Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study

Shuyi Wang(✉)[0000−0002−4467−5574], Shengyao Zhuang[0000−0002−6711−0955], and Guido Zuccon[0000−0003−0271−5563]

The University of Queensland, St Lucia, Australia
{shuyi.wang, s.zhuang, g.zuccon}@uq.edu.au

Abstract. Online Learning to Rank (OLTR) optimizes ranking models using implicit users' feedback, such as clicks, directly manipulating search engine results in production. This process requires OLTR methods to collect user queries and clicks; current methods are not suited to situations in which users want to maintain their privacy, i.e. not sharing data, queries and clicks.
Recently, the federated OLTR with evolution strategies (FOLtR-ES) method has been proposed to provide a solution that can meet a number of users' privacy requirements. Specifically, this method exploits the federated learning framework and ϵ-local differential privacy. However, the original research study that introduced this method only evaluated it on a small Learning to Rank (LTR) dataset and with no conformity with respect to current OLTR evaluation practice. It further did not explore specific parameters of the method, such as the number of clients involved in the federated learning process, and did not compare FOLtR-ES with the current state-of-the-art OLTR method. This paper aims to remedy to this gap.
Our findings question whether FOLtR-ES is a mature method that can be considered in practice: its effectiveness largely varies across datasets, click types, ranker types and settings. Its performance is also far from that of current state-of-the-art OLTR, questioning whether the maintained of privacy guaranteed by FOLtR-ES is not achieved by seriously undermining search effectiveness and user experience.

Keywords: Online learning to rank · Federated machine learning · Differential privacy

1 Introduction

Online learning to rank (OLTR) exploits users queries and interactions with search engine result pages (SERPs) to iteratively train and update a ranker in production [16]. In particular, OLTR relies on implicit user feedback from interactions on SERPs, e.g., clicks, rather than editorial relevance labels. Several methods for OLTR exist that attempt to address the specific challenges of online learning [7, 9, 16, 30]: from making sense of the implicit feedback, to exploring the space of feature weights, accounting for biases in the click signal, reducing the impact of the online learning process on user experience, among others.

An aspect that has not been received wide attention in OLTR is how the privacy of users could be guaranteed. Current OLTR methods in fact assume that a central server collects all queries and interactions of all users of the search system, and it is this central server that is responsible for the indexing of the collection, the training of the ranker and the production of the SERPs. A recent work by Kharitonov, however, has attempted to provide a mechanism for OLTR that preserves the privacy of users [10]. The method, called FOLtR-ES, relies on the federated learning paradigm [27],

in which data (collection, queries, interactions) is maintained at each client's side along with a copy of the ranker, and updates to the rankers that are learned from the interaction on the client side are shared to the central server, which is responsible for aggregating the update signal from clients and propagate the aggregated ranker update. In this specific case, all users observe and act on the same feature space; each user however retains control of their own data, which includes the collection, the queries and the interactions. FOLtR-ES uses evolutionary strategies akin to those in genetic algorithms to make client rankers explore the feature space, and a parametric privacy preserving mechanism to further anonymise the feedback signal that is shared by clients to the central server.

This paper aims to replicate and then reproduce the experiments from the original work of Kharitonov [10], investigating the effect different configurations of that federated OLTR method have on effectiveness and user experience, extending and generalising its evaluation to different settings commonly used in OLTR and to different collections. Specifically, we address the following research questions:

- **RQ1:** *Does the performance of FOLtR-ES generalise beyond the MQ2007/2008 datasets?* The original method was only evaluated using MQ2007/2008 [18], while current OLTR practice is to use larger datasets that are feature richer and that contain typical web results.
- **RQ2:** *How does the number of clients involved in FOLtR-ES affect its performance?* FOLtR-ES was previously evaluated using a set number of clients involved in the federated OLTR process ($n = 2,000$), and it was left unclear whether considering more or less client would impact performance.
- **RQ3:** *How does FOLtR-ES compare with current state-of-the-art OLTR methods?* Compared to OLTR methods, FOLtR-ES preserves user privacy, but it is unclear to what expense in terms of search performance: the original work compared FOLtR-ES to rankers in non-federated settings, but the rankers used in there were not the current state-of-the-art in OLTR.
- **RQ4:** *How does FOLtR-ES performance generalise to the evaluation settings commonly used for OLTR evaluation, i.e. measuring offline and online performance, with respect to nDCG and with relevance labels?* The original evaluation of FOLtR-ES considered an unusual setting for OLTR, consisting of using MaxRR [19] as evaluation measure in place of nDCG, computed on simulated clicks instead of on relevance labels.

The results of our empirical investigation of FOLtR-ES help understanding the specific settings in which this technique works, and the trade-offs between user privacy and search performance (in terms of effectiveness and user experience). They also unveil that more work is require to devise effective federated methods for OLTR that can guarantee some degree of user privacy without sensibly compromising search performance.

2 Federated OLTR with Evolution Strategies

We provide a brief overview of the FOLtR-ES method, which extends online LTR to federated learning; this is done by exploiting evolution strategies optimization, a widely used paradigm in Reinforcement Learning. The FOLtR-ES method consists of three parts. First, it casts the ranking problem into the federated learning optimization setting. Second, it uses evolution strategies to estimate gradients of the rankers. Finally, it introduces a privatization procedure to further protect users' privacy.

2.1 Federated Learning Optimization Setting

The federated learning optimization setting consists in turn of several steps, and assumes the presence of a central server and a number of distributed clients. First, a client downloads the most recently updated ranker from the server. Afterwards, the client observes B user interactions (search queries and examination of SERPs) which are served by the client's ranker. The performance metrics of these interactions are averaged by the client and a privatized message is sent to the centralized server. After receiving messages from N clients, the server combines them to estimate a single gradient g and performs an optimization step to update the current ranker. Finally, the clients download the newly updated ranker from the server.

2.2 Gradient Estimation

The method assumes that the ranker comes from a parametric family indexed by vector $\theta \in R^n$. Each time a user u has an interaction a, the ranking quality is measured; this is denoted as f. The goal of optimization is to find the vector θ^* that can maximize the mean of the metric f across all interactions a from all users u:

$$\theta^* = \arg\max_{\theta} F(\theta) = \arg\max_{\theta} \mathbb{E}_u \mathbb{E}_{a|u,\theta} f(a; \theta, u) \tag{1}$$

Using Evolution Strategies (ES) [20], FOLtR-ES considers a population of parameter vectors which follow the distribution with a density function $p_\phi(\theta)$. The objective aims to find the distribution parameter ϕ that can maximize the expectation of the metric across the population:

$$\mathbb{E}_{\theta \sim p_\phi(\theta)} \left[F(\theta) \right] \tag{2}$$

The gradient g of the expectation of the metric across the population (Equation 2) is obtained in a manner similar to REINFORCE [24]:

$$\begin{aligned} g &= \nabla_\phi \mathbb{E}_\theta[F(\theta)] = \nabla_\phi \int_\theta p_\phi(\theta) F(\theta) d\theta = \int_\theta F(\theta) \nabla_\phi p_\phi(\theta) d\theta = \\ &= \int_\theta F(\theta) p_\phi(\theta) \left(\nabla_\phi \log p_\phi(\theta) \right) d\theta = \mathbb{E}_\theta \left[F(\theta) \cdot \nabla_\phi \log p_\phi(\theta) \right] \end{aligned} \tag{3}$$

Following the Evolution Strategies method, FOLtR-ES instantiates the population distribution $p_\phi(\theta)$ as an isotropic multivariate Gaussian distribution with mean ϕ and fixed diagonal covariance matrix $\sigma^2 I$. Thus a simple form of gradient estimation is denoted as:

$$g = \mathbb{E}_{\theta \sim p_\phi(\theta)} \left[F(\theta) \cdot \frac{1}{\sigma^2} (\theta - \phi) \right] \tag{4}$$

Based on the federated learning optimization setting, θ is sampled independently on the client side. Combined with the definition of $F(\theta)$ in Equation 1, the gradient can be obtained as:

$$g = \mathbb{E}_u \mathbb{E}_{\theta \sim p_\phi(\theta)} \left[\left(\mathbb{E}_{a|u,\theta} f(a; \theta, u) \right) \cdot \frac{1}{\sigma^2} (\theta - \phi) \right] \tag{5}$$

To obtain the estimate $\hat{g}$ of g from Equation 5, $\hat{g} \approx g$, the following steps are followed: (i) each client u randomly generates a pseudo-random seed s and uses the seed to sample a perturbed model

$\theta_s \sim \mathbb{N}\left(\phi, \sigma^2 I\right)$, (ii) the average of metric f over B interactions is used to estimate the expected loss $\hat{f} \approx \mathbb{E}_{a|u,\theta_s} f(a; \theta_s, u)$ from Equation 5, (iii) each client communicates the message tuple $(s, \hat{f})$ to the server, (iv) the centralized server computes the estimate $\hat{g}$ of Equation 5 according to all message sent from the N clients.

To reduce the variance of the gradient estimates, means of antithetic variates are used in FOLtR-ES: this is a common ES trick [20]. The algorithm of the gradient estimation follows the standard ES practice, except that the random seeds are sampled at the client side.

2.3 Privatization Procedure

To ensure that the clients' privacy is fully protected, in addition to the federated learning setting, FOLtR-ES also proposes a privatization procedure that introduces privatization noise in the communication between the clients and the server.

Assume that the metric used on the client side is discrete or can be discretized if continuous. Then, the metric takes a finite number (n) of values, $f_0, f_1, ..., f_{n-1}$. For each time the client experiences an interaction, the true value of the metric is denoted as f_0 and the remaining $n-1$ values are different from f_0. When the privatization procedure is used, the true metric value f_0 is sent with probability p. Otherwise, with probability $1-p$, a randomly selected value $\hat{f}$ out of the remaining $n-1$ values is sent. To ensure the same optimization goal described in Section 2.2, FOLtR-ES assumes that the probability $p > 1/n$.

Unlike other federated learning methods, FOLtR-ES adopts a strict notion of ϵ-local differential privacy [10], in which the privacy is considered at the level of the client, rather than of the server. Through the privatization procedure, ϵ-local differential privacy is achieved, and the upper bound of ϵ is:

$$\epsilon \leq log \frac{p(n-1)}{1-p} \tag{6}$$

This means that, thanks to the privatization scheme, at least $log[p(m-1)/(1-p)]$-local differential privacy can be guaranteed. At the same time, any ϵ-local differential private mechanism also can obtain ϵ-differential privacy [3].

3 Experimental Settings

3.1 Datasets

The original work of Kharitonov [10] conducted experiments on the MQ2007 and MQ2008 learning to rank datasets [18], which are arguably small and outdated. In our work, we instead consider more recent and lager datasets: MSLR-WEB10k [18] and Yahoo! Webscope [1], which are commonly-used in offline and online learning to rank [6,7,16,30]. Compared to MQ2007/2008, both MSLR-WEB10k and Yahoo! use 5-level graded relevance judgements, ranging from 0 (not relevant) to 4 (perfectly relevant). Each dataset contains many more queries and corresponding candidate documents than MQ2007/2008: MSLR-WEB10k has 10,000 queries, with each query having 125 assessed documents on average, while Yahoo! has 29,921 queries with 709,877 documents. In addition, both datasets have much richer and numerous features. MSLR-WEB10k has 136 features and Yahoo! 700. For direct comparison with the original FOLtR-ES work, we also use MQ2007/2008.

Table 1: The three click model instantiations used for the MSLR-WEB10K and Yahoo! datasets.

	$p(click = 1\|R)$					$p(stop = 1\|R)$				
R	0	1	2	3	4	0	1	2	3	4
$perf$	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
nav	0.05	0.3	0.5	0.7	0.95	0.2	0.3	0.5	0.7	0.9
inf	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

3.2 Simulation

It is common practice in OLTR to use LTR datasets and simulate user interactions [6, 22]. This is because no public dataset with LTR features and clicks is available; in addition OLTR methods directly manipulate the rankings that have to be shown to users, so even if a public dataset with LTR features and clicks was to be available, this could not be used for OLTR. Thus, we simulate users and their reaction with the search results using labelled offline learning to rank datasets, akin to previous work [6, 22].

For the experiment, we follow the same method used by the original FOLtR-ES work. We sample B queries for each client randomly and use the local perturbed model to rank documents. The length for each ranking list is limited to 10 documents. After simulating users clicks, we record the quality metric for each interaction and perform the privatization procedure with probability p. Next, we send the averaged metric and pseudo-random seed to optimize the centralized ranker. Finally, each client receives the updated ranker.

For simulating users' clicks, we use the Cascade Click Model (CCM) [5], as in the original FOLtR-ES work. We run instances of CCM using the same click probabilities and stop probabilities for MSLR-WEB10K and Yahoo!. Under CCM, the users are assumed to examine a SERP from top to bottom. Each document is examined and clicked with click probability $P(click = 1|r)$, conditioned on the relevance label r. After a click occurs, the user stops with stop probability $P(stop = 1|r)$, or continues otherwise. It is common practice in OLTR to consider three instantiations of the CCM: a $perfect$ user with very reliable feedback, a $navigational$ user searching for reasonably relevant documents, and an $informational$ user with the noisiest feedback among three instantiations. Table 1 summarises the parameters of three click models. For simulating clicks for the MQ2007/2008, we use the same parameter settings from Table 1 in the original FOLtR-ES paper [10]: these are partially different from those used for MSLR-WEB10K and Yahoo! because relevance labels in these datasets are five-graded, while they are three-graded in MQ2007/2008.

3.3 Evaluation metric

For direct comparison with the original FOLtR-ES work, we use the reciprocal rank of the highest clicked result in each interaction (MaxRR [19]). This metric is computed on the clicks produced by the simulated users on the SERPs.

The evaluation setting above is unusual for OLTR. In RQ4, we also consider the more commonly used normalised Discounted Cumulative Gain (nDCG), as FOLtR-ES is designed to allow optimization based on any absolute measures of ranking quality. We thus record the nDCG@10 values from the relevance labels of the SERP displayed to users during interactions. This is referred to as online nDCG and the scores represent users' satisfaction [6]. We also record the nDCG@10 of the final learned ranker measured a heldout test set: this is refer to as offline nDCG.

3.4 FOLtR-ES and Comparison OLTR Methods

In all experiments, we adopt the same models and optimization steps used by Kharitonov [10], and rely on the well document implementation made publicly available by the author. The two ranking models used by FOLtR-ES are a linear ranker and a neural ranker with a single hidden layer of size 10. For optimization, we use Adam [11] with default parameters.

To study how well FOLtR-ES compares with current state-of-the-art OLTR (RQ3), we implemented the Pairwise Differentiable Gradient Descent (PDGD) [16]. Unlike many previous OLTR methods that are designed for linear models, PDGD also provides effective optimization for non-linear models such as neural rankers. During each interaction, a weighted differentiable pairwise loss is constructed in PDGD and the gradient is directly estimated by document pairs preferences inferred from user clicks. PDGD has been empirically found to be significantly better than traditional OLTR methods in terms of final convergence, learning speed and user experience during optimization, making PDGD the current state-of-the-art method for OLTR [7, 16, 30].

4 Results and Analysis

4.1 RQ1: Generalisation of FOLtR-ES performance beyond MQ2007/2008

For answering RQ1 we replicate the results obtained by Kharitonov [10] on the MQ2007 and MQ2008 datasets; we then reproduce the experiment on MSLR-WEB10k and Yahoo datasets, on which FOLtR-ES has not been yet investigated, and we compare the findings across datasets. For these experiments we use antithetic variates, set the number of interactions $B = 4$ and simulate 2,000 clients, use MaxRR as reward signal and for evaluation on clicked items.

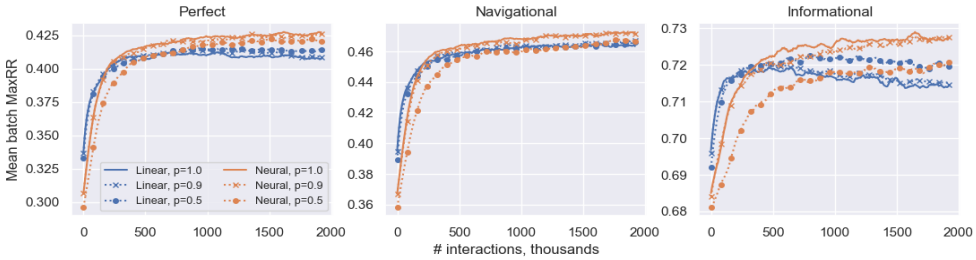
Figure 1a reports the results obtained by FOLtR-ES on the MQ2007 dataset[1] with respect to the three click models considered, various settings for the privatization parameter p, and the two FOLtR-ES methods (linear and neural). Our results fully replicate those of Kharitonov [10] and indicate the following findings: (1) FOLtR-ES allows for the iterative learning of effective rankers; (2) high values of p (lesser privacy) provide higher effectiveness; (3) the neural ranker is more effective than the linear ranker when $p \to 1$ (small to no privacy), while the linear model is equivalent, or better (for informational clicks) when $p = 0.5$.

However, not all these findings are applicable to the results obtained when considering MSLR-WEB10k and Yahoo!, which are displayed in Figures 1b and 1c. In particular, we observe that (1) the results for MSLR-WEB10k (and to a lesser extent also for Yahoo!) obtained with the informational click model are very unstable, and, regardless of the click model, FOLtR-ES requires more data than with MQ2007/2008 to arrive at a stable performance, when it does; (2) the neural ranker is less effective than the linear ranker, especially on MSLR-WEB10k. We believe these findings are due to the fact that query-document pairs in MSLR-WEB10k and Yahoo! are represented by a larger number of features than in MQ2007/2008. Thus, more data is required for effective training, especially for the neural model; we also note that FOLtR-ES is largely affected by noisy clicks in MSLR-WEB10k.

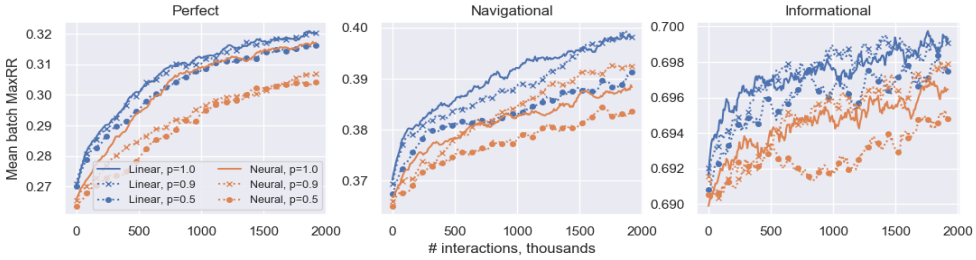
4.2 RQ2: Effect of number of clients on FOLtR-ES

To answer RQ2 we vary the number of clients involved in FOLtR-ES; we investigate the values {50, 1,000, 2,000}. Kharitonov [10] used 2,000 in the original experiments, and the impact of the number

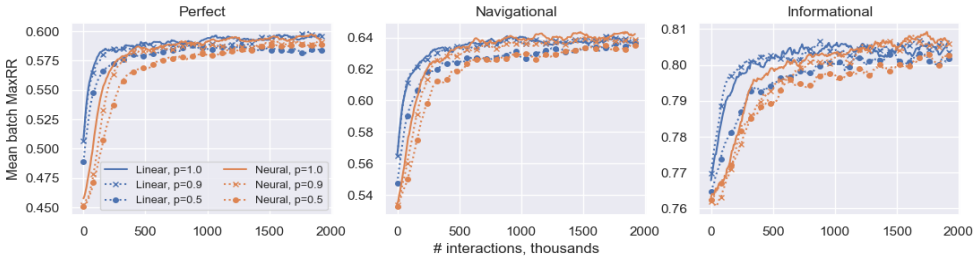
[1] Similar results were obtained for MQ2008 and are omitted for space reasons.



(a) Mean batch MaxRR for MQ2007.



(b) Mean batch MaxRR for MSLR-WEB10k.

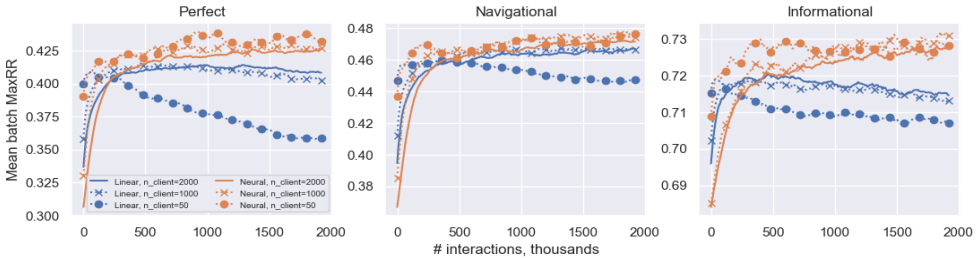


(c) Mean batch MaxRR for Yahoo!.

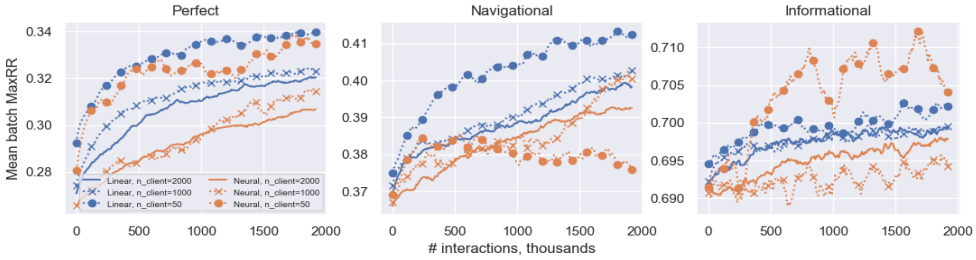
Fig. 1: Results for RQ1: performance of FOLtR-ES across datasets under three different click models (averaged across all dataset splits).

of clients has not been studied. To be able to fairly compare results across number of clients, we fixed the total number of ranker updates to 2,000,000; we also set $B = 4$ and $p = 0.9$. We perform these experiments on all three datasets considered in this paper, but we omit to report results for Yahoo! due to space limitations.

The results of these experiments are reported in Figure 2, and they are mixed. For MQ2007, the number of clients have little effect on the neural ranker used in FOLtR-ES, although when informational clicks are provided this ranker is less stable, although often more effective, if very few clients (50) are used. Having just 50 clients, instead, severally hits the performance of the linear ranker, when compared with 1,000 or 2,000 clients. The findings on MSLR-WEB10k, however, are different. In this dataset, a smaller number of clients (50), is generally better than larger numbers, both for linear and neural ranker. An exception to this is when considering navigational clicks: in this case the linear ranker obtains by far the best performance with a small number of clients, but



(a) Mean batch MaxRR for MQ2007.



(b) Mean batch MaxRR for MSLR-WEB10k.

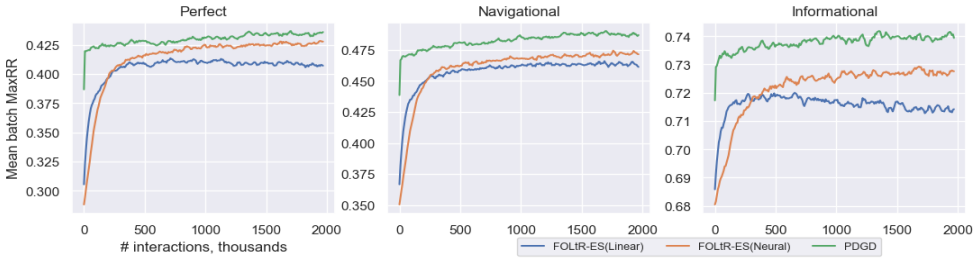
Fig. 2: Results for RQ2: performance of FOLtR-ES with respect to number of clients (averaged across all dataset splits).

the neural ranker obtains the worst performance. This suggest that the number of clients greatly affects FOLtR-ES: but trends are not consistent across click types and datasets.

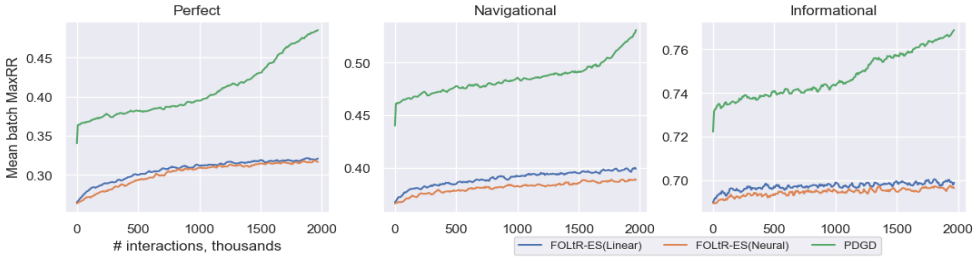
4.3 RQ3: Comparing FOLtR-ES to state-of-the-art OLTR methods

The original study of FOLtR-ES did not compared the method with non-federated OLTR approaches. To contextualise the performance of FOLtR-ES and to understand the trade-off between privacy and performance when designing FOLtR-ES, we compare this method with the current state-of-the-art OLTR method, the Pairwise Differentiable Gradient Descent (PDGD) [16]. For fair comparison, we set the privatization parameter $p = 1$ (lowest privacy) and the number of clients to 2,000. In addition note that in normal OLTR settings, rankers are updated after each user interaction: however in FOLtR-ES, rankers are updated in small batches. For fair comparison, we adapt PDGD to be updated in batch too. Instead of updating the ranker after each interaction (batch size 1), we accumulate gradients computed on the same batch size as for FOLtR-ES. Specifically, with 2000 clients for FOLtR-ES, the batch size of each update is 8,000 iterations ($4 \times 2{,}000$). We then compute the updated gradients for PDGD on 8,000 interactions too. We perform these experiments on all three datasets considered in this paper, but we omit to report results for Yahoo! due to space limitations.

Results are shown in Figure 3: regardless of linear or neural ranker, FOLtR-ES is less effective than PDGD. The gap in performance is greater in larger datasets like MSLR-WEB10k than in the smaller MQ2007/2008. This gap becomes even bigger, especially for the first iterations, if the PDGD ranker was updated after each iteration (not shown here), rather than after a batch has



(a) Mean batch MaxRR for MQ2007.



(b) Mean batch MaxRR for MSLR-WEB10k

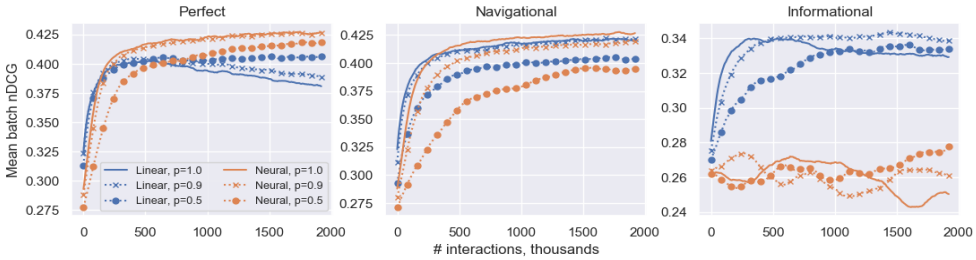
Fig. 3: Results for RQ3: performance of FOLtR-ES and PDGD across datasets with privatization parameter $p = 1$ and 2,000 clients (averaged across all dataset splits).

been completed. This highlights that FOLtR-ES has the merit of being the first privacy preserving federated OLTR approach available; however, more work is needed to improve the performance of FOLtR based methods so as to close the gap between privacy-oriented approaches and centralise approaches that do not consider user privacy.

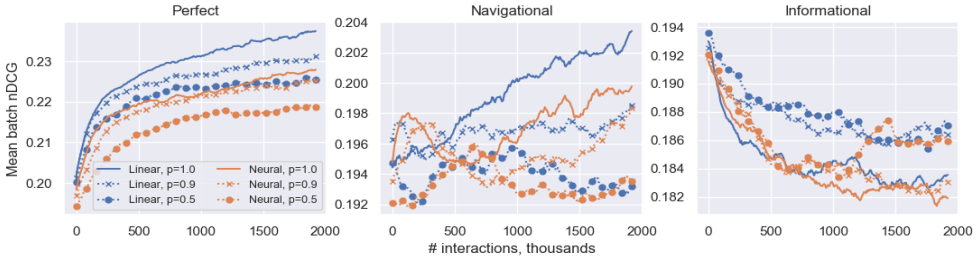
4.4 RQ4: Extending FOLtR-ES evaluation to common OLTR practice

In the original work and in the sections above, FOLtR-ES was evaluated using MaxRR computed with respect to the clicks performed by the simulated users (click models). This is an unusual evaluation for OLTR because: (1) usually nDCG@10 is used in place of MaxRR as metric, (2) nDCG is computed with respect to relevance labels, and not clicks, and on a withheld portion of the dataset, not on the interactions observed – this is used to produce learning curves and is referred to as offline nDCG, (3) in addition online nDCG is measured from the relevance labels in the SERPs from which clicks are obtained, and either displayed as learning curves or accumulated throughout the sessions – these values represent how OLTR has affected user experience. We then consider this more common evaluation of OLTR next, where we set the number of clients to 2,000 and experiment with $p = \{0.5, 0.9, 1.0\}$; we omit to report results for Yahoo! due to space limitations.

Results are reported in Figure 4. It is interesting to compare these plots with those in Figure 1, that relate to the unusual (for OLTR) evaluation setting used in the original FOLtR-ES work. By comparing the figures, we note that for MQ2007, FOLtR-ES can effectively learn rankers for perfect and navigational clicks. However, when the clicks become noisier (informational clicks), then FOLtR-ES learning is effective for the linear ranker but no learning occurs for the neural ranker: this



(a) Mean batch nDCG@10 for MQ2007.



(b) Mean batch nDCG@10 for MSLR-WEB10k.

Fig. 4: Results for RQ4: performance of FOLtR-ES in terms of online nDCG@10 computed using relevance labels and the SERPs used for obtaining user iterations (averaged across all dataset splits).

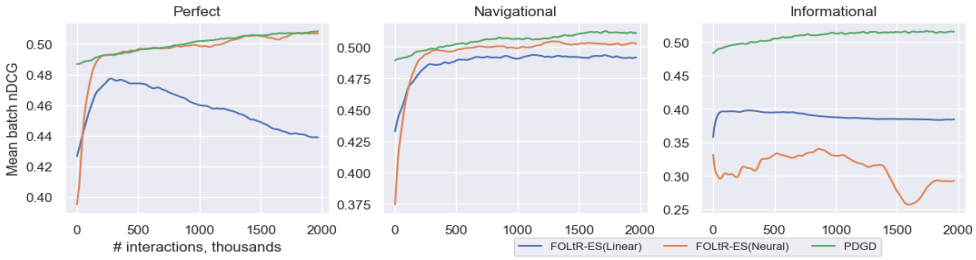
is unlikely in the evaluation settings of the original work (Figure 1). We note this finding repeating also for MSLR-WEB10k, but this time this affects both linear and neural rankers; we also note that the online performance in MSLR-WEB10k on navigational clicks is also quite unstable and exhibits little learning for specific values of p and ranker type. The online performance on MSLR10k for informational clicks (noisiest clicks) even exhibits a decreasing trend as iterations increase.

We further investigate the performance of FOLtR-ES with respect to offline nDCG@10. Results are shown in Figure 5, and are plotted along with the offline nDCG@10 of PDGD for additional context. Also the offline performance confirm that FOLtR-ES does not provide stable learning across click settings, datasets and ranker types. We also note that the performance of PDGD are sensibly higher than that of FOLtR-ES, apart for the neural ranker on MQ2007 when perfect and navigational clicks are considered.

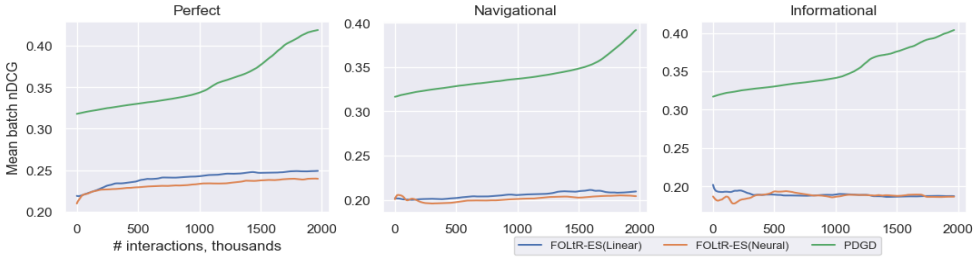
These findings suggest that FOLtR-ES is yet far from being a solution that can be considered for use in practice, and more research is required for devising effective federated, privacy-aware OLTR techniques.

5 Related Work

Learning to rank (LTR) consists of the application of supervised machine learning techniques to learn a ranking function from a set of labelled query-document pair examples, represented by features. A key limitation of LTR is the reliance on explicit relevance annotations (labels), which require substantial effort and cost to collect [1, 18]. Editorial labelling also poses ethical issue when



(a) Mean batch nDCG@10 for MQ2007.



(b) Mean batch nDCG@10 for MSLR-WEB10k.

Fig. 5: Results for RQ4: performance of FOLtR-ES and PDGD in terms of offline nDCG@10 with privatization parameter $p = 1$ and 2,000 clients (averaged across all dataset splits).

needing labels for private data [23], e.g., emails; in addition user preferences may not agree with that of annotators [21] and these labels cannot reflect evolving user preferences and search intents [15].

The use of implicit feedback in the form of, e.g., clicks has been suggested as a way to go beyond the above limitations [8]; this is the type of signal that the methods studied in this paper consider. This setting however presents a number of challenges: clicks are affected by a number of biases and noise, e.g., position bias and noisy clicks [4, 9, 17]. Approaches that exploit click feedback can be divided into counterfactual learning to rank (CLTR) [9] and online learning to rank (OLTR) [28]. CLTR relies on historical click through logs, treated as pure binary relevance labels, and commonly inverse propensity scoring (IPS) is used to re-weight clicks to minimise the impact of biases. Rankers are then trained in an offline manner and deployed online after training. OLTR instead, interactively updates rankers after each user interaction, in an online manner, and rankers explicitly manipulate SERPs to guide the learning process. This is the setup we consider in this paper, where rankers are iteratively updated in an online fashion following user interactions. A key aspect of OLTR is that the online interventions performed by rankers to guide the learning process carry the risk of displaying non optimal SERPs directly to the user, thus hurting user experience. It is important then for OLTR to rapidly learn a high quality ranker so as to not displaying low quality SERPs to a large number of users.

Little attention has been put on the fact that OLTR requires the search engine to monitor and collect user behaviour, thus not being appropriate when users want to preserve their privacy. In fact, current OLTR methods consider a central server that produces SERPs, collects queries and implicit user feedback, and updates a central ranker. An exception is the work of Kharitonov [10], considered in this paper, that instead exploits federate learning to de-centralise the collection of user

data and computation of gradient updates to the ranker; a central server is still required, but this only observes the federated gradient updates, which are then applied to the central ranker which is then distributed to the clients at each update iteration (more details in Section 2). Federated (machine) learning was recently introduced by Konecny et al. [12, 13]; in this framework models are learnt based on datasets distributed across different locations (clients) without the need to share the actual data, and with mechanisms to guarantee data leakage [27]. Privacy preservation is a topic of growing interest in information retrieval, with related workshops and tutorials being held in relevant venues [25, 26], but its main focus so far has been on query log anonymisation and privacy-preservation when sharing logs [2, 14, 29], rather than on integrating privacy preservation mechanisms within the ranking algorithms, as the work of Kharitonov instead does [10].

6 Conclusions

In this paper we considered the federated online learning to rank with evolutionary strategies (FOLtR-ES) method recently proposed by Kharitonov [10]. This is an interesting method because privacy requirements have been so far ignored in OLTR, and FOLtR-ES represents the first method of its kind.

We set to explore four research questions related to FOLtR-ES. RQ1 aimed to investigate the generalisability of the original results obtained by FOLtR-ES on the MQ2007/2008 dataset to other datasets used in current OLTR practice. Our experiments on MQ2007/2008 show consistent findings with that of Kharitonov [10]. However, when larger LTR datasets are considered, results change. In particular, the neural ranker used in FOLtR-ES is less effective than the linear ranker, especially on MSLR-WEB10k.

RQ2 aimed to investigate the effect varying the number of clients involved in FOLtR-ES has on the effectiveness of the method. Our experiments show mixed results with respect to the number of clients: the effect largely varies depending on dataset, ranker type and click settings.

RQ3 aimed to compare FOLtR-ES with current OLTR state-of-the-art methods to understand the gap required to be paid for maintaining privacy. Our experiments show that FOLtR-ES lags behind the current OLTR state-of-the-art in terms of ranking performance: differences become more substantial when noisy clicks or larger datasets are considered.

RQ4 aimed to investigate the generalisability of the original results obtained for FOLtR-ES to common evaluation practice in OLTR. Our experiments show that if the common evaluation settings used in OLTR are used to evaluate FOLtR-ES, then thee method shows high variability in effectiveness across datasets, rankers and clicks types – and overall that FOLtR-ES is unreliable on large datasets and noisy clicks. This finding suggests that more research and improvements are needed before a federated OLTR method, and FOLtR-ES in particular, can be used in practice.

Code, experiment scripts and further results are provided at https://github.com/ielab/foltr.

Acknowledgements. Shuyi Wang is sponsored by a China Scholarship Council (CSC) scholarship. Associate Professor Guido Zuccon is the recipient of an Australian Research Council DECRA Research Fellowship (DE180101579) and a Google Faculty Award.

References

1. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Chapelle, O., Chang, Y., Liu, T. (eds.) Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010. JMLR Proceedings, vol. 14, pp. 1–24. JMLR.org (2011)
2. Cooper, A.: A survey of query log privacy-enhancing techniques from a policy perspective. ACM Transactions on the Web (TWEB) **2**(4), 1–27 (2008)
3. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science **9**(3-4), 211–407 (2014)
4. Guan, Z., Cutrell, E.: 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. pp. 417–420 (2007)
5. Guo, F., Liu, C., Wang, Y.M.: Efficient multiple-click models in web search. In: Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009. pp. 124–131. ACM (2009)
6. Hofmann, K., Schuth, A., Whiteson, S., De Rijke, M.: 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. pp. 183–192 (2013)
7. Jagerman, R., Oosterhuis, H., de Rijke, M.: 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. pp. 15–24 (2019)
8. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 133–142 (2002)
9. Joachims, T., Swaminathan, A., Schnabel, T.: Unbiased learning-to-rank with biased feedback. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. pp. 781–789 (2017)
10. Kharitonov, E.: Federated online learning to rank with evolution strategies. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. pp. 249–257 (2019)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Konecný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
13. Konecný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
14. Korolova, A., Kenthapadi, K., Mishra, N., Ntoulas, A.: Releasing search queries and clicks privately. In: Proceedings of the 18th international conference on World wide web. pp. 171–180 (2009)
15. Lefortier, D., Serdyukov, P., De Rijke, M.: Online exploration for detecting shifts in fresh intent. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 589–598 (2014)
16. Oosterhuis, H., de Rijke, M.: Differentiable unbiased online learning to rank. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1293–1302 (2018)
17. Pan, B., Hembrooke, H., Joachims, T., Lorigo, L., Gay, G., Granka, L.: In google we trust: Users' decisions on rank, position, and relevance. Journal of computer-mediated communication **12**(3), 801–823 (2007)
18. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. arXiv preprint arXiv:1306.2597 (2013)
19. Radlinski, F., Kleinberg, R., Joachims, T.: Learning diverse rankings with multi-armed bandits. In: Proceedings of the 25th international conference on Machine learning. pp. 784–791 (2008)
20. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864 (2017)
21. Sanderson, M.: Test collection based evaluation of information retrieval systems. Now Publishers Inc (2010)

22. Schuth, A., Oosterhuis, H., Whiteson, S., de Rijke, M.: Multileave gradient descent for fast online learning to rank. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016. pp. 457–466. ACM (2016)
23. Wang, X., Bendersky, M., Metzler, D., Najork, M.: 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. pp. 115–124 (2016)
24. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3-4), 229–256 (1992)
25. Yang, G.H., Zhang, S.: Differential privacy for information retrieval. In: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval. pp. 325–326 (2017)
26. Yang, H., Soboroff, I., Xiong, L., Clarke, C.L., Garfinkel, S.L.: Privacy-preserving ir 2016: Differential privacy, search, and social media. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 1247–1248 (2016)
27. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
28. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 1201–1208 (2009)
29. Zhang, S., Yang, H., Singh, L.: Anonymizing query logs by differential privacy. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 753–756 (2016)
30. Zhuang, S., Zuccon, G.: Counterfactual online learning to rank. In: European Conference on Information Retrieval. pp. 415–430. Springer (2020)