Effective and Privacy-preserving Federated Online Learning to Rank

Shuyi Wang, Bing Liu, Shengyao Zhuang, Guido Zuccon
The University of Queensland
Brisbane, Australia
{shuyi.wang,bing.liu,s.zhuang,g.zuccon}@uq.edu.au

ABSTRACT

Online Learning to Rank (OLTR) has been primarily studied in the centralised setting, where a central server is responsible to index the searchable data, collect the users' queries and search interactions, and optimize ranking models. A drawback of such a centralised OLTR paradigm is that it cannot guarantee user's privacy as all data (both the searchable one and the one related to user interactions) is collected by the server.

In this paper, we propose a Federated OLTR method, called FPDGD, which leverages the state-of-the-art Pairwise Differentiable Gradient Descent (PDGD) and adapts it to the Federated Averaging framework. For a strong privacy guarantee, we further introduce a noise-adding clipping technique based on the theory of differential privacy to be used in combination with FPDGD.

Empirical evaluation shows FPDGD significantly outperforms the only other federated OLTR method. In addition, FPDGD is more robust across different privacy guarantee requirements than the current method: our method is therefore more reliable for real-life applications.

CCS CONCEPTS

• **Information systems** → **Combination, fusion and federated search**; **Retrieval models and ranking**.

KEYWORDS

Federated Online Learning to Rank; Online Learning to Rank; Differential Privacy; Privacy Aware Information Retrieval

ACM Reference Format:
Shuyi Wang, Bing Liu, Shengyao Zhuang, Guido Zuccon. 2021. Effective and Privacy-preserving Federated Online Learning to Rank. In *Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '21), July 11, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3471158.3472236

1 INTRODUCTION

This paper considers the problem of devising effective online learning to rank (OLTR) methods embedded in a federated system.

OLTR methods allow to learn a ranker from observing users queries and interactions with search engine result pages (SERPs), e.g., clicks. This is achieved by iteratively train and update a ranker in production. By using user interactions rather than explicit relevance labels, OLTR overcomes a number of drawbacks associated with traditional learning to rank approaches such as the high cost and time required by the editorial annotation effort, the fact that the intent identified by editors may not be the one the user had in mind [23], and the issues associated with rapid changes of intents underlying queries [32].

In traditional OLTR, search results are produced by a centralised search service, which also keeps track of user queries and interactions that are used as signal for learning an effective ranker [11, 19, 24, 27, 30, 31]. This centralised search service has also a complete index of the data to be searched (though likely this index is distributed and replicated across servers and data centres). A drawback of this solution is that the search service has access to the indexed data, and also actively collects users queries and interactions such as clicks: thus *user privacy is limited* in that all this user data is collected and exploited by the search service. So, for example, if users were interested for the search service to offer search functionalities on their private data (e.g. emails, desktop files) they need to surrender this to the search service. Similarly, the OLTR-based search service would also not meet the expectations of those users that wish for their search behaviour not to be collected to protect their privacy.

In a federated setting, pictured in Figure 1, data on which to search, along with user queries and interactions are withheld from the central server, and so is also the responsibility of producing search results. Search is instead performed within the user device ❷,

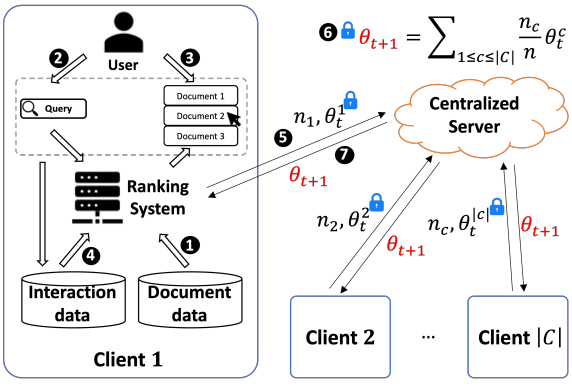


Figure 1: Schematic representation of the federated online learning to rank (FOLTR) setting.

ICTIR '21, July 11, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8611-1/21/07…$15.00
https://doi.org/10.1145/3471158.3472236

which also indexes the users data ❶ and collects users interactions ❸ and performs the online updates required by the OLTR method ❹. Updates are then shared by several users devices with a central server ❺. It is the responsibility of the central server to combine the ranker updates from different clients to produce a new version of the ranker ❻. This new version is then distributed to the users devices ❼. The advantage of a federated OLTR solution over a traditional OLTR approach is that neither user data nor users queries and interactions are seen by or shared with anyone, thus potentially *preserving user privacy*.

Federated learning approaches to search and recommendation have only recently emerged. In the area of OLTR, FOLtR-ES, proposed by Kharitonov [13] and then extended by Wang et al. [28], is the only federated method available. FOLtR-ES uses evolutionary strategies similar 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 privacy preserving mechanism is required to protect the gradient updates from privacy attacks [3]. In fact, despite the federate learning process not involving the sharing of user data, a malicious attacker that is able to observe the ranker model and gradient updates can potentially reconstruct the data that produced such an update [7], thus finally exposing user data.

However, the effectiveness of the current approach for federating OLTR (i.e., FOLtR-ES) exhibits large gaps in performance compared to current state-of-the-art, not federated, OLTR methods [28]. It is this gap that we want to fill in this paper. Specifically, in this paper, we put forward the following contributions:

- We cast the current state-of-the-art online LTR approach, the Pairwise Differentiable Gradient Descent (PDGD) [19], into the federated learning paradigm, specifically using the federated averaging algorithm. To do so, we devise a federated gradient update for PDGD. Our federated PDGD provides large, significant gains in effectiveness over the current federated OLTR method (FOLtR-ES [13, 28]), and its effectiveness generalises across datasets.
- We apply differential privacy on top of the proposed federated PDGD to secure the gradient updates performed in the federated environment against privacy attacks. We further investigate the trade-off between privacy preservation and ranker effectiveness.

2 RELATED WORKS

OLTR. Online Learning to Rank (OLTR) differs from traditional Learning to Rank (LTR) methods in that it makes use of user online feedback (such as clicks) as the optimization goal in order to tackle some data-annotation drawbacks of LTR. In fact, LTR relies on explicit relevance annotations (labels), which often imply high annotation costs. In addition, editorial annotation may be of ethical concern when private data, such as emails, need to be labelled [29]. Furthermore, user preferences may not agree with that of annotators [23], users' search intents may change from time to time [14, 32], and explicit relevance labels may not account for these aspects.

The Dueling Bandit Gradient Descent (DBGD) uses online evaluation to unbiasedly compare two or more rankers given a user interaction [30]. Subsequent work developed more reliable or more efficient online evaluation methods, including Probabilistic Interleaving (PIGD) [11], and its extension, the Probabilistic Multileaving (PMGD) [20], which compares multiple rankers at each interaction, resulting in the best DBGD-based algorithm. However, this method suffers from a high computational cost because it requires sampling ranking assignments to infer outcomes [31]. Further variations that reuse historical interaction data to accelerate the learning in DBDG have also been investigated [10].

The Pairwise Differentiable Gradient Descent method (PDGD) [19] constructs a pairwise gradient to update the ranking model according to the click feedback. Compared to previous OLTR methods, PDGD has showcased faster learning (higher online nDCG) and convergence to higher effectiveness (higher offline nDCG). Unlike other methods, PDGD can be applied to any differentiable ranking model and it is an unbiased method. Oosterhuis and de Rijke trained a linear and a neural ranker and both showed effective results [19]. PDGD is the current state-of-the-art OLTR method.

Federated learning with privacy protection. *Federated Learning* [16] considers a machine learning setting where several data owners (clients) collaboratively train a model without sharing the data. To this aim, a coordinator (central server) is required and each participating client sends what is essential for the training (for example, the local gradient) to the central server. The Federated Averaging (FedAvg) algorithm [16] relies on local stochastic gradient descent (SGD), which is computed by each client, while the centralised server performs the global model update by weighted averaging the client's updates.

Differential privacy [4, 5] is a method often used to ensure user privacy preservation in federated learning (the alternative is the encryption of the messages). We shall review the notion of ϵ-differential privacy in Section 3.3. Abadi et al. [1] have developed the algorithmic techniques for combining SGD-based deep learning methods with differential privacy. Differential privacy has also been added to FedAvg algorithm to provide user-level differential privacy guarantees for language models [17], as for these models it has been shown that the gradients or shared model's weights can reveal sensitive information regarding the clients' data [6, 7, 25].

Federated OLTR. OLTR has been primarily investigated in a centralised paradigm, where a central server holds the data to be searched and collects the users' search interactions (e.g., queries, clicks). The training of the ranker is also conducted by the server. This centralised paradigm is not suited to a privacy-preserving situation where each client does not want to share the data on which it wants to search, along with queries and other search interactions.

Recently, the Federated OLTR with Evolutionary Strategies method (FOLtR-ES) [13] has been proposed to address the above privacy-preserving issues. FOLtR-ES extends the OLTR optimization scenario to the Federated SGD [16] and uses Evolution Strategies as the optimization method [22]. To defend the gradient updates from a malicious attacker, FOLtR-ES also includes a privatization procedure, which leverages the theory of ϵ-local differential privacy [5]. FOLtR-ES is shown to perform well on small-scale datasets (MQ2007) for the MaxRR metric, although performance degrades

when privacy preservation is required [13]. However, the effectiveness of FOLtR-ES is not consistent across other, larger-scale datasets nor when typical OLTR metrics are considered (nDCG) [28].

3 FEDERATED PAIRWISE DIFFERENTIABLE GRADIENT DESCENT

In this section we propose a novel federated version of the current state-of-the-art OLTR method, the Pairwise Differentiable Gradient Descent (PDGD) [19], based on the Federated Averaging approach [16] (Section 3.2). We then show how the gradient update of our federated OLTR method can be secured using differential privacy (Section 3.3). First though, we provide a brief introduction to the PDGD method (Section 3.1).

3.1 Pairwise Differentiable Gradient Descent

Consider optimizing a ranking model $f_\theta(\mathbf{d})$ which sorts documents by decreasing output scores, where $\mathbf{d}$ represents the features of a query-document pair. The goal of an OLTR algorithm is to find the parameter vector θ for which the ranker displays an optimal ranking list.

Given this generic formulation of the ranking problem, PDGD applies a Plackett-Luce (PL) model to the ranking function $f_\theta(\cdot)$, resulting in a distribution over the document set D:

$$P(d|D) = \frac{e^{f_\theta(d)}}{\sum_{d' \in D} e^{f_\theta(d')}}. \tag{1}$$

A ranking R of length k is then created by sampling from the distribution k times. With d_i representing the document at position i, the probability of ranking R is computed as:

$$P(R|D) = \prod_{i=1}^{k} P(d_i|D \setminus \{d_1, ..., d_{i-1}\}). \tag{2}$$

After receiving the displayed list, the user may click on some of its documents. The method assumes that clicked documents represent a preference by the user over unclicked ones. For example, if d_k is clicked while d_l is not, the document preference inferred from this click event is $d_k >_c d_l$ and the probability that the preferred document d_k is sampled before d_l is increased. Based on this, PDGD adopts a pairwise optimization and estimates the gradient of the user preferences by the weighted sum:

$$\nabla f_\theta(\cdot) \approx \sum_{d_k >_c d_l} \rho(d_k, d_l, R, D)[\nabla P(d_k > d_l)]. \tag{3}$$

From [26], the probability that the preferred document d_k is sampled before d_l is:

$$P(d_k > d_l) = \frac{P(d_k|D)}{P(d_k|D) + P(d_l|D)} = \frac{e^{f(d_k)}}{e^{f(d_k)} + e^{f(d_l)}}. \tag{4}$$

Thus, the gradient estimation equals:

$$\nabla f_\theta(\cdot) = \sum_{d_k >_c d_l} \rho(d_k, d_l, R, D) \frac{e^{f_\theta(d_k)} e^{f_\theta(d_l)}}{(e^{f_\theta(d_k)} + e^{f_\theta(d_l)})^2} (f_\theta{}'(d_k) - f_\theta{}'(d_l)). \tag{5}$$

The function ρ is applied to reweigh the click preferences by the ratio between the occurring probabilities of R or $R^*(d_k, d_l, R)$, which is the reversed pair ranking for R, i.e. the same ranking as R except the position of d_k and d_l are swapped. This is done to reduce the position bias caused by the current ranking R. The reweighing function is defined as:

$$\rho(d_k, d_l, R, D) = \frac{P(R^*(d_k, d_l, R)|D)}{P(R|D) + P(R^*(d_k, d_l, R)|D)}. \tag{6}$$

Algorithm 1 details the PDGD method.

Algorithm 1 Pairwise Differentiable Gradient Descent(PDGD) [19]

Input: initial weights: θ_1; scoring function: f; learning rate η.
for $t \leftarrow 1 \ldots \infty$ **do**
  $q_t \leftarrow receive\_query(t)$ *// obtain a query from a user*
  $D_t \leftarrow preselect\_documents(q_t)$ *// preselect documents for query*
  $\mathbf{R}_t \leftarrow sample\_list(f_{\theta_t}, D_t)$ *// sample list according to Eq. 1,2*
  $\mathbf{c}_t \leftarrow receive\_clicks(\mathbf{R}_t)$ *// show result list to the user*
  $\nabla f_{\theta_t} \leftarrow \mathbf{0}$ *// initialize gradient*
  for $d_k >_\mathbf{c} d_l \in \mathbf{c}_t$ **do**
    $w \leftarrow \rho(d_k, d_l, R, D)$ *// initialize pair weight (Eq. 6)*
    $w \leftarrow w \frac{e^{f_{\theta_t}(\mathbf{d}_k)} e^{f_{\theta_t}(\mathbf{d}_l)}}{(e^{f_{\theta_t}(\mathbf{d}_k)} + e^{f_{\theta_t}(\mathbf{d}_l)})^2}$ *// pair gradient (Eq. 5)*
    $\nabla f_{\theta_t} \leftarrow \nabla f_{\theta_t} + w(f'_{\theta_t}(\mathbf{d}_k) - f'_{\theta_t}(\mathbf{d}_l))$ *// model gradient(Eq. 5)*
  $\theta_{t+1} \leftarrow \theta_t + \eta \nabla f_{\theta_t}$ *// update the ranking model*

Algorithm 2 FederatedAveraging PDGD.
- set of clients participating training: C, each client is indexed by c;
- local interaction set: B, number of local interactions: n_c.

Server executes:
  initialize θ_0; scoring function: f; learning rate: η
  for each round $t = 1, 2, \ldots$ **do**
    for each client $c \in C$ **in parallel do**
      $\theta^c_{t+1}, n_c \leftarrow \text{ClientUpdate}(c, \theta_t)$
    $\theta_{t+1} \leftarrow \sum_{c=1}^{|C|} \frac{n_c}{n} \theta^c_{t+1}$

ClientUpdate(c, θ): *// Run on client c*
  for each local update i from 1 to B **do**
    $\theta \leftarrow \theta + \eta \nabla f_\theta$ *//PDGD update shown in Algorithm. 1*
  return (θ, n_c) to server

3.2 Federated Averaging for PDGD

The Federated Averaging algorithm [16] is a popular approaches for Federated Learning. It has the advantages of being robust to unbalanced and non-IID data and can significantly reduce the communication costs between clients and the central server. Next, we provide a working description of the algorithm.

Generally speaking, the optimization step of a generic machine learning algorithm consists of minimizing a loss function f:

$$\min_{\theta \in \mathbb{R}^d} f(\theta) \qquad \text{where} \qquad f(\theta) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\theta). \tag{7}$$

For each data example (x_i, y_i), $f_i(\theta) = loss(x_i, y_i; \theta)$. Stochastic gradient descent (SGD) is commonly used to solve this optimization problem. By using SGD with a fixed learning rate η, the model is updated as:

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla f(\theta_t). \tag{8}$$

To perform the training process in federated manner, we assume that $|C|$ clients hold their own training dataset. For each client c holding the local dataset D_c, the volume of the dataset is $n_c = |D_c|$. The loss function Eq. 7 can be re-written as:

$$f(\theta) = \sum_{c=1}^{|C|} \frac{n_c}{n} F_c(\theta) \quad \text{where} \quad F_c(\theta) = \frac{1}{n_c} \sum_{i \in D_c} f_i(\theta). \tag{9}$$

Based on Eq. 9, $\nabla f(\theta_t) = \sum_{c=1}^{|C|} \frac{n_c}{n} \nabla F_c(\theta)$. On each client c, the average gradient on the local data for the current model is denoted as $g_c = \nabla F_c(\theta_t)$. Then, the local model is updated using $\theta_{t+1}^c \leftarrow \theta_t - \eta g_c$ and the global model is updated by weighted averaging all local models:

$$\theta_{t+1} \leftarrow \sum_{c=1}^{|C|} \frac{n_c}{n} \theta_{t+1}^c. \tag{10}$$

To do so, each client locally takes one step of gradient descent on the current model using its local data and the server then uses the weighted average of the local models as the global updated model. This process forms the FederatedSGD (FedSGD) algorithm [16]. Federated Averaging (FedAvg) extends FederatedSGD by adding more updating times to each client: the local update $\theta_{t+1}^c \leftarrow \theta_t - \eta g_c$ is performed within a minibatch portion of the local dataset, before the averaging step.

In our approach (FPDGD), we implement PDGD using FedAvg. In Algorithm 2, each client considers B interactions and updates the local ranker using PDGD gradients accordingly. After the local update is finished, each client returns the trained weights to the server. The server then aggregates those local messages by averaging the weights and sends back to the clients the latest ranker.

3.3 Securing the Gradient: Differential Privacy for FPDGD

A key motivation for adopting a federated learning approach is to protect users' private data: this is achieved by avoiding having to share data across clients or with a server – instead, only gradient updates are shared. However, in the federated learning setting, the model parameters can be attacked to reveal sensitive information about the training data [6, 7].

To provide a strong privacy guarantee for the proposed federated PDGD, we introduce a noise-adding clipping technique [17, 18] which is based on the theory of differential privacy [4, 5].

Definition 3.1. (ϵ-Differential Privacy): A randomized mechanism $\mathcal{M}: \mathcal{D} \rightarrow \mathcal{R}$ with a domain $\mathcal{D}$ (e.g., possible training datasets) and range $\mathcal{R}$ (e.g., all possible trained models) satisfies ϵ-differential privacy when there exists $\epsilon > 0$ such that:

$$P[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} P[\mathcal{M}(D_2) \in S] \tag{11}$$

where $P[\cdot]$ is a probability, D_1 and D_2 are any two datesets differing for only one data instance, and S denotes all subsets of possible outputs that $\mathcal{M}$ produces: $S \subseteq \mathcal{R}$. The parameter ϵ represents the privacy budget, where the lower the value of ϵ, the higher the privacy guarantee.

Definition 3.2. (Global Sensitivity): For any real-valued query function $t: \mathcal{D} \rightarrow \mathcal{R}$, where $\mathcal{D}$ denotes the set of all possible datasets, the global sensitivity Δ of t is defined as:

$$\Delta = \max_{D_1, D_2} |t(D_1) - t(D_2)| \tag{12}$$

for all $D_1, D_2 \in \mathcal{D}$.

Since there is no priori knowledge about the sensitivity of PDGD, we clip the weights outputed by PDGD to the bound $\frac{\Delta}{2}$, as in Eq. 13, so that the parameter updating can meet the global sensitivity Δ. It should be noticed that the sensitivity Δ is used as hyper-parameter and will be determined through grid search for a given privacy level ϵ.

$$\theta = \theta \cdot \min(1, \frac{\Delta}{2 \cdot \parallel \theta \parallel}). \tag{13}$$

The Laplacian mechanism preserves ϵ-differential privacy [4] through leveraging random noise X sampled from a symmetric Laplacian distribution. The probability density function $f(x)$ of a zero-mean Laplacian distribution is:

$$f(x) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \tag{14}$$

where λ is the scale parameter of the Laplacian distribution. Given Δ of the query function t, and the privacy loss parameter ϵ, the Laplacian mechanism $\mathcal{M}$ uses random noise X drawn from the Laplacian distribution with scale $\lambda = \frac{\Delta}{\epsilon}$.

In the federated learning setting, each client contributes a portion of differentially private noise so that the sum of all contributions results in a differentially private noise value. Previous work has used Gamma random variables to generated the Laplace Random variable [18]. Specifically, a Laplace Random variable can be generated from the sum of n Gamma random variables:

$$L(\mu, \lambda) = \frac{\mu}{n} + \sum_{p=1}^{n} \gamma_p - \gamma_p' \tag{15}$$

where μ is the mean parameter and λ is the scale parameter of the Laplacian mechanism.

γ_p and γ_p' are Gamma random variables with probability density function defined as:

$$f(x) = \frac{(1/s)^{1/n}}{\Gamma(1/n)} x^{1/n-1} e^{-x/s} \tag{16}$$

Algorithm 3 FederatedAveraging PDGD with differential privacy.
- set of clients participating training: C, each client is indexed by c;
- local interaction set: B, number of local interactions: n_c.

Server executes:
initialize θ_0; scoring function: f; learning rate: η
for each round $t = 1, 2, \ldots$ **do**
  for each client $c \in C$ **in parallel do**
    $\theta_{t+1}^c, n_c \leftarrow \text{ClientUpdate}(c, \theta_t)$
  $\theta_{t+1} \leftarrow \sum_{c=1}^{|C|} \frac{n_c}{n} \theta_{t+1}^c$

ClientUpdate(c, θ): // *Run on client c*
for each local update i from 1 to B **do**
  $\theta \leftarrow \theta + \eta \nabla f_\theta$ //*PDGD update shown in Algorithm. 1*
$\theta \leftarrow \theta \cdot \min(1, \frac{\Delta}{2 \cdot \|\theta\|})$ //*clip the weights*
return $(\theta + \gamma - \gamma', n_c)$ to server //*γ,γ' gamma noise sampled from Eq. 16*

where $1/n$ is the shape parameter, s is the scale parameter and $\Gamma(p) = \int_0^\infty x^{p-1}e^{-x}\,dx$. In our federated setting, n equals to number of clients $|C|$ and s is equivalent to λ of the Laplacian distribution.

We set $\mu = 0$ and make use of this differential privacy technique so that each client adds $\gamma_p - \gamma'_p$ noise to each gradient update. This process is shown in Algorithm 3.

4 EXPERIMENTAL SETUP

We rely on the typical evaluation setup used by previous OLTR research [9, 19, 20] to empirically investigate the effectiveness of the proposed FPDGD approach, and how it compares to other methods. This consists of use standard learning to rank datasets, simulate user interactions with SERPs, i.e. simulate clicks, and measure both online and offline performance.

Datasets. We use the MQ2007 [21] and MSLR-WEB10k [21] datasets. While other, larger datasets for learning to rank are currently available, our choice considered the trade-off between representativeness and the high computational costs associated with experimenting with the federated OLTR methods on larger datasets.

MQ2007 is relatively small and has fewer assessed documents per query, and was chosen to allow direct comparison with previous work on federated OLTR [13], which mainly used MQ2007, and because this dataset, being small, allows for computationally treatable experiments. The dataset, compiled using data from the TREC 2007 Million Query Track [2], contains 1,700 queries, divided into 5 folds, with associated relevance assessments, expressed on a 3-level scale, from *not relevant* (0) to *very relevant* (2). Each query-document pair is represented by a 46-dimensional feature vector.

The MSLR-WEB10k is larger and more recent than MQ2007, and it was chosen as previous work suggested findings for federated OLTR observed on MQ2007 may not generalise on MSLR-WEB10k [28]. The MSLR-WEB10k, constructed from a retired labelling set of a Microsoft web search engine, contains 10,000 queries (divided into 5 folds), and on average each query is associated with 125 assessed documents on a 5 point scale. Query-document pairs in MSLR-WEB10k have a much richer representation (136 features) than in MQ2007.

User simulations. Both the querying and clicking behaviour of users are simulated in our experiments.

For the querying behaviour, for each client participating in the federated OLTR, we sample B queries randomly, in line with previous work on FOLTR [13, 28]. For each query, we use the local ranking model (i.e. that held by the client) to rank documents; we limit SERP to 10 documents.

For the click behaviour, we rely on the Cascade Click Model (CCM) [8]. Under CCM, users are assumed to examine a SERP from top to bottom. Each document is inspected and clicked with click probability $P(click = 1|r)$, conditioned on the relevance label r. After a click occurs, the user stops scanning with stopping 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 click feedback among the three instantiations. The values used for $P(click = 1|r)$ and $P(stop = 1|r)$ for CCM in the two considered datasets are reported in Table 1.

Federated setup. We simulate the federated OLTR scenario as follows. Each client holds a copy of the current ranker. For each client, we consider $|B|$ user queries along with the respective interactions, during which the local ranker is optimised using the simulated user clicks. After $|B|$ interactions have occurred, the client sends the local message (updated weights) to the central server. The central server optimises the global ranker by aggregating the local messages and sends the newly-updated ranker back to each client.

In our experiments for MQ2007, unless otherwise specified, we simulate $|C| = 1,000$ clients with each client performing $|B| = 4$ interactions (queries) locally to contribute to each global model update. We restrict the total interaction budget to 4 million queries, which results in 1,000 global updates. For MSLR-10K dataset, unless otherwise specified, we also simulate $|C| = 1,000$ clients but with only $|B| = 2$ local interactions and we set the total interaction budget to 400,000 queries; thus resulting in 200 global updates.

Baselines To investigate the impact of the federated averaging process on PDGD, we compare FPDGD against the original, centralised, PDGD model [19]. We follow the original paper and set the learning rate $\eta = 0.1$. For both method, we train a linear model as the ranker. In federated OLTR, global rankers are updated in batches, that is, the central server updates rankers after each client has executed B searches and the local ranker updates have been sent to the server. However, in centralised OLTR settings, rankers are updated after each user interaction (batch size = 1). For fair comparison, we adapt PDGD to also be updated in batch by accumulating gradients updates.

At the time of writing, FOLtR-ES is the only federated OLTR algorithm proposed in the literature [13], and it constitutes a natural baseline for comparison with FPDGD, thus allowing to compare different federated OLTR methods. For FOLtR-ES, we train a linear model to make fair comparison with our method trained on linear ranker. We set the learning rate $\eta = 0.001$ and choose the reciprocal rank of the highest clicked result (MaxRR) in an interaction as the optimization metric in FOLtR-ES.

Evaluation measures. To evaluate OLTR and FOLTR methods we rely on the evaluation practice from previous OLTR work, which consists of measuring the rankers' offline and online performance.

To measure offline performance, we average nDCG@k of the global ranker over the queries in the held-out test set. We record this performance value after each update of the global ranker and the final performance is also recorded. We consider $k = 10$ as the number of documents in a ranked list for a query.

To measure online performance, we compute the cumulative discounted nDCG@k for the rankings displayed during the training phase [9]. For t-th query in a sequence of T queries, with R^t representing the ranking list displayed to the user, online performance is computed as:

$$Online\_Performance = \sum_{t=1}^{T} nDCG(R^t) \cdot \gamma^{t-1}. \tag{17}$$

This measure indicates the quality of the user experience during training; the discount factor γ puts more importance to the interactions in the early phases of training. Following previous work [19], we set $\gamma = 0.9995$.

Table 1: Settings of the CCM click models for MSLR-WEB10k (MQ2007) dataset. Note in MQ2007, only three-level of relevance is used. We demonstrate the values for MQ2007 in bracket.

	$P(click = 1 \mid r)$					$P(stop = 1 \mid click = 1, r)$				
Click models	$r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$
perfect	0.0 (0.0)	0.2 (0.5)	0.4 (1.0)	0.8 (-)	1.0 (-)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (-)	0.0 (-)
navigational	0.05 (0.05)	0.3 (0.5)	0.5 (0.95)	0.7 (-)	0.95 (-)	0.2 (0.2)	0.3 (0.5)	0.5 (0.9)	0.7 (-)	0.9 (-)
informational	0.4 (0.4)	0.6 (0.7)	0.7 (0.9)	0.8 (-)	0.9 (-)	0.1 (0.1)	0.2 (0.3)	0.3 (0.5)	0.4 (-)	0.5 (-)

Each experiment is repeated 25 times, spread evenly over all dataset training folds. All evaluation results are averaged and statistical significant differences between system pairs are evaluated using two-tailed Student's t-test with Bonferroni correction.

Choosing privacy parameters The differential privacy process in FPDGD is controlled by the parameter ϵ (Section 3.3). To make comparison with the baseline FOLtR-ES method fair, we utilize the FOLtR-ES's privacy parameter p to fix an upper bound on the value of ϵ. In FOLtR-ES, the higher the p value, the lower the privacy. In our experiments, we set $p \in \{0.25, 0.5, 0.9, 1.0\}$; these settings correspond to upper bound values of $\epsilon \in \{1.2, 2.3, 4.5, 10\}$. For a given privacy level ϵ, we search the best setting of sensitivity Δ through grid search in $\{1, 3, 5, 7, 9\}$. Then, for each $\epsilon \in \{1.2, 2.3, 4.5, 10\}$, we get the corresponding $\Delta \in \{3, 3, 5, 5\}$.

5 RESULTS

5.1 Overall ranking performance

Figure 2 displays the offline performance (nDCG@10) of the proposed FPDGD method across different settings of differential privacy ϵ, along with the effectiveness of the non-federated PDGD method and the baseline federated FOLtR-ES method (across different settings of differential privacy p). The online performance of these methods are reported in Table 2.

Impact of federation on PDGD. We start the analysis of the results by commenting on the impact of federation on the PDGD method, focusing on the offline performance (Figure 2). When PDGD is given the same number of updates than its federated counterpart, its performance are lower. This is not surprising: the non-federated PDGD only uses the click data from one search interaction (query) for each update, and thus it is updated only 200 times in MSLR-WEB10k (1,000 in MQ2007). While FPDGD is also updated 200 times (1,000), when an update is performed this considers 2 interactions per client and 1,000 clients – thus the total interaction budget assigned to FPDGD in MSLR-WEB10k is $200 \times 2 \times 1,000 = 400,000$ (4 million in MQ2007) compared to that of 200 for PDGD (1,000 for MQ2007). However if PDGD is given the same interaction budget as its federated counterpart, then PDGD does perform better than FPDGD. This difference is due to PDGD being able to perform a ranker update at each interaction (query), while FPDGD needs to wait the completion of batch size $\times$ clients $= 2 * 1,000 = 2,000$ interactions in MSLR-WEB10k for doing an update (4, 000 in MQ2007). Similar insights are found when examining online performance (Table 2).

Impact of privacy. Next, we analyse the impact of differential privacy on FPDGD, where lower values of ϵ correspond to higher privacy guarantees. We find that differential privacy has little impact on the performance of FPDGD (both online and offline) for MSLR-WEB10k. For MQ2007, lower differential privacy values tend to provide lower performance, though not significantly worse.

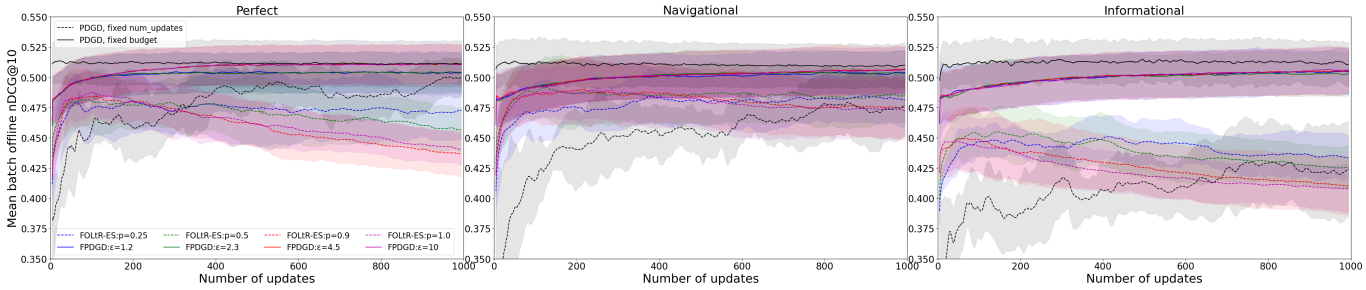
Comparison with FOLtR-ES. When comparing the proposed federated approach, FPDGD, with the currently only other federated OLTR method, FOLtR-ES, we note that FPDGD consistently outperforms FOLtR-ES on offline performance across datasets, click models, and differential privacy settings (Figure 2). Gains over FOLtR-ES are specifically notable for all MSLR-WEB10k experiments and for the informational click model for MQ2007. The fact that the performance of FOLtR-ES are particularly poor in MSLR-WEB10k finds confirmation in previous work, which shows FOLtR-ES has poor performance on large datasets [28]. In addition, in MQ2007, FOLtR-ES exhibits a surprisingly decreasing trend as global model updates increase, especially for higher values of differential privacy p ($p = 0.9, 1.0$), when the perfect and informational click models are considered. Similar observations can be made when considering online performance on MSLR-WEB10k dataset (Table 2): FPDGD is significantly better than FOLtR-ES under all three click models within different privacy budgets. However, on MQ2007, our method is not consistently better than FOLtR-ES. When the navigational click model is considered, FPDGD is significantly outperformed by FOLtR-ES. We believe the results are caused by the random noise from the sampling strategy of PDGD that occurs in the early training interactions: results on small-scale datasets like MQ2007 tend to be easily affected by this issue. In depth analysis of the results on MQ2007[1] reveal that for this dataset FOLtR-ES has better online performance than FPDGD at the very early search interactions, with FPDGD soon catchup with, and then outperforming, FOLtR-ES in later interactions.

Evaluation using MaxRR. In our experiments, we used nDCG@10 as evaluation measure and optimisation criteria for the OLTR process, in line with previous work on OLTR. FOLtR-ES however was previously evaluated on MaxRR [13] (though Wang et al. also reports nDCG@10 evaluation [28]), which is used internally by FOLtR-ES as optimisation criteria. For completeness, in Figure 3 we compare FPDGD and FOLtR-ES when MaxRR is used for evaluation. We find that the trends observed when evaluating with nDCG@10 (Figure 2) are also found when MaxRR is used.

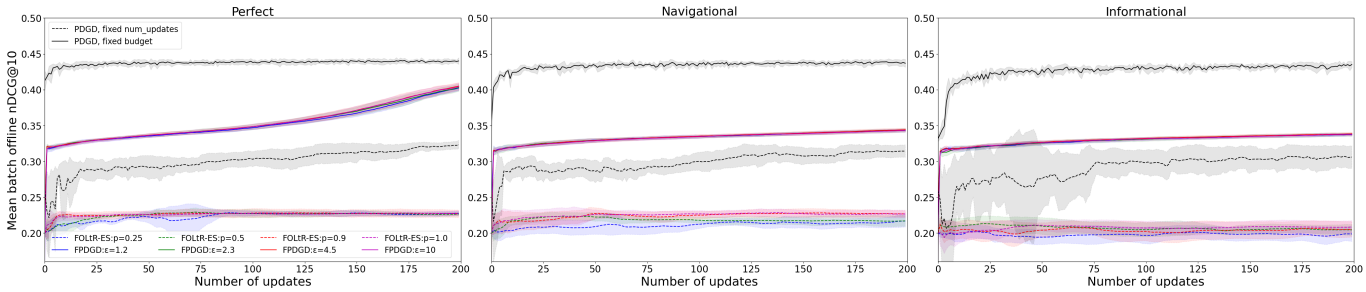
5.2 Influence of number of clients

To study the influence of the number of clients participating in our federated setup, we vary the number of clients in $|C| \in \{10, 100, 1000\}$. We report the offline nDCG@10 results obtained on MSLR-10k in

[1] Not shown here, but will be made available as online appendix.



(a) **MQ2007**



(b) **MSLR-WEB10k**

Figure 2: Offline performance (nDCG@10) across datasets, under different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

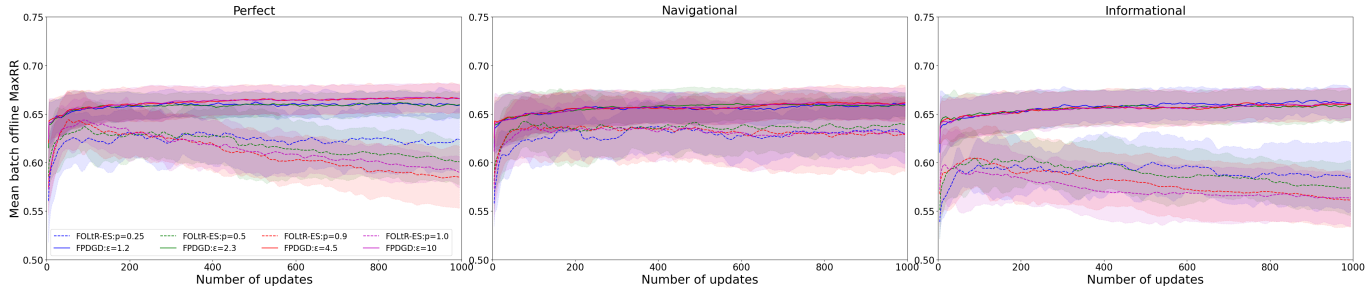
Table 2: Online performance for each dataset under different instantiations of CCM (Table 1). Significant gains and losses of FPDGD over FOLtR-ES (baselines) are indicated by $^{\triangle}$, $^{\triangledown}$ ($p < 0.05$) and $^{\blacktriangle}$, $^{\blacktriangledown}$ ($p < 0.01$), respectively.

		$\epsilon = 1.2$	$\epsilon = 2.3$	$\epsilon = 4.5$	$\epsilon = 10$
		MQ2007			
perfect	FPDGD	296.03 $^{\blacktriangledown}$	296.09 $^{\blacktriangledown}$	313.28	313.26
	FOLtR-ES	316.66	317.93	313.80	312.29
navigational	FPDGD	293.29 $^{\blacktriangledown}$	293.42 $^{\blacktriangledown}$	303.80 $^{\blacktriangledown}$	303.82 $^{\blacktriangledown}$
	FOLtR-ES	319.90	325.04	324.33	323.05
informational	FPDGD	291.84	292.02	301.45 $^{\blacktriangle}$	301.30 $^{\blacktriangle}$
	FOLtR-ES	292.58	294.78	288.57	285.96
		MSLR-10k			
perfect	FPDGD	54.62 $^{\blacktriangle}$	54.61 $^{\blacktriangle}$	54.64 $^{\blacktriangle}$	54.61 $^{\blacktriangle}$
	FOLtR-ES	39.35	40.50	40.87	41.14
navigational	FPDGD	52.33 $^{\blacktriangle}$	52.30 $^{\blacktriangle}$	52.30 $^{\blacktriangle}$	52.29 $^{\blacktriangle}$
	FOLtR-ES	38.55	39.59	40.32	40.47
informational	FPDGD	51.11 $^{\blacktriangle}$	51.16 $^{\blacktriangle}$	51.14 $^{\blacktriangle}$	51.18 $^{\blacktriangle}$
	FOLtR-ES	37.26	37.18	37.21	37.53

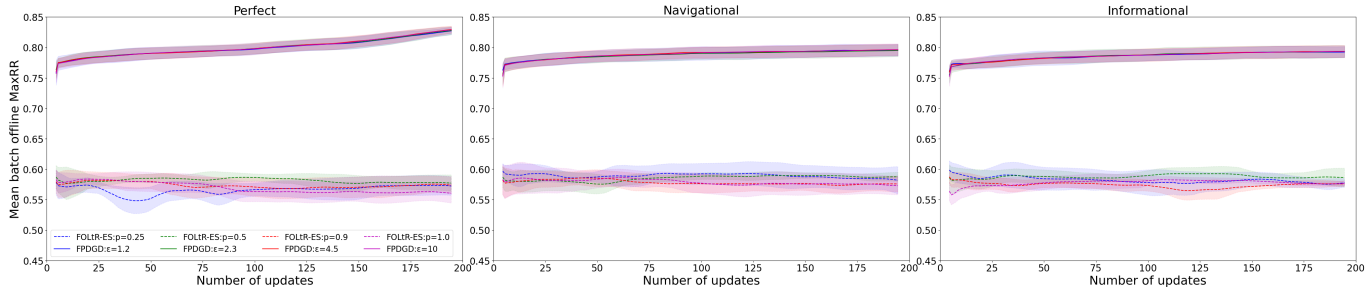
Figure 4 (similar conclusions are identified when considering online performance and MQ2007[2]). We consider two versions of FPDGD: one that uses the introduced differential privacy mechanism with $\Delta = 5$ and $\epsilon = 4.5$ (labelled *w DP* in Figure 4), and one for which differential privacy is turned off (*w/o DP*).

When FPDGD does not rely on differential privacy, the more clients participate to the federated system the better, although differences are minor and not statistically significant. However, if differential privacy is used, when only a small number of clients participates to the federation, then FPDGD performs poorly. We believe that this is so because the differential privacy process adds noise to the local gradient updates; this noise hinders the convergence of the global model when relying on very little interaction data for updating (i.e., 10 clients × 2 batch size = 20 queries). In this case, the noise introduced by the differential privacy process has a large impact. Conversely, differential privacy has little to no impact on performance if numerous clients participate ($|C| = 100, 1000$). Across all settings, we note that past a certain point, adding more clients to the system has little impact – this finding is confirmed by

[2]Not shown here, but will be made available as online appendix.



(a) **MQ2007**



(b) **MSLR-WEB10k**

Figure 3: Offline performance in terms of MaxRR across datasets, under three different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

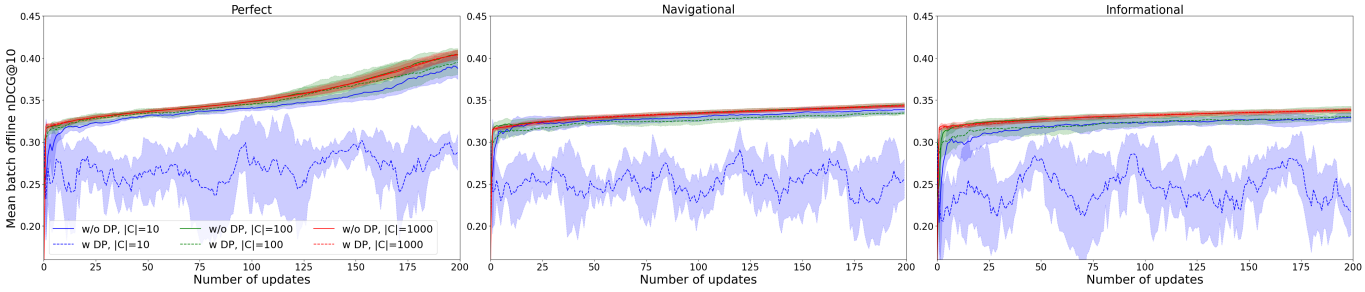


Figure 4: Offline performance on MSLR-WEB10k under three different click models, averaged across all dataset splits and experimental runs, for different number of clients $|C|$. We study FPDGD with differential privacy (*w DP*) and without (*w/o DP*).

prior work in general federated learning, which showed diminishing returns (because of the communication costs) for adding more clients beyond a certain point [16].

5.3 Influence of batch size

We further study the influence of the local batch size ($|B|$) on FPDGD by varying $|B| \in \{2, 4, 8\}$. In our federated setting, each client parallelly conducts FPDGD's model update based on the local interactions B (i.e. queries) and sends the updated model to the server. Note that within each batch of $|B|$ interactions, the client updates the local model $|B|$ times. The differential privacy settings are set to $\Delta = 5, \epsilon = 4.5$.

The offline nDCG@10 performance of FPDGD across different B settings are shown in Figure 5 for MSLR-WEB10k[3] The results suggest that the larger the batch size, the better the offline performance. This trend is especially significant on the perfect click model, although models trained on different batch sizes tend to ultimately converge after enough updates have been made. For the informational click models these differences are less remarked (though still significant), and convergence is long delayed. These results should not be surprising: a larger batch size means that more interactions (queries) have been used during a local update, e.g., 200 interactions × 2 batch size = 400 queries per client vs. 200 interactions × 8 batch size = 1,600 queries per client.

However, when the budget is maintained equal across different batch sizes, and thus smaller batch size means more frequent updates, then the performance obtained across different values of $|B|$ is quite similar (and no significant differences found), although a small batch size guarantees earlier updates, thus resulting in stronger initial performance. This trend is shown in Figure 6 for the offline performance on MSLR-WEB10K. The stronger initial

[3]Similar findings for online performance and MQ2007, not shown here.

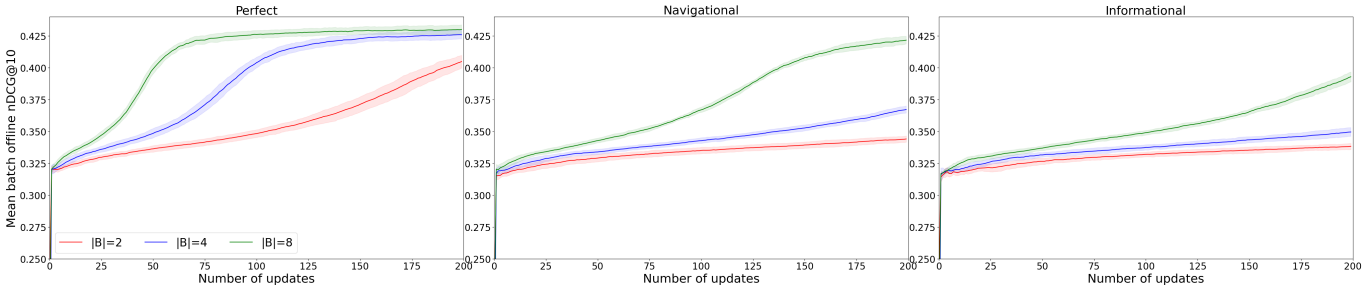


Figure 5: Investigation of the influence of batch size $|B|$ on offline performance for MSLR-WEB10k under three different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

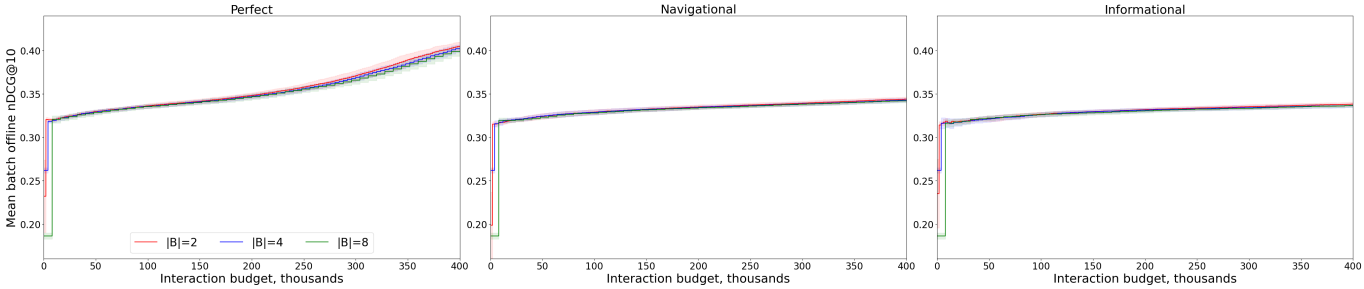


Figure 6: Investigation of the influence of batch size $|B|$ on offline performance for MSLR-WEB10k under three different click models, averaged across all dataset splits and experimental runs, under *fixed budget*.

performance, which is due to the initial model update occurring earlier when the batch size is small, does result in a slightly higher online performance when a small batch size is used.

6 CONCLUSIONS

In this paper, we considered the problem of effective online learning to rank within a privacy-preserving environment, where the user private data (e.g., the data to be searched, their queries and click interactions) never leave the user's own device. To this aim, we cast the Pairwise Differentiable Gradient Descent method (PDGD) [19], which represents the current state-of-the-art in OLTR, to the Federated Learning framework by using the Federated Averaging algorithm, thus proposing the federated Pairwise Differentiable Gradient Descent method (FPDGD). To further enhance the privacy of the method, we overlaid an ϵ-differential privacy method to the proposed FPDGD, thus enforcing a level of privacy guarantee on the local gradient updates. The main features of our method are: (1) we use PDGD as the core OLTR optimisation algorithm – this method is shown to perform well and converge faster than previous methods and is unbiased with respect to users' preferences; (2) the FedAvg framework is easy to implement and it has been shown to converge for non-IID data[4] [12, 15]; (3) we use differential privacy to protect the communication of the local gradient updates between the clients and the server.

We further demonstrate the effectiveness of FPDGD through empirical experiments comparing the method to its non-federated counterpart (PDGD) and the other only federated OLTR method, FOLtR-ES. Compared to the non-federated PDGD, our method converges slower when the same total interaction budget (i.e. number of queries) is considered: in particular, the difference is significant when differential privacy is considered – this is the price to pay for not sharing the clients' data with the central sever (federated setup) and for protecting the local gradient updates (differential privacy). When compared with FOLtR-ES, the proposed method showcases higher performance. This resonates with previous work that showed the performance of FOLtR-ES to be highly variable across datasets and settings, making the method unstable and of risky use in practice [28]. FPDGD instead represents a reliable, stable, and secured alternative to non-federated state-of-the-art OLTR methods.

In future work we plan to evaluate FPDGD on a wider selection of large datasets and explore a large number fo search interactions, to confirm the positive results obtained on MQ2007 and MSLR-WEB10k. The original PDGD method was investigated with respect to both a linear ranker (which we used also for FPDGD) and a neural ranker – in future we plan to investigate the impact a neural ranker has on FPDGD. Finally, we also plan to study the effect on non-IID data on FOLTR and in particular on our FPDGD.

The source code that implements FPDGD, along with experiment code and additional evaluation plots are made available at http://ielab.io/FPDGD.

ACKNOWLEDGMENTS

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).

[4] i.e., data that is non independent and identically distributed; for example in the context of federated OLTR this means that different clients hold data that largely differs in its feature distribution.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] James Allan, Ben Carterette, Javed A Aslam, Virgil Pavlu, Blagovest Dachev, and Evangelos Kanoulas. 2007. *Million query track 2007 overview.* Technical Report. MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE.

[3] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984* (2018).

[4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[5] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.

[7] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients–How easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053* (2020).

[8] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *Proceedings of the second acm international conference on web search and data mining*. 124–131.

[9] Katja Hofmann. 2013. Fast and reliable online learning to rank for information retrieval. In *ACM SIGIR Forum*, Vol. 47. ACM New York, NY, USA, 140–140.

[10] 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*. 183–192.

[11] Katja Hofmann, Shimon Whiteson, and Maarten De Rijke. 2011. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 249–258.

[12] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. 2019. First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715* (2019).

[13] 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.

[14] 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.

[15] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[16] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[17] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*.

[18] Vaikkunth Mugunthan, Anton Peraire-Bueno, and Lalana Kagal. 2020. Privacyfl: A simulator for privacy-preserving and secure federated learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3085–3092.

[19] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *Proceedings of the 2018 ACM on Conference on Information and Knowledge Management*. ACM.

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

[21] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[22] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[23] Mark Sanderson. 2010. *Test collection based evaluation of information retrieval systems*. Now Publishers Inc.

[24] 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.

[25] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[26] Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. 2015. Online Rank Elicitation for Plackett-Luce: A Dueling Bandits Approach. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 604–612.

[27] 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*. 835–844.

[28] Shuyi Wang, Shengyao Zhuang, and Guido Zuccon. 2021. Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study. In *European Conference on Information Retrieval*.

[29] 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.

[30] 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.

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

[32] 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*.