for navigational queries on MSLR-WEB30K, we can conclude that none of the models can be considered somehow outstanding or can be preferred over other models for each measure and fold. Except for a few cases, the proposed models are not significantly improving over the baseline and this can be due to the small amount of relevant documents included in the datasets. As discussed above, models just need to place the single

**Table 7** Models evaluation on MSLR-WEB30K navigational queries with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.5126 | 0.5108 | 0.5288 | 0.5009 | **0.51** | 0.5126 |
| | nMCG-MART | **0.5156** | 0.5093 | 0.5277 | **0.5012** | 0.5065 | 0.512 |
| | mseT200_ndcgT300 | 0.5143 | 0.5124 | **0.5336** | 0.501 | 0.5042 | 0.5131 |
| | recallT300_ndcgT200 | 0.5078 | 0.5109 | 0.5248 | 0.4997 | 0.5048 | 0.5096 |
| | recallT400_ndcgT100 | 0.508 | 0.5103 | 0.5264 | 0.497 | 0.5022 | 0.5088 |
| | recallT300_nmcgT200 | 0.5118 | **0.5166**$^{\bullet,+}$ | 0.5282 | 0.4997 | 0.5057 | 0.5124$^{+}$ |
| | recallT400_nmcgT100 | 0.5132$^{+}$ | 0.5159$^{\bullet,+}$ | 0.5314$^{+}$ | 0.4989 | 0.508$^{+}$ | **0.5135**$^{+}$ |
| nDCG@100 | LambdaMart | 0.6862 | 0.6845 | 0.698 | 0.6803 | 0.6857 | 0.6869 |
| | nMCG-MART | 0.6889 | 0.6847 | 0.6981 | 0.6803 | 0.6838 | 0.6872 |
| | mseT200_ndcgT300 | **0.6891** | 0.6854 | **0.7022**$^{\bullet}$ | **0.682** | 0.6851 | **0.6888**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6875 | 0.6851 | 0.6958 | 0.6802 | 0.685 | 0.6867 |
| | recallT400_ndcgT100 | 0.6869 | 0.6858 | 0.6959 | 0.679 | 0.6839 | 0.6863 |
| | recallT300_nmcgT200 | 0.687 | **0.6879**$^{\bullet}$ | 0.6973 | 0.6799 | 0.6847 | 0.6873 |
| | recallT400_nmcgT100 | 0.6882 | 0.6873 | 0.6992$^{+}$ | 0.6797 | **0.6866**$^{+}$ | 0.6882$^{+}$ |
| Recall@10 | LambdaMart | 0.3483 | 0.3271 | 0.3601 | 0.3299 | **0.3354** | 0.3402 |
| | nMCG-MART | 0.3473 | 0.3251 | 0.3613 | 0.33 | 0.3323 | 0.3392 |
| | mseT200_ndcgT300 | **0.3547**$^{\bullet}$ | 0.3304 | **0.3658**$^{\bullet}$ | **0.333** | 0.3307 | **0.3429**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.3472 | 0.329 | 0.3572 | 0.3324 | 0.3321 | 0.3396 |
| | recallT400_ndcgT100 | 0.3495 | 0.3267 | 0.3594 | 0.3301 | 0.3289 | 0.3389 |
| | recallT300_nmcgT200 | 0.3502 | 0.332 | 0.3616 | 0.3287 | 0.3297 | 0.3404 |
| | recallT400_nmcgT100 | 0.3514 | **0.3322**$^{+}$ | 0.3634 | 0.3291 | 0.3249 | 0.3402 |
| Recall@100 | LambdaMart | 0.9196 | 0.9127 | 0.9236 | 0.9165 | 0.9147 | 0.9174 |
| | nMCG-MART | 0.9196 | 0.9138 | 0.9238 | 0.9173 | 0.9143 | 0.9178 |
| | mseT200_ndcgT300 | 0.9223 | **0.9157**$^{\bullet}$ | **0.9259**$^{\bullet}$ | 0.9188 | **0.9182**$^{\bullet}$ | **0.9202**$^{\bullet}$ |
| | recallT300_ndcgT200 | **0.9238**$^{\bullet}$ | 0.9152 | 0.9219 | 0.9197 | 0.9179$^{\bullet}$ | 0.9197$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.9232$^{\bullet}$ | 0.9153 | 0.9219 | **0.92** | 0.9177$^{\bullet}$ | 0.9196$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.9218 | 0.9128 | 0.9222 | 0.9147 | 0.9158 | 0.9174 |
| | recallT400_nmcgT100 | 0.9221 | 0.9131 | 0.9228 | 0.916 | 0.9163 | 0.918 |
| ERR@10 | LambdaMart | 0.5038 | 0.4962 | 0.5173 | **0.4871** | 0.4987 | 0.5006 |
| | nMCG-MART | **0.5094** | 0.4967 | 0.5171 | 0.4863 | 0.494 | **0.5007** |
| | mseT200_ndcgT300 | 0.5023 | 0.494 | **0.5207** | 0.4815 | 0.488 | 0.4973 |
| | recallT300_ndcgT200 | 0.4992 | 0.4954 | 0.5078 | 0.4808 | 0.4886 | 0.4944 |
| | recallT400_ndcgT100 | 0.4962 | 0.4972 | 0.5064 | 0.4767 | 0.4863 | 0.4926 |
| | recallT300_nmcgT200 | 0.5014 | **0.5035**$^{\bullet,+}$ | 0.5138 | 0.4851 | 0.4937 | 0.4995$^{+}$ |
| | recallT400_nmcgT100 | 0.5025$^{+}$ | 0.5002 | 0.5185$^{+}$ | 0.4821 | **0.4993**$^{+}$ | 0.5005$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is highlighted in bold

highly relevant document at the beginning of the ranking and they are all comparable in achieving this task.

Finally, Table 8 reports the evaluation scores for each measure and fold with respect to informational queries on MSLR-WEB30K. When nDCG@10 is considered, CM approaches trained with nMCG are the best performing models, significantly improving over the baseline on each fold. Moreover, when the average across folds is considered they are also significantly better than CM approaches trained with nDCG. This further confirms our hypothesis that combining CM with the user dynamics represent a successful approach, especially in terms of nDCG@10.

We can observe a similar trend when Recall@10 is used as evaluation measure. In this case, mseT200_ndcgT300 is the best performing model when we consider the average across folds and it is significantly better than the baseline. However, when we consider the scores on each single fold, mseT200_ndcgT300 is the best model on fold 4 and fold 5, and similarly recallT400_nmcgT100 is the best performing model on fold 2 and fold 3. With respect to the average across folds, recallT400_nmcgT100 is the second best performing model, significantly improving over the baseline and over recallT400_ndcgT100.

**Table 8** Models evaluation on MSLR-WEB30K informational queries with respect to different measures

| Measure | Model | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Average |
|---|---|---|---|---|---|---|---|
| nDCG@10 | LambdaMart | 0.5156 | 0.5186 | 0.5186 | 0.5179 | 0.5143 | 0.517 |
| | nMCG-MART | 0.5172 | 0.5183 | 0.5189 | 0.5179 | 0.5146 | 0.5174 |
| | mseT200_ndcgT300 | 0.5202$^{\bullet}$ | 0.5201 | 0.5202 | 0.5213$^{\bullet}$ | 0.5164$^{\bullet}$ | 0.5196$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.5202$^{\bullet}$ | 0.5202 | 0.5197 | 0.5212$^{\bullet}$ | 0.5185$^{\bullet}$ | 0.52$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.5202$^{\bullet}$ | 0.5191 | 0.5197 | 0.5206$^{\bullet}$ | 0.5181$^{\bullet}$ | 0.5195$^{\bullet}$ |
| | recallT300_nmcgT200 | **0.5214**$^{\bullet}$ | **0.5213**$^{\bullet}$ | 0.5219$^{\bullet,+}$ | **0.5225**$^{\bullet}$ | **0.5193**$^{\bullet}$ | **0.5213**$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.521$^{\bullet}$ | 0.5205 | **0.522**$^{\bullet,+}$ | 0.5218$^{\bullet}$ | 0.5185$^{\bullet}$ | 0.5207$^{\bullet,+}$ |
| nDCG@100 | LambdaMart | 0.6926 | 0.6939 | 0.6936 | 0.6931 | 0.6923 | 0.6931 |
| | nMCG-MART | 0.6938$^{\bullet}$ | 0.6933 | 0.6939 | 0.693 | 0.6929 | 0.6934 |
| | mseT200_ndcgT300 | **0.697**$^{\bullet}$ | **0.6962**$^{\bullet}$ | **0.6956**$^{\bullet}$ | **0.6956**$^{\bullet}$ | 0.6955$^{\bullet}$ | **0.696**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.6965$^{\bullet}$ | 0.6951 | 0.6951$^{\bullet}$ | 0.6956$^{\bullet}$ | **0.6962**$^{\bullet}$ | 0.6957$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.6965$^{\bullet}$ | 0.6945 | 0.6947 | 0.6953$^{\bullet}$ | 0.6959$^{\bullet}$ | 0.6954$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.6963$^{\bullet}$ | 0.6951$^{\bullet}$ | 0.6954$^{\bullet}$ | 0.6952$^{\bullet}$ | 0.6956$^{\bullet}$ | 0.6955$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.6964$^{\bullet}$ | 0.6946 | 0.6954$^{\bullet}$ | 0.6947$^{\bullet}$ | 0.6953$^{\bullet}$ | 0.6953$^{\bullet}$ |
| Recall@10 | LambdaMart | 0.3594 | 0.3727 | 0.3636 | 0.3701 | 0.3695 | 0.3671 |
| | nMCG-MART | 0.3602 | 0.3736 | 0.3645 | 0.3713 | 0.3698 | 0.3679 |
| | mseT200_ndcgT300 | 0.3637$^{\bullet}$ | 0.3747 | 0.3659 | **0.3761**$^{\bullet}$ | **0.3722**$^{\bullet}$ | **0.3705**$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.362 | 0.3746 | 0.365 | 0.3735$^{\bullet}$ | 0.3709 | 0.3692$^{\bullet}$ |
| | recallT400_ndcgT100 | **0.3637**$^{\bullet}$ | 0.373 | 0.3652 | 0.3728$^{\bullet}$ | 0.3705 | 0.369$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.3634$^{\bullet}$ | 0.3744 | 0.3656 | 0.373$^{\bullet}$ | 0.3713 | 0.3695$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.3634$^{\bullet}$ | **0.3755**$^{\bullet,+}$ | **0.3665**$^{\bullet}$ | 0.3736$^{\bullet}$ | 0.3709 | 0.37$^{\bullet,+}$ |
| Recall@100 | LambdaMart | 0.902 | 0.9038 | 0.904 | 0.9015 | 0.906 | 0.9035 |
| | nMCG-MART | 0.9021 | 0.9033 | 0.9041 | 0.9019 | 0.9065 | 0.9036 |
| | mseT200_ndcgT300 | **0.9053**$^{\bullet}$ | 0.9065$^{\bullet}$ | 0.9066$^{\bullet}$ | **0.9048**$^{\bullet}$ | 0.9088$^{\bullet}$ | 0.9064$^{\bullet}$ |
| | recallT300_ndcgT200 | 0.9048$^{\bullet}$ | 0.9067$^{\bullet}$ | **0.9071**$^{\bullet}$ | 0.9045$^{\bullet}$ | **0.9092**$^{\bullet}$ | **0.9064**$^{\bullet}$ |
| | recallT400_ndcgT100 | 0.9052$^{\bullet}$ | **0.9068**$^{\bullet}$ | 0.9068$^{\bullet}$ | 0.9042$^{\bullet}$ | 0.9091$^{\bullet}$ | 0.9064$^{\bullet}$ |
| | recallT300_nmcgT200 | 0.9037$^{\bullet}$ | 0.9052$^{\bullet}$ | 0.905 | 0.9021 | 0.9081$^{\bullet}$ | 0.9048$^{\bullet}$ |
| | recallT400_nmcgT100 | 0.9039$^{\bullet}$ | 0.9052$^{\bullet}$ | 0.9053$^{\bullet}$ | 0.9026$^{\bullet}$ | 0.9081$^{\bullet}$ | 0.905$^{\bullet}$ |
| ERR@10 | LambdaMart | 0.5688 | **0.5688** | 0.5714 | 0.5676 | 0.5633 | 0.568 |
| | nMCG-MART | 0.5694 | 0.5671 | **0.5728** | 0.5674 | 0.5639 | 0.5681 |
| | mseT200_ndcgT300 | 0.5711 | 0.5672 | 0.5687 | 0.5663 | 0.565 | 0.5676 |
| | recallT300_ndcgT200 | 0.5672 | 0.562 | 0.5641 | 0.5644 | 0.5635 | 0.5643 |
| | recallT400_ndcgT100 | 0.5653 | 0.5609 | 0.5632 | 0.5623 | 0.562 | 0.5627 |
| | recallT300_nmcgT200 | **0.5719**$^{+}$ | 0.5679$^{+}$ | 0.5716$^{+}$ | **0.5706**$^{+}$ | **0.5674**$^{\bullet,+}$ | **0.5699**$^{\bullet,+}$ |
| | recallT400_nmcgT100 | 0.5708$^{+}$ | 0.5664$^{+}$ | 0.5702$^{+}$ | 0.5672$^{+}$ | 0.565$^{+}$ | 0.5679$^{+}$ |

Statistically significant improvements with $p = 0.05$ over the baseline LambdaMart are marked with $^{\bullet}$. Statistically significant improvements with $p = 0.05$ over the model recallT$\cdots$_ndcgT$\cdots$ are marked with $^{+}$. The best score is highlighted in bold

ERR@10 further highlights this trend: recallT300_nmcgT200 is the best performing model on three folds out of five and when the average across folds is considered. Moreover, it is the only model which is significantly better than the baseline. Both recallT300_nmcgT200 and recallT400_nmcgT100 are significantly improving over recallT300_ndcT200 and recallT400_ndcgT100 both on each separate fold and on the average across folds.

However, as it was previously observed, when the cut-off is increased to 100, CM approaches using nMCG are less effective. Thus, when nDCG@10 is considered, mseT200_ndcgT300 is the best performing model, followed by CM approaches with Recall and nDCG. When using Recall@100, the ranking of models is reversed and CM approaches trained with Recall and nDCG are the best performing models followed by mseT200_ndcgT300 (even if their scores are equals up to the third decimal digit). Although CM approaches trained with nMCG are not the best performing models when the cut-off is set at 100, they are still significantly better then the LambdaMart baseline.

Therefore, we can conclude that with informational queries CM approaches exploiting the user dynamics represent the best models or are among the best ones when the focus is on the top 10 positions, meaning that exploiting the user dynamics is beneficial for identifying the most relevant documents and moving them towards the top of the ranking.

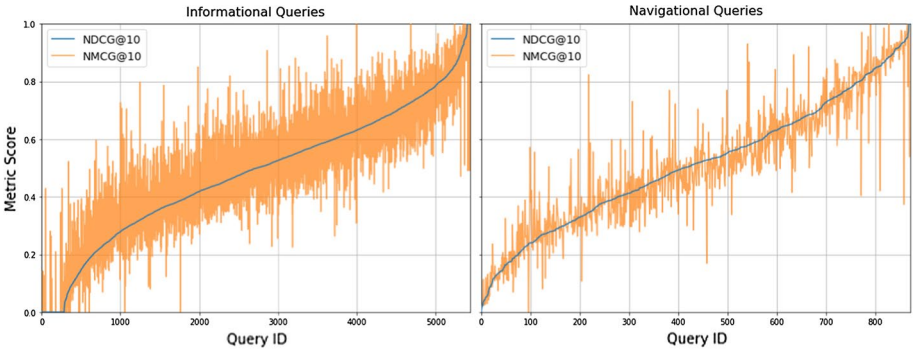



**Fig. 2** Comparison of nDCG@10 versus nMCG@10 score distributions for informational and navigational queries

As discussed above, the user dynamics account only for the first 10 rank positions and an extension of the user dynamics further deep in the ranking might potentially improve the performance of these models, even at higher cut-off thresholds. Moreover, even if the proposed CM approaches using nMCG are not the best performing models with respect to nDCG@100 or Recall@100, they are still significantly improving over the LambdaMart baseline.

Finally, in Fig. 2 we further investigate how the nMCG metric impacts on the model training process compared to nDCG. Specifically, we computed the distribution of nDCG@10 scores generated by the LambdaMart model over the test queries of the Fold 1 of the MSLR-WEB30K dataset, and the distribution of nMCG@10 scores generated by the nMCG-MART model (we observed similar results on other folds). While the score distributions cannot be compared directly as they refer to different IR metrics, they can be used to analyse the behaviour of the two models. Figure 2 shows, distinguishing between informational and navigational queries, the nDCG@10 scores by LambdaMart for each query, where queries are sorted from least to best performing, and the nMCG@10 scores by nMCG-MART for the same queries in the same order. We can observe two very different behaviours. For navigational queries the two distributions are similar showing some correlation. This means that for whichever query LambdaMart is able to provide an high nDCG@10 then also nMCG-MART is accurate. Indeed, the user dynamics enforced by nMCG is very similar to the nDCG discount: the first rank position is highly promoted compared to the others as also shown in Fig. 1. For informational queries, the two distributions show instead a more limited correlation: queries where LambdaMart is accurate might not be the best queries for nMCG-MART and viceversa. This means that, to some extent, the nMCG-MART model devotes its focus and training efforts on a different subset of queries, due to the different user dynamics of the nMCG metric. This results in a better accuracy across different metrics, other than nMCG, as we observed in this sub section.

### 4.4 RQ3: Tree-wise performance comparison

In this section we analyse the tree-wise performance of the proposed models. Figure 3 compares the performance of different models on MSLR-WEB30K for fold 1. On each plot, the $x$-axis reports the number of trees while the $y$-axis reports nDCG@10 score. Since

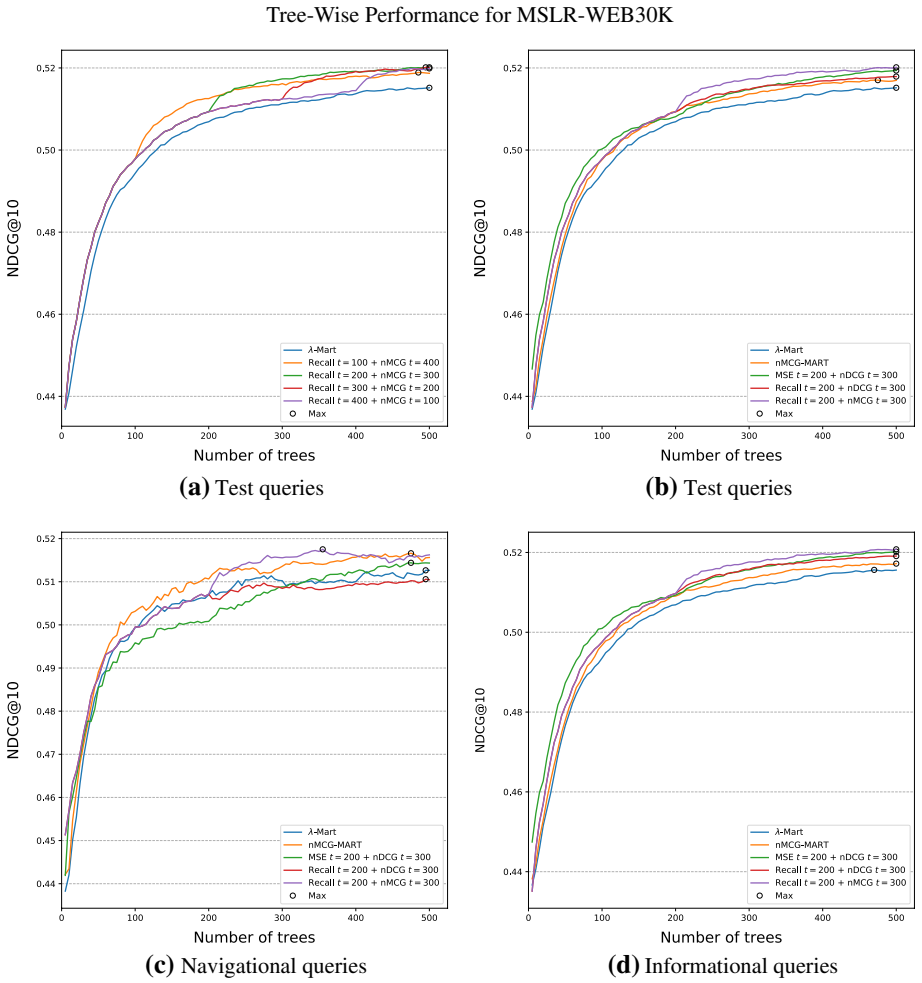



**Fig. 3** Comparison of continuation methods approaches trained with nMCG and with different switching points (a) and comparison of different models on test queries (b), just on navigational queries (c) and just on informational queries (d). All the figures refers to MSLR-WEB30K Fold 1 and all the models are evaluated with respect to nDCG@10

the trend of the proposed models is similar across folds and datasets, we do not report the same figure for each fold and for MSLR-WEB10K, but we choose fold 1 as the explanatory example.

Figure 3a shows the tree-wise performance of different CM approaches trained with Recall, followed by nMCG, together with LambdaMart as baseline. For the CM approaches, we varied the switching point $T_0$ in the set $\{100, 200, 300, 400\}$. You can note that all the CM approaches are always better than the baseline, as the number of trees increases. Moreover, each boosting of the performance corresponds exactly to the switching point $T0$. Therefore, replacing the objective function from Recall to nMCG is beneficial for the learning algorithm at any switching point. Indeed, the learning algorithm gets an

immediate boosting at the switching point and then persists in learning with a decreasing rate until a plateau is reached.

Figure 3b–d, compare `recallT200_nmcgT300` with `recallT200_ndcgT300`, `mseT200_ndcgT300`, nMCG-MART and LambdaMart as baselines. Figure 3b shows the tree-wise performance of each model on the whole test set. Again, you can clearly note that all the proposed models are better than the baseline as the number of trees increases. Focusing on the CM approaches leveraging the user dynamics, it is interesting how these methods achieve the greatest boosting when the objective function is replaced, i.e. at $T - 0 = 200$. Indeed, when the number of trees is lower than 200, the best performing model is `mseT200_ndcgT300`, meaning that MSE leads to better nDCG@10 scores than Recall. However, after the switching point `recallT200_nmcgT300` becomes the best performing model, thanks to nMCG which increases nDCG@10 scores more than nDCG. This supports once more our hypothesis that combining CM approaches with nMCG represents a successful strategy.

Figure 3d corresponds to Fig. 3b, but considers just informational queries. The general trend of each model is quite similar to the one illustrated in Fig. 3b, before the switching point MSE leads to better performance, while after the switching point nMCG allows the CM to achieve higher scores than nDCG, as the number of trees increases. This is perfectly aligned with the results reported in Table 4 and in Table 8, where the outcome of the models comparison is similar for the whole dataset and for informational queries with respect to nDCG@10.

Finally, Fig. 3c corresponds to Fig. 3b, but considers just navigational queries. It is extremely evident how the lines representing each model are not as smooth as in Fig. 3b or Fig. 3c. This perfectly mirrors the results reported in Table 7, which demonstrate how the best performing model strongly depends on the fold and none of the models is stable across the folds with respect to nDCG@10. Indeed, in Fig. 3c all the models lines cross each other multiple times and the best performing model depends on the number of trees. With less than 200 trees `mseT200_ndcgT300` is the best model, then `recallT200_nmcgT300` is the best model and, finally, with more than 400 trees `mseT200_ndcgT300` is again the best performing model. Therefore, Fig. 3c further highlights that the considered models struggle in learning how to rank documents when the amount of relevance available is limited. As future work we plan to investigate different combinations of CM approaches to achieve better performances even on navigational queries.

## 5 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. Building on the results of our previous work, we modeled the user dynamics in scanning a ranked result list with Markov chains and integrated the resulting model in the loss function driving the learning process. Moreover, with the aim of better capturing the weak relevance signals hidden in the features representing the training samples, we designed the iterative learning of ranking functions based on forests of decision trees as a two step process, where the initial trees are trained by minimizing a simpler, possibly smoother objective function, while the remaining trees are trained by optimizing nDCG or our new nMCG measure aware of the user dynamics.

To assess our proposal we conducted an exhaustive set of reproducible experiments based on publicly available datasets and code. We swiped all the hyperparameters to devise the setting of the baselines and our proposed algorithms providing the best performance on the validation datasets. 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 Chuklin et al. (2015) 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. Moreover, we plan to investigate more complex continuation methods, as for example with three steps instead of two, to understand whether there is a performance boosting with respect to each switching point and whether changing the objective function is always beneficial for the learning algorithm.

**Acknowledgements** This paper is partially supported by the BIGDATAGRAPES (EU H2020 RIA, Grant Agreement No. 780751) and the OK-INSAID (MIUR-PON 2018, Grant Agreement No. ARS01_00917) 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

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior. In E. N. Efthimiadis, S. Dumais, D. Hawking, & K. Järvelin (Eds.), *Proceedings of 29th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2006)*, ACM Press, New York, USA, pp. 19–26.

Allan, J., Aslam, J. A., Sanderson, M., Zhai, C., & Zobel, J. (Eds.). (2009). *Proceedings of 32nd annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2009)*, ACM Press, New York, USA.

Allgower, E. L., & Georg, K. (1980). *Numerical continuation methods an introduction*. Heidelberg: Springer.

Beitzel, S. M., Jensen, E. C., Frieder, O., Grossman, D., Lewis, D. D., Chowdhury, A., & Kolcz, A. (2005). Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR '05)*, ACM, New York, NY, USA, pp. 581–582. https://doi.org/10.1145/1076034.1076138.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In A. Danyluk, L. Bottou, M. L. Littman (Eds.), *Proceedings of 26th annual international conference on machine learning (ICML 2009)*, ACM Press, New York, USA, pp. 41–48.

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, *36*(2), 3–10.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. In S. Dzeroski, L. De Raedt, & S. Wrobel (Eds.), *Proceedings of 22nd international conference on machine learning (ICML 2005)*, ACM Press, New York, USA, pp. 89–96.

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An overview*. Technical report, Microsoft Research, MSR-TR-2010-82.

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. C. Platt, & T. Hoffman (Eds.), *Proceedings of 19th international conference on neural information processing systems (NIPS 2006)*, MIT Press, Cambridge, MA, USA, pp. 193–200.

Chapelle, O., Chi, M., & Zien, A. (2006). A continuation method for semi-supervised SVMs. In W. Cohen, A. Moore (Eds.), *Proceedings of 23rd annual international conference on machine learning (ICML 2006)*, ACM Press, New York, USA, pp. 185–192.

Chapelle, O., Joachims, T., Radlinski, F., & Yue, Y. (2012). Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, *30*(1), 6.

Chapelle, O., Metzler, D., Zhang, Y., & Grinspan, P. (2009). Expected reciprocal rank for graded relevance. In D. W. L. Cheung, I. Y. Song, W. W. Chu, X. Hu, & J. J. Lin (Eds.), *Proceedings of 18th international conference on information and knowledge management (CIKM 2009)*, ACM Press, New York, USA, pp. 621–630.

Chapelle, O., & Zhang, Y. (2009). A dynamic bayesian network click model for web search ranking. In J. Quemada, G. León, Y. Maarek, & W. Nejdl (Eds.), *Proceedings of 18th international conference on World Wide Web (WWW 2009)*, ACM Press, New York, USA, pp. 1–10.

Chen, B., & Xiu, N. (1999). A global linear and local quadratic noninterior continuation method for nonlinear complementarity problems based on Chen–Mangasarian smoothing functions. *SIAM Journal on Optimization*, *9*(3), 605–623.

Chen, X., Gupta, A. (2015). Webly supervised learning of convolutional networks. In R. Bajcsy, G. Hager, Y. Ma, K. Ikeuchi, C. Schnörr, J. Sivic, & R. Vidal (Eds.), *Proceedings of 2015 IEEE international conference on computer vision (ICCV)*, IEEE Computer Society, Los Alamitos, CA, USA, ICCV '15, pp. 1431–1439.

Chuklin, A., Markov, I., & de Rijke, M. (2015). *Click models for web search*. San Rafael: Morgan & Claypool Publishers.

Coleman, T. F., & Wu, Z. (1996). Parallel continuation-based global optimization for molecular conformation and protein folding. *Journal of Global Optimization*, *8*(1), 49–65.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537.

Craswell, N., Zoeter, O., Taylor, M., & Ramsey, B. (2008). An experimental comparisonof click position-bias models. In (Najork et al., 2008), pp. 87–94.

Donmez, P., Svore, K. M., Burges, C. J. C. (2009). On the local optimality of LambdaRank. In (Allan et al., 2009), pp. 460–467.

Dou, Z., Song, R., Yuan, X., & Wen, J. R. (2008). Are click-through data adequate for learning web search rankings? In J. G. Shanahan, S. Amer-Yahia, I. Manolescu, Y. Zhang, D. A. Evans, A. Kolcz, K. S. Choi, & A. Chowdhury (Eds.), *Proceedings of 17th international conference on information and knowledge management (CIKM 2008)*, ACM Press, New York, USA, pp. 73–82.

Ferrante, M., Ferro, N., & Maistro, M. (2014). Injecting user models and time into precision via Markov chains. In S. Geva, A. Trotman, P. Bruza, C. L. A. Clarke, & K. Järvelin (Eds.), *Proceedings of 37th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2014)*, ACM Press, New York, USA, pp. 597–606.

Ferro, N., Lucchese, C., Maistro, M., & Perego, R. (2017). On including the user dynamic in learning to rank. In N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, & R. W. White (Eds.), *Proceedings of 40th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2017)*, ACM Press, New York, USA, pp. 1041–1044.

Ferro, N., Lucchese, C., Maistro, M., & Perego, R. (2018). Continuation methods and curriculum learning for learning to rank. In A. Cuzzocrea, J. Allan, N. W. Paton, D. Srivastava, R. Agrawal, A. Broder, M. J. Zaki, S. Candan, A. Labrinidis, A. Schuster, & H. Wang (Eds.), *Proceedings of 27th international conference on information and knowledge management (CIKM 2018)*, ACM Press, New York, USA, pp. 1523–1526.

Hofmann, K., Schuth, A., Whiteson, S., & de Rijke, M. (2013). Reusing historical interaction data for faster online learning to rank for IR. In S. Leonardi, A. Panconesi, P. Ferragina, A. Gionis (Eds.), *Proceedings of 6th ACM international conference on web searching and data mining (WSDM 2013)*, ACM Press, New York, USA, pp. 183–192.

Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Proceedings of 27th international conference on neural information processing systems (NIPS 2014)*, Vol. 2, MIT Press, Cambridge, MA, USA, pp. 2042–2050.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 422–446.

Jiang, D., Pei, J., & Li, H. (2013). Mining search and browse logs for web search: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *4*(4), 57:1–57:37.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In O. Zaïane, R. Goebel, D. Hand, D. Keim, & R. Ng (Ed.), *Proceedings of 8th ACM SIGKDD international conference on knowledge discovery and data mining (KDD 2002)*, ACM Press, New York, USA, pp. 133–142.

Joachims, T., & Radlinski, F. (2007). Search engines that learn from implicit feedback. *Computer*, *40*(8), 34–40.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., & Gay, G. (2005). Accurately interpreting click through data as implicit feedback. In R. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, & J. Tait (Eds.),

*Proceedings 28th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2005)*, ACM Press, New York, USA, pp. 154–161.

Joachims, T., Swaminathan, A., & Schnabel, T. (2017). Unbiased learning-to-rank with biased feedback. In M. de Rijke, M. Shokouhi, A. Tomkins, M. Zhang (Eds.), *Proceedings of 10th ACM international conference on web searching and data mining (WSDM 2017)*. ACM Press, New York, USA, pp. 781–789.

Liu, T. Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval (FnTIR)*, *3*(3), 225–331.

Lucchese, C., Orlando, S., Perego, R., Silvestri, F., & Tolomei, G. (2013). Discovering tasks from search engine query logs. *ACM Transactions on Information System (TOIS)*, *31*(3), 14:1–14:43.

Mehrotra, R., Awadallah, A. H., & Yilmaz, E. (2018) Learnir: Wsdm 2018 workshop on learning from user interactions. In *Proceedings of the eleventh ACM international conference on web search and data mining*, ACM, New York, NY, USA, WSDM '18, pp. 797–798. https://doi.org/10.1145/3159652.3160598.

Moré, J. J., & Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal on Optimization*, *7*(3), 814–836.

Nagamune, R. (2003). A robust solver using a continuation method for Nevanlinna–Pick interpolation with degree constraint. *IEEE Transactions on Automatic Control (TAC)*, *48*(1), 113–117.

Najork, M., Broder, A., & Chakrabarti, S. (Eds.). (2008). *Proceeding of 1st ACM international conference on web searching and data mining (WSDM 2008)*, ACM Press, New York, USA.

Norris, J. R. (1998). *Markov chains*. Cambridge: Cambridge University Press.

Qin, T., & Liu, T. Y. (2013). Introducing LETOR 4.0 Datasets. Information Retrieval (csIR). arXiv:1306.2597.

Qu, M., Tang, J., & Han, J. (2018). Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In Y. Chang, C. Zhai, Y. Liu, & Y. Maarek (Eds.), *Proceedings of 11th ACM international conference on web searching and data mining (WSDM 2018)*, ACM Press, New York, USA, pp. 468–476.

Rabani, E., Reichman, D. R., Krilov, G., & Berne, B. J. (2002). The calculation of transport properties in quantum liquids using the maximum entropy numerical analytic continuation method: Application to liquid para-hydrogen. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, *99*(3), 1129–1133.

Sakai, T., & Dou, Z. (2013). Summaries, ranked retrieval and sessions: A unified framework for information access evaluation. In G. J. F. Jones, P. Sheridan, D. Kelly, M. de Rijke, & T. Sakai (Eds.), *Proceedings of 36th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2013)*. ACM Press, New York, USA, pp. 473–482.

Schuth, A., Hofmann, K., Whiteson, S., & de Rijke, M. (2013). Lerot: An online learning to rank framework. In K. Balog, D. Elsweiler, E. Kanoulas, L. Kelly, & M. D. Smucker (Eds.), *Proceedings of 1st workshop on living labs for information retrieval evaluation (LL 2013)*, ACM Press, New York, USA, pp. 23–26.

Schuth, A., Oosterhuis, H., Whiteson, S., & de Rijke, M. (2016). Multileave gradient descent for fast online learning to rank. In P. N. Bennett, V. Josifovski, J. Neville, & F. Radlinski (Eds.), *Proceedings of 9th ACM international conference on web searching and data mining (WSDM 2016)*. ACM Press, New York, USA, pp. 457–466.

Serdyukov, P., Craswell, N., & Dupret, G. (2012). WSCD2012: Workshop on web search click data 2012. In E. Adar, J. Teevan, E. Agichtein, Y. Maarek (Eds.), *Proceedings of 5th ACM international conference on web searching and data mining (WSDM 2012)*. ACM Press, New York, USA, pp. 771–772.

Silvestri, F. (2009). Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval (FnTIR)*, *4*(1–2), 1–174.

Smucker, M. D., Allan, J., & Carterette, B. A. (2007). A comparison of statistical significance tests for information retrieval evaluation. In M. J. Silva, A. A. F. Laender, R. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, & A. Falcão (Eds.), *Proceeding of 16th international conference on information and knowledge management (CIKM 2007)*, ACM Press, New York, USA, pp. 623–632.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). SoftRank: Optimizing non-smooth rank metrics. In (Najork et al., 2008), pp. 77–86.

Teodorescu, I. (2009). Maximum likelihood estimation for Markov Chains. Computation (statCO). arXiv :0905.4131.

Wang, H., Langley, R., Kim, S., McCord-Snook, E., & Wang, H. (2018). Efficient exploration of gradient space for online learning to rank. In: K. Collins-Thompson, Q. Mei, B. Davison, Y. Liu, & E. Yilmaz (Eds.), *Proceedings of 41th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2018)*, ACM Press, New York, USA, pp. 145–154.

White, R. W., Bailey, P., & Chen, L. (2009). Predicting user interests from contextual information. In: (Allan et al., 2009), pp. 363–370.

Wu, Q., Burges, C. J. C., Svore, K. M., & Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, *13*(3), 254–270.

Yilmaz, E., Shokouhi, M., Craswell, N., & Robertson, S. (2010). Expected browsing utility for web search evaluation. In J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, A. An (Eds.), *Proceedings of 19th international conference on information and knowledge management (CIKM 2010)*, ACM Press, New York, USA, pp. 1561–1565.

Zhang, Y., Park, L. A. F., & Moffat, A. (2010). Click-based evidence for decaying weight distributions in search effectiveness metrics. *Information Retrieval*, *13*(1), 46–69.

Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., & Sun, G. (2007). A general boosting method and its application to learning ranking functions for web search. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.) *Proceedings of 20th international conference on neural information processing systems (NIPS 2007)*, MIT Press, Cambridge, MA, USA, pp. 1697–1704.

Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., & Wen, Z. (2017). Online learning to rank in stochastic click models. In D. Precup, & Y. Whye Teh (Eds.), *Proceedings of 34th annual international conference on machine learning (ICML 2017)*, ACM Press, New York, USA, pp. 4199–4208.