# FedPS: A Privacy Protection Enhanced Personalized Search Framework

Jing Yao
School of Information
Renmin University of China
Beijing, China
jing_yao@ruc.edu.cn

Zhicheng Dou
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
dou@ruc.edu.cn

Ji-Rong Wen
Beijing Key Laboratory of Big Data Management and Analysis Methods, Key Laboratory of Data Engineering and Knowledge Engineering, MOE
Beijing, China
jirong.wen@gmail.com

## ABSTRACT

Personalized search returns each user more accurate results by collecting the user's historical search behaviors to infer her interests and query intents. However, it brings the risk of user privacy leakage, and this may greatly limit the practical application of personalized search. In this paper, we focus on the problem of privacy protection in personalized search, and propose a privacy protection enhanced personalized search framework, denoted with **FedPS**. Under this framework, we keep each user's private data on her individual client, and train a shared personalized ranking model with all users' decentralized data by means of federated learning. We implement two models within the framework: the first one applies a personalization model with a personal module that fits the user's data distribution to alleviate the challenge of data heterogeneity in federated learning; the second model introduces trustworthy proxies and group servers to solve the problems of limited communication, performance bottleneck and privacy attack for FedPS. Experimental results verify that our proposed framework can enhance privacy protection without losing too much accuracy.

## CCS CONCEPTS

• **Information systems** → **Personalization**.

## KEYWORDS

personalized search; privacy protection; federated learning

**ACM Reference Format:**
Jing Yao, Zhicheng Dou, and Ji-Rong Wen. 2021. FedPS: A Privacy Protection Enhanced Personalized Search Framework. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449936

## 1 INTRODUCTION

Personalized search tailors document lists for each user based on the user's interests to satisfy her information need behind the query which might be ambiguous [19, 40]. Many studies have been proposed, including traditional methods relying on features [8, 9, 19, 42, 43] and learning based models that employ deep learning to mine user preferences [21, 26, 47, 48, 50]. Existing models exploit users' personal information, such as historical query sequences and click behaviors, to infer their interests and real intent under a query. This raises the risk of user privacy leakage [1, 23, 37]. At present, the problem of user privacy protection is receiving more and more attention. Many countries formulate some privacy laws [23]. In this paper, we focus on the privacy protection issue in personalized search, and investigate the possibility of implementing a personalized search model without exposing user privacy.

Current solutions for privacy protection in search mainly consider the identifiability and linkability of privacy [1]. Identifiability means who is the user. Linkability is the possibility of inferring the user's interests from the observed query behaviors. Some studies utilize anonymous user id or group id to mask user identities [37, 52]. But some users may issue private queries (such as their names), so that the user's interests, gender and other information can still be deduced from the log. For example, it was shown that detailed user profiles can be constructed based on the published AOL anonymous dataset [5]. To reduce the linkability, the query obfuscation solution is explored [1, 4, 33, 49]. It aims to hide the user's real search intent among a set of noisy queries. Though great effects of privacy protection have been achieved by these methods, they still expose the user's search behaviors to the server and collect the logs to train a personalization model on the server. As web technology develops, there are more malicious attackers on the web, and the obtained personal data can be used in various ways. According to [23], most users are worried about their personal data being collected, exploited or released. Currently, users access search engines through their own Internet devices with certain computing, storage and communication capabilities, such as smartphones. Thus, we can use these devices to store the privacy-sensitive data and complete some computing tasks locally.

In personalized search, every user owns a personal query log that contains the user's detailed query behaviors. We first mine the user's preferences from the log as her interest profile. For example, SLTB model [9] extracts topic-based and clicked-based features from the search history, and PEPS [47] trains personal word embeddings with the user's individual log. Then, a personalized ranking model is used to calculate scores for candidate documents based on the user profile to generate personalized results. This process mainly involves three materialized components: the query log, the created user profile, and the trained personalized ranking model. With regard to the contained privacy, the most sensitive part is the original query log of each user, which might contain much


*WWW '21, April 19–23, 2021, Ljubljana, Slovenia*

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449936

personal information; the second is the user profile built on the search history, which explicitly or implicitly reflects user interests. The personalized ranking model has no direct access to the original log hence it contains much less privacy. To prevent user privacy leakage, *we claim that we are not supposed to record user search logs and construct user profiles on a remote server – we can only store these data on the corresponding user's client devices.* Thus, on each client, we only have a user's personal data whose amount is very limited. It is infeasible to solely use these data to train a reliable personalized model for the user. To jointly train a shared high-quality personalization model with query logs distributed on all users' devices, we adapt federated learning to personalized search and design a privacy protection enhanced framework, referred to as **FedPS**. With this framework, we are able to train a reliable personalized search model with all users' knowledge without exposing their original logs and profiles, which enhances privacy protection and saves the bandwidth of exchanging query logs.

In FedPS, the client submits the user issued query to the search engine along with cover queries to obscure the real intent, then the personalized ranking model deployed locally re-ranks the returned documents of the real query and shows the result to the user. The query logs, as well as the built user profile, are stored on the client. All clients and a server cooperate to train a shared personalized ranking model. In each step, the server samples several clients and sends the current model to them; these clients update the model with local data and upload the parameter updates; the server aggregates all updates to get a new model. We implement two models within the general framework. In the first one, namely FedPSFlat, we adapt the state-of-the-art personalized search model PEPS [47] to make it privacy compatible. We select PEPS because it uses personal word embeddings that could be stored and updated locally and is free of data from other users, and it mitigates the data heterogeneity problem in federated learning. But there is only one central server in FedPSFlat and contacting numerous clients could cause a performance bottleneck for the server. Besides, some clients may have limited communication or computation during training. Thus, in the second model FedPSProxy, we introduce group servers and trustworthy proxies to improve the flatten FedPS to a hierarchical structure. The servers do not connect to the clients but proxies, and those poorly connected clients could transfer their model computing tasks to the proxy. This model solves performance pressure, limited communication and privacy attack for FedPS. Experiments on two log datasets prove that FedPS protects user privacy without losing too much search accuracy.

Our main contribution is summarized as: (1) We explore a crucial issue in personalized search – privacy protection, which is one of the bottlenecks in the practical application of personalized search. This is the first time the privacy issue is considered in a deep personalized search model. (2) We give a detailed analysis on the data used and generated in personalized search, and evaluate their privacy sensitivity. Based on the analysis, we design a privacy protection enhanced framework **FedPS** by adapting federated learning to state-of-the-art personalized search and implement two specific models. Experiments confirm that our framework can protect privacy without affecting the ranking quality too much. (3) The first model FedPSFlat alleviates the data heterogeneity challenge of federated learning. The FedPSProxy introduces trustworthy proxies and group servers to relieve performance pressure for the server and promote privacy protection.

In the rest of the paper, related works are reviewed in Section 2. The FedPS framework is presented in Section 3. In Section 4 and 5, we introduce the experiments. The paper is concluded in Section 6.

## 2 RELATED WORK

### 2.1 Personalized Search

Personalized search has been widely studied due to its ability to return users more satisfying search results. The basic idea is inferring user interests from the search history and re-ranking the general search results based on the interests. Most early models were derived from heuristic methods or used features to analyze user preferences. For example, Dou et al. [19] proposed P-Click to re-rank documents by how many times the user has clicked them in the search history. Some models applied a topic model to obtain topic-based features from the clicked documents which are used to express user interests [13, 22, 38, 42–44]. SLTB [9] combined both the click-based and topic-based features extracted from the search history with the learning-to-rank (LTR) algorithm LambdaRank [12]. Moreover, some works [7, 16] demonstrated that the user's location, reading level and other features are also helpful for personalization.

Deep learning has been widely applied in personalized search due to its representation learning ability for mining potential user preferences. Song et al. [39] used the individual data to adapt the global model. Ge et al. [21] designed a hierarchical RNN with query-aware attention to capture sequential information hidden in the history and dynamically build user profile according to the current query. Lu et al. [26] and Yao et al. [48] respectively leveraged the generative adversarial network and reinforcement learning to help construct better user profiles. Besides, some works [27, 47, 50] attempted to disambiguate the current query with personal word embeddings for each user, search context or knowledge graph. All these models make improvements in personalization, but they ignore the privacy protection issue. In this paper, we design a privacy protection enhanced personalization framework.

### 2.2 Privacy Protection in Personalization

The inherent tension between personalization and user privacy originates from the essence of personalization techniques, which track the user's search process to infer her query intents and interests. As people become more concerned about their privacy [23], privacy protection in personalization receives widespread attentions.

Shen et al. [37] defined four levels of privacy protection in personalized search: pseudo identity, group identity, no identity and no personal information. Various approaches are proposed to achieve different levels of protection. They mainly focus on the identifiability and linkability of user privacy. Anonymous user id [17], group user id which is shared by a group of users [52] and peer-to-peer query schemes [14, 18] where each user submits queries issued by other users were exploited to mask the user identity. Considering the raw text may contain user privacy information, Li et al. [24] and Bendersky et al. [6] converted the original texts into anonymous n-grams or generalized attribute values. To prevent the disclosure of specific user interests from the collected user query log, many

studies added noise to the recorded data [15, 33]. Additional fake queries are generated along with the real query to obscure the user's query intents [1, 4, 36]. These approaches play a role in privacy protection, but all users' original query logs are still centrally collected by them to train the model.

Federated learning [28] is a great technique to train a shared model with all users' data distributed on their individual devices, without the need to centrally store these data. It maintains a global model on the central server; each client trains the global model with the local data and sends the model update to the server; then all these updates are used to improve the current global model. This method guarantees the security of each participant's original data compared to traditional machine learning. However, it has also been demonstrated that the model updates might leak the user privacy [30]. Consequently, several defense methods were proposed, such as Multi-Party Computation (MPC) [11, 20], Homomorphic Encryption (HE) [32] and Differential Privacy (DP) [29]. Secure MPC [20] is a class of cryptography technique for many participants to jointly train a model in a peer-to-peer topology. DP [29] protects privacy by adding random noise to the uploaded data, which has an impact on model accuracy. Our privacy protection enhanced framework is designed based on federated learning.

## 3 OUR PROPOSED APPROACH

### 3.1 Problem Formulation

In personalized search, we first analyze the user's search history to build a user interest profile, then the personalized ranking model customizes a document list for the user based on her user profile. This process mainly involves the user's original search log, user profile, a personalized ranking model and some shared assisted data such as term frequencies, word embeddings, etc. We carefully analyze their contents and list the involved user privacy below.

• **User's original search log**, including all queries issued by the user, browsed document lists and click behaviors. Search log is the most privacy-sensitive data in personalized search, and studies [5] have shown that information (such as name, residence, and hobbies) of some users can be identified by analyzing their issued queries.

• **User profile** constructed from the search log. Most personalized search models build user profiles to represent user interests. Different formats of user profiles are used in existing works. Typical profiles include term, topic, click distributions [9, 43], sequential search behavior representation vectors [21], and personal word embeddings [47]. User profiles are usually **aggregated** vectorized representations of user behaviors, so they contain less private information than original query logs, but are still privacy-sensitive.

• **A personalized ranking model** that calculates the personalized score for candidate documents based on the query and user profile. Parameters of the model mainly reflect the personalized ranking strategy with extracted features or representation vectors as the input. Thus, the model contains little user privacy.

• **Other auxiliary data** used to help ranking, such as the shared word embeddings [21, 47, 50]. The association between user privacy and these data is determined by the specific model.

According to the analysis above, the user's original query log contains the most private information, followed by the user profile. Stated in Section 1, to protect privacy, we are not supposed to collect

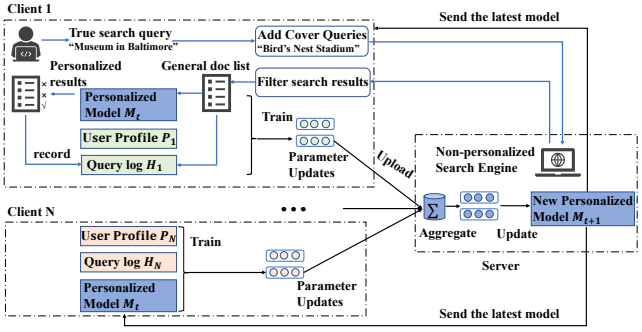


**Figure 1: The FedPS framework. The blue lines illustrate the data stream of the use process, and the black lines show the data stream of the training process.**

search logs and build user profiles on a remote server. Currently, users access the search engine through their own client devices with certain computing, storage and communication capabilities, denoted as $C_1, C_2, \ldots, C_N$. Thus, we store the user's search history $H$, build the user profile $P$ and personalize the general search results with a personalized ranking model on the client. To train the personalized ranking model, the safest method is to train a personal model on each user's client with local data. However, training a reliable neural model usually depends on a lot of samples, and the data of a single user is not enough. Therefore, we apply federated learning to personalized search to train a shared high-quality personalization model with rich data distributed on all users' clients.

In the following, we will describe our proposed framework FedPS and two different implementations within this framework.

### 3.2 FedPS —— The Framework

The FedPS framework is shown in Figure 1, and the process is: the user enters a query on her own client; the client sends the issued query accompanied with several cover queries to the search engine to obtain the relevant documents; then the personalized ranking model deployed on the client adjusts the document list of the real query and presents the personalized result to the user. After the user gives feedback on the result, the issued query, general document list, personalized document list and the user's clicks are recorded in the local query log. During the whole process, the user's complete query log is only stored on the client.

We adopt the query obfuscation method [4, 36, 49] to hide the user's genuine query intent among a set of generated unrelated cover queries on the server side. The number of cover queries should not be too large which will otherwise affect the server's response speed. We first train an LDA (Latent Dirichlet Allocation) [10] topic model on the whole document set to infer the topic proportion of the issued query, which represents the user's real search intent. Then, cover queries are generated on different topics with similar entropy on account of plausibility. The specific method is not the key problem of this paper, and more details can be found in [4, 49]. Different from recording cover queries into the search log on the server, which will distort the user interest profile and have negative impacts on personalization, we just add cover queries

when submitting the query to the search engine. After all the search results are returned to the client, we filter those fake queries and only record the real query into the local log. Therefore, the user profile in our framework will not be influenced by cover queries.

All clients cooperate to train the personalized ranking model, and a central server is responsible for controlling the whole collaboration process. We optimize the personalized ranking model referring to FedAvg [28]. FedAvg is one of the widely used distributed optimization algorithms in federated learning. Suppose that a certain amount of log data has been stored on each client, and there is a randomly initialized personalized ranking model on the server. The model here can be any learning based personalization model. At this point, all the $N$ clients begin to communicate with the server to train the model for a total of $R$ rounds. In each round, we ensure that every client updates the model once, and complete the model training on all clients in $\frac{N}{K}$ steps. The operations in each step $t$ are: **First**, the server samples $K$ clients and sends the latest personalized ranking model $M_t$ to them. **Second**, each of the selected clients receives the latest model $M_t$ from the server, and updates the model for $E$ epochs with the training samples $D$ and user profile $P$ constructed on the local log data $H$. Mini-batch stochastic gradient descent (SGD) optimization algorithm is applied. Then, all the selected clients send the parameter updates of the personalized ranking model back to the server, with the history $H$, training samples $D$ and user profile $P$ kept locally. **Third**, the server aggregates the parameter updates from all the selected clients, and applies the aggregation to update the current model $M_t$ to a new one $M_{t+1}$. During the above joint model training process, some clients may have limited and unreliable communication because mobile devices are frequently offline or in poorly connected environments. Furthermore, the available computing resources on some clients may be not enough to complete the model training task. FedPS trains the shared personalization model in a synchronous way. To prevent the server from waiting too long for the clients with a poor connection or computation, we set maximum response time $T_{max}$, and ignore the model updates of those clients without response during this period. The whole federated training process is described in Algorithm 1.

Note that if the parameter $K = 1$, the federated training algorithm degrades to a stream method. All available clients are accessed one by one to update the model trained by the last client. Due to the model is updated by only one user at each time, the parameter updates may expose the user's data features and privacy to subsequent users. Therefore, we don't consider the case of $K = 1$, and select more than one client in each step. If $K = N$, the algorithm is similar to full-batch gradient descent, where we are required to consider all clients and spend a lot of time for calculation and communication in every step. To balance effects and efficiency, we usually set $K$ as an appropriate value between 1 and $N$. We also conduct experiments to evaluate different values of $K$ in Section 5.2.

After the $R$ rounds of model training with the existing log data on all clients are completed, the server broadcasts the trained personalization model to clients for subsequent use. Users will continuously perform search on their own devices and generate new query logs. Thus, we can further update the personalized ranking model with new data. Both the general document lists returned by the search engine and locally personalized results are recorded on the client. In order to remove the impact of the previous personalized results on the subsequent updating of the personalized ranking model, we always utilize the non-personalized data.

**Algorithm 1** Federated Training of Personalization Model

$B$ is the local batch size, $E$ is the local model update epoch, and $\eta$ is the learning rate.
Randomly initialize the personalized ranking model $M_1$
**for** each round $r = 1, 2, \ldots, R$ **do**
  **for** each step in $t = 1, 2, \ldots, \frac{N}{K}$ **do**
    Server samples $K$ different clients
    Wait updates from each of the $K$ selected clients for $T_{max}$
    **for** each client $j \in \{1, 2, \ldots, K\}$ **in parallel do**
      Receive $M_t$ from the central server
      Construct training samples $D$ and user profile $P$ on $H$
      $M^j_{t+1} \leftarrow SGD(M_t, D, P, B, E, \eta), n_j = \|D\|$
      Send parameter updates $\Delta M^j_{t+1}$ to the central server
    **end for**
    $M_{t+1} = M_t + \sum_{j=1}^{K} \frac{n_j}{n}(\Delta M^j_{t+1}), n = \sum_{j=1}^{K} n_j$ ($\Delta M^j_{t+1}$ of clients exceeding $T_{max}$ are NULL.)
  **end for**
**end for**
Broadcast the trained model $M$ to all clients

We consider two ways to update the model. The first is an online training method that the client sends an update application to the server if the amount of newly generated data is enough. Then, the client updates the current global model with new data and uploads the parameter updates to the server. After the server receives parameter updates from $K$ clients, it aggregates all these updates to generate a new model and distributes the new model to all clients. However, there are numerous users; if each client communicates with the server to update the current model once some new data is generated, a relatively high communication cost will be required and the model performance may also be unstable. In addition, the frequency of searching and the amount of newly generated data are very unbalanced among users. Some users frequently issue new queries and their corresponding clients incrementally update the personalized ranking model will make the model biased towards these active users and the overall performance will be worse. On account of these problems, we propose a more feasible model updating method that reduces the communication cost and makes the model perform more stable. We set a fixed time interval to update the model (such as three days), which is determined by comprehensively considering the communication, computing resources and users' search frequency. During the time interval, all clients employ the model trained in the previous stage without updates. After a time interval, the server starts a task to jointly train a new personalized ranking model from scratch with the current log data on all clients for $R$ rounds, as stated in Algorithm 1. With the training completed, the server broadcasts the new model to all clients for the use of the next stage. Compared with the online incremental update, this method updates the model with data from all clients in every stage, which makes the model better adapt to the global data distribution and achieve better overall performance. To speed up

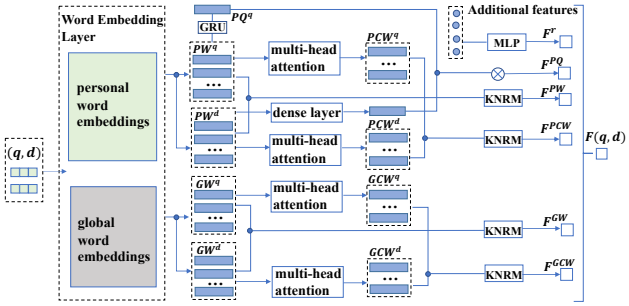


**Figure 2: Structure of the personalized search model employed in FedPSFlat.**

the updating process and save computing resources, we can also choose to update the model of the previous stage with only the newly generated data in this stage.

So far we have described the FedPS framework. In the next sections, we implement two models to cope with some challenges, such as data heterogeneity and communication efficiency.

## 3.3 FedPSFlat: the Flatten FedPS

In this implementation, we specially employ a personalized search model with a personal module to tackle the challenge of data heterogeneity in federated learning, introduced in the next.

*3.3.1 Personalized Search Model.* During search process, different users tend to click different documents under the same query. Therefore, the whole logs of all users naturally suffer from non-IID distribution. This is one of the critical challenges for standard federated learning which trains a single global model for all clients [25]. Multi-task learning is used to model this statistical heterogeneity by treating the model training on every client as a separate task. Focusing on state-of-the-art personalized search model PEPS [47], it sets up a module which contains personal word embeddings for each user trained from the user's own search data to disambiguate the query keywords. In this case, the personal word embedding module can adapt to the user's personal data distribution, and it is promising to alleviate the problem of data heterogeneity like multi-task learning. We adapt PEPS as the personalized search model in this implementation, shown in Figure 2. The main modules are briefly introduced as follows, and more details can be referred to [47].

**Word embedding layer.** There is a global word embedding matrix and a personal word embedding matrix in this layer. The global word embeddings are shared and updated with all users' data. The personal word embeddings are different for different users, which are trained with merely the corresponding user's data to contain the user interested word meanings and can be used as the interest profile. Without centrally collected query logs, we initialize the global word embeddings with the word2vec [31] model trained on the document collection or the Wikipedia corpus. As for the personal word embeddings, we use the global word2vec model or that trained with the user's individual search log for initialization.

**Matching & Ranking.** Through the word embedding layer, we are able to convert the query $q$ and document $d$ into vectors. Five kinds of text representations of different aspects are considered, including the personal and global word vectors $PW^q, PW^d, GW^q, GW^d$, personal and global contextual representations $PCW^q, PCW^d, GCW^q$, $GCW^d$ to clarify the keywords by modeling the interactions between contexts with multi-head self-attention [41], and personalized query representation $PQ^q$.

With these text representations, we calculate the personalized scores and re-rank the candidate documents. As for the four types of word representations, we use the neural matching component KNRM [46] to compute the interactive matching scores, i.e. $F^{PW}$, $F^{PCW}$, $F^{GW}$, $F^{GCW}$. For the query representation $PQ^q$, the cosine similarity with the document is calculated as $F^{PQ}$. In addition, a series of click and topic features are fed into an MLP to get a relevance score $F^r$. Finally, all the scores are combined through an MLP layer to compute the personalized score of the document as:

$$F(q,d) = \text{MLP}(F^{PW}, F^{PCW}, F^{GW}, F^{GCW}, F^{PQ}, F^r). \quad (1)$$

We use the pairwise learning-to-rank algorithm LambdaRank [12] to train the model. For each client, document pairs are created on the local query log.

*3.3.2 Model Training.* We train the personalized search model with Algorithm 1. Considering user privacy, we have to make a detailed discussion about the parameters to be exchanged. Generally, all the parameters of the above model can be divided into personal word embeddings, global word embeddings and parameters of the ranking module. The personal word embeddings are updated with the user's individual data and used as the user profile which contain a wealth of user privacy. Thus, we should keep this part of parameters on the client. Parameters of the personalized ranking model contain the least privacy. We can upload these parameters to the server for aggregation to obtain a more reliable personalized ranking model. In addition, the global word embeddings are knowledge shared by all users, but the updates of this module could also reflect the word distribution of the user's query log. Therefore, whether and how to upload the global word embeddings depends on the requirement of privacy protection. We consider the following three cases.

**Case1**: Ignore the small amount of user privacy that may be contained in the updates of global word embeddings and upload the complete parameter updates.

**Case2**: Taking into account that some words with low frequency or specific information may leak personal privacy or interests, we sample some words that appear frequently in the document collection, and only upload the updates of these words.

**Case3**: In order to strictly avoid user privacy leakage, we do not upload parameter updates of global word embeddings, and keep the pre-trained global word embeddings fixed on the server side.

Although the whole global word embeddings have a large number of parameters, we only upload the embedding updates, which usually focus on a few terms and will not cause too much communication pressure and bandwidth expense.

## 3.4 FedPSProxy: Hierarchical FedPS with Proxy

In FedPSFlat, all clients communicate with the central server to jointly train the personalized ranking model. This setting mitigates the privacy risks associated with centrally collecting data from each device, but there are still some problems. (1) The only central

server may become the performance bottleneck in the training process. In actual application, there is a large number of clients. Communicating and exchanging data with all clients will take the server a lot of time and computing resources. A failure of the central server will interrupt all communications in the current training step. (2) There exist potential risks of privacy attacks. After a client updates the model with local data, the parameter updates can reflect some information about the user's data. Directly uploading the updates to the server will provide the malicious server with an opportunity for privacy attack. Thus, it is necessary to mask the contained private information or break the link between the user and the corresponding parameter updates. Currently, there are several defense techniques to cope with privacy leakage in federated learning, including Multi-Party Computation (MPC) [20], Homomorphic Encryption (HE) [32] and Differential Privacy (DP) [29]. But encryption methods increase communication and computation cost, which grows quadratically with the number of clients especially costs for the central server. And DP leads to a loss of the model accuracy.(3) Some client devices with limited communication, computing or storage abilities may slow down the entire training process, because FedPS trains the global model in a synchronous way. In Section 3.2, we claim to set maximum response time and discard the model updates of those stragglers, but this loses training data and may reduce search accuracy.

We attempt to address the above issues by introducing proxies and group servers to improve the flatten FedPS where all clients communicate with the central server into a hierarchical structure. Figure 3 presents FedPSProxy, a four-layer implementation of FedPS. We can build more layers based on the number of users and privacy requirements. The central server sets a series of group servers to relieve the communication pressure. Each group server is responsible for communicating with a part of clients. And this structure can dynamically adapt to large-scale users scenario by increasing group servers. In addition, the clients can not directly access the server but through a proxy for privacy protection. We are supposed to ensure that the proxies are trustworthy and reliable. Referring to edge computation [45, 51], we set proxies as the edge gateway at home (the safest), edge server of a work organization and so on. Under this layered model, there are some changes in the operations of the server and clients. The specific changes and the benefits they can bring to eliminate the above problems are as follows:

**Communicating with server/clients.** In FedPSProxy, the central server cannot directly connect to the clients. When a client wants to submit an application of training or updating the model to the server, it first sends the message to the corresponding proxy. Then, the proxy aggregates all applications from clients in its scope and sends a message to the group server, which indicates how many clients made a request. The central server receives information about how many devices are connected from group servers, and samples a batch of available clients. Each group server broadcasts the latest global model to the selected proxies connected to it. Then, each proxy sends the model to the selected clients within its scope. Compared with client devices, the number of proxies is much smaller. Furthermore, there are a series of group servers responsible for communication, instead of a single central server. Thus, the FedPSProxy model solves the communication bottleneck

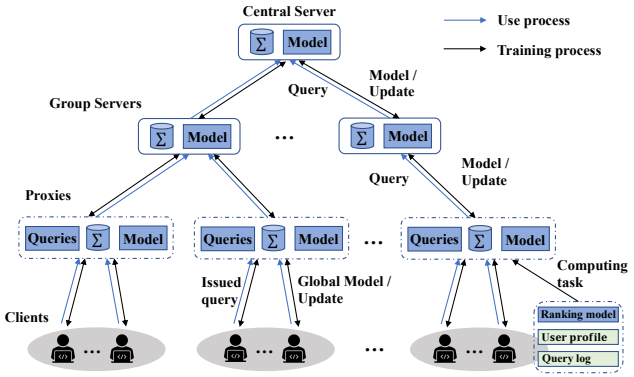


**Figure 3: Architecture of the four-layer FedPSProxy model. The blue lines show the data stream of the use process, and the black lines show the data stream of the training process.**

for the central server and improves efficiency since multiple group servers and proxies can work in parallel.

**Uploading the model updates.** When a client uploads the parameter updates to the server, it first sends the updates to the proxy, then the proxy aggregates the updates from all selected clients within its scope and uploads the aggregation to the group server. Finally, the central server combines parameter updates on all group servers. This method breaks the link between the model updates and the user on the server side, avoiding user privacy leakage. For example, a single user's updates of the global word embeddings can be used to infer the words that frequently appear in the user's query log, but the aggregated word embedding updates of multiple users within a proxy are not easy to expose the data information of a single user. In addition, if users require stronger privacy protection and do not expect to expose parameter updates to the proxy, it is more feasible and efficient to apply encryption algorithms such as MPC among small-scale users under the proxy, which can save a lot of communication and computation costs.

**Uploading issued queries.** Stated in Section 3.2, when the client submits a issued query to the server to obtain non-personalized search results, we apply the query obfuscation method to mask the user's real query intents. In FedPSProxy, the clients under a proxy upload their issued queries to the server through the same proxy, which hides each single user in a group of users and protects the user privacy. In such case, there is no necessity to upload additional cover queries, promoting the response speed and reducing bandwidth expense.

**Offloading computing task.** In our FedPSProxy model, the proxies are generally trustworthy and reliable in communication (e.g., edge server of the company). Thus, the client devices with weak communication, storage or computing abilities can offload their tasks composed of the local query log, user profile and personalized ranking model to the proxies. The proxy with stronger computing power will help train the model and generate personalized search results. If the user is still worried about the security of proxies, they can also choose to keep their original log data and the input layers on the client, and offload the remaining model layers and calculation to the proxy. In general, this method solves

**Table 1: Statistics of the experimental datasets.**

| Type | AOL | Commercial |
|---|---|---|
| Time Span | Mar–May 2006 | Jan–Feb 2013 |
| User Num | 118,067 | 5,998 |
| Query Num | 3,461,637 | 738,731 |
| Session Num | 391,893 | 275,910 |
| Avg/Max Query Per User | 29.32 (1,449) | 138.93 (3,207) |
| Avg/Max Query Per User-Day | 2.043 (358) | 5.713 (153) |

**Table 2: KL-Divergence of different privacy protection techniques. The best results are shown in bold.**

| Model | AOL Dataset | | Commercial Dataset | |
|---|---|---|---|---|
| | KL-Divergence | | KL-Divergence | |
| GroupUser | 2.715 | -61.72% | 2.854 | -65.84% |
| CoverQuery | 5.071 | -28.50% | 1.563 | -81.29% |
| FedPSFlat | **7.092** | – | 8.356 | - |
| FedPSProxy | 5.403 | -23.82% | **9.073** | 8.58% |

the delay and data loss in the federated training process caused by stragglers, making the FedPS framework more feasible and effective. For the clients with limited capabilities and resources, this method also relieves their burden of communication and model calculation.

To conclude, FedPSProxy addresses the following problems: the performance bottleneck of the central server in large-scale user scenario, the limited connection and computation of clients, and the privacy attack of the flatten model.

# 4 EXPERIMENTAL SETTINGS

## 4.1 Dataset and Evaluation Metrics

Lacking the condition to conduct experiments with real client devices, we employ two widely used non-personalized query logs for simulation. Statistics of the two datasets are shown in Table 1.

**AOL Dataset** is a public three-month log [34]. Each record is identified by an anonymous user id based on which we divide all the data into query logs of different users. We follow [2, 3] to process the original AOL log. First, the query sequences are split into sessions with borders decided by the difference between two consecutive queries. Then, we cut the whole log into the background set and experimental set. The background set contains the first five weeks log, used to build interest profiles. The last eight weeks experimental data is separated into the training, validation and testing set with the proportion 6:1:1. In the AOL set, only the URLs of clicked documents are recorded, based on which we crawled the title of the corresponding documents. Then, we use the same method as [2, 3] to rank all documents with BM25 [35] and sample the top documents as the candidates. 50 documents are selected for each test query, while 5 candidates per training/validation query to speed up training. To ensure the validity of personalization, we filter users without enough background set or training set.

**Commercial dataset** includes query logs in Jan. and Feb. 2013. Each query corresponds to a complete document list and the information of click behaviors. Following [21, 26], we view the documents with more than 30 seconds of dwelling time as satisfied clicked documents. With 30 minutes of inactivity as the interval, we separate the query sequences into sessions. Then, the log of the first six weeks is used as historical data to mine user interests; the remaining two weeks data is split into training, validation and testing set with 4:1:1 ratio.

**Evaluation Metrics** With the clicked documents as relevant and other candidate documents as irrelevant [48], we choose three common metrics to evaluate the ranking results: MAP, MRR and P@1. A more reliable metric for personalized search P-Improve is also employed. Following [47, 50], we only use P-Improve on the commercial dataset whose recorded document lists were really presented to users. In addition to ranking quality, we also define a metric to measure the privacy protection capability of models, which is the Kullback-Leibler divergence between the real user profile and the observed profile on the server [4]. The larger the KL divergence, the less privacy is disclosed. Since user interests are mainly reflected in the issued queries and clicked documents, we use the distribution of all terms in the issued queries and clicked documents as the user profile.

## 4.2 Baselines and Our Models

The original rankings of the AOL set are generated by BM25 [35] and that of the commercial set is returned by the search engine. Besides, we select other baselines, including neural ranking models, personalized search models and privacy enhanced models.

**KNRM** [46]: It is a neural model using kernels to extract features from interactions between the query and document for ranking.

**HRNN** [21]: This work employed a hierarchical RNN with query-aware attention to build the short-term and long-term user profiles.

**HTPS** [50]: Inspired by BERT, Zhou et al. [50] proposed to encode historical queries as the context with the transformer [41] to enhance the representation of the current query for disambiguation.

**PEPS** [47]: It trained personal word embeddings for each user to clarify the personalized meaning of some keywords. We turn off its query reformulation module in our implementation.

**GroupUser** [52]: A group of users share a single identity and their search logs are recorded together to build user profile at the group level, hiding the privacy information of each single user.

**CoverQuery** [1, 4]: This approach first applies a topic model to infer the user's query intent, then generates several noisy queries from unrelated topics to conceal the user's real intentions.

The two privacy protection baselines are both applied to PEPS.

**FedPSFlat** and **FedPSProxy** are our proposed models.

As for the personalized search model, we train a 100-dim word2vec model on the document set to initialize the global and personal word embeddings. Other hyper-parameters are set the same as PEPS [47]. We adopt Adam optimizer to train the personalization model on clients. In each update step, we sample 10 clients to update the model for 1 epoch. The models are trained on the whole training set and evaluated on the test set. We assume 10% of the clients are in poor communication during training. We define 100 users as a group in the GroupUser approach. And we separate 100 users under each proxy, 1,000 proxies under each group server in FedPSProxy. One query is randomly generated as the cover query.

**Table 3: Overall performance of models. The percentages are computed based on the corresponding original model. † means no significant differences from the original model with paired t-test at p<0.05 level. Results of the best privacy enhanced model are shown in bold.**

| Model | AOL Dataset | | | | | | Commercial Dataset | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | | MRR | | P@1 | | MAP | | MRR | | P@1 | | P-Imp. | |
| Original Rank | .2504 | -64.8% | .2596 | -64.2% | .1534 | -75.5% | .7399 | -9.8% | .7506 | -9.6% | .6162 | -15.1% | - | - |
| KNRM | .4291 | -39.7% | .4391 | -39.4% | .2704 | -56.9% | .4916 | -40.1% | .5001 | -39.8% | .2849 | -60.7% | .0655 | -74.1% |
| HRNN | .5423 | – | .5545 | – | .4854 | – | .8065 | – | .8191 | – | .7127 | – | .2404 | – |
| FedPS & HRNN | .5419† | -0.07% | .5541† | -0.07% | .4852† | -0.04% | .8061† | -0.05% | .8186† | -0.06% | .7125† | -0.03% | .2402† | -0.08% |
| HTPS | .7091 | – | .7231 | – | .6272 | – | .8219 | – | .8315 | – | .7287 | – | .2548 | – |
| FedPS & HTPS | .7085† | -0.08% | .7226† | -0.07% | .6268† | -0.06% | .8213† | -0.07% | .8309† | -0.07% | .7282† | -0.07% | .2545† | -0.12% |
| PEPS | .7115 | – | .7246 | – | .6266 | – | .8206 | – | .8305 | – | .7256 | – | .2533 | – |
| GroupUser | .6917 | -2.78% | .7037 | -2.88% | .6106 | -2.55% | .8145 | -0.74% | .8248 | -0.69% | .7204 | -0.72% | .2411 | -4.8% |
| CoverQuery | .7101† | -0.20% | .7232† | -0.19% | .6252† | -0.22% | .8201† | -0.06% | .8299† | -0.07% | .7151† | -0.07% | .2528† | -0.20% |
| FedPSFlat | .7074† | -0.58% | .7205† | -0.57% | .6229† | -0.59% | .8195† | -0.13% | .8294† | -0.13% | .7247† | -0.12% | .2525† | -0.32% |
| FedPSProxy | **.7112**† | -0.04% | **.7242**† | -0.06% | **.6262**† | -0.06% | **.8204**† | -0.02% | **.8304**† | -0.01% | **.7254**† | -0.03% | **.2532**† | -0.04% |

# 5 EXPERIMENTAL RESULTS AND ANALYSIS

## 5.1 Overall Performance

The overall results, including the privacy protection effect and ranking quality, are shown in Table 2 and Table 3. We can observe:

(1) **The FedPS framework not only protects the user privacy best, but also achieves better personalization results than other privacy enhanced personalized models**. In Table 2, FedPS models get the best result of KL-Divergence which is used to measure privacy protection capability. The GroupUser and CoverQuery use some noise to hide the user's true identity and query intentions but still collect user search logs on the server side. However, under our framework, the user's original log data is guaranteed not to be exposed at all so user privacy is better protected. Besides, FedPSProxy outperforms both the GroupUser and CoverQuery methods on all ranking metrics on the two datasets. Due to noise is added to the user's search log in the two baselines but FedPS exploits the real query log on the client side to analyze the user's interests, FedPS performs more accurate personalization. We find that the CoverQuery method also has no significant loss of the ranking results though some fake queries are added to the log. We deduce it may because that the fake queries increase training data of the model, thereby reducing the impacts of vague user profiles.

(2) **The FedPS framework is adaptive to various learning-based personalized search models and attains comparable results to the original models trained from the centrally stored data.** We apply the FedPS framework to several state-of-the-art personalization models, including HRNN, HTPS and PEPS, and compare them with the original models. As shown in Table 3, there is no much difference between the results of them and the original models on all ranking metrics, confirmed by paired t-test at $p < 0.05$ level. This proves that our privacy protection framework will not cause much loss to ranking quality.

(3) **FedPSProxy outperforms FedPSFlat.** In this experiment, we set 10% of the clients in poor connection or offline during the training process. For FedPSProxy, these stragglers can transfer their computing tasks to the corresponding proxy which helps complete model training. But FedPSFlat ignores these clients as well as the training data on them, thus affecting the model performance. In actual application, there are indeed a certain percent of clients with poor communication ability, so FedPSProxy with some error tolerance shows better applicability and feasibility.

To summarize, **our FedPS framework indeed has the ability to protect user privacy without impacting the model accuracy too much, and it greatly adapts to various learning based personalized search models.**

**Table 4: The performance of FedPSProxy with different $K$. $K$ is the number of clients sampled in each step. 'Rounds' is the total rounds of model training until convergence. 'Steps' is the total number of model update steps.**

| $K$ | Rounds ($R$) | | Steps ($\frac{N}{K} * R$) | | Best MAP |
|---|---|---|---|---|---|
| 5 | 3 | 1x | 15762 | 1x | 0.7112 |
| 10 | 3 | 1x | 7881 | 0.5x | 0.7112 |
| 20 | 4 | 1.3x | 5254 | 0.33x | 0.7112 |
| 30 | 6 | 2x | 5254 | 0.33x | 0.7079 |
| 100 | 8 | 2.7x | 2101 | 0.13x | 0.6997 |

## 5.2 Study of Different Parameters

In Algorithm 1, there are two main parameters to be determined, i.e. the number of sampled clients and the epoch the model is trained locally in each step. We experiment with different parameters and illustrate the results in Table 4 and Table 5. 'Rounds' is the total rounds of model training until convergence, used to measure the communication and computation cost. 'Steps' represents the total number of model update steps. In each step, the selected clients work in parallel, so 'Steps' can be a measure of wall-clock time cost.

**Number of sampled clients $K$.** In each joint training step, we sample more than one clients to update the current model with their own data locally, then merge their updates to generate a new global model, which avoids exposing the parameter updates of a single user to other users. Presented in Table 4, we set $K$ as 5,10,20,30 and 100. As $K$ increases, we observe that the total number of training steps decreases, saving wall-clock time. However, a larger value of $K$ makes the model converge slowly and requires more rounds

**Table 5: The performance of FedPSProxy with different $E$. $E$ represents the epoch of local model training in each step.**

| $E$ | Rounds ($R$) | | Steps ($\frac{N}{K} * R$) | | Best MAP |
|---|---|---|---|---|---|
| 1 | 4 | 1x | 10508 | 1x | 0.7112 |
| 2 | 3 | 0.75x | 7881 | 0.75x | 0.7091 |
| 3 | 3 | 0.75x | 7881 | 0.75x | 0.7088 |

**Table 6: Communication time for joint model training in a round by different FedPS models. $T$ represents the time to exchange model parameters, and $d$ is the delay time due to poor communication.**

| Settings | delay ratio=10% | | delay ratio=20% | |
|---|---|---|---|---|
| | $K = 100$ | $K = 1000$ | $K = 100$ | $K = 1000$ |
| FedPSFlat | $236134T + 11551d$ | | $236134T + 23557d$ | |
| FedPSProxy | $47240T$ | $6902T$ | $44878T$ | $6664T$ |

of communication and model training. Moreover, the model with smaller $K$ tends to achieve greater results. The possible reason is that the gradients generated by various clients with non-i.i.d data might be quite different from each other and it would be less effective to update the model by simply aggregating these gradients, causing the model turn into worse performance.

**Local epoch $E$.** For each selected client, it can update the model with local data for $E$ epochs. We conduct experiments with $E = 1, 2, 3$ respectively. $K$ is set as 10. Observing the results in Table 5, we find developing local update on each client for more than one epoch could reduce the number of training rounds and steps, saving communication and wall-clock time costs. But there are some losses on the model's effect. We infer it may due to that more local training epochs in each step cause the model to overfit to the limited local data, thus decreasing the global performance of the model.

This study shows different hyper-parameters have various effects on the model accuracy, time and computation resources. We should take these factors into account to select appropriate parameters.

## 5.3 Effect of Proxy and Group Server

In order to test whether FedPSProxy has the ability to reduce the performance burden of the central server and promote the federated training efficiency, we compare the required communication time to jointly train the personalization model for a round with FedPSFlat and FedPSProxy. We focus on the time spent for broadcasting the current global model and uploading the parameter updates, both of which are assumed equal to $T$. We consider 118,065 users in the AOL log. FedPSFlat samples $K$ clients in each training step, while FedPSProxy first samples $K/10$ proxies, then samples 10 clients in the scope of each proxy. Moreover, we discuss that 10% or 20% of the devices have a poor connection and suffer a delay $d$ in each step. In FedPSProxy, such stragglers offload their computing tasks to the proxies so we ignore the time cost of data exchange between these clients and the corresponding proxy. The comparison of communication time is shown in Table 6.

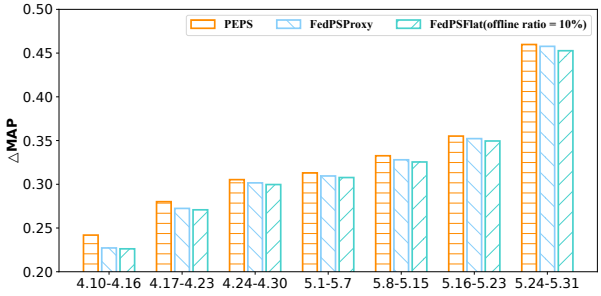


**Figure 4: Results of continuous model updating simulation.**

From Table 6, we see that the training process of FedPSFlat can be easily slowed down by the devices with limited connection, costing much additional delay time. In contrast, FedPSProxy is almost not affected by these errors, showing robust performance and higher efficiency. The value before $T$ is used to measure the communication time of data exchange among the server, group servers, proxies and clients. FedPSProxy greatly reduces the communication time of the central server compared to FedPSFlat, and the reduction is greater as the number of sampled clients $K$ gets larger, significantly relieving the communication pressure of the server.

## 5.4 Simulation of Continuous Model Updating

In the previous experiments, the personalization model is trained on the whole training set, without subsequent updates. In this section, we use the data of AOL from $3^{rd}$ Apr to $31^{st}$ May to simulate the continuous model updating process described in our FedPS framework. We cut all log into 8 stages with one week as a stage. First, the personalized search model is initialized with the first week data. Then, after each week, we apply the currently available logs to retrain the model from scratch, and evaluate the trained model on the data of the next stage. We consider 10% of the clients are offline or in poor connection. The improvement on MAP of PEPS and our two models over the original ranking are displayed in Figure 4.

Generally, the results of all models improve as the data increases with stages. FedPSProxy performs inferior to PEPS trained with the centrally collected data in early stages. It may due to the training data on each client is very limited in early stages, and using the limited local data to update the model could cause the model to overfit to the individual data. In later stages with enough local data, FedPSProxy almost achieves similar results as PEPS. FedPSFlat performs a little worse than FedPSProxy, because it ignores the offline clients during training while computing tasks of these stragglers are transferred to the proxies in FedPSProxy. As the data on each client increases, the effect differences between the two FedPS models caused by discarding training data of the offline clients get larger.

## 6 CONCLUSION

In this study, we focus on the privacy protection issue in personalized search, and propose a privacy protection enhanced framework FedPS for learning based personalized search models. It stores the user's privacy-sensitive query log and interest profile on the client, and employs federated learning to jointly train a shared personalized ranking model with all clients and their decentralized data.

Within this framework, we design two implementations, FedPSFlat and FedPSProxy. FedPSFlat eliminates the challenge of data heterogeneity. The second model improves FedPSFlat by introducing proxies and group servers to promote privacy protection and relieve the performance pressure caused by large-scale clients. Experimental results confirm that our framework protects user privacy without affecting the model accuracy too much. In the future, we will explore techniques for stronger user privacy protection.

## ACKNOWLEDGMENTS

Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370 and No. 61832017, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, and Shandong Provincial Natural Science Foundation under Grant ZR2019ZD06.

## REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Intent-aware Query Obfuscation for Privacy Protection in Personalized Web Search. In *SIGIR 2018*. ACM, 285–294.
[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-Task Learning for Document Ranking and Query Suggestion. In *ICLR 2018,*.
[3] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *Proceedings of SIGIR 2019*.
[4] Wasi Uddin Ahmad, Md. Masudur Rahman, and Hongning Wang. 2016. Topic Model based Privacy Protection in Personalized Web Search. In *SIGIR 2016*. ACM.
[5] Michael Barbaro and Tom Zeller. 2006. A Face is exposed for AOL searcher no. 4417749. *New York Times* (2006).
[6] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from User Interactions in Personal Search via Attribute Parameterization. In *WSDM 2017*. ACM, 791–799.
[7] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *SIGIR 2011*.
[8] Paul N. Bennett, Krysta Marie Svore, and Susan T. Dumais. 2010. Classification-enhanced ranking. In *Proceedings WWW 2010*. 111–120.
[9] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR '12*. 185–194.
[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent Dirichlet Allocation. In *NIPS 2001*. 601–608.
[11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*.
[12] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *(ICML 2005)*. 89–96.
[13] Mark James Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *CIKM 2010*.
[14] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartí. 2009. Preserving user's privacy in web search engines. *Comput. Commun.* 32, 13-14 (2009).
[15] Gang Chen, He Bai, Lidan Shou, Ke Chen, and Yunjun Gao. 2011. UPS: efficient privacy protection in personalized web search. In *SIGIR 2011*. ACM, 615–624.
[16] Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Sebastian de la Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *CIKM 2011*. 403–412.
[17] Roger Dingledine. 2011. Tor and Circumvention: Lessons Learned - (Abstract to Go with Invited Talk). In *CRYPTO 2011 (Lecture Notes in Computer Science, Vol. 6841)*. Springer, 485–486.
[18] Josep Domingo-Ferrer, Maria Bras-Amorós, Qianhong Wu, and Jesús A. Manjón. 2009. User-private information retrieval based on a peer-to-peer community. *Data Knowl. Eng.* 68, 11 (2009), 1237–1252.
[19] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW 2007*.
[20] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018. A Pragmatic Introduction to Secure Multi-Party Computation. *Found. Trends Priv. Secur.* 2, 2-3 (2018), 70–246.
[21] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In *CIKM 2018*.
[22] Morgan Harvey, Fabio Crestani, and Mark James Carman. 2013. Building user profiles from topic models for personalised search. In *CIKM'13*. 2309–2314.
[23] Alfred Kobsa. 2007. Privacy-Enhanced Web Personalization. In *The Adaptive Web, Methods and Strategies of Web Personalization (Lecture Notes in Computer Science, Vol. 4321)*. Springer, 628–670.
[24] Cheng Li, Mingyang Zhang, Michael Bendersky, Hongbo Deng, Donald Metzler, and Marc Najork. 2019. Multi-view Embedding-based Synonyms for Email Search. In *SIGIR 2019*. ACM, 575–584.
[25] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749
[26] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. PSGAN: A Minimax Game for Personalized Search with Limited and Noisy Click Data. In *SIGIR 2019*. 555–564.
[27] Shuqi Lu, Zhicheng Dou, Chenyan Xiong, Xiaojie Wang, and Ji-Rong Wen. 2020. Knowledge Enhanced Personalized Search. In *SIGIR 2020*. ACM, 709–718.
[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS 2017 (Proceedings of Machine Learning Research, Vol. 54)*. PMLR, 1273–1282.
[29] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR 2018*.
[30] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 691–706.
[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR 2013, Proceedings*.
[32] Shiho Moriai. 2019. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. In *ARITH 2019*. IEEE, 198.
[33] HweeHwa Pang, Xuhua Ding, and Xiaokui Xiao. 2010. Embellishing Text Search Queries To Protect User Privacy. *Proc. VLDB Endow.* 3, 1 (2010), 598–607.
[34] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Infoscale 2006*. 1.
[35] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.
[36] David Sánchez, Jordi Castellà-Roca, and Alexandre Viejo. 2013. Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines. *Inf. Sci.* 218 (2013), 17–30.
[37] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2007. Privacy protection in personalized search. *SIGIR Forum* 41, 1 (2007), 4–17.
[38] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *CIKM 2007*.
[39] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep RankNet for personalized search. In *WSDM 2014*. 83–92.
[40] Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *WSDM 2011*. 85–94.
[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*.
[42] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search Personalization with Embeddings. In *ECIR 2017*.
[43] Thanh Tien Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal Latent Topic User Profiles for Search Personalisation. In *ECIR 2015*. 605–616.
[44] Ryen W. White, Wei Chu, Ahmed Hassan Awadallah, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *WWW '13*. 1411–1420.
[45] Qiong Wu, Kaiwen He, and Xu Chen. 2020. Personalized Federated Learning for Intelligent IoT Applications: A Cloud-Edge based Framework. *CoRR* abs/2002.10671 (2020).
[46] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR 2017*. 55–64.
[47] Jing Yao, Zhicheng Dou, and Ji-Rong Wen. 2020. Employing Personal Word Embeddings for Personalized Search. In *SIGIR 2020*. ACM, 1359–1368.
[48] Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2020. RLPer: A Reinforcement Learning Model for Personalized Search. In *WWW '20*. ACM / IW3C2, 2298–2308.
[49] Puxuan Yu, Wasi Uddin Ahmad, and Hongning Wang. 2018. Hide-n-Seek: An Intent-aware Privacy Protection Plugin for Personalized Web Search. In *SIGIR 2018*. ACM, 1333–1336.
[50] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding History with Context-aware Representation Learning for Personalized Search. In *SIGIR 2020*.
[51] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. 2019. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proc. IEEE* 107, 8 (2019), 1738–1762.
[52] Yun Zhu, Li Xiong, and Christopher Verdery. 2010. Anonymizing user profiles for personalized web search. In *WWW 2010*. ACM, 1225–1226.