Federated Online Learning to Rank with Evolution Strategies

Eugene Kharitonov
Facebook AI Research
kharitonov@fb.com

ABSTRACT

Online Learning to Rank is a powerful paradigm that allows to train a ranking model using only online feedback from its users. In this work, we consider Federated Online Learning to Rank setup (FOLtR) where on-mobile ranking models are trained in a way that respects the users' privacy. We require that non-privatized user data, such as queries, results, and their feature representations are never communicated for the purpose of the ranker's training. We believe this setup is interesting, as it combines unique requirements for the learning algorithm: (a) preserving the user privacy, (b) low communication and computation costs, (c) learning from noisy bandit feedback, and (d) learning with non-continuous ranking quality measures.

We propose a learning algorithm FOLtR-ES that satisfies these requirements. A part of FOLtR-ES is a privatization procedure that allows it to provide ϵ-local differential privacy guarantees, i.e. protecting the clients from an adversary who has access to the communicated messages. This procedure can be applied to any absolute online metric that takes finitely many values or can be discretized to a finite domain.

Our experimental study is based on a widely used click simulation approach and publicly available learning to rank datasets MQ2007 and MQ2008. We evaluate FOLtR-ES against offline baselines that are trained using relevance labels, linear regression model and RankingSVM. From our experiments, we observe that FOLtR-ES can optimize a ranking model to perform similarly to the baselines in terms of the optimized online metric, Max Reciprocal Rank.

KEYWORDS

online learning-to-rank; federated learning; evolution strategies

ACM Reference Format:
Eugene Kharitonov. 2019. Federated Online Learning to Rank with Evolution Strategies. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290968

1 INTRODUCTION

Mobile devices are a major Internet-access tool, hence numerous ranking models are executed and trained on the data that comes from mobile devices. Typically, to support machine-learned models that operate on user data, data streams are sent to a central server, where they are used to train the models. However, it might be preferable for the data not to leave the user's device, either to preserve the user's privacy or due to limited bandwidth. A learning paradigm where a client only sends what is sufficient to perform the coordinated optimization is called *Federated Learning* [28].

In this work, we extend the Online Learning to Rank problem to the Federated Learning setup. We refer to it as Federated Online Learning to Rank (FOLtR). This problem arises when a mobile application has a machine-learned ranking model and we want to train it using the user interaction feedback. At the same time, we require that non-privatized user data (queries, documents seen and clicked, and their feature representations) never leave the user's device. An example could be a Wikipedia application with its index stored on-device. Whenever the user submits their query to the application, the retrieval and ranking is done locally. Periodically, the application communicates a statistic to the central server and downloads an updated model when it is available. This server optimizes the ranker in a centralized fashion using the statistics collected from the users. Moreover, FOLtR can be combined with a Private Information Retrieval [7, 43] system that provides differential privacy at the retrieval time.

We believe our considered setup is important and general, but it is also challenging. Indeed, any online learning to rank algorithm needs to handle noisy online feedback, non-continuous quality metrics, and explore the space of potential rankers [15]. A FOLtR algorithm also needs to ensure a low communication cost and a guaranteed privacy.

In this paper, we describe a learning algorithm, FOLtR-ES, that follows the Federated Online Learning to Rank scenario. At the core of this algorithm is a zero-order optimization method, Evolution Strategies (ES) [38, 39]. Our proposed algorithm incurs little communication that does not depend on the size of the ranker's model and can optimize non-continuous quality measures. Further, it includes a privatization procedure that provides ϵ-local differential privacy guarantees for the ranking quality metrics that can take finitely many values. In our experimental study, we use publicly available learning to rank datasets MQ2007 and MQ2008 [35] and a widely accepted protocol for simulating user clicks [14]. From our experiments, we observe that our proposed learning algorithm achieves quality that is similar to that of linear baselines (linear regression and RankingSVM) trained against the document-query relevance labels. These relevance labels are unavailable to our algorithm, as it trains directly from the user interactions. Finally, we show that our proposed algorithm can successfully train non-linear ranking models, and it is robust to the noise that comes from the privatization procedure.

Overall, our contributions in this paper are the following:

(1) We propose the FOLtR-ES algorithm which operates in the FOLtR setup and combines Evolution Strategies optimization with a privatization procedure;

(2) We demonstrate that it achieves ϵ-local differential privacy and perform both analytic and simulation analysis of the privacy guarantees;
(3) We experimentally study the optimization performance of the proposed algorithm and its trade-offs.

The code for the experiments in this paper is available at https://github.com/facebookresearch/foltr-es.

2 BACKGROUND

The literature closely related to our work comes from two areas: online learning to rank and federated/privacy-aware learning.

Online learning to rank An initial approach for online learning to rank was based on inferring document preferences from click feedback and use of RankingSVM [17]. Later, work in this area concentrated on optimizing ranked lists assuming some structure on the documents [36, 42] or a fixed stochastic model of the user interaction with a result page [24, 25, 47].

Since absolute click feedback is affected by inter-query and inter-user variance, pairwise interleaving evaluation methods were proposed [5, 17]. Yue and Joachims [45] proposed an algorithm for online learning from pairwise feedback, *Dueling Bandit Gradient Descent* (DBGD). This algorithm became a foundation for multiple applications and extensions. Examples include training parameters of BM25 [41], adapting DBGD to multileaving [40] and Probabilistic Interleaving [32].

There are important differences between the existing online learning to rank literature and our paper. The existing online learning to rank literature only deals with the centralized learning setup, where ranker's training algorithm is aware of the user's queries and clicks. Next, our learning algorithm is free of assumptions about the way the user interacts with the result lists and the quality measures optimized, hence it seamlessly addresses the inter-dependence of the results, the diversity requirements, the position bias, etc.

While recent centralized online learning to rank algorithms, such as Multileave Gradient Descent [40], concentrate on learning with interleaving/multileaving feedback, we believe this setup is limiting. It puts the burden of proving that the interleaving methods are applicable and unbiased on the application owner. Indeed, interleaving methods were only validated in the document [5] and image search [20] verticals of large search engines, and validating those methods for complex layouts (e.g. those considered in [31]) might be problematic. Hence, we argue that the ability to directly optimize the absolute measures of ranking quality (e.g. time before the first click [37]) is of a higher practical interest. We leave the adaptation of FOLtR-ES to the interleaving feedback as a potential direction of future work.

Further, there is a line of work that highlights the connection between Reinforcement Learning and the (online) learning to rank setup (e.g. [15, 16, 31]). In particular, Oosterhuis and de Rijke [31] study the offline learning to rank problem in the presence of complex layouts, where the use of interleaving is impractical. As FOLtR-ES is based on a standard Reinforcement Learning algorithm, ES, we believe that it is a perfect match for applying [31] in the privacy-aware online setup.

Federated and privacy-aware learning The first federated learning algorithm, Federated Averaging, was proposed in [28]. Under this algorithm, each on-device model is firstly updated by several SGD steps, then client models are averaged in a data center. Finally, the clients download the aggregated model. Konečný et al. [22] discuss further steps on improving communication efficiency of federated learning. Further, McMahan et al. [27] studied the problem of learning word-level language models in federated learning setup. In the latter work, both Federated Averaging and Federated SGD (essentially, large-batch SGD) are discussed alongside with their privacy guarantees. Abadi et al. [1] develop tools to make SGD-based optimization of deep learning models (ϵ, δ)-differential private.

We build upon the existing work on Federated Learning and extend it along two directions. Firstly, we extend it to the online learning to rank problem, which has its own challenges: (a) bandit feedback & the need for exploration, and (b) non-continuous optimized measures of quality. Secondly, the previously studied Federated Learning setups operate in the trusted curator ϵ-differential private [10] regime, only providing guarantees against the private data being extracted from the (aggregated) learned models. Consequently, they do not protect against adversaries who can access the communicated messages, e.g. having access to the central optimization server. In contrast, FOLtR-ES provides the client-level ϵ-local differential privacy [8, 11], where non-privatized data never leaves the client's device.

Finally, the notion of differential privacy proved to be useful in Information Retrieval community[1] as a tool for preserving user privacy when publishing query logs [23, 46] and finding the most popular documents [2].

3 FEDERATED ONLINE LEARNING TO RANK WITH EVOLUTION STRATEGIES

We split the description of our proposed FOLtR-ES algorithm in several parts. Firstly, we describe the optimization scenario (Section 3.1). Next, we discuss the use of Evolution Strategies to estimate and communicate gradients (Section 3.2). Finally, we introduce a way to ensure the users' privacy (Section 3.3).

3.1 Optimization Scenario

The optimization scenario we consider is not different from one used in Federated SGD [27]. We assume that the ranking model is applied locally on the client, and there are means to ensure the users' privacy at the retrieval time, either by doing it entirely on-device or by using Private Information Retrieval system [7, 43].

After installing an application, the client downloads the most recent ranking model. After the client performed B interactions[2] ($B = 1, 2, \ldots$), the performance metrics for these interactions are averaged locally and a message sent to the centralized server. After collecting messages from N clients, the server combines them in a single gradient estimate $\hat{g}$ and performs an optimization step, forming a new model. Finally, the clients download the latest model from the server.

[1]See also https://privacypreservingir.org/
[2]A single event of submitting a query and interacting with its results we call an *interaction*.

3.2 Getting a Gradient Estimate

Consider a ranking model that comes from a parametric family indexed by a vector $\theta \in \mathbb{R}^n$. For instance, it could be a neural ranking model, similar to RankNet [3], with the parameter vector encoding weights and biases of all layers of the network.

Whenever a user u has an interaction a with the ranker's output, they experience ranking quality f (note that f is not necessarily continuous w.r.t. θ). We are interested in finding the model vector θ^* that maximizes the mean of the metric f across all interactions a of all users u:

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

Under Evolution Strategies (ES) [39], instead of considering a single model parameter vector θ, we consider a population of those, represented by a distribution with a density function $p_\phi(\theta)$, which is in turn parameterized by ϕ. Next, instead of optimizing $F(\theta)$, one looks for the distribution parameter ϕ that maximizes the expectation of the metric across the population:

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

As the mean of a random variable is always smaller or equal than its maximal value, the expectation (2) is a lower bound of the value of $F(\theta)$ within the support of the distribution with density $p_\phi(\theta)$.

The gradient g of the objective Equation (2) is obtained in a fashion similar to REINFORCE [44]:

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

We have a freedom of choosing $p_\phi(\theta)$ and a typical approach is to use a fully factorized Gaussian distribution[3] with a fixed diagonal covariance matrix $\sigma^2 I$ and the mean ϕ [39]. Equation (3) obtains a simple form:

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

Further, by plugging the definition of $F(\theta) = \mathbb{E}_u\ \mathbb{E}_{a|u,\theta}\ f(a;\theta,u)$ from Equation (1), we get:

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

We will sample θ independently on each client, hence we can rearrange expectations:

$$g = \mathbb{E}_u \mathbb{E}_{\theta \sim p_\phi(\theta)}\left[\left(\mathbb{E}_{a|u,\theta}\ f(a;\theta,u)\right)\cdot\frac{1}{\sigma^2}(\theta-\phi)\right] \tag{6}$$

The estimate of the gradient $\hat{g} \approx g$ from Equation (6) can be obtained as follows: (a) each client u randomly generates a pseudo-random seed s, (b) uses this seed s to sample a perturbed model $\theta_s \sim \mathbb{N}(\phi, \sigma^2 I)$, (c) estimates the expected loss $\hat{f} \approx \mathbb{E}_{a|u,\theta_s}\ f(a;\theta_s,u)$ by averaging over B interactions a, (d) communicates the tuple $(s, \hat{f})$ to the server. Now, given the tuple $(s, \hat{f})$, the central server can generate the same model θ_s and, knowing ϕ, re-calculate the inner expectation Equation (6) and average it across the users, getting the estimate $\hat{g}$ of g.

The communication via random seeds is often used in the ES literature [39]. Assuming that s and $\hat{f}$ take 4 bytes each, the whole message size would be 8 bytes. Importantly, this size is independent of the size of the model vector θ.

Another ES trick [26, 39] we use is to reduce the variance of the gradient estimates by means of antithetic variates. To do so, one uses symmetrically opposite perturbations $\theta + \sigma \cdot \nu_s$ and $\theta - \sigma \cdot \nu_s$ ($\nu_s \sim_s \mathbb{N}(0,1)$) in a single batch. This would require sending two floats (the mean metric value for each direction) per message.

We note that the client-side computational overhead is small even for large and complex models, as it does not require performing backpropagation on-device to calculate gradients as in other Federated Learning setups [27, 28].

The above described algorithm only differs from the standard ES by the fact that the random seeds are generated by clients. The client-side generation fits FOLtR setup better than the centralized generation [39], as requires less synchronization.

3.3 Privatization procedure

Intuitively, as only a randomly generated pseudo-random seed and a value of the function f is shared with the central server, our scheme gives a user a plausible deniability of the submitted query and hence the clicked results. In the following, we aim to formalize and amplify these guarantees by developing a *privatization* procedure [8] that injects privatization noise in the communicated values.

Clearly, this procedure has to depend on the nature of the application and the metric used. We assume that the metric used takes a finite number of values, $f_0, f_1, \ldots, f_{n-1}$. Popular metrics [5, 36] such as the position of the first or the last click, their reciprocal ranks, and clickthrough rate fall into this category. Continuous metrics can be discretized. Alternatively, our privatization procedure can be extended for the continuous case by e.g. using Laplace noise [11]. We leave that as a direction of future work.

The privatization we use is akin to the randomized response technique [12]. W.l.o.g. we assume that after the user interaction occurred, we obtain the value of the metric f_0. Instead of transmitting this value, we toss a coin. If we obtain heads (with probability p), we send the true value of the metric, f_0. Otherwise, with probability $1-p$, we uniformly at random select a value $\hat{f}$ out of $n-1$ values which are *different* from f_0, and send it.

The expected value of the privatized metric f_p, $\mathbb{E}(f_p|f_0)$ is:

$$\mathbb{E}(f_p|f_0) = p\cdot f_0 + \frac{1-p}{n-1}\sum_{i\neq 0} f_i = p\cdot f_0 - \frac{1-p}{n-1}f_0 + \frac{1-p}{n-1}\sum_i f_i$$
$$= f_0\cdot\left[p - \frac{1-p}{n-1}\right] + C = f_0\cdot\frac{pn-1}{n-1} + C, \tag{7}$$

where C is some constant that is independent from the user interactions. From Equation (7) we see that the expectation of the privatized metric f_p is equal to the original metric, shifted by C and scaled by $(pn-1)/(n-1)$. This leads us to the observation:

[3]The convergence of this type of optimization algorithms is studied by Nesterov and Spokoiny [29].

Fact 1. *When $p > 1/n$, the parameter vector θ^* that maximizes Equation (2) with the expectation of the privatized metric $\mathbb{E}(f_p|f)$ as a metric, would also maximize it for the non-privatized metric f.*

In practice, we do not calculate the expectation of the privatized metric separately, using a single-sample 'estimate' per interaction instead. In the following, we always assume that $p > 1/n$.

4 PRIVACY ANALYSIS

In the Federated Learning literature, privacy is often studied through the lenses of ϵ-differential privacy [10, 27, 28]. This approach considers privacy guarantees *after the data from the clients was aggregated* by the central server (curator). This leaves a possible attack direction where the adversary has an access to the central server itself and can intercept messages, e.g. being a malicious employee of the company. We resort to a more strict notion of ϵ-local differential privacy [8, 11, 12]. The main difference is that the privacy is considered at the level of an individual client, not at the aggregated level. Any ϵ-local differential private mechanism is also necessarily ϵ-differential private [11, Chapter 12].

We consider an adversary who aims to learn if the user u has submitted a query q_1 or a query q_2. This adversary has intercepted the content of the user's communicated feedback, i.e. the pseudo-random seed and the value of the quality metric of the user u. For simplicity, we assume that (a) the feedback represents a single query interaction, and (b) given the intercepted random seed, the adversary is able to reconstruct the result lists for q_1 and q_2, since they have an access to the central model. Moreover, the adversary has a 'freedom' to select q_1 and q_2 to be the most 'dissimilar' queries. We believe that these assumptions simplify the adversary's task.

Definition 1. *An algorithm $\mathcal{A}$ is called ϵ-local differential private [8], if for any two inputs (i.e. queries) q_1 and q_2 and any possible output $o \in Im(\mathcal{A})$, the following relation holds:*

$$P(\mathcal{A}(q_1) = o) \leq e^{\epsilon} \cdot P(\mathcal{A}(q_2) = o) \tag{8}$$

Lower values of ϵ imply higher levels of privacy.

In our case, $\mathcal{A}$ outputs a tuple o of the pseudo-random seed s and the metric value $\tilde{f}$. Under our assumptions, the pseudo-random seed is used by the adversary to recover the ranked lists for q_1 and q_2. Since it is random itself and independent from the query, we are interested in finding how much a value of the metric can reveal:

$$\epsilon = \max_{q_1, q_2, \tilde{f}} \log \frac{P\left(\tilde{f} \mid q_1\right)}{P\left(\tilde{f} \mid q_2\right)} \tag{9}$$

The ratio in Equation (9) would be maximized if we can select q_1, q_2, and $\tilde{f}$ in such a way that $P(\tilde{f}|q_1)$ achieves its maximum and $P(\tilde{f}|q_2)$ is simultaneously minimized. From our privatization procedure in Section 3.3, it actually follows that no metric value $\tilde{f}$ can be transmitted with probability $P(\tilde{f}|q)$ higher than p or lower than $(1-p)/(n-1)$ (we assume that $p > 1/n$).

To illustrate that, we choose a query q and assume that some metric value f_0 is generated by the user with probability p_0, $0 \leq p_0 \leq 1$. Then, f_0 is transmitted with probability $P_T(f_0|q) = p_0 \cdot p + (1-p_0) \cdot (1-p)/(n-1)$. It is clear that, as p_0 varies, $P_T(f_0|q)$ is bounded between p and $(1-p)/(n-1)$. Next, we notice that p is the upper bound: $p > 1/n = (n-1)/(n(n-1)) = (1-1/n)/(n-1) > (1-p)/(n-1)$.

We conclude that $P_T(f_0)$ is maximized when p_0 is equal to 1 and minimized when p_0 is 0, i.e. $(1-p)/(n-1) \leq P_T(f_0) \leq p$.

Putting the obtained bounds in Equation (9), we obtain:

$$\epsilon \leq \log \frac{p}{(1-p)/(n-1)} = \log \frac{p(n-1)}{1-p} \tag{10}$$

The bound obtained in Equation (10) effectively corresponds to the case where the user would *never* produce f_0 for q_2 ($p_0 = 0$) and will *always* produce f_0 for q_1 ($p_0 = 1$). Due to the privatization, however, the user still gets privacy guarantees. The assumption of existence of such q_1, q_2, and f_0 might be too conservative in practice. In Section 6 we run a simulation study to check the bound (10).

We conclude that, given our assumptions, our privatization scheme allows achieving at least $\log[p(n-1)/(1-p)]$-local differential privacy, with p being the privatization scheme parameter ($p > 1/n$) and n equal to the number of possible values of the metric. Higher the number of values n the metric can return and higher the parameter p, lower the level of privacy we get. At the same time, as p approaches $1/n$ from above, the privacy loss ϵ tends to zero.

Finally, we note that our analysis ignores that the same queries and results might repeat across interactions. Accounting for that requires a more sophisticated models of the user's behaviour.

Table 1: Clicking $p(c|r)$ and stopping $p(s|r)$ probability parameters for the three click model instantiations we use. The parameters are identical to those used in [14].

	Click probability			Stopping probability		
	r = 0	r = 1	r = 2	r = 0	r = 1	r = 2
Perfect	0.0	0.5	1.0	0.0	0.0	0.0
Navigational	0.05	0.5	0.95	0.2	0.5	0.9
Informational	0.4	0.7	0.9	0.1	0.3	0.5

5 EVALUATION SETUP

The main goals of our evaluation study are: (a) to investigate how various algorithm trade-offs influence the learning performance of FOLtR-ES, and (b) to find if FOLtR-ES can train highly effective rankers. In the evaluation, we aim to closely repeat the operational scenario for the FOLtR setup: the perturbed models are serving the client requests, the clients communicate messages to the server, which performs the optimization. Unfortunately, there is no publicly available search log datasets that would allow online learning to rank with real data. Thus, we chose to simulate the user behavior using publicly available offline learning to rank datasets. This approach is widely used in the online learning to rank literature [14, 40, 41].

We are mainly interested in the online learning performance, i.e. how the user experience changes during the course of learning. Indeed, a specifics of online learning to rank is that the optimization is performed against the actual users. Hence, we want not to damage the user experience during the training itself. This important property of online learning to rank was highlighted, for instance, in [30], where some form of online loss is controlled for. At the same time, the generalization capabilities of the trained models are less important, as they are trained and deployed for the same population of the users.

Simulating users At each step, each user samples B queries uniformly at random from a dataset of queries, ranks the available results according to the locally perturbed model, and simulates clicks on them. Next, the quality measure values are aggregated across B queries and sent to the optimization algorithm with the corresponding pseudo-random seed. When antithetic variates are used, the first and the second half of queries are ranked with antithetic perturbations. In this case, we send one averaged metric value per direction. In all experiments, we simulate 2,000 of clients.

Simulating user clicks Given a ranked list of documents with provided relevance labels, we simulate the user by running an instance of Cascade Click Model (CCM) [13]. While more advanced click models exist (e.g. [6]), we use it to be compatible with an existing online learning to rank literature [14, 32, 40].

Firstly, we cut top ten results from the ranked list, so that a typical ranking scenario is represented. Next, we examine the results from top to bottom, one-by-one. When a document with relevance label r is examined, we sample a clicking event with probability of $P(c|r)$. After a click, the user stops with probability $P(s|r)$ and continues otherwise. This model is parameterized by clicking and stopping probabilities, $P(c|r)$ and $P(s|r)$, specified for three relevance levels r, $r \in \{0, 1, 2\}$. Following [14], we consider three instantiations of the click model, namely, *Perfect*, *Navigational*, and *Informational* models. These models differ in the amount of noise in the click feedback and model different user behavior. The parameters of these click models are reported in Table 1.

Models We experiment with two ranking models. The first model is a linear ranker without the bias term. The second model is a neural network with a single hidden layer of size 10 and a ReLU non-linearity. All parameters of the models are initialized with zeros.

Baselines Since to the best of our knowledge no other algorithms were proposed to operate in FOLtR, we use baselines as means to set a level of 'reasonable' performance on the dataset.

The first baseline, MSE, is a linear regression model (as implemented in scikit-learn [34]) that is fit to predict relevance of a document to a query given the feature representation of the query-document pair. The model is obtained by minimizing the mean squared loss across all query-document pairs in the training set. The second baseline is a RankingSVM model [17, 18]. We use the author's implementation.[4]

These baselines are trained using relevance labels, that are unavailable for FOLtR-ES. Instead, FOLtR-ES has to infer the relative usefulness of the ranked lists from the privatized click feedback.

The state-of-the-art performance in offline learning to rank is achieved by gradient boosted regression trees (GBRT) [4] that are trained against massive datasets with relevance labels. Typically, that implies large-scale centralized applications, such as ranking models for major search engines. As GBRT rankers rely on a completely different class of the models and a usage scenario, we do not consider a GBRT baseline.

[4]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 2: Estimating the privacy loss ϵ for MaxRR (smaller is better). The values obtained from Equation (10) and the estimates for the Perfect model coincide due to the properties of the model.

	Privatization parameter p					
	0.25	0.5	0.75	0.90	0.95	0.99
Perfect (= Theoretical)	0.51	1.61	2.71	3.81	4.55	6.20
Navigational	0.47	1.52	2.58	3.65	4.39	6.00
Informational	0.28	1.00	1.70	2.56	3.13	4.39

Datasets We use public datasets, MQ2007 and MQ2008 [35]. MQ2007 (MQ2008) dataset contains 1692 (784) unique queries, 69623 (15211) relevance labels. Each query-document pair is represented by 46 features. The datasets are provided with five splits (20-20-60), which we use in our experiments. We do not apply any additional preprocessing, such as feature normalization.

Quality Metric Generally, FOLtR-ES allows us to optimize any absolute measures of ranking quality. This gives us a freedom to optimize the online quality measures we actually care about. An example could be the ground-truth online measures used in the A/B testing, such as user engagement metrics or their combinations [9, 19]. However, we cannot generate realistic session or multi-session user data using publicly available data, hence we restrict ourselves to the measures that operate on the single result list level. Out of the metrics discussed and found to be sensitive in [5], we select the reciprocal rank of the highest clicked result (MaxRR) [36] in an interaction. To achieve a higher level of privacy, we only consider clicks on the first ten results ($n = 11$, including the non-click event).

Since our setup assumes training using the feedback of the actual user population, we report the learning curves that illustrate the changes of the users' experience over the training time. As a result, we both optimize and report the mean value of quality metrics over the batch of interactions. Since the MSE and SVMRank baselines are trained on the entire training set, we simply report their average performance.

Optimization As an optimizer, we use Adam [21] with the default parameters, as implemented in Pytorch [33]. We do not apply any regularization. We fix σ to be 10^{-2} and never tune it. We turn the regularization off for MSE and set the RankingSVM's parameter C to 10^3, effectively allowing them to overfit to the training dataset. It is not an issue in the case where only one population of users exists and we are interested in a high level of performance on it.

6 RESULTS AND DISCUSSION

We split the evaluation in three logical parts. At first, we illustrate how tight is the theoretical bound on the value of ϵ that we obtained in Section 4. Next, we study the influence of the FOLtR-ES parameters on its performance: the use of antithetic variates, the number of interactions in the on-device aggregated feedback B, and the privatization parameter p. We also fix our ranker to be linear. In these exploratory experiments, we use the validation query set from Fold 1 of MQ2007. After running these exploratory experiments, we compare FOLtR-ES to the baselines.

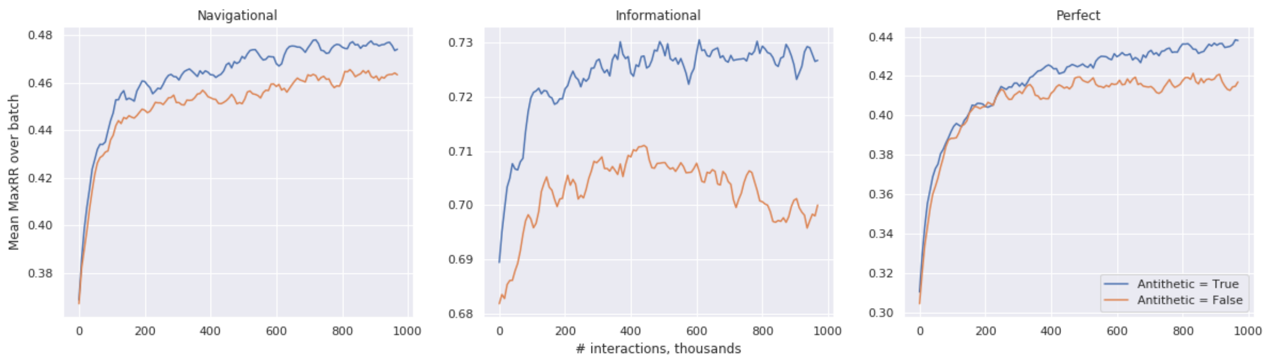

Figure 1: Influence of antithetic variates on the optimization performance (measured by mean batch MaxRR). The learning curves are smoothed by applying sliding average over four measurements.

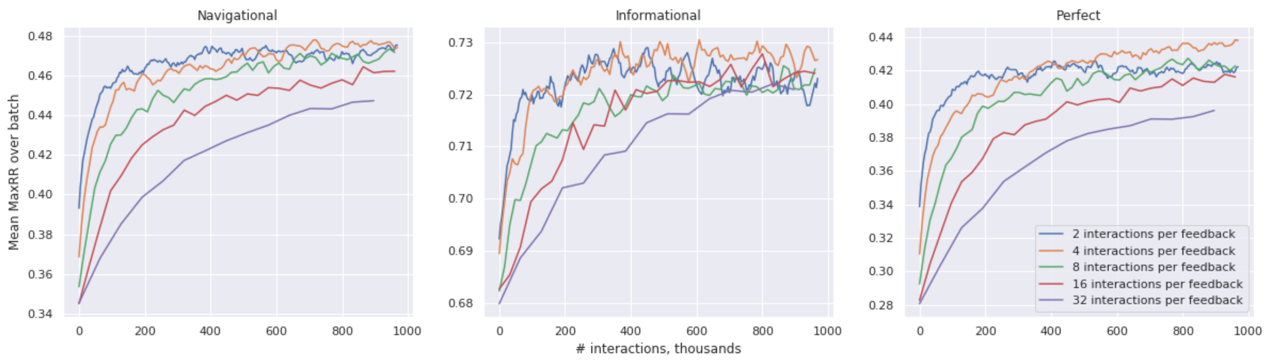

Figure 2: Sensitivity of the mean batch MaxRR to the number of sessions aggregated into a single communicated message. The learning curves for the cases of 2, 4, and 8 interaction per feedback message are smoothed by applying the sliding average over 8, 4, and 2 measurements, respectively.

6.1 How tight is the ϵ estimate?

To investigate empirically how close the obtained estimate of ϵ (Equation (10)) is to the real values, one would need an access to a dataset of queries q along with their metric distributions $p(f|q)$. Reliably estimating this distribution for MaxRR would be only possible for a few of very frequent queries. Hence, we resort to simulating the user click interactions. Conditioned on the relevance labels of the documents $\{0, 1, 2\}$ in a ranked list, *Navigational*, *Informational*, and *Perfect* user models output distributions of clicks on the positions. In turn, we use these distributions to determine the distribution of the values of MaxRR.

To generate these distributions, we enumerated all $3^5 = 243$ possible placements of relevance labels $\{0, 1, 2\}$ in a list of length 5. Then, we considered all possible pairs of these ranked lists (pretending that they belong to different queries), and selected those that maximize Equation (9) for a particular combination of p and the user model. We report the resulting ϵ values for MaxRR in Table 2.

From Table 2 we see that the theoretical estimate we obtained is close to the estimated value for the Navigational model. However, for Informational it is very far from being tight. Consequently, when the users behave close to the Informational model, the privacy guarantees are better than those we obtained in Section 4.

6.2 Hyperparameter analysis

Antithetic variates To investigate the impact of antithetic variates on the training performance, we train the linear ranker twice, with and without variance reduction. We set $B = 4$ for the both trainings and turn off the privatization. When the antithetic variates are used, first $B/2$ and the last $B/2$ of interactions are performed with antithetic perturbations. The training trajectories are reported in Figure 1. We observe that the antithetic variates are indeed useful and considerably speed up the optimization, in particular, for the Informational click model. In the remaining experiments, we always use antithetic variates.

Feedback size sensitivity Next, we investigate how the optimization quality is affected by the number of interactions B aggregated in a single communication message. With large B, the variance due to the query variability vanishes. On the other hand, that increases the overall batch size, which could be detrimental

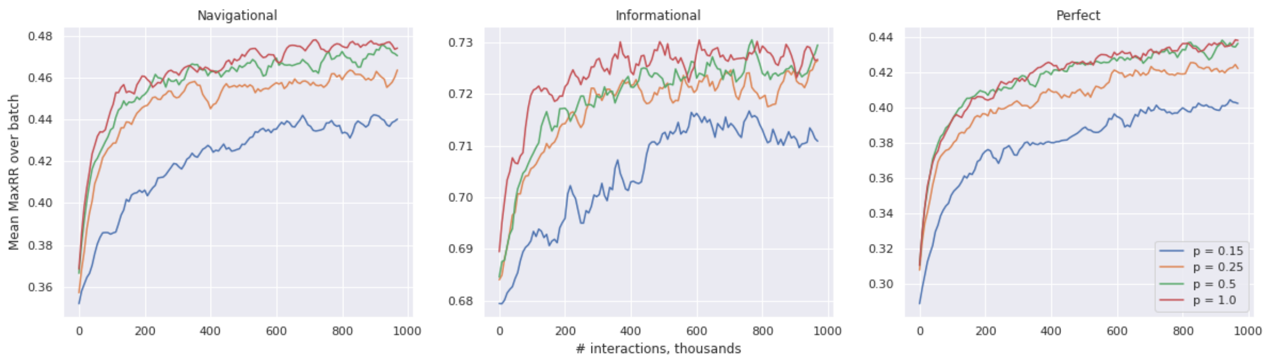

Figure 3: Trade-off between the privacy and the optimization performance. Higher values of p correspond to lower privacy. The learning curves are smoothed by applying sliding average over four measurements.

for the stochastic optimization. Again, we turn the privatization procedure off. We vary B across $\{2, 4, 8, 16, 32\}$.

The learning curves are reported in Figure 2. For each of the considered user click models (Navigational, Perfect, and Informational), we observe that the metric values for all values of B are close, within 0.05 absolute points. At the same time, larger values of B ($B = 16$ and $B = 32$) lead to slower optimization in comparison to smaller values $\{2, 4, 8\}$. For simplicity, in the following experiments we use $B = 4$, as it is among the top-performing values for the Navigational and Informational models, and the performs best in the case of the Perfect model.

Privatization vs. optimization speed trade-off While our privatization procedure should not affect the optimal solution (Section 3.3), it necessarily slows down the optimization due to the increased noise. To study the privatization vs. optimization trade-off, we vary the privatization parameter p across $\{0.15, 0.25, 0.5, 1.0\}$ and train as before (use antithetic variates, $B = 4$). While we train models using privatized feedback, we the reported optimization performance is non-privatized, as it reflects the actual user experience. We report the results in Figure 3.

As expected, we observe that the higher values p perform better. Surprisingly, there is very little difference in performance between $p = 0.5$ and $p = 1.0$. When p is set to low values, $p = 0.15$ and $p = 0.25$, the convergence is considerably slowed down. However, the rankers achieve reasonable performance. We believe that is interesting, as in these regimes the true metric values are reported only in 15% ($p = 0.15$) and 25% ($p = 0.25$) of the messages.

6.3 Ranking quality

In the next experiment, we aim to study the ranking performance of FOLtR-ES. We use antithetic variates and $B = 4$. For this experiment, we to set p to 0.9. The FOLtR-ES models are trained on the privatized feedback, but we report the non-privatized values, as they reflect what the users actually experience. We test two models, the linear and two-layer neural net rankers. As before, we fix $\sigma = 10^{-2}$. To improve readability, we smooth the learning curves by applying sliding average over four measurements.

The learning curves across five splits of MQ2007 and MQ2008 datasets are reported in Figure 4 and 5, respectively. We observe that, under the Navigational and Perfect users models, the FOLtR-ES-trained rankers perform at least as good as MSE on MQ2007 and as good as RankingSVM on MQ2008. Moreover, often only relatively few interactions (less than 500K) are needed to get a well-performing ranker. Considering the Informational user model, we notice a larger difference between the rankers. Across all folds and datasets, two-layer FOLtR-ES ranker achieves higher performance than linear, one-layer ranker. In some cases (Folds 1 & 2, MQ2008) it achieves the level of the best baseline. Since the Informational

Table 3: The expectation of the MaxRR metric, averaged over dataset splits. In bold are values that achieve a stat. sig. difference between the two-layer model trained FOLtR-ES and RankingSVM (paired t-test, $p < 0.05$).

	Navigational	Informational	Perfect
	MQ2007, train		
MSE	0.470	0.732	0.424
SVMRank	0.480	**0.737**	0.436
FOLtR-ES Linear	0.469	0.715	0.411
FOLtR-ES Two-Layer	0.479	0.732	0.434
	MQ2007, test		
MSE	0.462	0.727	0.415
SVMRank	0.477	0.736	0.432
FOLtR-ES Linear	0.467	0.712	0.402
FOLtR-ES Two-Layer	0.474	0.730	0.428
	MQ2008, train		
MSE	0.457	0.730	0.408
SVMRank	0.471	0.736	0.424
FOLtR-ES Linear	0.469	0.720	0.415
FOLtR-ES Two-Layer	**0.481**	0.734	**0.430**
	MQ2008, test		
MSE	0.450	0.726	0.400
SVMRank	0.470	0.735	0.422
FOLtR-ES Linear	0.458	0.715	0.401
FOLtR-ES Two-Layer	0.464	0.731	0.414

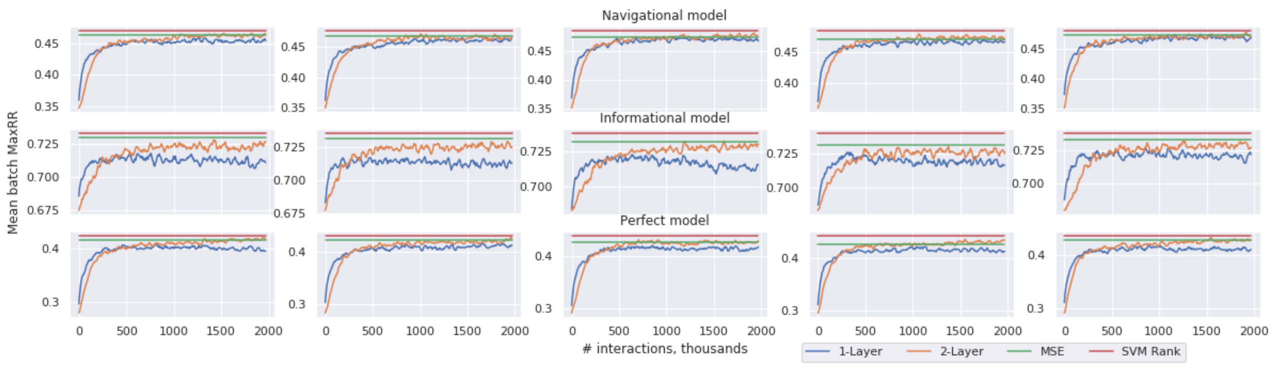

Figure 4: Mean batch MaxRR. Each column corresponds to the dataset split (MQ2007).

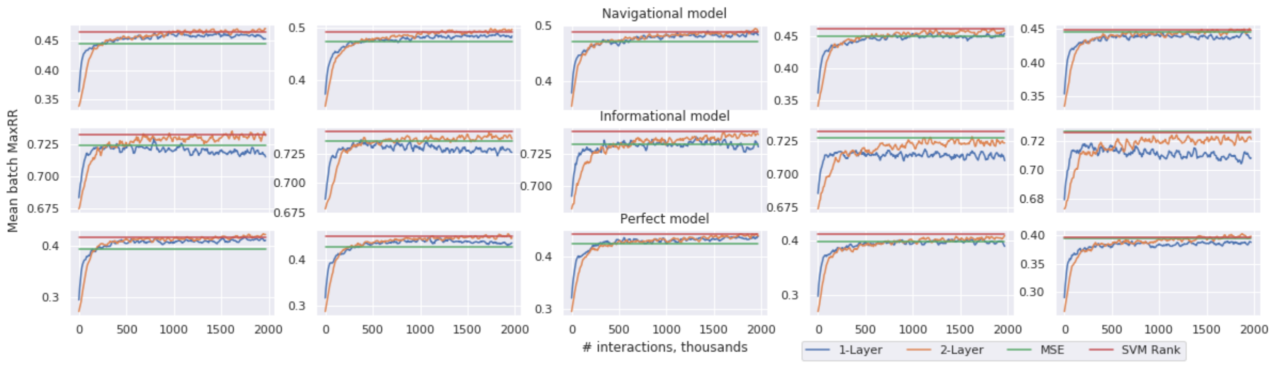

Figure 5: Mean batch MaxRR. Each column corresponds to the dataset split (MQ2008).

model simulates the most noisy user feedback, we hypothesize that the combination of the exploration, privatization, and click noise made the optimization problem harder in this case.

As we discussed in Section 5, the on-training performance is more important than the generalization capabilities of the trained models. However, it is still interesting to check if the trained models can generalize. To study this, we run the following experiment. For each of the reported learning curves, we take the 'final' model that corresponds to their (unperturbed) population means θ at the end of the training. Next, we use these models to get the ranked lists for all queries in the training and test sets for their corresponding folds. Further, we apply the user click model that was used for training to calculate the expectation of the MaxRR metric for these ranked lists. Finally, we average these values over all queries. We report the resulting metric values in Table 3.

From Table 3 we firstly observe that, again, RankingSVM outperforms MSE in every case. Similarly, the two-layer FOLtR-ES model also consistently get higher expected MaxRR than the linear model. Out of $3 \times 4 = 12$ user model and query set combinations, in three cases there are statistically significant differences between SVM-Rank and the two-layer model ($p < 0.05$, paired t-test). Out of those, Two-Layer FOLtR-ES performs better in two combinations. Overall, this supports the observation that FOLtR-ES can train models of the quality similar to those of the baselines.

We believe the results of this experimental comparison are non-trivial. Indeed, the baselines have access to noise-free document labels, while FOLtR-ES has to deal with (a) click and query sampling noise, (b) privatization noise, (c) non-continuous evaluation metric.

In this section, we investigated the performance of FOLtR-ES when varying a set of parameters: the use of antithetic variates, the number of locally aggregated sessions per feedback message B, the privatization parameter p. We compared FOLtR-ES-trained linear and two-layer models against two baselines, MSE and RankingSVM. Our experiments demonstrate that FOLtR-ES can train models that perform similarly to the baselines that have access to noise-free relevance labels.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we formulated the Federated Online Learning to Rank problem (FOLtR). In this problem, we train a ranker using online user feedback, in such a way that only minimal privatized data is communicated from the user device to the central server. In Section 3, we proposed an algorithm FOLtR-ES, that operates in this scenario. FOLtR-ES is a practical, easy-to-implement algorithm that has a set of useful properties. It communicates only small messages to the central server (8-12 bytes), never transmits non-privatized user data, and can optimize non-continuous absolute metrics. In

Section 4, we demonstrated that for a metric that takes n values, one can achieve $\log[p(n-1)/(1-p)]$-local differential privacy, with p controlling how often the true metric value is transmitted. We additionally performed a simulation study (Section 6), showing that this estimate is tight for the users that behave close to the Navigation model, but is less so for the Informational model.

We studied the optimization performance of the proposed algorithm and its modifications (Section 6). We used two publicly available datasets (MQ2007 & MQ2008). To simulate the user's click behavior, we followed a widely accepted click simulation approach.

Firstly, we demonstrated that, given a fixed number of interactions, the use of antithetic variates allows us to train more effective models. Next, we found that FOLtR-ES is relatively insensitive to the number of interactions used in the client-side aggregation. Further, we demonstrated that FOLtR-ES is robust to the privatization noise, being able to train rankers even when only 15% of the communicated measurements are true. Finally, we compared the linear and two-layer models trained by FOLtR-ES against RankingSVM and MSE baselines that are trained using relevance labels. We found out that FOLtR-ES-trained models perform close to the baselines both in online performance and on the hold-out queries.

We believe this research opens a set of interesting research questions. There are numerous possibilities to improve FOLtR-ES in terms of improving privacy and convergence speed, extend it to different types of feedback (e.g. interleaving). However, a bigger question is: what is needed to build practical differentially private search systems and how can we build them?

Acknowledgements The author is immensely grateful to Nicolas Usunier for his thoughtful feedback.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *ACM CCS*.

[2] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. 2017. BLENDER: enabling local search with a hybrid differential privacy model. In *Proc. of the 26th USENIX Security Symposium*. 747–764.

[3] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *ICML*.

[4] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*.

[5] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *TOIS* (2012).

[6] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *WWW*.

[7] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private information retrieval. In *FOCS*.

[8] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html.

[9] Georges Dupret and Mounia Lalmas. 2013. Absence time and user engagement: evaluating ranking functions. In *WSDM*.

[10] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*.

[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends Theoretical Computer Science* (2014).

[12] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *SIGSAC*.

[13] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *WSDM*.

[14] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*.

[15] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval* 16, 1 (2013), 63–90.

[16] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. *arXiv:1803.00710* (2018).

[17] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*.

[18] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *SIGKDD*.

[19] Eugene Kharitonov, Alexey Drutsa, and Pavel Serdyukov. 2017. Learning sensitive combinations of A/B test metrics. In *WSDM*.

[20] Eugene Kharitonov, Craig Macdonald, Pavel Serdyukov, and Iadh Ounis. 2015. Generalized team draft interleaving. In *CIKM*.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).

[22] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv:1610.05492* (2016).

[23] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. 2009. Releasing search queries and clicks privately. In *WWW*.

[24] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading bandits: Learning to rank in the cascade model. In *ICML*.

[25] Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvari. 2018. TopRank: A practical algorithm for online stochastic ranking. *arXiv:1806.02248* (2018).

[26] Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search provides a competitive approach to reinforcement learning. *arXiv:1803.07055* (2018).

[27] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private language models without losing accuracy. *arXiv:1710.06963* (2017).

[28] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629

[29] Yurii Nesterov and Vladimir Spokoiny. 2011. *Random gradient-free minimization of convex functions.* Technical Report. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE).

[30] Harrie Oosterhuis and Maarten de Rijke. 2017. Balancing speed and quality in online learning to rank for information retrieval. In *CIKM*.

[31] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for Relevance and Display Preferences in Complex Presentation Layouts. *arXiv:1805.02404* (2018).

[32] Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke. 2016. Probabilistic Multileave Gradient Descent. In *Advances in Information Retrieval*.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

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

[36] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *ICML*.

[37] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How does clickthrough data reflect retrieval quality?. In *CIKM*.

[38] I. Rechenberg and M. Eigen. 1973. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. *Frommann-Holzboog Stuttgart* (1973).

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

[40] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *WSDM*.

[41] Anne Schuth, Floor Sietsma, Shimon Whiteson, and Maarten De Rijke. 2014. Optimizing base rankers using clicks. In *ECIR*.

[42] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. 2013. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *JMLR* 14 (2013).

[43] Raphael R Toledo, George Danezis, and Ian Goldberg. 2016. Lower-cost epsilon-private information retrieval. *arXiv preprint arXiv:1604.00223* (2016).

[44] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* (1992).

[45] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*.

[46] Sicong Zhang, Hui Yang, and Lisa Singh. 2015. Applying epsilon-differential private query log releasing scheme to document retrieval. In *SIGIR Workshop on PIR*.

[47] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online learning to rank in stochastic click models. *arXiv:1703.02527* (2017).