How do Online Learning to Rank Methods Adapt to Changes of Intent?

Shengyao Zhuang
The University of Queensland
Brisbane, QLD, Australia
s.zhuang@uq.edu.au

Guido Zuccon
The University of Queensland
Brisbane, QLD, Australia
g.zuccon@uq.edu.au

ABSTRACT

Online learning to rank (OLTR) uses interaction data, such as clicks, to dynamically update rankers. OLTR has been thought to capture user intent change overtime – a task that is impossible for rankers trained on statistic datasets such as in offline and counterfactual learning to rank. However, this feature has never been demonstrated and empirically studied, as previous work only considered simulated online data with a single user intent, or real online data with no explicit notion of intents and how they change over interactions. In this paper, we address this gap by study the capability of OLTR algorithms to adapt to user intent change.

Our empirical experiments show that the adaptation to intent change does vary across OLTR methods, and is also dependent on the amount of noise in the implicit feedback signal. This is an important result, as it highlights that intent change adaptation should be studied alongside online and offline performance.

Investigating how OLTR algorithms adapt to intent change is challenging as current LTR datasets do not explicitly contain the required intent data. Along with the main findings reported in this paper related to intent change, we also contribute a methodology to investigate this aspect of OLTR methods. Specifically, we create a collection for OLTR with explicit intent change by adapting an existing TREC collection to this task. We further introduce methods to model and simulate click behaviour related to intent change. We further propose novel evaluation metrics tailored to study different aspects of how OLTR methods adapt to intent change.

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; **Retrieval models and ranking**.

KEYWORDS

Online Learning to Rank, Intent Change, Web Search

ACM Reference Format:
Shengyao Zhuang and Guido Zuccon. 2021. How do Online Learning to Rank Methods Adapt to Changes of Intent?. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462937

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462937

1 INTRODUCTION

Online Learning to Rank (OLTR) refers to techniques that learn a ranker from implicit user feedback, like clicks, rather than from explicit relevance judgements, and they do so in an online manner, that is, by iteratively manipulating the ranking shown to users to both provide high quality results and obtaining feedback to further improve the ranker. Although compared to explicit relevance judgements, implicit feedback presents noise and a number of biases [13, 19, 30], OLTR methods have been shown to overcome this problem and learn effective rankers, especially from click feedback [14, 15, 27, 29, 35, 43]. By using implicit feedback, OLTR provides a number of advantages compared to learning methods based on explicitly labelled data. For example, OLTR does not require costly and lengthy relevance labels to be collected [15], and it can be applied to scenarios where privacy is paramount [21].

In addition to advantages related to the costs and privacy of collecting implicit relevance labels, the literature has claimed that implicit feedback would allow OLTR to adapt to changes in intents for users queries that occur throughout the search interactions [17, 27]. That is, for example, if a document may have been relevant to a query few months ago, that same document may not be relevant any more now: this may be because of temporal factors, breaking news, or user preference changes. This change in intent would be sensed by the OLTR ranker from the implicit feedback, e.g., click signal, and the ranker updated to reflect such change. Although it appears intuitive to argue for this property of OLTR methods, it is unclear how current OLTR methods do perform with respect to intent change, and if the level of intent change adaptability varies across methods. This is because OLTR experiments were conducted either by considered simulated online data with only a single user intent, or real online data with no explicit notion of intents and how they change over interactions. In this paper, we aim to investigate how existing OLTR algorithms adapt to user intent change and evaluate two aspects related to intent change – the sensitivity of OLTR methods to intent change (*nDCG_Drop@k*) and the average loss due to intent change (*nDCGΔ@k*).

An impediment to the investigation of how OLTR adapts to changes of intent is that there is no dataset available that explicitly models this aspect. To overcome this, we leverage a public IR dataset and devise a method to create an OLTR dataset for the investigation of intent changes. In addition, we introduce two evaluation metrics that quantify the impact changes of intent have, facilitating the comparison of different OLTR methods.

Our investigation considers five focused research questions aimed at understanding the role intent change has in OLTR studies. While

previous work has claimed OLTR methods adapt to intent change, this property has not been formally investigated. Our first research question aims to fill this gap:

RQ1: How sensitive are OLTR methods to intent change? How well do they adapt? To answer this research question, we simulate intent variation during the online learning process and measure the drop in effectiveness of the ranker when the new intent is introduced, and whether the ranker recovers from this drop.

In practice, intent changes occur following different patterns. At time, changes are abrupt; other times they occur over time, and two or more intents co-exist across a common interval of interactions. This leads to the next research question:

RQ2: How intent change patterns affect OLTR methods? To answer this research question, we simulate different patterns of intent change and measure how effectiveness in terms of intent change adaptation varies across patterns.

At times, intent changes are temporal, and in particular some are periodic [20, 25, 37, 41]. For example, a query may have intent A for several months a year, but for a period the intent underlying a query switches to B, for then moving back to A again. This leads us to the next research question:

RQ3: How well does OLTR adapt to new re-occurrences of intents already observed in past iterations, but that did not occur in the near past? To answer this question, we simulate variations of intent of the pattern $A \rightarrow B \rightarrow A$ and measure the difference in OLTR effectiveness between the last iteration on intent A before switching to B and the first iteration on intent A after switching back from B. We also measure the difference in effectiveness produced by switching to B for a period of time compared to train only on A.

Most OLTR research has been investigated with respect to simple linear rankers. Only recently have more complex rankers, such as neural rankers, been studied in the context of OLTR, e.g. [21, 27]. This motivates us to investigate more complex rankers with respect to intent change:

RQ4: Do neural rankers adapt better to intent change than simpler rankers (e.g., linear) in OLTR? To answer this, we compare the results obtained for linear rankers in RQ1 with the corresponding results in the same settings when using neural rankers.

It is often the case in practice that multiple intents do occur concurrently in query logs. The experimental setup we put forward in this paper offers us the opportunity to further investigate OLTR effectiveness in light of the different intents underlying a query. To demonstrate the larger applicability of our framework of investigation, we further investigate the following research question:

RQ5: How does OLTR performance vary between rankers learnt separately per-intent vs. by combining all intents? In other words, we explore whether it is better to have a single ranker learnt across all intents, or a set of intent-specific rankers, if we were equipped with reliable methods to detect intents at query time so as to route a query to the ranker purposely learnt to setup the specific intent for the query.

2 RELATED WORK

In Learning to Rank (LTR) a ranker is constructed by training a model in an off-line manner, using labelled data consisting of query-document pairs [24]. The use of labelled data and the offline learning process have three main drawbacks: (1) relevance judgements are onerous to collect [5, 31, 32], both in terms of time and cost, (2) the labelling by editors of private documents, e.g., emails, raises privacy concerns [39], (3) this often resolves in a creation of a static dataset which does not adapt to changes in intent for the users queries – this may render some documents that have been labelled relevant now, irrelevant in a future time.

In response to the limitations of traditional LTR methods, the application of LTR to online settings (OLTR) has been considered, where rankers are trained interactively with user interactions [40]. Unlike methods in the offline setting, OLTR controls what rankings to display and updates the ranker's weights according to the user's responses. This online training mechanism allows learning algorithms to quickly adapt to changes that happen in the learning signals. This is unlike other approaches that attempt to exploit user clicks to train rankers in place of editorial labels, but do so still in an offline, batch-mode manner, e.g., counterfactual LTR [1, 16].

The Dueling Bandit Gradient Descent (DBGD) [40] is a representative OLTR method. It models OLTR as a dueling bandit problem and uses online evaluation where preference between a candidate and the current production rankers is inferred with user clicks on interleaved rankings. Recent works in OLTR have built upon DBDG and online evaluation. Schuth et al. [36] introduced Multileave Gradient Descent (MGD): this method replaces the interleaving in DBGD with multileaving. Other extensions of this method focus on improving efficiency and effectiveness of multileaving comparison [26] and reduce variance of gradient estimate by online evaluation [38]. A more recent work, called Counterfactual Online Learning to Rank (COLTR), further suggested to replace online evaluation with counterfactual evaluation [43]. Since this method does not require interleaving or multilieaving, it can evaluate a large number of candidate rankers at each interaction time step, thus being computationally efficient. The current state-of-the-art OLTR method is Pairwise Differentiable Gradient Descent (PDGD) [27]. Instead of estimating gradient updates from a pool of candidate rankers, PDGD directly estimates gradients based on document pair preferences inferred from user clicks. Empirical results have shown that PDGD significantly outperforms DBDG based methods [27, 28]. It is important to note that, instead of conducting experiments with real-world users, all research mentioned above has evaluated OLTR methods using synthetic user interaction data. In such an approach, user clicks are generated based on the relevance labels recorded in existing offline LTR datasets. However, relevance labels in those datasets are collected from adhoc search tasks, and thus exhibiting a single intent per query. Thus, existing OLTR methods have been evaluated assuming the presence of a single intent, and the effect of intent change on OLTR has so far been ignored.

Previous work that examined intents in offline LTR has done so in the context of search result diversification [33, 34, 42] and fresh intent detection [10, 22]. A recent OLTR method has been proposed for search engine result diversification [23].

3 DATASET FOR INTENT CHANGE

In this section, we discuss our framework for studying intent change in OLTR. This consists of (1) a dataset of query-document pairs for which feature representations are computed to be used for ranking, coupled with indications of multiple intents for each queries

Table 1: Average number of relevant documents per intent before re-balancing.

intent id	1	2	3	4
Avg. rel per query	71.6	52.0	42.0	26.4

Table 2: Instantiations of the SDBN click model.

	$p(c = 1\|rel(d))$		$p(s = 1\|rel(d))$	
$rel(d)$	0	1	0	1
perfect	0.0	1.0	0.0	0.0
navigational	0.05	0.95	0.2	0.9
informational	0.3	0.7	0.1	0.5

and relevance assessments tailored to intents (Section 3.1), (2) a mechanism to simulate clicks (Section 3.2), and (3) mechanisms to simulate changes in intents (Section 3.3).

3.1 Dataset for LTR

There is no available dataset for OLTR that explicitly records intents for users queries and that presents changes of intent across interactions for the same query. Currently available OLTR datasets consist of query-document pairs with associated feature vectors and relevance labels that are used to train the rankers: a click model is applied to simulate clicks based on the relevance labels and rankings. We set to create one such dataset, but which does explicitly encode an array of intents for each query.

The TREC Web Track 2009 to 2012 collection does provide a large set of web pages (ClueWeb09) along with 200 queries [8]. The diversity task in this TREC track provides for each query an array of intent descriptions, ranging from 4 intents up to 10 intents. Furthermore, query-document pairs relevance judgements are provided per-intent. This provides a good collection from which to derive an OLTR collection for intent change because intents are explicitly defined and relevance judgements are provided with respect to the intents. We note that TREC collections have been used before in the context of OLTR research [32], while ClueWeb09 has been used for LTR research [2, 12] but not for OLTR.

To use Clueweb09 in OLTR experiments, we generated features for query-document pairs to be used in the LTR process. For this, we rely on existing features like PageRank and the Waterloo spam scores [9], and we create a feature extraction pipeline to generate other common LTR features, e.g. field-based BM25 scores, TF-IDF scores, URL length, query length, etc.. Each query-document pair in our dataset is represented by 105 features; the full feature description along with code for the feature generation pipeline can be found at https://github.com/ielab/OLTR-intent-change.

The TREC diversity track data we use to define our intents provides a different number of intents per query, but all queries have a minimum of 4 intents. To simplify the settings of our experiments, then, for each query we select the first four intents as defined in the TREC files, so that all queries in the dataset for OLTR have the same number of intents. Then, we collate all documents that have an explicit relevance judgement for at least one of the selected intents for a query, and compute their feature representation with respect to the give query. Note that, for a given query, a document may be relevant to more than one intent: this however does not affect the feature representation as we use a mix of query dependent and query independent features, but no feature is intent dependent. The created dataset consists of 200 queries with 4 intents each and on average 512 candidate documents per query. Table 1 reports the average number of relevant documents per query for across the selected intents. We observe that the average number of relevant documents per intent varies largely across intents: for example, intent 4 has less than half the number of relevant documents than intent 1. Because of this distribution of relevance labels, optimizing a ranker for intent 4 may be a harder task compared to optimizing it for intent 1. To avoid this bias, for each query we re-label the original intent number by randomly assign it a new intent id. This is possible because intents are dependent across documents, but independent across queries. This trick ensures that the relevance is consistent across an intent for a query, but balances the number of relevant documents per intent id across the dataset. In our experiments, we shall repeat this process of re-balancing 25 times, thus giving rise to 25 OLTR experiments, for which results are averaged.

3.2 Synthetic click pattern generation

Using a LTR dataset with heuristically initialized click models to simulate user clicks is the standard experimental setting used in OLTR [7]. We follow the same procedure (with the difference that for us clicks are dependent also on intents, more on this in Section 3.3). First, a query is uniformly sampled from the dataset along with its candidate document set. Then, the OLTR method generates a ranking list based on its current model parameters. User click behaviour on this ranking is then simulated by the click model. Finally, the learning algorithm updates the ranking function according to the observed clicks. This experimental pipeline allows to avoid testing OLTR methods on real-world users by replacing users with click models. In addition, user click behaviour can be controlled using different click models.

We use the *Simplified Dynamic Bayesian Network* (SDBN) click model [6] to simulate clicks (click labels): this is a standard click model used extensively in OLTR [14, 15, 27, 29, 35, 38, 43] and allows for clear control of the point when intent change occurs. Alternative click models, like neural click models [3, 4] might better fit users' click behaviour, however it is unclear how they can be used in a controlled manner to control for intent change. In addition, these models require to initialize all parameters from scratch, a task typically done using existing logs: but logs with an explicit account of intent change do not currently exist.

SDBN assumes users always exam documents in the ranking list from top to bottom and decide whether to click on an observed document d with a probability $P(c = 1|rel(d))$ where $rel(d)$ is the relevance label for document d in the LTR dataset. After a document is clicked, the user may decide to stop the search session with a certain probability $P(s = 1|rel(d))$. This click model has been shown to perform well in the task of click prediction for web search, suggesting it reasonably models real-world user click behaviour [7]. Table 2 reports the values for the SDBN click model parameters. We consider three click settings: perfect (no noise), navigational and informational (noisiest click model). These settings are in line with previous work on OLTR. According to the perfect click model, users examine all documents provided in the ranking list, always click on relevant documents and never click on non-relevant documents. Thus, the perfect click model omits click noise, allowing us to investigate the performance of OLTR algorithms under a perfect

relevance environment. The navigational instance, which refers to the navigational search task, models users who are seeking for a particular relevant document. In this setting, high probability is assigned to the user stopping the search session after finding a high relevant document. Finally, the informational click setting models a user searching for an an array of information regarding a topic: in this settings clicks are noisy and there is a lower probability of stopping after encountering a relevant document, compared with the navigational setting. As per typical OLTR practice, we only present 10 documents in the ranking list to collect click data: this setting introduces selection bias in the OLTR process.

3.3 Non-stationary intent environment and synthetic intent change

Previous work on OLTR has either assumed a stationary intent environment, i.e., without intent change, e.g., using the LETOR 4.0 dataset [32], or has relied on datasets released by web search engines that may have exhibited a non-stationary intent environment, but intents were not explicitly captured and thus neither the stationariness of the environment can be determined nor OLTR can be studied in presence of intent changes. Our dataset instead contains explicit per-intent query-document relevance labels, but it lacks how intents are distributed over interactions. For this, we define five non-stationary user intent environments.

The non-stationary user intent environments used here influence the relevance of a document to a query: documents are regarded relevant if they are relevant to the intent associated to the query being used at the specific interaction. Formally, the relevance of a document d to a given query is decided by $rel_i(d)$ where $i \in \{1, 2, 3, 4\}$ is the intent label. Consequently these intent environments also influence thee synthetic clicks generated by the settings specified in Section 3.2. Specifically when simulating user clicks with respect to intents, if $rel_i(d) \neq rel_{i+1}(d)$, then if the current intent changes from i to $i + 1$ (intent change), then also $rel_i(d)$ will change to $rel_{i+1}(d)$, and so will change the click probability for document d as the click probability depends on the relevance label.

Since we use click models to simulate clicks, we have control on when intents change. We model the current intent $\mathcal{I}$ as a random variable with value i decided by a discrete probability distribution $P_t(\mathcal{I} = i)$, where $\sum_i P_t(\mathcal{I} = i) = 1$, i is the intent label, and t is the time step. Before the user issues a query at a time step t, we sample the user intent label i according to P_t. Then user clicks on the SERP are generated with respect to the sampled intent. Next, we describe the five non-stationary user intent environments used in this work.

Abrupt change: In this environment, we only use one intent at the time to simulate clicks and change from an intent to another abruptly at some point in the interaction history. At any one point in time, only one intent is used. For example, we only use intent 1 to simulate user clicks at the start of the process; we then suddenly change to intent 2 after n query interactions. While this intent environment unlikely to occur in practice, as it requires all users to switch intent for the same query at a given point in time, it is a simple and controlled environment in which to investigate how OLTR algorithms adapt to intent change.

For this environment, we set an intent change time point c so that, at time step $t < c$, the sampling probability for the current intent i is equal to 1 ($P_{t<c}(\mathcal{I} = i) = 1$) and that of other intents is equal to 0 ($P_{t<c}(\mathcal{I} \neq i) = 0$). This means we only sample intent i before the time step c. At time step c, we change the current intent i to $i + 1$ by setting $P_{t\geq c}(\mathcal{I} = i + 1) = 1$ and $P_{t\geq c}(\mathcal{I} \neq i + 1) = 0$.

Smooth abrupt change: In this environment, we consider that at any one given point in time all available intents are possible, but with different probability. In particular, for each time period we set a specific intent to dominate, i.e., be more likely to occur, while the remaining intents can occur but with a low probability. After the period is concluded, there is an abrupt change in the dominating intent. This environment aims to simulate the situation in which, for example, a breaking news leads to a large change in intent for the majority of the users, but a small proportion of users maintain their intent unchanged. For this intent change environment, before the change point c, we set intent sampling probability for the dominating intent i to $P_{t<c}(\mathcal{I} = i) = 0.7$ and for the other intents to $P_{t<c}(\mathcal{I} \neq i) = 0.1$. After the dominating intent changes to $i + 1$, the sampling probabilities become $P_{t\geq c}(\mathcal{I} = i + 1) = 0.7$ and $P_{t\geq c}(\mathcal{I} \neq i + 1) = 0.1$.

Intent leaking: Another possible type of intent change is that a user's intent gradually changes over time to another – this long-term behaviour models the great majority of intent changes occurring on the Web. To test OLTR methods on this type of intent change environment, we introduce an intent leaking mechanism where we linearly decrease the proportion of the dominating intent and linearly increase the proportion assigned to another intent until eventually it dominates. Unlike the aforementioned two intent change environments, in this setup, intents change only a little bit after each interaction (time step) and never change abruptly. To model the intent leaking environment, instead of an abrupt changing point, the sampling probability of the dominating intent linearly "leaks" to another intent at each time step. To achieve this, we set a leaking rate η for this environment where $P_{t+1}(\mathcal{I} = i) = P_t(\mathcal{I} = i) - \eta$ and $P_{t+1}(\mathcal{I} = i + 1) = P_t(\mathcal{I} = i + 1) + \eta$.

Abrupt Swap: This is a special abrupt change environment where we only consider two intents (for simplicity, although we could have considered more intents) and intent changes occur repeatedly across these two intents. In this type of environment, we aim to investigate how OLTR algorithms adapt to an intent that has been already observed before in a different period in the historical click log. This environment simulates period intents associated with a query. This intent change mechanism is formally expressed as the abrupt intent change, except that the intent changes from i to $i + 1$ and then back to i.

Mixed Intents: In real Web data, it is often the case that multiple intents occur within the same time period. To test OLTR methods in presence of mixed intents, we uniformly sample intents for each query during training. Changes of intents occur throughout. For this, we uniformally set the sampling probability for all intents to 0.25 throughout training (all intents have equal chance to be sampled at any time step).

4 INTENT CHANGE EVALUATION MEASURES

To measure the performance of OLTR methods at any time step, we use the relevance judgments of the current intent and measure the nDCG@10 obtained by the learned rankers over all observed

queries during training. This evaluation is referred to as offline performance [15]. In order to investigate the impact of intent change, we also report the offline performance of rankers that have been trained under a stationary user intent environment. Note that, for measuring the offline performance, we are not using a held-out unseen test set to evaluate rankers. This is because intents are independent for different queries: this means there is no correlation for the same intent label between two different queries, and thus rankers' performance on unseen queries for an intent label are not necessarily aligned with the same intent label of queries observed during training. In addition to the offline performance, we also consider user experience during training. We evaluate this by averaging the nDCG@10 values of the displayed rankings during training. This is referred to as online performance [15].

The evaluation metrics mentioned above are standard OLTR metrics; however, they do not provide a clear picture of the impact of intent change on the OLTR methods' effectiveness. To address this, we devise two evaluation metrics for OLTR tailored to measure impact on effectiveness due to intent changes; these metrics are the immediate drop due to intent change (*nDCG_Drop@k*) and the average long-term loss in effectiveness (*nDCGΔ@k*).

Specifically, (normalised) *nDCG_Drop@k* evaluates the immediate sensitivity of OLTR methods to intent change:

$$nDCG\_Drop@k = \frac{max(nDCG@k(\theta_{t=c-1}) - nDCG@k(\theta_{t=c}), 0)}{nDCG@k(\theta_{t=c-1})} \quad (1)$$

where c is the intent change point, and θ_t is the ranker at time step t. This evaluation metric computes the difference in nDCG between that obtained by the ranker at the time step in which intent change occurred (point c) and that obtain at the time step immediately before the intent change. Intuitively, we expect that the nDCG score of the ranker will suddenly drop at the time step in which intent change occurs. Larger drops indicate that the ranker is affect by intent change more (larger *nDCG_Drop@k*). The max function is used to maintain the drop value not negative (in the unlikely case of a gain).

(normalised) *nDCGΔ@k*, instead, aims to measure the average loss of the ranker due to intent change:

$$nDCG\Delta@k = \frac{1}{T}\sum_{t\geq c}^{T} \frac{max(nDCG@k(\theta_t^{(i)}) - nDCG@k(\theta_t), 0)}{nDCG@k(\theta_t^{(i)})} \quad (2)$$

where i is the intent after intent change, and $\theta_t^{(i)}$ is the ranker at time step t trained under the stationary intent i (i.e. a ranker that has undergone as many iterations but that has only observed clicks with respect to intent i, that is, without experiencing intent change). The nDCG value of $\theta_t^{(i)}$ can be considered as the skyline of the nDCG value of θ_t at time t, since $\theta^{(i)}$ has never been affected by intent change. The max function keeps losses not negative. The final score represents the average normalized loss across all time steps. It is important to note that, being a loss, the smaller the value of *nDCGΔ@k*, the better.

nDCGΔ@k is based on the following intuition: After intent change occurs, the distribution of relevant documents for each query changes suddenly, thus the offline performance of the current ranker ($nDCG@k(\theta_t)$) is likely to drop suddenly. This drop however will not occur for the ranker that is trained with a fixed intent (stationary). If an OLTR method can rapidly adapt to changes intents, then $nDCG@k(\theta_t)$ should increase after each time step subsequent to an intent change, and eventually match $nDCG@k(\theta_t^{(i)})$ – in other words, losses should approach zero.

5 COMMON EXPERIMENT SETTINGS

OLTR methods. We investigate the impact intent change has on OLTR using three representative OLTR methods. The first method we consider is *Dual Bandit Gradient Descent (DBGD)* [18]. This is a popular OLTR method that uses interleaving for online evaluation. For this method, we use the same parameters settings reported in previous work [29], where the step size is set to $\delta = 1$ and the learning rate to $\alpha = 0.01$. The second baseline is the recently proposed *Counterfactual Online Learning to Rank (COLTR)* [43]. This method replaces online evaluation with counterfactual evaluation, delivering a large improvement in effectiveness over DBGD. For COLTR we use the best parameter settings reported in the original paper [43]. The third method is *Pairwise Differentiable Gradient Descent (PDGD)* [27] which is currently the state-of-the-art OLTR algorithm. For PDGD, we set the learning rate to $\alpha = 0.1$. Except for RQ4, we use these OLTR methods to train a linear ranker. While these methods have been thoroughly compared in previous OLTR work [14, 15, 18, 27, 29, 35, 36, 38, 43], they have never been studied with respect to non-stationary environments (intent change).

Intent changes. In Section 3.3 we have defined five different non-stationary intent environments. Unless indicated differently, we set the intent change environments as follows. Abrupt change and abrupt swap are characterised by the sampling probability for the current intent i, which is set to $P_{t<c}(\mathcal{I} = i) = 1$, and that of other intents, which is set to $P_{t<c}(\mathcal{I} \neq i) = 0$. For smooth abrupt change, we set intent sampling probability for the dominating intent i to $P_{t<c}(\mathcal{I} = i) = 0.7$ and for the other intents to $P_{t<c}(\mathcal{I} \neq i) = 0.1$ before the change in intent from i to $i+1$ occurs (time step c). After the dominating intent changes to $i+1$, we set $P_{t\geq c}(\mathcal{I} = i+1) = 0.7$ and $P_{t\geq c}(\mathcal{I} \neq i+1) = 0.1$. Intent leaking is characterised by the leaking rate η, which we set to $0.6/50,000$, i.e., 60% of the current intent will be leaked to the next intent over 50,000 impressions. For mixed intents, we uniformally sample intents with a 0.25 probability throughout the OLTR process.

For all intent change configurations we consider four intents. Unless differently indicated, we set the duration of an intent to 50,000 consecutive query interactions (time steps). This configuration allows OLTR methods to converge at a stable level before any intent change occurs.

Runs repetition and statistical analysis. All experiments are repeated 25 times with different random seeds. Results are averaged and a paired, two-tailed Student's t-test is used for significance testing.

6 RESULTS AND ANALYSIS

6.1 OLTR Sensitivity to intent change

To answer RQ1, we use the abrupt intent change setting, as we are interested in how the methods considered in this study (DBGD, COLTR, PDGD) perform under extreme intent change conditions.

Figure 1 compares the learning curves for offline nDCG@10 for three different click settings. When the perfect click setting is used (no click noise), there are sudden drops in effectiveness

Table 3: *nDCG_Drop*@10 for the considered OLTR methods: the lower the loss value, the better. Three intent change points are evaluated. Significant losses for PDGD over other methods are indicated by † ($p < 0.01$).

	Method	c_1	c_2	c_3
Perfect	DBGD	0.035	0.027	0.025
	COLTR	0.035	0.028	0.025
	PDGD	$0.070^{\dagger}$	$0.067^{\dagger}$	$0.049^{\dagger}$
Navig.	DBGD	0.034	0.042	0.023
	COLTR	0.032	0.024	0.022
	PDGD	$0.062^{\dagger}$	$0.073^{\dagger}$	$0.043^{\dagger}$
Inform.	DBGD	0.034	0.034	0.016
	COLTR	0.029	0.027	0.018
	PDGD	$0.039^{\dagger}$	$0.038^{\dagger}$	$0.036^{\dagger}$

associated to each intent change point. Rankers then gradually adapt to the new intent (past the abrupt change). This behaviour is observed for every OLTR method. However, when the navigational and informational click settings is used, this trend becomes less stark for PDGD and largely vanishes for DBGD and COLTR.

The *nDCG_Drop*@10 values in Table 3 further confirm these observations. Values for PDGD are much higher than those for BDGD and COLTR, across all click settings.

These observations suggest that PDGD is affected by intent change more than the other methods. This is because PDGD can quickly and very effectively train the ranker to fit a specific intent (as suggested by the high nDCG@10 values). Thus, when the intent changes, the ranker learnt with PDGD does not generalize well to the new intent, exhibiting a large drop in performance.

Nevertheless, even if PDGD is affected by intent change the most, its performance still remains higher than that of the other methods also in presence of intent change (although differences between PDGC and COLTR are less remarkable for navigational and informational settings). In fact, while PDGD almost always outperforms DBGD and COLTR, COLTR is more competitive under noisy click settings. These results are in agreement with the findings from Zhuang and Zuccon [43]. With this respect, Table 4 reports the online nDCG@10 scores of the methods within each intent period. PDGD significantly outperforms the other methods, indicating it provides the best user experience. Surprisingly, although COLTR has much better offline nDCG curves than DBGD when noisy clicks are used, the online performance is worse.

In summary, for RQ1 we conclude that, when the perfect clicks setting is used, OLTR methods are more sensitive to intent changes than when noisy clicks are present. On the other hand, the state-of-the-art OLTR method, PDGD, is much more sensitive than the other methods, although it also has the best performance in terms of offline and online nDCG.

Given the positive results obtained with PDGD, for the next research questions, we only report results related to PDGD.

6.2 Impact of intent change patterns

To answer RQ2, we consider the *nDCG*Δ@10 values obtained by PDGD under the abrupt change, smooth abrupt change and intent leaking non-stationary intent environments. We also compare the learning curves obtained by the ranker trained using the current intent with that of the ranker trained by maintaining a fixed intents

Table 4: Online nDCG@10. Significant gains for PDGD over other methods are indicated by † ($p < 0.01$).

	Method	intent1	intent2	intent3	intent4
Perfect	DBGD	0.280	0.303	0.315	0.318
	COLTR	0.329	0.330	0.337	0.339
	PDGD	$0.354^{\dagger}$	$0.357^{\dagger}$	$0.361^{\dagger}$	$0.363^{\dagger}$
Navig.	DBGD	0.246	0.270	0.281	0.286
	COLTR	0.265	0.271	0.279	0.283
	PDGD	$0.306^{\dagger}$	$0.317^{\dagger}$	$0.322^{\dagger}$	$0.325^{\dagger}$
Inform.	DBGD	0.226	0.260	0.271	0.276
	COLTR	0.226	0.239	0.247	0.252
	PDGD	$0.322^{\dagger}$	$0.314^{\dagger}$	$0.328^{\dagger}$	$0.332^{\dagger}$

throughout all iterations performed so far. The latter is a ranker that has never been exposed to changes on intent. For fair comparison, we use the same random seed for each runs, so that the queries the rankers observe are the same.

Figure 2 compares the offline nDCG learning curves across different intent change environments and click settings. For rankers trained with fixed intents, we only plot the curve obtained within the intent period their intent refers to.

From the figure, we observe that the drop in performance at each intent change point is less abrupt when more noisy clicks are present: this is valid across all intent change environments. This can be explained as follows: noisy clicks are likely to make PDGD fitting a particular intent less well, but at the same time providing higher generalizability across intents. This finding aligns with the results of Section 6.1. In addition, we observe that the size of the drops is smaller when the smooth abrupt change environment is used; and no obvious drop is observed for intent leaking. This finding is expected: in both settings all intents have a chance of being selected at any time, and thus the change in click pattern that occurs at a change point is less sudden.

Although the learning curves always increase after an intent change, rankers trained in the intent change environments always perform worse than rankers trained with fixed intents.

In summary, for RQ2, we found that for perfect clicks and abrupt intent change, the effect of intent change is substantial (large drops of performance). However, this effect is less remarked for noisy click settings and smoother intent change environments.

6.3 Adaptation to new intent re-occurrences

To answer RQ3, we consider experiments performed using the abrupt swap environment. We aim to investigate how well PDGD adapts to a newly occurring intent given enough training steps, and how well it adapts to an intent it has observed before.

We consider a long-term intent swap and a short-term intent swap. For the long-term intent swap, we change the intent changes from intent1 in the first period to intent2 in the second and then again to intent1 in the third period. To ensure that the ranker coverages after each change, we set the number of iterations between two change points to 500,000.

For the short-term intent swap, we again switch intent between intent1 and intent2, but we repeat this change for five times, each time after 50,000 iterations. This also allows us to investigate how PDGD performs when frequent periodic intent changes occur.

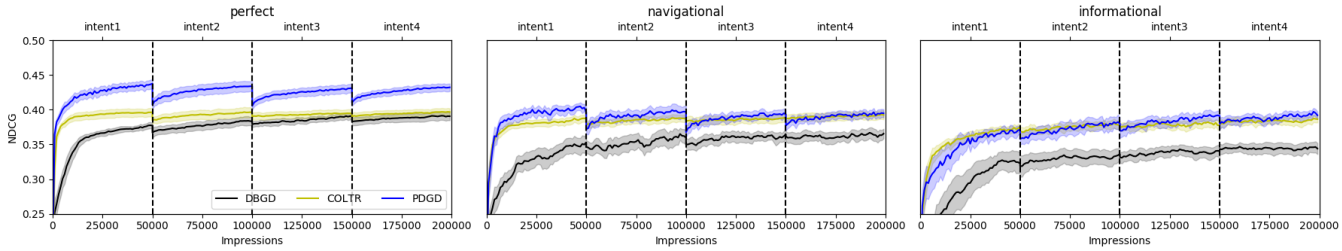

Figure 1: Offline nDCG learning curves in the abrupt change intent environment and for different click settings.

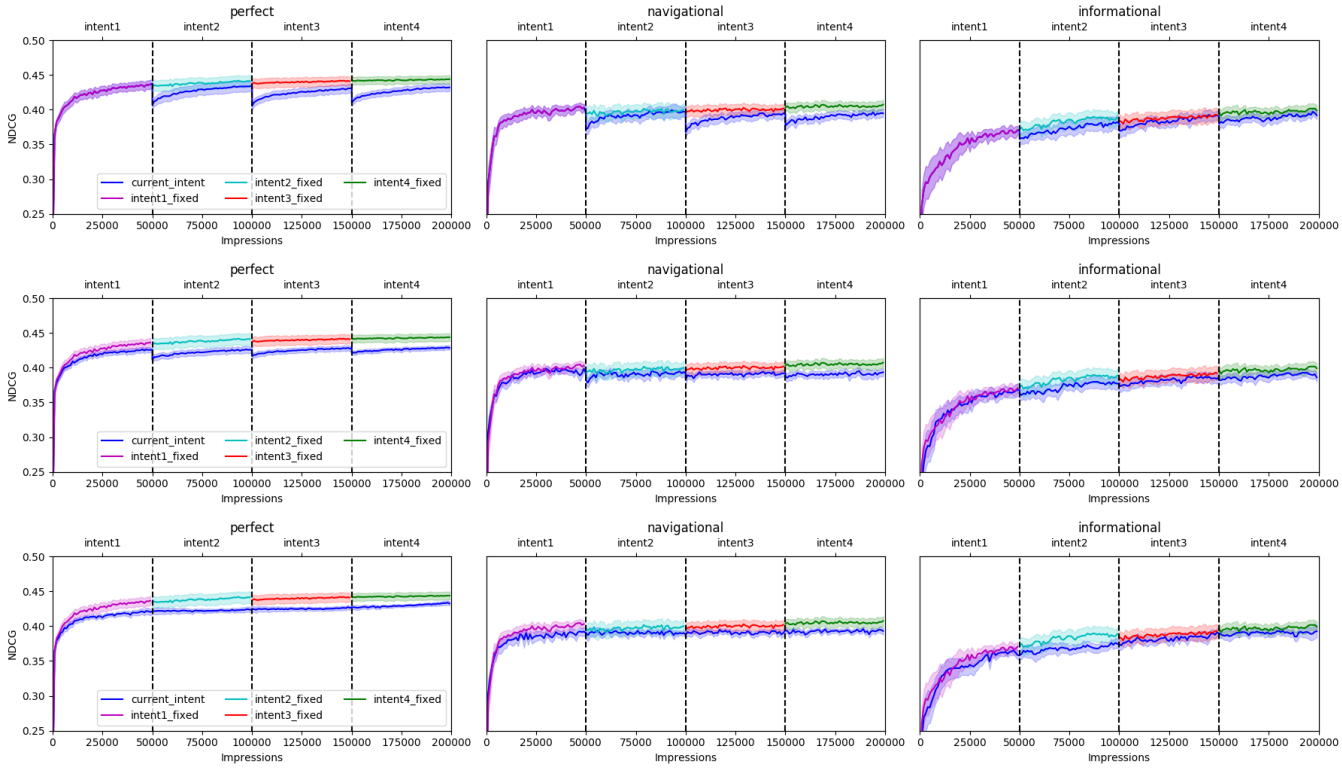

Figure 2: Offline nDCG@10 for PDGD under fixed intents vs. different intent change environments: abrupt change (first row), smooth abrupt change (second row), and intent leaking (third row).

Figure 3 compares the learning curves for the long-term swap experiments. With navigational and informational clicks, PDGD fails to adapt and reach the same performance as the ranker trained with fixed intent. However, with perfect clicks, PDGD can adapt well and converge to the same level as the ranker trained with fixed intent for the intent that it observed first (i.e., intent1).

A similar finding is present also in the short-term experiments, shown in Figure 4. PDGD appears to perform better for the intent that is first observed at the start of training. To confirm this, we calculate $nDCG\Delta@10$ for each intent period in the short-term runs: the results are reported in Table 5. From the table, it is clear that the performance of PDGD adapting to intent2 is always worse than that obtained when adapting to intent1. It is important to note that these results still hold after experiments are ran such that intent2 is the first intent and intent1 is the second: in this case, PDGD performs better for intent2.

In summary, for RQ3, we find that PDGD adapts better for the first training intent. When given enough perfect clicks, despite the intent changes, PDGD can even achieve the same performance of the ranker trained with the fixed intent. However, PDGD adapts worse to multiple repetition of the second observed intent.

6.4 Neural vs. Simpler Rankers

A common perception with neural rankers is that they require more training samples than linear rankers to perform well because they have more parameters needing to be trained. Thus, in the context of non-stationary environments like that of our intent change framework, we would expect neural rankers to adapt worse than simpler linear rankers to intent change.

To answer RQ4, we use PDGD to train a neural ranker with two hidden layers (64 hidden units), following the original PDGD work [27]. The offline nDCG learning curve of the neural and linear rankers are compared in Figure 5. As training begins (intent 1), the linear ranker learns faster than the neural ranker, thus matching our expectation above that neural rankers require more data for training. However, for intents past the first, and regardless of changes in

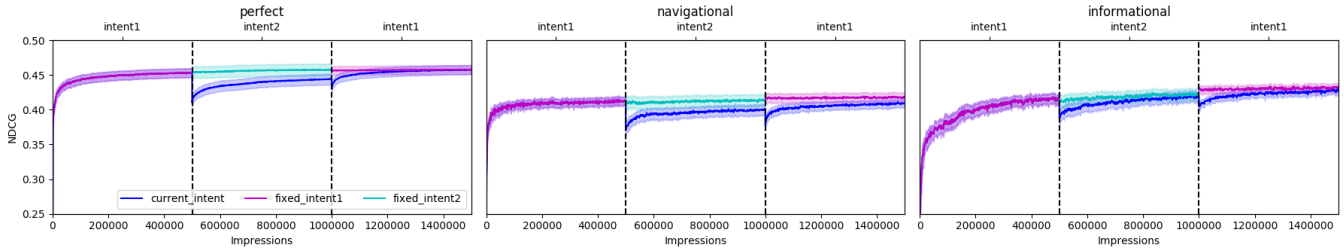

Figure 3: Long-term offline NDCG for PDGD when abrupt intent swap is considered and across different click settings.

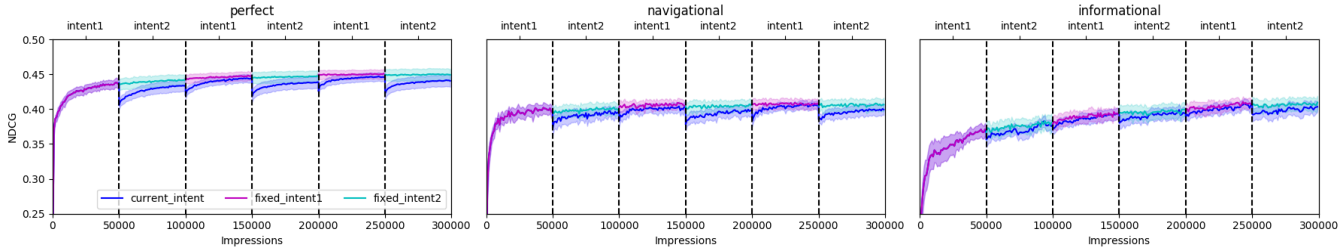

Figure 4: Short-term offline NDCG for PDGD when abrupt intent swap is considered and across different click settings.

intents, the neural ranker converges at a higher score than the linear ranker when the perfect click setting is used. When other click settings are used, the neural ranker converges at the same level of the linear ranker. Figure 6 further compares the same neural ranker trained when intent change occurs vs. when intents have been fixed throughout all iterations. This result shows that, except for the perfect click setting, the neural ranker is only minimally affected by intent change.

In summary, we found that neural rankers are less affected by intent changes than simpler linear rankers, and they adapt to intent change best when no click noise is present.

6.5 Per-Intent vs. Combination Learning

To answer RQ5, we create a mixed intent environment with four randomly sampled intents. We use PDGD to train a single ranker and four intent-specific rankers learnt separately according to the intents sampled at each time step. Note that the number of training examples for the single ranker is four times larger than that of the intent-specific rankers: this is because only one intent-specific ranker has been trained at each time step. When reporting offline nDCG at each time step, we assume that we know which intent occurred at each time step and we compare the performance of the single ranker against the intent-specific ranker for that time step.

Figure 7 compares the offline nDCG learning curves during 2 million iterations. The single ranker outperforms the intent-specific rankers at the early stage of training across all click settings. However, for perfect and navigational click settings, intent-specific rankers surpass the single ranker very quickly; the speed becomes much slower when noisy clicks are used.

Table 6 further reports the online nDCG scores of the single ranker and intent-specific rankers. The user experience aligns with the offline nDCG learning curves shown in Figure 7, except for navigational clicks: although the intent-specific rankers outperform the single ranker in terms of offline nDCG, no statistical differences are found for online nDCG.

Table 5: $nDCG\Delta@10$ of PDGD under short-term abrupt swap.

	intent1	intent2	intent1	intent2	intent1	intent2
Perfect	0.0	0.043	0.030	0.044	0.031	0.044
Navig.	0.0	0.046	0.031	0.046	0.028	0.045
Inform.	0.0	0.035	0.034	0.039	0.031	0.045

Table 6: Online nDCG@10 of PDGD for single ranker vs. intent-specific rankers. †: significant gains ($p < 0.01$)

Click setting	single	intent-specific
Perfect	0.371	$0.375^{\dagger}$
Navigational	0.334	0.332
Informational	$0.346^{\dagger}$	0.337

In summary, for RQ5, we found that, given limited iterations and when the perfect click setting is used, OLTR performs better with rankers trained specifically on different intents. However, when the informational click setting is used, the ranker trained with mixed intents performs best. In our experiments we assumed a perfect intent predictor was available; this is unlike to be the case in reality and thus the advantage of intent-specific rankers is likely limited.

7 CONCLUSIONS

Adapting to changes in the intents that underlay a user's query (non-stationary relevance assessments) is one of the key benefits put forward by online LTR. However, the extent to which OLTR methods adapt to such intent changes, and whether methods differ in how well they adapt, has not been investigated so far. In this paper we perform the first, thorough investigation of the impact of intent change on OLTR performance.

Current OLTR studies use static offline LTR datasets to synthetically generate user click data and do not have an explicit notion of intents and non-stationary relevance. In this way, they either consider truly stationary environments (as is for those datasets

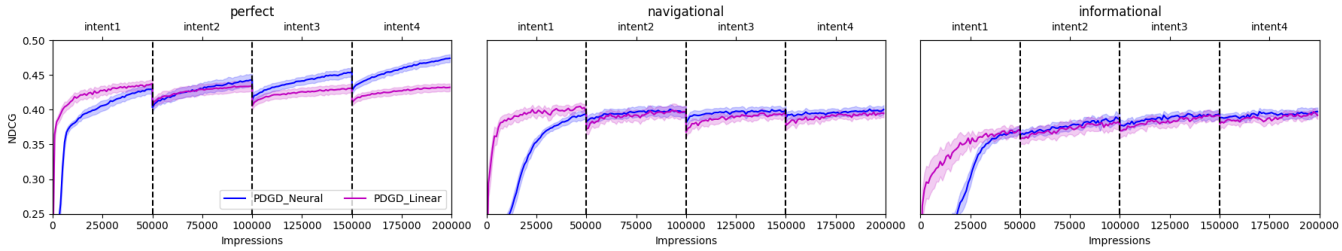

Figure 5: Offline nDCG learning curves for neural ranker vs. linear ranker under PDGD.

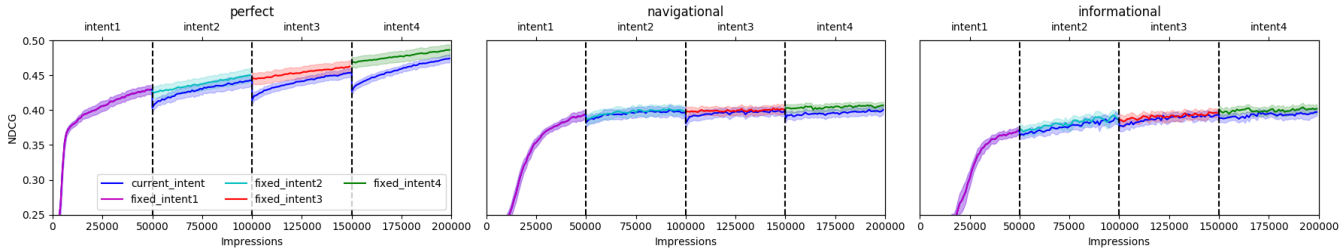

Figure 6: Offline nDCG learning curves for neural ranker under PDGD.

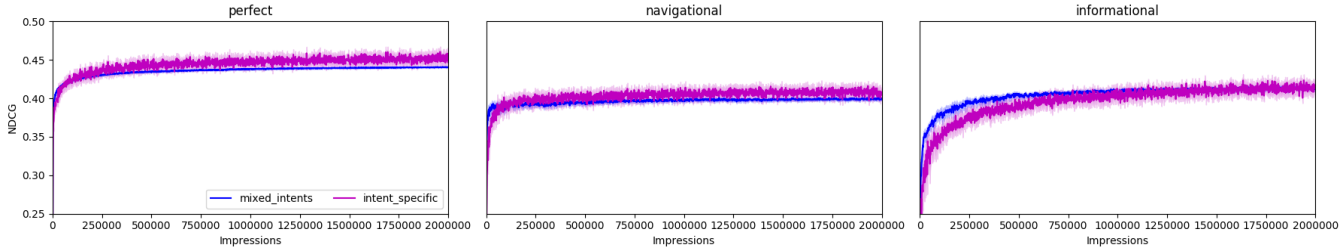

Figure 7: Offline nDCG learning curves for PDGD under mixed intent vs intent-aware.

based on TREC data [32]), or consider environments that may have exhibited non-stationary relevance, but this has not explicitly captured and modelled (as is for current web based datasets like iStella or Yahoo [5, 11]). Thus, these datasets cannot be used to simulate and study the effect of intent change on OLTR methods.

To study OLTR methods under synthetic intent change environments, we construct an OLTR dataset in which intents are explicitly associated to queries and relevance labels are provided with respect to intents. We then design 5 different synthetic non-stationary intent change environments so to investigate the performance of OLTR methods under extreme conditions and real-world scenarios. We further propose two novel evaluation metrics to better quantify the effect of intent change: *nDCG_Drop@k* for measuring how sensitive OLTR methods are to intent change, and *nDCGΔ@k* for measuring the average loss of rankers due to intent change.

Our experimental results demonstrate that OLTR methods have the ability to adapt to intent changes and non-stationary environments; however, their performance is affected. In particular, we find that OLTR methods exhibits different levels of sensitivity to intent change. The state-of-the-art PDGD is the most sensitive to intent change, but it also performs best compared to the other methods, even after the intent has changed. The investigation of different synthetic intent change environments shows that the sensitivity of OLTR varies across different non-stationary scenarios. OLTR methods usually cannot adapt well and rapidly to a new, fresh intent, thus not matching the performance the ranker would have obtained if trained with that intent since the start.

Our study is not without limitations. Because of the limited size of the underlying collection we used in our experiments, the TREC diversity task, we could only rely on 200 unique queries: a limited number compared with modern LTR datasets [5, 11], but larger than the initial datasets used in LTR. However, there are currently no datasets with intent change for OLTR and we thus believe our work provides the first stepping stone for studying this problem.

On the other hand, because the intents of different queries are independent, we could only evaluate rankers on queries that were observed during training. As a result, rankers may overfit training queries, and it is unclear how OLTR would adapt to intent change for unseen queries. Similarly, more complex intent change environments could be considered, e.g. simulated through Markov chains. Although worthwhile for future work, we believe our current setting strikes the balance between being grounded on previous work (SDBN used throughout OLTR literature) and allowing for the careful control of the impact of intent change on OLTR. In light of the results reported in this paper, these limitations provide the motivation for creating large scale LTR collections that explicitly encompass intent information.

Acknowledgement. Dr Guido Zuccon is the recipient of Australian Research Council DECRA Research Fellowship DE180101579.

REFERENCES

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 5–14.

[2] Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 997–1000.

[3] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[4] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A click sequence model for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 45–54.

[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*. 1–24.

[6] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. 1–10.

[7] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services* 7, 3 (2015), 1–115.

[8] Charles L Clarke, Nick Craswell, and Ellen M Voorhees. 2012. *Overview of the TREC 2012 web track*. Technical Report. NATIONAL INST OF STANDARDS AND TECHNOLOGY GAITHERSBURG MD.

[9] Gordon V Cormack, Mark D Smucker, and Charles LA Clarke. 2011. Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval* 14, 5 (2011), 441–465.

[10] Na Dai, Milad Shokouhi, and Brian D Davison. 2011. Learning to rank for freshness and relevance. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 95–104.

[11] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 1–31.

[12] Maik Fröbe, Janek Bevendorff, Jan Heinrich Reimer, Martin Potthast, and Matthias Hagen. 2020. Sampling Bias Due to Near-Duplicates in Learning to Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1997–2000.

[13] Zhiwei Guan and Edward Cutrell. 2007. An eye tracking study of the effect of target rank on web search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 417–420.

[14] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing Historical Interaction data for faster Online Learning to Rank for IR. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 183–192.

[15] Katja Hofmann, Shimon Whiteson, and Maarten De Rijke. 2011. Balancing exploration and exploitation in learning to rank online. In *European Conference on Information Retrieval*. Springer, 251–263.

[16] Rolf Jagerman and Maarten de Rijke. 2020. Accelerated Convergence for Counterfactual Learning to Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 469–478.

[17] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 15–24.

[18] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 133–142.

[19] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.

[20] Rosie Jones and Fernando Diaz. 2007. Temporal profiles of queries. *ACM Transactions on Information Systems (TOIS)* 25, 3 (2007), 14–es.

[21] Eugene Kharitonov. 2019. Federated online learning to rank with evolution strategies. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 249–257.

[22] Damien Lefortier, Pavel Serdyukov, and Maarten De Rijke. 2014. Online exploration for detecting shifts in fresh intent. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 589–598.

[23] Chang Li, Haoyun Feng, and Maarten de Rijke. 2020. Cascading Hybrid Bandits: Online Learning to Rank for Relevance and Diversity. In *Fourteenth ACM Conference on Recommender Systems*. 33–42.

[24] Tie-Yan Liu. 2011. *Learning to rank for information retrieval*. Springer Science & Business Media.

[25] Donald Metzler, Rosie Jones, Fuchun Peng, and Ruiqiang Zhang. 2009. Improving search relevance for implicitly temporal queries. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 700–701.

[26] Harrie Oosterhuis and Maarten de Rijke. 2017. Sensitive and scalable online evaluation with theoretical guarantees. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 77–86.

[27] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1293–1302.

[28] Harrie Oosterhuis and Maarten de Rijke. 2019. Optimizing Ranking Models in an Online Setting. In *European Conference on Information Retrieval*. Springer, 382–396.

[29] Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke. 2016. Probabilistic multileave gradient descent. In *European Conference on Information Retrieval*. Springer, 661–668.

[30] Bing Pan, Helene Hembrooke, Thorsten Joachims, Lori Lorigo, Geri Gay, and Laura Granka. 2007. In Google we trust: Users' decisions on rank, position, and relevance. *Journal of computer-mediated communication* 12, 3 (2007), 801–823.

[31] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[32] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[33] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*. 784–791.

[34] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2011. Intent-aware search result diversification. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 595–604.

[35] Anne Schuth, Robert-Jan Bruintjes, Fritjof Buüttner, Joost van Doorn, Carla Groenland, Harrie Oosterhuis, Cong-Nguyen Tran, Bas Veeling, Jos van der Velde, Roger Wechsler, et al. 2015. Probabilistic multileave for online retrieval evaluation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 955–958.

[36] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 457–466.

[37] Milad Shokouhi. 2011. Detecting seasonal queries by time-series analysis. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 1171–1172.

[38] Huazheng Wang, Sonwoo Kim, Eric McCord-Snook, Qingyun Wu, and Hongning Wang. 2019. Variance Reduction in Gradient Exploration for Online Learning to Rank. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*.

[39] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 115–124.

[40] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1201–1208.

[41] Ruiqiang Zhang, Yuki Konda, Anlei Dong, Pranam Kolari, Yi Chang, and Zhaohui Zheng. 2010. Learning recurrent event queries for web search. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. 1129–1139.

[42] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. 2014. Learning for search result diversification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 293–302.

[43] Shengyao Zhuang and Guido Zuccon. 2020. Counterfactual Online Learning to Rank. In *European Conference on Information Retrieval*. Springer, 415–430.