THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Effective and Secure Federated Online Learning to Rank

Shuyi Wang

M.S.

0000-0002-4467-5574

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2024

School of Electrical Engineering and Computer Science

Abstract

Online Learning to Rank (OLTR) optimises ranking models using implicit user feedback, such as clicks. Unlike traditional Learning to Rank (LTR) methods that rely on a static set of training data with relevance judgements to learn a ranking model, OLTR methods update the model continually as new data arrives. Thus, it addresses several drawbacks such as the high cost of human annotations, potential misalignment between user preferences and human judgments, and the rapid changes in user query intents. However, OLTR methods typically require the collection of searchable data, user queries, and clicks, which poses privacy concerns for users.

Federated Online Learning to Rank (FOLTR) integrates OLTR within a Federated Learning (FL) framework to enhance privacy by not sharing raw data. While promising, FOLTR methods currently lag behind traditional centralised OLTR due to challenges in ranking effectiveness, robustness with respect to data distribution across clients, susceptibility to attacks, and the ability to unlearn client interactions and data. This thesis comprehensively investigates these challenges and proposes effective and secure FOLTR methods.

First, we address the effectiveness of FOLTR by identifying the limitations of existing work and proposing a new method, termed Federated Pairwise Differentiable Gradient Descent (FPDGD). By adapting Pairwise Differentiable Gradient Descent (PDGD) to the Federated Averaging framework, we significantly improve upon the previous method, as demonstrated by empirical evaluations.

Next, we examine the robustness of FOLTR under non independent and identically distributed (non-IID) data across clients. We identify four data distribution scenarios that can lead to non-IID issues and evaluate their impact on ranking performance. Additionally, we assess common federated learning approaches to mitigate these non-IID challenges in FOLTR.

We also investigate data and model poisoning attack strategies and their impact on FOLTR effectiveness. We explore robust aggregation rules for federated learning to counter these attacks, providing insights into the effectiveness of attack and defense methods in FOLTR systems and identifying key factors influencing their success.

Finally, we propose an efficient unlearning method to remove a client's contributions from the FOLTR system using historical local updates. To verify the unlearning process, we introduce a novel evaluation approach based on poisoning attacks.

In summary, this thesis presents a comprehensive study on Federated Online Learning to Rank, addressing its effectiveness, robustness, security, and unlearning capabilities, thereby expanding the landscape of FOLTR.

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Publications included in this thesis

1. [1] **Shuyi Wang**, Shengyao Zhuang, and Guido Zuccon, Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study, *In European Conference on Information Retrieval (ECIR)*, pages 134-149, 2021

2. [2] **Shuyi Wang**, Bing Liu, Shengyao Zhuang, and Guido Zuccon, Effective and Privacy-preserving Federated Online Learning to Rank, *In Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 3–12, 2021

3. [3] **Shuyi Wang**, and Guido Zuccon, Is Non-IID Data a Threat in Federated Online Learning to Rank?, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2801–2813, 2022

4. [4] **Shuyi Wang**, and Guido Zuccon, An Analysis of Untargeted Poisoning Attack and Defense Methods for Federated Online Learning to Rank Systems, *In Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 215–224, 2023

5. [5] **Shuyi Wang**, Bing Liu, and Guido Zuccon, How to Forget Clients in Federated Online Learning to Rank?, *In European Conference on Information Retrieval (ECIR)*, pages 105–121, 2024

Submitted manuscripts included in this thesis

No manuscripts submitted for publication.

Other publications during candidature

No other publications.

Contributions by others to the thesis

The works included in this thesis were done under the supervision of Prof. Guido Zuccon and A/Prof. Bevan Koopman.

Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

Research involving human or animal subjects

No animal or human subjects were involved in this research.

Acknowledgments

I would like to take this opportunity to express my gratitude to everyone who has contributed to my research and personal growth during my PhD journey.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Guido Zuccon. His unwavering support, insightful guidance, and continuous encouragement have been invaluable throughout my PhD. I am deeply impressed by his profound knowledge, dedication to research, and enthusiasm for work. I would like to thank him for providing detailed feedback on my research, encouraging me when I encountered bottlenecks or faced paper rejections, offering generous support for my academic travels, and providing invaluable suggestions on career development. My co-advisor, A/Prof. Bevan Koopman, has also been a tremendous support, offering valuable feedback and engaging insightful discussions about my research. I have greatly benefited from his numerous talks on research methods and academic writing during our group meetings. Their expertise and dedication have profoundly shaped my research, and I am deeply grateful for their mentorship.

My special thanks to Prof. Shane Culpepper, Prof. Gabriele Tolomei and Dr. Harrie Oosterhuis for being my PhD committee. Thank you for your valuable feedback and discussion on improving this thesis.

I would like to extend my sincere thanks to my colleagues and friends in the ielab group: Ahmed, Anton, Daniel, Hang, Harry, Ismail, Joel, Jonathan, Katya, Kim, Linh, Shengyao, Shuai, Shuoqi, Sitthi, Suresh, Watheq, Xinyu. Their constructive feedback and discussions have made my research experience both enjoyable and enriching. I have truly enjoyed our lunch talks, coffee breaks, and group activities on the weekends. I am deeply impressed by their intelligence, hard work, and kindness. I want to thank friends that I met at UQ: Kexuan, Can, Shijie, Tianyu, Pengfei. I enjoyed our board games, road trips, and hanging out at weekends and holidays. Thank you for generously offering support to me and sharing your thoughts on academic life and career development. I wish you all a bright future in your continuing research and career. Further thanks to friends from Europe: Sophia, Gineke, Wojciech, Ferdinand and Maik. It was nice to meet your in Brisbane and thank you for the friendly company in the office. I would also like to thank the numerous others who have inspired, encouraged, and helped me along the way. Your kindness and support have been a source of strength and motivation for me.

I am deeply grateful to my parents and family for their unconditional love and support. Thank you for your understanding, encouragement, and for being there through the ups and downs of this journey. Your patience and sacrifices have made this achievement possible. To my parents, your guidance and values have shaped who I am today, and I am forever indebted to you for your endless support.

Lastly, I would like to acknowledge the love and support provided by Bing. Your encouragement and patience have been vital to me, and without you, this journey would not have been possible.

Thank you all for being a part of this journey.

Financial support

This research was supported by a scholarship from China Scholarship Council and a Google PhD Fellowship.

To my family for their love and support

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations	
LTR	Learning to Rank
CLTR	Counterfactual Learning to Rank
OLTR	Online Learning to Rank
ULTR	Unbiased Learning to Rank
PDGD	Pairwise Differentiable Gradient Descent
FL	Federated Learning
FedAvg	Federated Averaging
FOLTR	Federated Online Learning to Rank
FOLtR-ES	Federated OLTR with Evolutionary Strategies method
FPDGD	Federated Pairwise Differentiable Gradient Descent
non-IID	non independent and identically distributed
SERPs	search engine result pages
CCM	Cascade Click Model
nDCG	normalised Discounted Cumulative Gain
PBM	Position-Based Model
PLMs	Pretrained Language Models

Chapter 1

Introduction

1.1 Background

Ranking is a core task in Information Retrieval [1]. It involves ordering documents in a way that places the most relevant results at the top of the list in response to a user need, typically formulated by means of a search query. The effectiveness of ranking algorithms directly impacts the user experience and satisfaction of search engines and various information retrieval applications. Consequently, significant research efforts are dedicated to developing and optimizing ranking models, with a particular focus on Learning to Rank (LTR) methods that leverage machine learning techniques to improve ranking performance. These models score the relevance of each candidate document to a given query and rank them based on these scores. To explore the relevance between queries and documents, the data used for training typically consist of the multi-dimensional feature representations of each query-document pair along with corresponding relevance judgements. Various loss functions, such as *pairwise*, *pointwise*, and *listwise* [13], can be utilized to learn the ranking models, aiming to achieve better ranking effectiveness.

Although learning to rank methods have achieved exceptional ranking performance with the help of supervision, the necessity of human labeling presents a significant bottleneck, as data annotation is extremely costly in the field of information retrieval [14] and other machine learning tasks. In addition to being expensive and time-consuming to generate, human annotations may not always align with real users' preferences [14], and these labels sometimes fail to capture the evolving nature of user search intents [15, 16]. Additionally, human labeling raises ethical issues when applied to private data, such as emails, as users' privacy can not be guaranteed if the annotated documents contain sensitive personal information [17].

To address the document annotation challenges in learning to rank, two groups of methods have been developed by substituting human judgements with real users' feedback. When users interact

[1]The broader research area of Information Retrieval encompasses various domain-specific applications and multimedia documents, such as legal search [8], email search [9], image search [10], music search [11], etc. In this thesis, we focus on the general context of information retrieval known as document retrieval [12]. Examples of document include web pages, email, books, text messages, etc.

with a list of ranked documents, their actions (such as clicks) on the search engine result pages (SERPs) are recorded and used as relevance signals for training ranking models. One method is termed Counterfactual Learning to Rank (CLTR), which utilizes logged user interactions from previous ranking systems as the training set for a new ranking system, following the common offline training setting before deployment. The other method is called Online Learning to Rank (OLTR), which collects real-time user interactions with the ranking result pages and directly learns from them to update the ranking model. This procedure is conducted in an online manner, meaning the ranking system simultaneously handles real users' information needs and learns from user feedback to update itself.

Unlike previous LTR methods, CLTR and OLTR require real user interactions. Although inexpensive to obtain, user interactions contain a significant amount of noise and biases [18], making the translation of these interactions into relevance signals a not straightforward solution. Noise in user interactions arises from the randomness of user behaviour, i.e., users click for unexpected reasons or misclick some documents. In practice, averaging over many interactions can mitigate the influence of noisy interactions, making them less threatening to the system. However, addressing biases in user interactions is more complex, as they consistently stem from user behaviours under the way of how ranking lists are presented. One of the most studied biases is position bias, where users are more likely to examine and click on higher-ranked documents, not necessarily due to their relevance. Another is selection bias, which indicates that user interactions are limited to the presented results only. In the active research area of Unbiased Learning to Rank[2], more biases are being explored along with corresponding methods for remedying them. To handle the inherent biases in user interactions, CLTR methods build user models that explicitly account for biases to optimize ranking metrics unbiasedly using historical interactions. Differently, OLTR methods address biases by randomizing results through online interventions, i.e. actively controlling what is displayed to the user, and optimize ranking models using online interactions directly. OLTR involves continuously updating the ranking model as new data become available. The online aspect allows the model to adapt to changes in user preferences or document collections in real-time. This thesis focuses on study of OLTR methods.

One key issue that has been neglected in previous learning-from-user-interaction methods is that user interactions with ranking result pages always reflect user's preferences, posing a threat to their privacy. Figure 1.1 demonstrates an overview of an OLTR system where user interactions are gathered in an online manner. The ranking system is deployed on one or several centralised servers responsible for indexing the documents, producing the result lists, and collecting users' queries and search interactions to optimise the ranker. A drawback of this centralised paradigm is that user's privacy cannot be fully guaranteed. Firstly, the server maintains a record of the searchable documents in the form of an index. For web search where documents are crawled on the Internet, the threat to user privacy is limited as the documents are publicly accessible. However, it is not desirable when the documents to be searched are personal, such as in email search [9] or desktop search [21], or

[2]Unbiased Learning to Rank (ULTR) broadly includes two groups of methods: Counterfactual Learning to Rank and Online Learning to Rank, both of which share the same goal of learning unbiased user preferences from biased user interactions [19, 20].

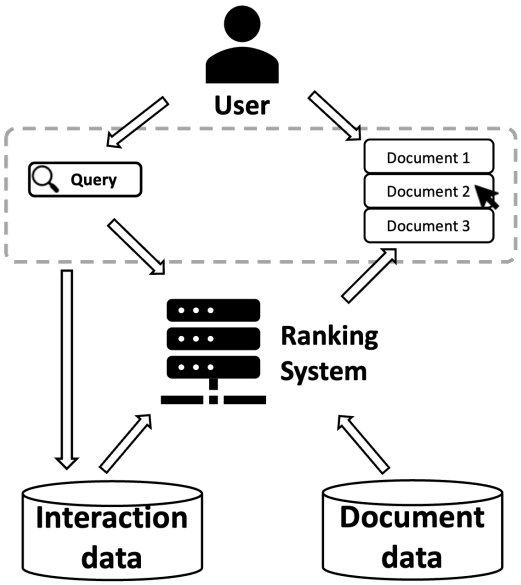


Figure 1.1: Illustration of Online Learning to Rank (OLTR) system.

when the data contains private or valuable information, such as in medical record search [22], or enterprise search [23]. Another potential privacy risk is that the central server also observes other information that the users may wish to keep private, such as the queries they issue and their clicks on search results, which reflect personal information needs, search interests, and user behaviour. This information can disclose users' demographic attributes, personal habits, political views, etc [24–26]. With growing awareness of protecting personal data and the enforcement of data protection regulations such as the General Data Protection Regulation (GDPR) in the European Union, and the California Consumer Privacy Act (CCPA) in California, privacy protection has become not only an ethical issue but also a legal requirement. Thus, the design of a responsible ranking system requires anonymising the user-generated data during search activities.

In this thesis, we focus on addressing the aforementioned privacy issues in the context of Online Learning to Rank. As an online method, OLTR has been shown to be more adaptive to changes in user search intents [16] compared to CLTR which is fully implemented offline. However, the current implementation of OLTR systems poses threats to user privacy: the privatization of user queries and interactions cannot be guaranteed, nor can the confidentiality of documents containing personal information, such as those in desktop or email search scenarios. Therefore, there is a need for methods that ensure more reliable, effective, and privacy-preserving OLTR systems.

1.2 Task Definition

To address privacy issues in OLTR, this thesis focuses on a new training pipeline that transitions previous centralised OLTR approaches into a Federated Learning (FL) setting, termed Federated Online Learning to Rank (FOLTR) [27]).

First proposed by McMahan et al. [7], Federated Learning addresses data privacy issues in training machine learning models. In the traditional setting, known as Centralised Machine Learning, training

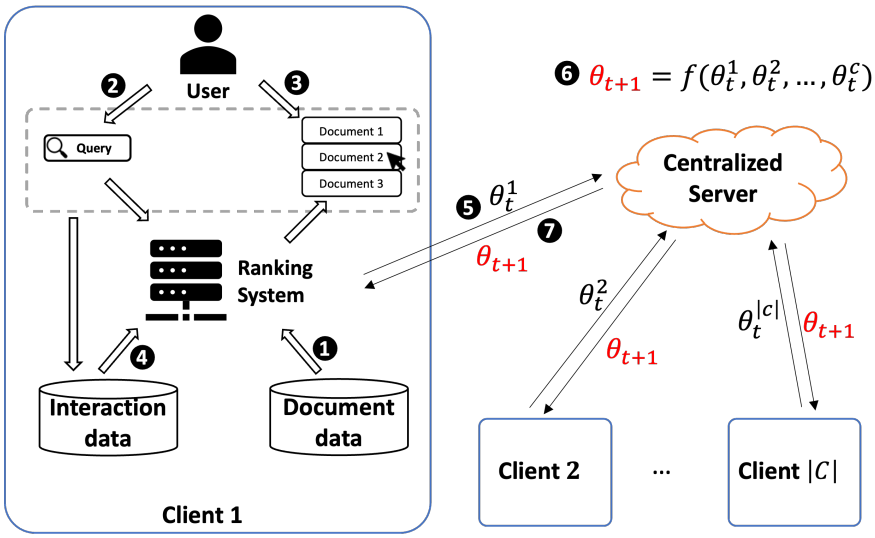


Figure 1.2: Schematic representation of Federated Online Learning to Rank (FOLTR) setting.

data are typically gathered by a central server before the learning process. This approach poses a threat to user privacy, especially if the training data contain sensitive information such as personal health records, user profiles, or financial transactions. For example, multiple hospitals want to collaboratively improve their machine learning models for diagnosing certain diseases. The sharing of and gathering patient data to a central server will be infeasible due to the data privacy restriction. With stricter data regulations and increased user awareness of data protection, obtaining such private data has become increasingly difficult and infeasible. Additionally, as these data are usually distributed across multiple devices, gathering them centrally is costly due to limited communication bandwidth. However, machine learning models, particularly deep learning models, require large amounts of data to achieve high performance, especially in today's natural language processing and computer vision tasks. Federated Learning provides a solution: instead of sharing data, the training is moved to the device of each data owner (the "client"). Each client trains a local model using its local dataset. All local models have the same structure and only the information needed for collaborative training is sent to a central server. In the most commonly used federated learning algorithm, FedAvg [7], model updates are communicated from each client to a central server, which takes the weighted average of the model updates from all clients to perform an update for the global model. The newly updated global model is then sent back to each client to replace the local model, and subsequent local training is based on this updated model. Federated Learning allows each client to benefit from the data of all participants through collaboration, without the threat of sharing local data.

In a typical Federated Online Learning to Rank (FOLTR) [3] setting, depicted in Figure 1.2, the documents to be searched, along with user queries and interactions, are withheld from the central server, and so is also the responsibility of producing search results. Instead, search is performed within the user device ❷, which also indexes the user's data ❶, collects users interactions ❸, and performs the online updates required by the OLTR method ❹. Updates from several user devices are then

[3] In general, FL has been approximately grouped into three types: horizontal FL, vertical FL and transfer FL [28]. In this thesis, we only consider horizontal FL for our FOLTR framework.

shared with a central server ❺. The central server is responsible for combining the ranker updates from different clients to produce a new version of the ranker ❻, which is then distributed to the user devices ❼. The advantage of a federated OLTR solution over a traditional OLTR approach is that neither user data nor user queries and interactions are seen by or shared with anyone, thus potentially preserving user privacy.

In this thesis, we conduct a comprehensive study of Federated Online Learning to Rank, focusing on addressing the limitation of previous methods and tackling unique challenges in this area.

1.3 Research Questions

Unlike most machine learning tasks, stochastic gradient descent (SGD) optimisation is not commonly used in Online Learning to Rank [29] due to two main reasons: 1) the algorithm learns directly from user interaction data in an online manner, and 2) pairwise, listwise and pointwise loss functions are commonly used in ranking tasks [13]. However, Federated Learning typically relies on local SGD under an offline training setting with local datasets on device. This brings specific challenges for adopting FL for OLTR, demonstrated also by the existence of just a single FOLTR method prior to the work in this thesis.

Challenge 1: Low Effectiveness in Balancing Ranking Performance and Privacy Protection. The only FOLTR method existing prior to the work of this thesis, FOLtR-ES [27], demonstrated the feasibility of conducting OLTR within the FL framework. Empirical studies and theoretical analyses have shown its effectiveness in producing ranking lists and ensuring user privacy protection with the help of differential privacy. This has highlighted the potential of addressing privacy issues in OLTR using FL. However, due to the lack of some commonly-used evaluation protocols in OLTR, this method requires to be further explored and verified its effectiveness. As we later demonstrate in this thesis, several limitations of FOLtR-ES exist: it performs poorly on large-scale datasets and other common evaluation practices in OLTR, it does not effectively optimize neural ranking models, and its ranking performance lags behind the state-of-the-art OLTR in centralised setting. As the optimal goal of OLTR is to serve users' information needs, the ranking performance should be given equal importance to user privacy protection. One key aspect neglected in the previous FOLTR method is addressing the biases in user interactions. Without accounting for bias interactions, the exploration of model updates can only reach to a suboptimal space. Therefore, how to effectively address biases from user interactions in a federated setting and simultaneously protect user privacy remain the highest priority challenge.

Challenge 2: Uncertain Robustness under Heterogeneous Clients. In a federated setting, training data is distributed across local devices, leading to data heterogeneity or non independent and identically distributed (non-IID) data. Since local data is not shared between clients and the central server, ignoring non-IID issues can result in divergence among local model updates, negatively impacting the convergence of the global model during federated aggregation. While several methods have addressed FL under non-IID data and proposed solutions for tasks in computer vision and natural language processing [6, 30–32], there is a lack of focus on this issue in FOLTR. Given that users have

different search intents and interaction behaviours, it further exacerbates the non-IID problem and complicates the development of robust and effective FOLTR systems. Due to the diversity of search activities and lack of non-IID benchmarks, the key factors that lead to non-IID issues and their impact on ranking performance remain unclear and require investigation.

Challenge 3: Potential Security Risk Posed by Malicious Clients. Existing FOLTR systems are not necessarily secure: the federated mechanism provides malicious clients with opportunities to attack the effectiveness of the global ranker by deliberately manipulating their local data (data poisoning) or local model updates (model poisoning). For example, malicious clients can send arbitrary weights to the server, disrupting the convergence of the global ranker after aggregation. This kind of attack, termed untargeted poisoning attack, aims to compromise the integrity of the global model trained federatively [33]. This issue is critical for federated learning systems but has not yet been studied on FOLTR where the impact of a poisoned ranking model is further exacerbated due to its online updating and service. To understand the vulnerability of existing FOLTR methods to these attacks and the effectiveness of related defense mechanisms, two main challenges exist: first, how to design and implement existing data and model poisoning attacks to the context of FOLTR; second, how to design defense strategies and validate their utility under various settings of attack.

Challenge 4: Addressing User's Right to Be Forgotten. Recent data protection legislations, such as the General Data Protection Regulation (GDPR) [34] and the California Consumer Privacy Act (CCPA) [35], emphasize a user's unconditional right to be forgotten. This means users can request that contributions made using their data be removed from learned models, a concept also known as machine unlearning [36] and federated unlearning [37]. A federated learning system should accommodate the possibility for clients to leave the federation and request the erasure of their data contributions. However, existing FOLTR systems lack an unlearning mechanism to forget certain users' contributions, making their design defective in this respect. Developing an effective and efficient unlearning mechanism presents two main challenges. The first challenge is how to efficiently unlearn without requiring an unreasonable amount of additional computation while ensuring the new ranker maintains comparable effectiveness to a model retrained from scratch. The second challenge is how to evaluate the effectiveness of an unlearning method. In a FOLTR system with many clients, the impact of removing a client's data on the ranker's effectiveness may be marginal or even unnoticeable. A natural question from a user leaving the federation is: how can it be proven that the impact of their data on the ranker has been erased? Therefore, an adequate evaluation method is required to verify whether an unlearning process effectively achieves forgetting.

In this thesis, we address the unique research gaps in FOLTR by focusing on the following research questions.

1.3.1 RQ1: How to design an effective and generalisable FOLTR method that can achieve competitive performance compared to state-of-the-art OLTR methods?

As the service for addressing users' information needs, the ranking effectiveness of FOLTR should be a primary concern. It is currently unknown if existing methods can achieve performance on par with centralised OLTR methods. If not, the key question is how to enhance the ranking effectiveness for FOLTR, ensuring that the use of federated learning for user privacy protection does not significantly compromise ranking performance. Additionally, exploring methods for integrating other data protection techniques, such as differential privacy [38], into FOLTR could provide feasible and adaptive solutions for balancing privacy protection and model utility. This RQ aligns with Challenge 1.

RQ1.1: Does the only existing FOLTR method (FOLtR-ES) generalize to other commonly-used datasets and metrics in OLTR?

With this RQ, we aim to conduct a comprehensive analysis of previous FOLTR method with a focus on ranking utility. Specifically, to compare the performance with centralised setting, we evaluate the existing FOLTR method on large-scale datasets and use commonly-used metrics against OLTR baselines trained in a centralised manner. A thorough understanding of the existing method, including their pros and cons, will inform the design of future methods in this field. This analysis will provide insights into the generalisability and effectiveness of FOLtR-ES across diverse scenarios.

RQ1.2: How to integrate the state-of-the-art OLTR method, PDGD, with the widely-used, easy-to-implement FedAvg framework while further protecting user privacy?

Unlike other applications of FL, OLTR methods handle noisy and biased user feedback in an online manner, and Stochastic Gradient Descent (SGD) is not commonly used for optimising ranking models in OLTR. In this RQ, we investigate whether the state-of-the-art OLTR method, Pairwise Differentiable Gradient Descent (PDGD) [39], can be integrated into the common FL framework to achieve a highly effective ranker. Simultaneously, methods for integrating other data protection techniques, such as differential privacy [38], will be explored to provide enhanced privacy guarantees. This research will focus on achieving a balance between ranking performance and privacy protection by adapting the OLTR method to fit the FL framework and enhancing it with additional privacy-preserving techniques.

1.3.2 RQ2: Are FOLTR methods robust to non-IID data among clients?

Within this RQ, we consider a scenario where the local training data are non independent and identically distributed (non-IID) across participating clients in FOLTR. This situation violates the common IID assumption in machine learning and can cause the global model to update in a biased direction [6]. Investigating this scenario is crucial because the learning of FOLTR relies on user-inputted queries, local documents, and relevance signals inferred from user interactions. Both search behaviours and

interactions with search engines result pages can vary among users, naturally leading to non-IID issues for FOLTR. Understanding the robustness of FOLTR methods under these conditions will help in designing more resilient and effective ranking systems that can handle diverse user behaviours and interaction patterns. This RQ is in line with Challenge 2.

RQ2.1: What are the potential types of non-IID data in FOLTR and their impact on model performance?

Since there has been no prior study addressing the non-IID challenge in FOLTR, the typical types of non-IID data specific to search scenarios remain unidentified, as do their potential impacts on ranking performance. In this RQ, we aim to identify and categorise types of non-IID data specific to search scenarios and understand which types cause significant degradation in ranking effectiveness compared to IID settings. By concretizing the types of non-IID data specific to search scenarios and analyzing their impacts, this RQ aims to uncover the potential challenges and degradations in ranking performance caused by non-IID issues in FOLTR systems.

RQ2.2: Do existing methods that deal with non-IID data in FL for other tasks generalise to FOLTR?

As demonstrated in this thesis, certain types of non-IID data negatively impact the ranking performance of FOLTR. The focus of this RQ is to investigate methods that can alleviate the impact of non-IID data. Specifically, we exploit existing methods proven effective in supervised federated learning under offline setting and migrate them into FOLTR, where training occurs online and relevance feedback is noisy and biased. By exploring and adapting existing methods for handling non-IID data in general FL to the context of FOLTR, this RQ aims to mitigate the negative impacts of non-IID data on ranking performance in FOLTR systems.

1.3.3 RQ3: Are FOLTR methods secure in the face of potential risks brought by malicious participants?

FOLTR relies on collaboration among participating clients. However, this exposes the system to potential risks from malicious participants who aim to obstruct the convergence of the global model and impair its performance by transmitting arbitrary or carefully crafted model weights. These are termed poisoning attacks with two main categories: *data poisoning*, where the attack is to inject perturbation into the training data without any modification upon the training process itself, and *model poisoning*, where the attackers directly manipulate the training process by altering the objective function, directly editing the model post-training, and so on. Given the lack of previous research on how these attacks affect FOLTR systems, within this RQ, we aim to provide a comprehensive understanding of potential risks brought by poisoning attacks and investigate methods for defending against them. This RQ aligns with Challenge 3.

RQ3.1: Can data poisoning and model poisoning strategies undermine ranking performance on FOLTR?

This research question explores the potential risks associated with poisoning attacks in the context of FOLTR. Inspired by the established data and model poisoning strategies within general FL, we tailor and perform them under the specific nuances of the FOLTR environment. Furthermore, we plan to examine several crucial factors that influence the extent of ranking performance degradation. Understanding these factors is expected to inform the development of more robust defensive strategies against such attacks.

RQ3.2: Can defense strategies mitigate the impact of malicious attacks in FOLTR?

As we demonstrate in this thesis, poisoning attacks can cause degradation to the ranking effectiveness under FOLTR. Within this RQ, we exploit defense strategies to alleviate the impact from malicious attacks. Specifically, we employ widely recognized robust aggregation rules as defensive measures and assess their performance in counteracting both data poisoning and model poisoning attacks. The evaluation considers various attack configurations to understand how these defenses can best maintain the integrity and effectiveness of the ranking system. Furthermore, to assess the feasibility of deploying these defensive strategies in real-world settings, we examine scenarios without any attacks to verify if there is any decline in effectiveness when these defenses are implemented.

1.3.4 RQ4: How to effectively and efficiently forget client data from the trained ranker when a client requests to exit the FOLTR system?

Users of FOLTR have the right to discontinue their participation when they no longer require the service. In response to the requirements driven by users' privacy concerns and recent data regulations, the model should eliminate any knowledge derived from the data of users who opt out. A straightforward method involves retraining the model from scratch without the data from the exiting users, but this approach is computationally expensive and time-consuming. To overcome this, methods for Machine Unlearning and Federated Unlearning have been studied. This research question delves into these methods which are underexplored within the context of FOLTR. Specifically, it assesses both the effectiveness — vital as FOLTR addresses users' information needs in real-time — and the efficiency — crucial as FOLTR operates in an online environment and need to adapt swiftly to dynamic conditions. This RQ aligns with Challenge 4.

RQ4.1: How to evaluate the effectiveness of unlearning?

Evaluating the effectiveness of unlearning presents significant challenges, as evidenced by existing literature. Several factors contribute to these challenges. First, due to the stochastic inherence of the learning processes of machine learning and federated learning, the learnt model is influenced by numerous configuration factors that are difficult to control. This variability can result in vastly

different models even when trained on the same dataset, making it unclear whether changes in the model are due to effective unlearning strategies or other variables. Second, the leaving of a small number of users may not significantly impact model performance, as the effect of one user's data can be relatively minor. Consequently, traditional evaluation settings and metrics are inadequate for assessing unlearning effectiveness accurately. Within this RQ, we aim to highlight these issues in the FOLTR context and develop a suitable evaluation framework that adequately measures the effectiveness of unlearning strategies.

RQ4.2: How to efficiently forget the client without requiring retraining from scratch?

In addition to effectiveness, the efficiency of unlearning is another question worth studying. Specifically, unlearning methods need to demonstrate efficiency compared to the baseline that consists of retraining from scratch. If not, the necessity for unlearning diminishes, rendering the method ineffective. Within this RQ, we explore unlearning methods tailored to FOLTR and analyse the factors influencing both the effectiveness and efficiency of unlearning within the established evaluation framework.

1.4 Contributions

In this thesis, we thoroughly explore the aforementioned research questions with the aim of developing effective, robust, secure federated online learning to rank methods.

We commence by examining the effectiveness of FOLTR, identifying shortcomings in existing work, and proposing an effective method named FPDGD. Addressing a crucial gap in previous research, we tackle the bias stemming from user implicit feedback by adapting the Pairwise Differentiable Gradient Descent method to the Federated Averaging framework. To ensure a robust privacy guarantee, we introduce a noise-adding clipping technique grounded in the theory of differential privacy to be used in combination with FPDGD. Our empirical evaluations demonstrate that FPDGD significantly outperforms the sole existing federated OLTR method. Moreover, FPDGD exhibits greater robustness across various privacy guarantee requirements than the current method, rendering it more dependable for real-world applications. This work has resulted in two publications:

1. Shuyi Wang, Shengyao Zhuang, and Guido Zuccon, Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study, *In European Conference on Information Retrieval (ECIR)*, pages 134-149, 2021

2. Shuyi Wang, Bing Liu, Shengyao Zhuang, and Guido Zuccon, Effective and Privacy-preserving Federated Online Learning to Rank, *In Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 3–12, 2021

We assess the resilience of FOLTR under non independent and identically distributed (non-IID) data across participating clients. Initially, we enumerate four potential data distribution settings that may give rise to non-IID issues. Subsequently, we delve into the impact of each setting on

ranking performance, pinpointing which data distribution may present challenges for FOLTR methods. This analysis constitutes an important contribution to the current landscape of FOLTR research as understanding the factors influencing performance, particularly the effects of non-IID data, is paramount for the deployment of FOLTR systems. This work has resulted in the following publication:

1. Shuyi Wang, and Guido Zuccon, Is Non-IID Data a Threat in Federated Online Learning to Rank?, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2801–2813, 2022

We examine both data and model poisoning attack strategies to showcase their impact on FOLTR search effectiveness. Additionally, we investigate the efficacy of robust aggregation rules in federated learning to counter these attacks. This exploration offers understanding of the effect of attack and defense methods for FOLTR systems, as well as identifying the key factors influencing their effectiveness. Our research contributes to a deeper understanding of the dynamics between attack and defense methods in FOLTR systems, providing insights for enhancing their resilience in real-world scenarios. This work has resulted in the following publication:

1. Shuyi Wang, and Guido Zuccon, An Analysis of Untargeted Poisoning Attack and Defense Methods for Federated Online Learning to Rank Systems, *In Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 215–224, 2023

Lastly, we study an effective and efficient unlearning method designed to remove a client's contribution without compromising the overall ranker effectiveness, and without necessitating a complete retraining of the global ranker. A pivotal challenge lies in assessing whether the model successfully unlearns the contributions from the client requesting removal (referred to as "the unlearned client"). To tackle this, we instruct the unlearned client to execute a poisoning attack through adding noise to its updates. Subsequently, we measure whether the impact of the attack diminishes after the unlearning process has been executed. Through empirical experiments, we showcase the effectiveness and efficiency of the unlearning strategy under various parameter settings combinations. This work has resulted in the following publication:

1. Shuyi Wang, Bing Liu, and Guido Zuccon, How to Forget Clients in Federated Online Learning to Rank?, *In European Conference on Information Retrieval (ECIR)*, pages 105–121, 2024

In summary, this thesis delivers a comprehensive examination on Federated Online Learning to Rank, addressing various unexplored areas concerning its effectiveness, robustness, and security. By delving into these domains, the research broadens the landscape of FOLTR, offering insights and advancements to the field.

1.5 Thesis Overview

The thesis is structured as follows:

Chapter 2: Background and Literature Review. This chapter provides the background and literature review of Learning to Rank, Online Learning to Rank, Federated Learning and related methods pertinent to this thesis. Specifically, an explanation of the relevant terminology, methodologies, evaluation protocols, and datasets is provided, which lays the groundwork for subsequent chapters.

Chapter 3: Effective and Privacy-Preserving FOLTR. We conduct an analysis of the existing FOLTR method and introduce our method designed for effective and privacy-preserving FOLTR. (Supporting paper: [1, 2])

Chapter 4: FOLTR with non-IID Data. We present our work, which introduces a terminology of non-IID data in FOLTR and analyses its performance under various federated aggregation rules. (Supporting paper: [3])

Chapter 5: FOLTR under Poisoning Attacks. We introduce our study, which investigates the potential risks posed by data and model poisoning attacks to FOLTR systems, along with the effectiveness of corresponding defense strategies. (Supporting paper: [4])

Chapter 6: Unlearning for FOLTR. We introduce our method for effective and efficient unlearning in FOLTR, which is the first study of its kind. (Supporting paper: [5])

Chapter 7: Conclusion. We conclude by providing a comprehensive summary of our findings, emphasizing our contributions to the field of FOLTR. We discuss avenues for further research, particularly in areas crucial for practical FL implementations for ranking tasks but still underexplored.

Chapter 2

Background and Literature Review

This chapter provides an overview of Learning to Rank, Online Learning to Rank methods, Federated Learning and other related works that are involved in this thesis. Additionally, the common experimental settings, including datasets, user simulation, and evaluation metrics, will be introduced in detail.

2.1 Learning to Rank

Learning to Rank (LTR) is the task to construct a ranking model with the aim to rank documents in a manner that optimally satisfies user queries [13]. To achieve this, the ranking function is learned with the help of machine learning techniques using a set of labelled query-document pair examples, represented by predefined features [40]. Specifically, LTR has evolved through three primary approaches: the pointwise, pairwise, and listwise approaches. Pointwise approaches, such as those presented in the works of Gao et al. [41], treat ranking as a regression or classification problem on individual documents. Pairwise methods, exemplified by the RankNet model developed by Burges et al. [42], focus on comparing pairs of documents to determine their relative orders. Listwise approaches, with models like ListNet [43] and LambdaRank [44], consider the entire list of documents when optimizing the ranking. Recent advancements in deep learning have further propelled LTR, integrating neural networks to enhance ranking accuracy and scalability. Despite these advancements, challenges such as data sparsity, scalability, and data annotation continue to drive research in this dynamic field.

A key limitation of LTR is the reliance on explicit relevance annotations, which require substantial effort and costs to collect [45]. Editorial labels also pose ethical issue when labelling private data [17] such as emails. In addition, user preferences may not agree with that of annotators [14] and these labels cannot reflect evolving user preferences and search intents [15]. To solve the data annotation issues for LTR, two group of methods have been proposed and investigated with the integration of user feedback. One method is termed Counterfactual Learning to Rank (CLTR) [46, 47], which improves ranking models using logged user interaction data from previous ranking systems, such as clicks, in a manner that accounts for potential biases and confounding factors by using techniques from causal

inference and counterfactual reasoning. The other method is called Online Learning to Rank (OLTR), which collects real-time user interactions and directly learns from them to update the ranking model. This procedure is conducted in an online and interactive environment, meaning the ranking system simultaneously handles real users' information needs and learns from user feedback to update itself.

2.2 Online Learning to Rank

Online Learning to Rank (OLTR) differs from traditional Learning to Rank (LTR) methods in that it makes use of user online feedback (such as clicks) as the optimization goal in order to tackle some data-annotation drawbacks of LTR.

The Dueling Bandit Gradient Descent (DBGD) uses online evaluation to unbiasedly compare two or more rankers given a user interaction [29]. Subsequent work developed more reliable or more efficient online evaluation methods, including Probabilistic Interleaving (PIGD) [48], and its extension, the Probabilistic Multileaving (PMGD) [49], which compares multiple rankers at each interaction, resulting in the best DBGD-based algorithm. However, this method suffers from a high computational cost because it requires sampling ranking assignments to infer outcomes [50]. Further variations that reuse historical interaction data to accelerate the learning in DBGD have also been investigated [51].

The Pairwise Differentiable Gradient Descent method (PDGD) [39] constructs a pairwise gradient to update the ranking model according to the click feedback. Compared to previous OLTR methods, PDGD has showcased faster learning (higher online performance) and convergence to higher effectiveness (higher offline performance). Unlike other methods, PDGD can be applied to any differentiable ranking model and it is an unbiased method. Oosterhuis and de Rijke trained a linear and a neural ranker and both showed effective results [39].

Counterfactual Online Learning to Rank (COLTR) [50] uses the same DBGD framework for updating the ranker interactively. Instead of interleaving or multileaving, the key difference of COLTR from DBGD-based methods is the use of counterfactual evaluation from Counterfactual Learning to Rank (CLTR) to replace online evaluation thus COLTR uses clicks collected by the current ranker to evaluate candidate rankers.

Another group of methods focus on leveraging powerful offline learning learning rank solutions to the online setting. PairRank [52] directly learns a pairwise logistic regression ranker online and explores the pairwise ranking space via a divide-and-conquer strategy based on the model's uncertainty about the documents' rankings. A follow-up work by Jia and Wang [53] is proposed to extend PairRank with a multi-layer neural ranker. However, this method suffers from high computational complexity caused by the exploration in the pairwise document ranking space. To solve the computational efficiency bottleneck, Jia et al. [54] propose an efficient exploration strategy based on bootstrapping: an ensemble of ranking models trained with perturbed user click feedback.

2.2.1 Evaluation Practice in OLTR

It is common practice in OLTR to simulate user interactions with the search results using labelled offline learning to rank datasets [39, 51, 55]. This is because no public dataset with LTR features and clicks is available; in addition OLTR methods directly manipulate the rankings that have to be shown to users, so even if a public dataset with LTR features and clicks was to be available, this could not be used for OLTR. This section introduces the common implementation and evaluation practice in OLTR, which this thesis also follows.

Datasets

In the study of OLTR, four learning-to-rank (LTR) datasets are commonly used: MQ2007 [40], MSLR-WEB10K [40], Yahoo [45], and Istella-S [56]. Each dataset contains query ids and candidate document lists for each query, which is formalized as exclusive query-document pairs. Each query-document pair is represented by a multi-dimensional feature vector and annotated relevance label of the corresponding document.

Among the selected datasets, MQ2007 is relatively small and has fewer assessed documents per query. The dataset, compiled using data from the TREC 2007 Million Query Track [57], contains 1,700 queries, divided into 5 folds, with associated relevance assessments, expressed on a 3-level scale, from *not relevant* (0) to *very relevant* (2). Each query-document pair is represented by a 46-dimensional feature vector.

Provided by commercial search engines, the other three datasets are larger and more recent. MSLR-WEB10K, constructed from a retired labelling set of a Microsoft web search engine, has 10,000 queries and each query is associated with 125 documents on average, each represented with 136 features. Yahoo has 29,900 queries and each query-document pair has 700 features. Istella-S is the largest, with 33,018 queries, 220 features, and an average of 103 documents per query. These three commercial datasets are all annotated for relevance on a five-grade-scale: from *not relevant* (0) to *perfectly relevant* (4). Both MQ2007 and MSLR-WEB10K datasets have five data splits (stored separately in five data folders) while Yahoo and Istella-S contain only one.

User simulations

In standard setup for user simulations in OLTR [39, 51, 55], user's queries are selected by randomly choosing from the dataset and user interactions on the search engine result pages (SERPs) are simulated accordingly. While there are many forms of user interactions, the most common and widely-used one is the click signal, which is also the only form of user interaction that is considered and simulated throughout this thesis.

In particular, user's clicks are simulated relying on the *Cascade Click Model* (CCM) click model [58] based the relevance label and ranking position of the candidate documents. Specifically, for each query, the SERPs are commonly limited to k documents. User clicks on the displayed ranking list are generated based on the CCM click model. Each user is assumed to inspect every

document displayed in a SERP from top to bottom while clicks the document with click probability $P(click = 1|rel(d))$ conditioned with the actual relevance label. After a click happens, the user will stop the browsing session with stop probability $P(stop = 1|click = 1, rel(d))$, or continue otherwise. In the aforementioned previous works, same as experiments in this thesis, three widely-used instantiations of CCM click model are considered: *perfect, navigational, informational*. A *perfect* user inspects every document in a SREP with high chance to click high relevant documents thus provides very reliable feedback. A *navigational* user also searches for reasonably relevant document but has a higher chance of stopping browsing after one click. An *informational* user typically clicks on many documents without a specific information preference thus provides the noisiest click feedback. The implementations of these click models are detailed in Table 2.1 and Table 2.2.

Table 2.1: The three click model instantiations used for the MQ2007 dataset. $r = rel(d)$ represents the relevance label for the query-document pair.

	$p(click = 1\|rel(d))$			$p(stop = 1\|click = 1, rel(d))$		
click model	$r = 0$	$r = 1$	$r = 2$	$r = 0$	$r = 1$	$r = 2$
$perfect$	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

Table 2.2: The three click model instantiations used for the datasets with five-grade-scale relevance annotation (MSLR-WEB10K, Yahoo!, and Istella-S datasets). $r = rel(d)$ represents the relevance label for the query-document pair.

	$p(click = 1\|rel(d))$					$p(stop = 1\|click = 1, rel(d))$				
click model	$r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$
$perfect$	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
navigational	0.05	0.3	0.5	0.7	0.95	0.2	0.3	0.5	0.7	0.9
$informational$	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

Evaluation metric

The evaluation practice from previous OLTR work consists of measuring the rankers' offline and online performance by using normalised Discounted Cumulative Gain (nDCG) [39, 53].

As each SERP is limited to k documents, $nDCG@k$ is used for evaluating the overall offline effectiveness of the ranker. Effectiveness is measured by averaging the $nDCG$ scores of the ranker on the queries in the held-out test dataset with the actual relevance label. This performance value is recorded after each update of the ranker and the final convergence performance is also recorded.

To measure online performance, it is common practice to compute the cumulative discounted $nDCG@k$ for the rankings displayed during the training phase [59]. For t-th query in a sequence of T queries, with R^t representing the ranking list displayed to the user, online performance is computed as:

$$Online\_Performance = \sum_{t=1}^{T} nDCG(R^t) \cdot \gamma^{t-1}. \tag{2.1}$$

This measure indicates the quality of the user experience during training; the discount factor γ puts more importance to the interactions in the early phases of training. Following previous work [39], in this thesis, we set $\gamma = 0.9995$.

In the common evaluation practice in OLTR, each experiment is repeated multiple times, spread evenly over all dataset training folds. All evaluation results are averaged and statistical significant differences between system pairs are evaluated using two-tailed Student's t-test with Bonferroni correction.

2.3 Federated Learning

2.3.1 Federated learning with privacy protection

Federated Learning [7] considers a machine learning setting where several data owners (clients) collaboratively train a model without sharing the data. To this aim, a coordinator (central server) is required and each participating client sends what is essential for the training (for example, the local gradient) to the central server. The Federated Averaging (FedAvg) algorithm [7] relies on local stochastic gradient descent (SGD), which is computed by each client, while the centralised server performs the global model update by weighted averaging the client's updates.

Differential privacy [38, 60] is a method often used to ensure user privacy preservation in federated learning (the alternative is the encryption of the messages). We shall review the notion of ε-differential privacy in Section 3.2.1. Abadi et al. [61] have developed the algorithmic techniques for combining SGD-based deep learning methods with differential privacy. Differential privacy has also been added to FedAvg algorithm to provide user-level differential privacy guarantees for language models [62], as for these models it has been shown that the gradients or shared model's weights can reveal sensitive information regarding the clients' data [63–65].

2.3.2 Federated OLTR

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

The Federated OLTR with Evolutionary Strategies method (FOLtR-ES) [27] has been the first method that was proposed to address the above privacy-preserving issues in OLTR. This method extends the OLTR optimization scenario to the Federated SGD [7] and uses Evolution Strategies as the optimization method [66]. To further protect the gradient updates from a malicious attacker, FOLtR-ES also includes a privatization procedure, which leverages the theory of ε-local differential privacy [38]. FOLtR-ES is shown to perform well on small-scale datasets (MQ2007) for the MaxRR metric, although performance degrades when privacy preservation is required [27]. However, the

effectiveness of FOLtR-ES is not consistent across other, larger-scale datasets nor when typical OLTR metrics are considered (nDCG) [1].

2.3.3 Federated Learning in other Information Retrieval Tasks

Aside from its usage in online learning to rank, recent works have applied federated learning in other information retrieval contexts [67]. Zong et al. [68] provide a solution for cross-modal retrieval in a distributed data storage scenario, which uses federated learning to reduce the potential privacy risks and the high maintenance costs encountered when dealing with large amount of training data. Wang et al. [69] study learning to rank (but not in the online setting we consider in this thesis) in a cross-silo federated learning setting; this work is aimed at helping companies that have access to limited labelled data to collaboratively build a document retrieval system, in an efficient way. Hartmann et al. [70] use federated learning to improve the ranking of suggestions in the Firefox URL bar, so that the training of the ranker on user interactions is performed in a privacy-preserving way; they show that this federated approach improves on the suggestions produced by the previously employed heuristics in Firefox. Yang et al. [71] describe the use of federated learning for search query suggestions in the Google Virtual Keyboard (GBoard) product. Here, a baseline model is used to identify relevant query suggestions given a user query; candidate suggestions are then filtered using a triggering model learnt using federated learning. Aside from the previous examples, federated learning has also seen adoption in the area of personalised search [72], which aims to return search results that cater to the specific user's interests. Yao et al. [73] recently proposed a privacy protection enhanced personalized search framework which adapts federated learning to the state-of-the-art personalized search model.

2.3.4 Federated Learning with non-IID data

Zhu et al. have compiled a comprehensive survey on the impact of non-IID data on federated learning [6], also reviewing the current research on handling these challenges. Early work from Zhao et al. [30] shows a deterioration of the accuracy of federated learning if non-IID or heterogeneous data is present; they also provide a solution to this problem by creating a small subset of globally shared data between all clients (local devices). Li et al. [74] analyse the convergence of the federated learning algorithm `FedAvg` [7] (which is a component of the FOLTR method we rely upon for investigation [3]) on non-IID data and empirically show that data heterogeneity slows down the convergence. This raised attention to the presence of non-IID data in federated learning.

Generally speaking, existing approaches for handling non-IID issues in federated learning can be classified into three categories: data-based approaches, algorithm-based approaches, and system-based approaches [6]. Data sharing [30] and data augmentation [75] are two kinds of typical data-based approaches. While they achieve state-of-the-art performance, they fundamentally conflict with the objective of federated learning: that of not sharing data across clients. This is because, for example, methods such as data sharing require a subset of private data to be shared across all clients. While proposals have been made to use synthetic, rather than real, data for the data sharing mechanism [76]

it is unclear (1) what the effectiveness loss of the sharing of synthetic data in place of real data is, and (2) whether the sharing of synthetic data could still jeopardise privacy as this synthetic data is typically generated from real data, and thus analysis of the synthetic data may reveal key aspects of and information contained in the real data. Algorithm-based approaches mainly focus on personalisation methods like local fine-tuning of a neural model [77] and Personalized FedAvg (Per-FedAvg) [78] – which are both limited mainly to neural models – or the casting of the federated learning process into a multi-task learning problem [79]. System based approaches adopt clustering [80] and tree-based structure [81] to deal with non-IID data. Limitations exist among all proposed approaches, and this is still a much unexplored line of research.

2.3.5 Poisoning Attacks on Federated Learning

Poisoning attacks on federated learning systems aim to compromise the integrity of the system's global model. Poisoning attacks can be grouped according to the goals of the attack into two categories: untargeted poisoning attacks, and targeted poisoning attacks (also known as backdoor attacks).

Targeted poisoning attacks aim to manipulate a global model according to the attacker's objectives, such as misclassifying a group of data with certain features to a label chosen by the attacker, while maintaining normal model effectiveness under other conditions. This is accomplished through backdoor attacks [33, 82], which are designed to allow the targeted manipulations to transpire stealthily and without detection.

In contrast, untargeted poisoning attacks (also known as Byzantine failures [83–87]) aim to decrease the overall effectiveness of the global model indiscriminately for all users and data groups. Current untargeted poisoning methods can be divided into two categories: data poisoning and model poisoning. Label flipping [88] is a representative data poisoning method: the labels of honest training data are changed without altering their features. Model poisoning, on the other hand, directly affects the local model updates before they are sent to the centralized server. For example, Baruch et al. [89] poisons the local model updates through the addition of noise computed from the variance between the before-attack model updates, while Fang et al. [86]'s attacks are optimized to undermine specific robust aggregation rules.

Among untargeted poisoning attacks on federated learning systems, model poisoning methods have been found to be the most successful [33]. In particular, data poisoning attacks have limited success when Byzantine-robust defense aggregation rules are in use [86]; we introduce these defense methods in Section 5.3. Furthermore, most data poisoning attacks assume that the attacker has prior knowledge about the entire training dataset, which is often unrealistic in practice.

In this thesis, we focus on untargeted poisoning attacks, delving into the effectiveness of both data poisoning and model poisoning methods. These attack methods are studied within the framework of a FOLTR system based on FPDGD, with and without the integration of defense countermeasures.

2.3.6 Machine Unlearning and Federated Unlearning

Machine Unlearning pertains to the removal of any evidence of a chosen data point from the model, a process commonly known as selective amnesia. Except ensuring the removal of certain data points from the model being the primary objective, the unlearning procedure should also do not affect the model's effectiveness. Machine unlearning has been explored in both centralised [36,90] and federated settings [37,91–93].

Methods in Machine Unlearning

Methods in machine unlearning can be broadly classified into two families: exact unlearning and approximate unlearning. Exact unlearning methods are designed to provide a theoretical guarantee that the methods can completely remove the influence of the data to be forgotten [94–98]; but a limitation of these methods is that they can only be applied to simple machine learning models. Approximate unlearning methods, on the other hand, are characterized by higher efficiency, which is achieved by relying on specific assumptions regarding the accessibility of training information, and by permitting a certain amount of reduction in the model's effectiveness [99–101]. These methods can be used on more complex machine learning models (e.g., deep neural networks). The existing unlearning methods for federated learning belong to the approximate unlearning family [37, 102, 103]. However, most of these methods are applied to classification tasks and no previous work considers either ranking or FOLTR systems.

Evaluation of Machine Unlearning

A key challenge posed by the task of unlearning in the federated learning context is how to evaluate whether an unlearning method has successfully removed the contributions from the client that requested to leave the federation. This evaluation is at times termed as unlearning verification [104], which specifically aims to certify that the unlearned model is unrelated to the data that needed to be removed. Due to the stochastic nature of the training for many machine learning models, it is difficult to distinguish the individual clients' models and their unlearned counterparts after a certain group of data is removed.

To evaluate the effectiveness of unlearning, a group of methods leverage membership inference attacks [37, 105, 106], i.e. a kind of attack method that predicts whether a data item belongs to the training data of a certain model [64]: it is then straightforward to adapt this type of attacks to verify if the unlearned samples participated in the unlearning process. However, conducting successful membership inference attacks needs subtle design and training of the inference model. The effectiveness of such attacks on FOLTR is unknown as it is an unexplored area. Thus, we do not consider this type of verification in this thesis.

Another type of methods is inspired from the idea of poisoning attacks. These methods add arbitrary noise [107] or backdoor triggers [91, 108–110] to the data that is needed to be deleted, with the aim of manipulating the effectiveness of the trained model. After the unlearning of poisoned

datasets has taken place, the impact from the arbitrary poisoning or backdoor triggers should be reduced in the unlearned model: the extent of this reduction (and thus model gains) determines the extent of the success of unlearning.

2.4 Chapter Summary

In this chapter, we introduce the basic concepts and related works for this thesis. In particular, we give an overview of existing methods in Online Learning to Rank and the common evaluation practice used in OLTR. We introduce a group of related works in Federated Learning with focus on privacy-protection techniques, dealing with non-IID data, poisoning attacks and defense, and unlearning methods. Furthermore, we summarise federated learning applications in other Information Retrieval and Recommendation tasks. In the next chapter, we start introducing our main work, findings and contributions for this research.

The following publication has been incorporated as Chapter 3.

1. [1] **Shuyi Wang**, Shengyao Zhuang, and Guido Zuccon, Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study, *In European Conference on Information Retrieval (ECIR)*, pages 134-149, 2021
2. [2] **Shuyi Wang**, Bing Liu, Shengyao Zhuang, and Guido Zuccon, Effective and Privacy-preserving Federated Online Learning to Rank, *In Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 3–12, 2021

Chapter 3

Effective and Privacy-Preserving Federated Online Learning to Rank

This chapter considers the problem of devising effective online learning to rank (OLTR) methods embedded in a federated system. Specifically, we investigate the following research questions:

RQ1: How to design an effective and generalisable FOLTR method that can achieve competitive performance compared to state-of-the-art OLTR methods?

RQ1.1: Does the existing FOLTR method (FOLtR-ES) generalize to other commonly-used datasets and metrics in OLTR?

RQ1.2: How to integrate the state-of-the-art OLTR method with the widely-used, easy-to-implement FedAvg framework while further protecting user privacy?

We investigate this problem into two parts. The first part correspond to Section 3.1 where we conduct reproduction of the previous FOLTR method with the focus on understanding its limitation to answer RQ1.1. The second part corresponds to Section 3.2 where we propose an effective FOLTR method with further privacy-preserving plugin and term it as FPDGD. Following common experimental analysis in OLTR and FL, we demonstrate the effectiveness improvement of FPDGD compared to the FOLtR-ES baseline, which address RQ1.2.

3.1 Limitations of FOLtR-ES

In the context of few attention on guaranteeing user's privacy in OLTR, Kharitonov proposed FOLtR-ES [27] as the first attempt in the area of FOLTR. The method relies on the common federated learning paradigm [28], in which data (collection, queries, interactions) is maintained at each client's side along with a copy of the ranker, and updates to the rankers that are learned from the interaction on the client side are shared to the central server, which is responsible for aggregating the update signal from clients and propagate the aggregated ranker update. In this specific case, all users observe and act on the same

feature space; each user however retains control of their own data, which includes the collection, the queries and the interactions. FOLtR-ES uses evolutionary strategies akin to those in genetic algorithms to make client rankers explore the feature space, and a parametric privacy preserving mechanism to further anonymise the feedback signal that is shared by clients to the central server.

However, the original research study that introduced this method only evaluated it on a small Learning to Rank (LTR) dataset and with no conformity with respect to current OLTR evaluation practice. It further did not explore specific parameters of the method, such as the number of clients involved in the federated learning process, and did not compare FOLtR-ES with the current state-of-the-art OLTR method.

To remedy the aforementioned gap, in this section, we replicate and then reproduce the experiments from the original work of Kharitonov [27], investigating the effect different configurations of that federated OLTR method have on effectiveness and user experience, extending and generalising its evaluation to different settings commonly used in OLTR and to different collections. Specifically, we address the following research questions:

RQ1.1.1: *Does the performance of FOLtR-ES generalise beyond the MQ2007/2008 datasets?* The original method was only evaluated using MQ2007/2008 [40], while current OLTR practice is to use larger datasets that are feature richer and that contain typical web results.

RQ1.1.2: *How does the number of clients involved in FOLtR-ES affect its performance?* FOLtR-ES was previously evaluated using a set number of clients involved in the federated OLTR process ($n = 2,000$), and it was left unclear whether considering more or less client would impact performance.

RQ1.1.3: *How does FOLtR-ES compare with current state-of-the-art OLTR methods?* Compared to OLTR methods, FOLtR-ES preserves user privacy, but it is unclear to what expense in terms of search performance: the original work compared FOLtR-ES to rankers in non-federated settings, but the rankers used in there were not the current state-of-the-art in OLTR.

RQ1.1.4: *How does FOLtR-ES performance generalise to the evaluation settings commonly used for OLTR evaluation, i.e. measuring offline and online performance, with respect to nDCG and with relevance labels?* The original evaluation of FOLtR-ES considered an unusual setting for OLTR, consisting of using MaxRR [111] as evaluation measure in place of nDCG, computed on simulated clicks instead of on relevance labels.

3.1.1 Federated OLTR with Evolution Strategies

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

Federated Learning Optimization Setting

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

Gradient Estimation

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

$$\theta^* = \arg\max_{\theta} F(\theta) = \arg\max_{\theta} \mathbb{E}_u \mathbb{E}_{a|u,\theta} f(a;\theta,u) \tag{3.1}$$

Using Evolution Strategies (ES) [66], FOLtR-ES considers a population of parameter vectors which follow the distribution with a density function $p_\phi(\theta)$. The objective aims to find the distribution parameter ϕ that can maximize the expectation of the metric across the population:

$$\mathbb{E}_{\theta \sim p_\phi(\theta)} \left[F(\theta) \right] \tag{3.2}$$

The gradient g of the expectation of the metric across the population (Equation 3.2) is obtained in a manner similar to REINFORCE [112]:

$$\begin{aligned} g &= \nabla_\phi \mathbb{E}_\theta [F(\theta)] = \nabla_\phi \int_\theta p_\phi(\theta) F(\theta) d\theta = \int_\theta F(\theta) \nabla_\phi p_\phi(\theta) d\theta = \\ &= \int_\theta F(\theta) p_\phi(\theta) \left(\nabla_\phi \log p_\phi(\theta) \right) d\theta = \mathbb{E}_\theta \left[F(\theta) \cdot \nabla_\phi \log p_\phi(\theta) \right] \end{aligned} \tag{3.3}$$

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

$$g = \mathbb{E}_{\theta \sim p_\phi(\theta)} \left[F(\theta) \cdot \frac{1}{\sigma^2} (\theta - \phi) \right] \tag{3.4}$$

Based on the federated learning optimization setting, θ is sampled independently on the client side. Combined with the definition of $F(\theta)$ in Equation 3.1, the gradient can be obtained as:

$$g = \mathbb{E}_u \mathbb{E}_{\theta \sim p_\phi(\theta)} \left[\left(\mathbb{E}_{a|u,\theta} f(a;\theta,u) \right) \cdot \frac{1}{\sigma^2} (\theta - \phi) \right] \tag{3.5}$$

To obtain the estimate $\hat{g}$ of g from Equation 3.5, $\hat{g} \approx g$, the following steps are followed: (i) each client u randomly generates a pseudo-random seed s and uses the seed to sample a perturbed model

$\theta_s \sim \mathbb{N}(\phi, \sigma^2 I)$, (ii) the average of metric f over B interactions is used to estimate the expected loss $\hat{f} \approx \mathbb{E}_{a|u,\theta_s} f(a;\theta_s,u)$ from Equation 3.5, (iii) each client communicates the message tuple $(s,\hat{f})$ to the server, (iv) the centralized server computes the estimate $\hat{g}$ of Equation 3.5 according to all message sent from the N clients.

To reduce the variance of the gradient estimates, means of antithetic variates are used in FOLtR-ES: this is a common ES trick [66]. The algorithm of the gradient estimation follows the standard ES practice, except that the random seeds are sampled at the client side.

Privatization Procedure

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

Assume that the metric used on the client side is discrete or can be discretized if continuous. Then, the metric takes a finite number (n) of values, $f_0, f_1, ..., f_{n-1}$. For each time the client experiences an interaction, the true value of the metric is denoted as f_0 and the remaining $n-1$ values are different from f_0. When the privatization procedure is used, the true metric value f_0 is sent with probability p. Otherwise, with probability $1-p$, a randomly selected value $\hat{f}$ out of the remaining $n-1$ values is sent. To ensure the same optimization goal described in Gradient Estimation of Section 3.1.1, FOLtR-ES assumes that the probability $p > 1/n$.

Unlike other federated learning methods, FOLtR-ES adopts a strict notion of ε-local differential privacy [27], in which the privacy is considered at the level of the client, rather than of the server. Through the privatization procedure, ε-local differential privacy is achieved, and the upper bound of ε is:

$$\varepsilon \leq log\frac{p(n-1)}{1-p} \tag{3.6}$$

This means that, thanks to the privatization scheme, at least $log[p(n-1)/(1-p)]$-local differential privacy can be guaranteed. At the same time, any ε-local differential private mechanism also can obtain ε-differential privacy [38].

3.1.2 Experimental Settings

Datasets

The original work of Kharitonov [27] conducted experiments on the MQ2007 and MQ2008 learning to rank datasets [40], which are arguably small and outdated. In our work, we instead consider more recent and larger datasets: MSLR-WEB10K [40] and Yahoo! Webscope [45], which are commonly-used in offline and online learning to rank [39, 47, 50, 51]. Compared to MQ2007/2008, both MSLR-WEB10K and Yahoo! contain many more queries and corresponding candidate documents. In addition, both datasets have much richer and numerous features. For direct comparison with the original FOLtR-ES work, we also use MQ2007/2008. Detailed descriptions about each dataset are discussed in Section 2.2.1.

Simulation

Akin to previous work [27, 51, 55], we simulate users and their reaction with the search results using labelled offline learning to rank datasets.

For the experiment, we follow the same method used by the original FOLtR-ES work. We sample B queries for each client randomly and use the local perturbed model to rank documents. The length for each ranking list is limited to 10 documents. We simulate users' clicks using the Cascade Click Model (CCM) as discussed in Section 2.2.1[1]. After simulating users clicks, we record the quality metric for each interaction and perform the privatization procedure with probability p. Next, we send the averaged metric and pseudo-random seed to optimize the centralized ranker. Finally, each client receives the updated ranker.

Evaluation metric

For direct comparison with the original FOLtR-ES work, we use the reciprocal rank of the highest clicked result in each interaction (MaxRR [111]). This metric is computed on the clicks produced by the simulated users on the SERPs.

The evaluation setting above is unusual for OLTR. In RQ1.1.4 of this section, we also consider the more commonly-used normalised Discounted Cumulative Gain (nDCG) metric, as FOLtR-ES is designed to allow optimization based on any absolute measures of ranking quality. We thus record the nDCG@10 values from the relevance labels of the SERP displayed to users during interactions. This is referred to as online nDCG and the scores represent users' satisfaction [51]. For online evaluation, we take the online $nDCG@10$ score averaged across all clients. We also record the nDCG@10 of the globally-updated ranker measured with a held-out test set: this is refer to as offline nDCG. We record the offline $nDCG@10$ score of the global ranker during each federated training update. Detailed implementation of these metrics are introduction in Section 2.2.1.

FOLtR-ES and Comparison OLTR Methods

In all experiments, we adopt the same models and optimization steps used by Kharitonov [27], and rely on the well document implementation made publicly available by the author. The two ranking models used by FOLtR-ES are a linear ranker and a neural ranker with a single hidden layer of size 10. For optimization, we use Adam [113] with default parameters.

To study how well FOLtR-ES compares with current state-of-the-art OLTR (RQ1.1.3), we implemented the Pairwise Differentiable Gradient Descent (PDGD) [39] (more details of this algorithm is introduced in Section 3.2.1). Unlike many previous OLTR methods that are designed for linear models, PDGD also provides effective optimization for non-linear models such as neural rankers. During each interaction, a weighted differentiable pairwise loss is constructed in PDGD and the gradient is directly estimated by document pairs preferences inferred from user clicks. PDGD has been empirically

[1]For simulating clicks for the MQ2007/2008, we use the same parameter settings from Table 1 in the original FOLtR-ES paper [27] (shown in Table 2.1): these are partially different from those used for MSLR-WEB10K and Yahoo! because relevance labels in these datasets are five-graded, while they are three-graded in MQ2007/2008.

found to be significantly better than traditional OLTR methods in terms of final convergence, learning speed and user experience during optimization, making PDGD the current state-of-the-art method for OLTR [39, 47, 50].

3.1.3 Results and Analysis

RQ1.1.1: Generalisation of FOLtR-ES performance beyond MQ2007/2008

For answering RQ1.1.1 we replicate the results obtained by Kharitonov [27] on the MQ2007 and MQ2008 datasets; we then reproduce the experiment on MSLR-WEB10K and Yahoo! datasets, on which FOLtR-ES has not been yet investigated, and we compare the findings across datasets. For these experiments we use antithetic variates, set the number of interactions $B = 4$ and simulate 2,000 clients, use MaxRR as reward signal and for evaluation on clicked items.

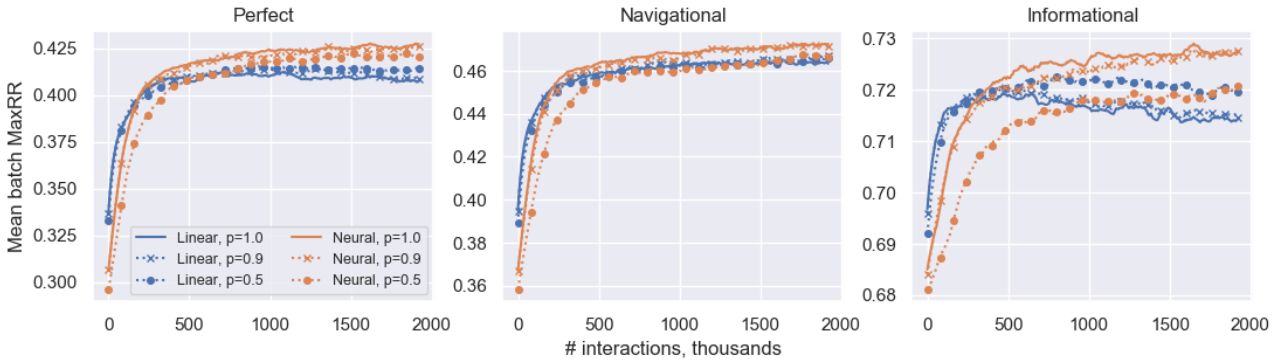
Figure 3.1a reports the results obtained by FOLtR-ES on the MQ2007 dataset with respect to the three click models considered, various settings for the privatization parameter p, and the two ranking models (linear and neural). Our results fully replicate those of Kharitonov [27] and indicate the following findings: (1) FOLtR-ES allows for the iterative learning of effective rankers; (2) high values of p (lesser privacy) provide higher effectiveness; (3) the neural ranker is more effective than the linear ranker when $p \to 1$ (small to no privacy), while the linear model is equivalent, or better (for informational clicks) when $p = 0.5$. Figure 3.1b reports similar results obtained from MQ2008 dataset.

However, not all these findings are applicable to the results obtained when considering MSLR-WEB10K and Yahoo!, which are displayed in Figures 3.1c and 3.1d. In particular, we observe that (1) the results for MSLR-WEB10K (and to a lesser extent also for Yahoo!) obtained with the informational click model are very unstable, and, regardless of the click model, FOLtR-ES requires more data than with MQ2007/2008 to arrive at a stable performance, when it does; (2) the neural ranker is less effective than the linear ranker, especially on MSLR-WEB10K. We believe these findings are due to the fact that query-document pairs in MSLR-WEB10K and Yahoo! are represented by a larger number of features than in MQ2007/2008. Thus, more data is required for effective training, especially for the neural model; we also note that FOLtR-ES is largely affected by noisy clicks in MSLR-WEB10K.

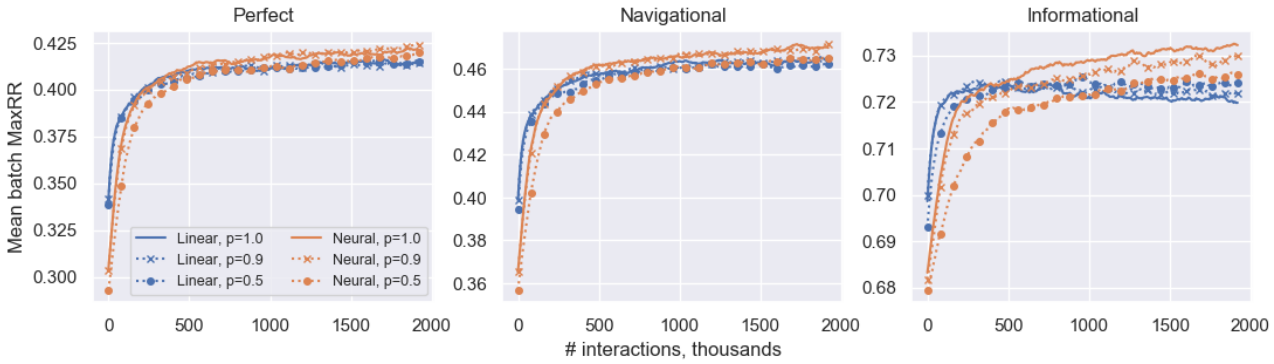
RQ1.1.2: Effect of number of clients on FOLtR-ES

To answer RQ1.1.2 we vary the number of clients involved in FOLtR-ES; we investigate the values $\{50, 1{,}000, 2{,}000\}$. Kharitonov [27] used 2,000 in the original experiments, and the impact of the number of clients has not been studied. To be able to fairly compare results across number of clients, we fixed the total number of ranker updates to 2,000,000; we also set $B = 4$ and $p = 0.9$.

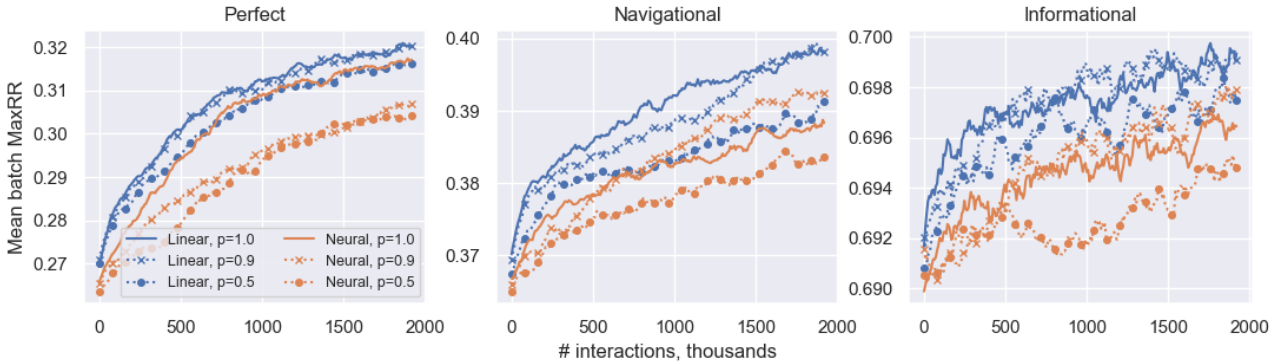
The results of these experiments are reported in Figure 3.2, and they are mixed. For MQ2007, the number of clients have little effect on the neural ranker used in FOLtR-ES, although when informational clicks are provided this ranker is less stable, although often more effective, if very few clients (50) are used. Having just 50 clients, instead, severally hits the performance of the linear ranker, when compared with 1,000 or 2,000 clients. For MQ2008, in general, with more clients, the ranker gains



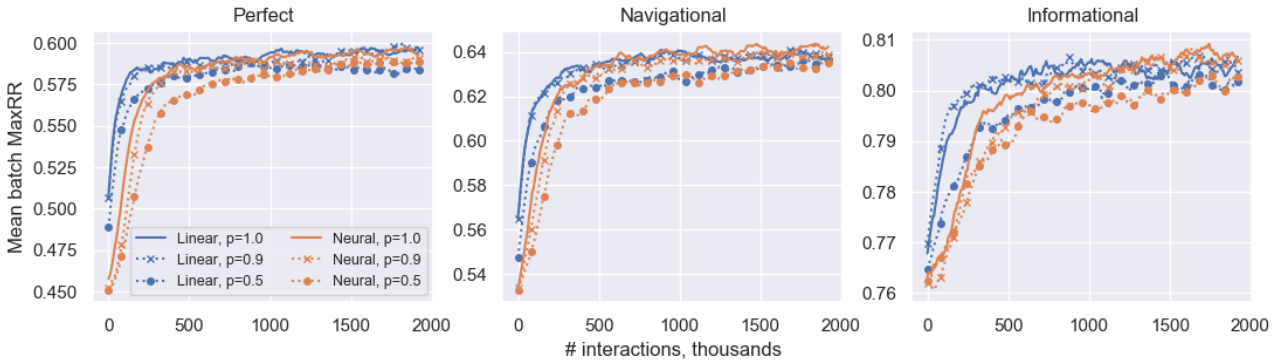
(a) Mean batch MaxRR for MQ2007.



(b) Mean batch MaxRR for MQ2008.

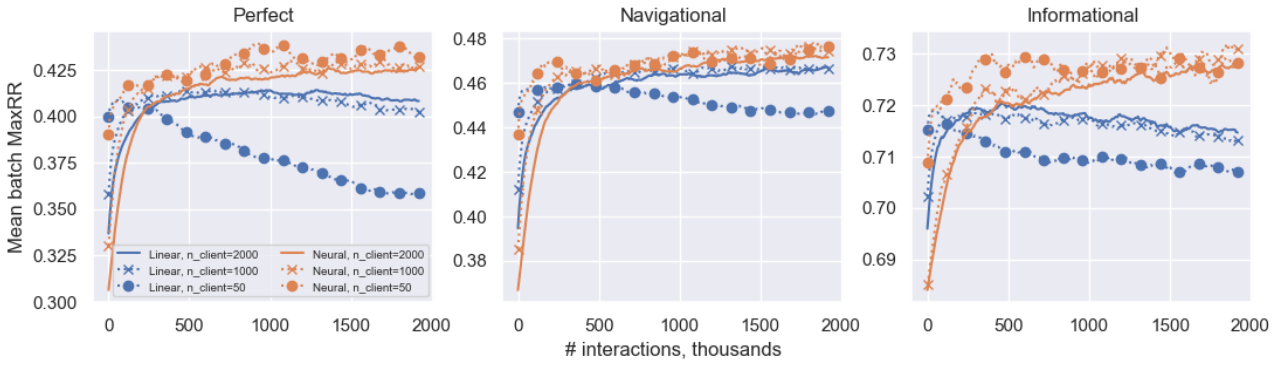


(c) Mean batch MaxRR for MSLR-WEB10K.

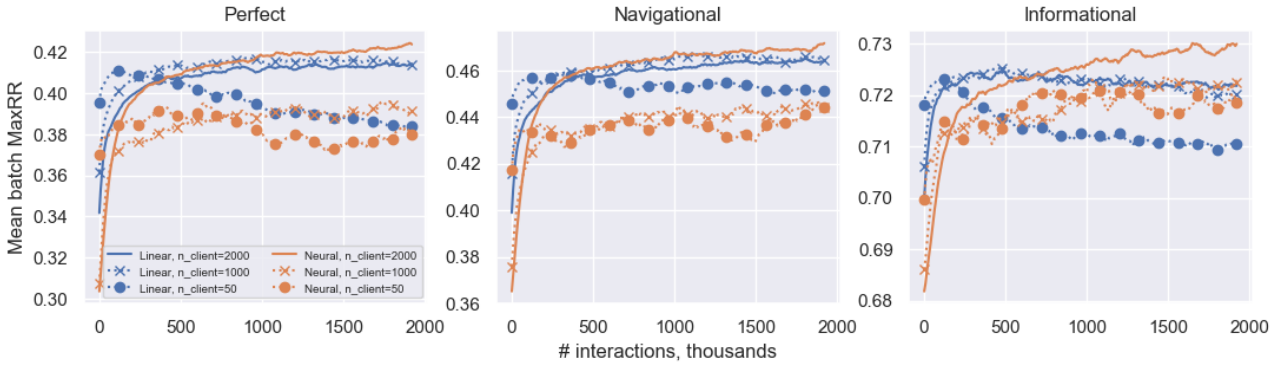


(d) Mean batch MaxRR for Yahoo!.

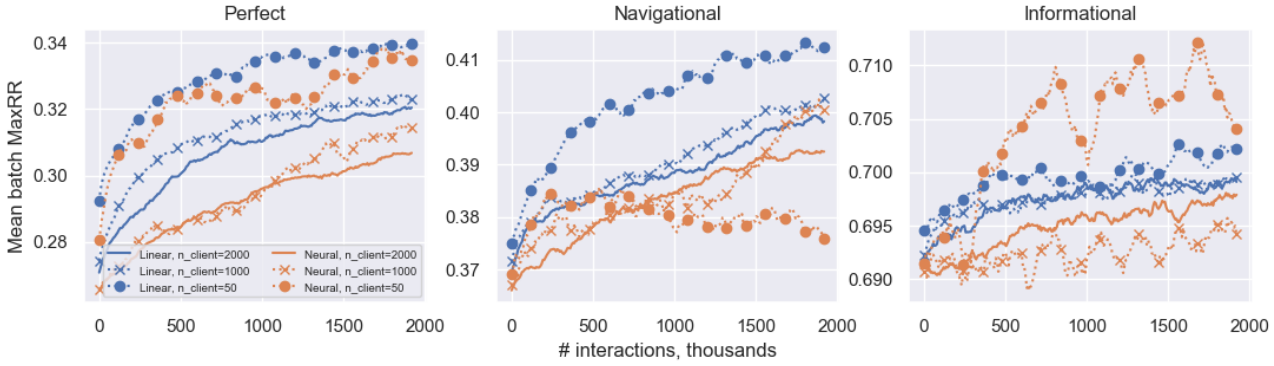
Figure 3.1: Results for RQ1.1.1: performance of FOLtR-ES across datasets under three different click models (averaged across all dataset splits).



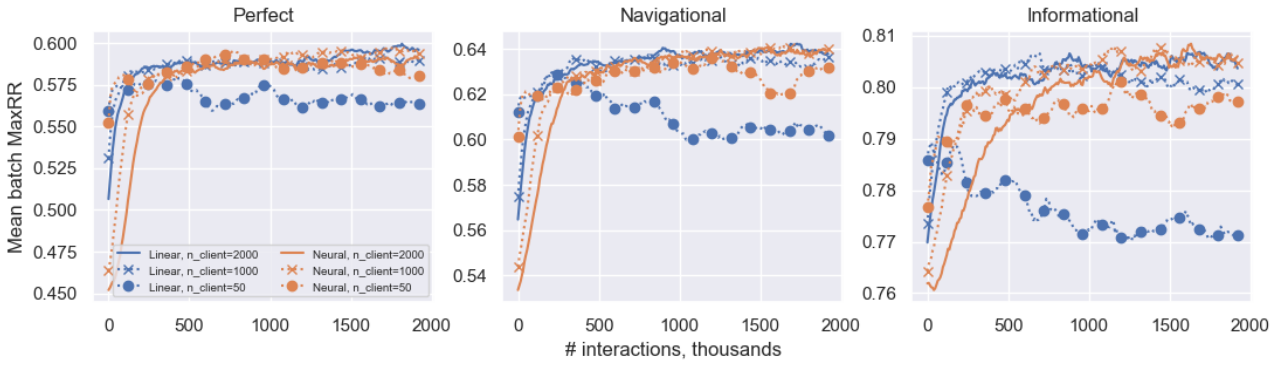
(a) Mean batch MaxRR for MQ2007.



(b) Mean batch MaxRR for MQ2008.



(c) Mean batch MaxRR for MSLR-WEB10K.



(d) Mean batch MaxRR for Yahoo!.

Figure 3.2: Results for RQ1.1.2: performance of FOLtR-ES with respect to number of clients (averaged across all dataset splits).

higher performance. A degradation trend is also found with only 50 clients for linear ranker on MQ2008. The findings on MSLR-WEB10K, however, are different. In this dataset, a smaller number of clients (50), is generally better than larger numbers, both for linear and neural ranker. An exception to this is when considering navigational clicks: in this case the linear ranker obtains by far the best performance with a small number of clients, but the neural ranker obtains the worst performance. For Yahoo! dataset, the number of clients has little impact on the ranking performance with exceptions on linear ranker with 50 clients. Under the noisiest informational clicks, the growing trend of ranking effectiveness is less stable. This suggest that the number of clients greatly affects FOLtR-ES: but trends are not consistent across click types and datasets.

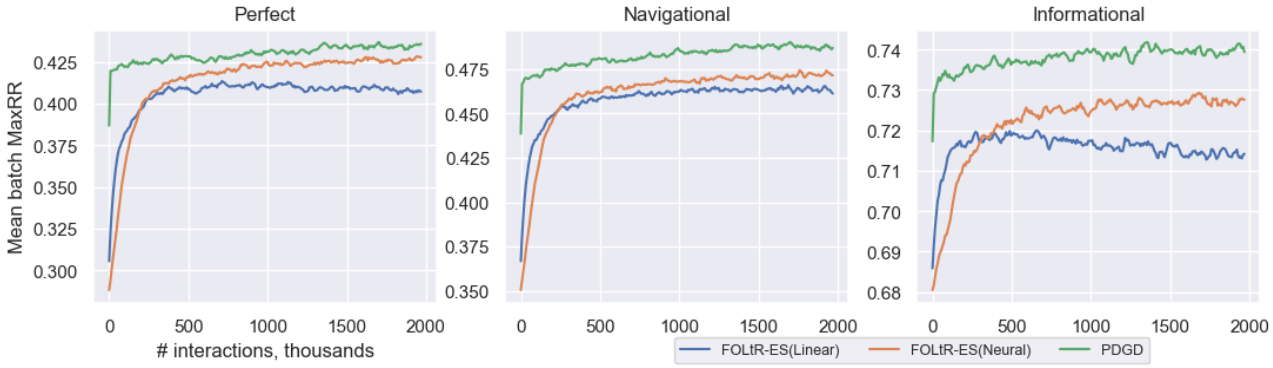
RQ1.1.3: Comparing FOLtR-ES to state-of-the-art OLTR methods

The original study of FOLtR-ES did not compare the method with non-federated OLTR approaches. To contextualise the performance of FOLtR-ES and to understand the trade-off between privacy and performance when designing FOLtR-ES, we compare this method with the current state-of-the-art OLTR method, the Pairwise Differentiable Gradient Descent (PDGD) [39]. For fair comparison, we set the privatization parameter $p = 1$ (lowest privacy) and the number of clients to 2,000. In addition note that in normal OLTR settings, rankers are updated after each user interaction: however in FOLtR-ES, rankers are updated in small batches. For fair comparison, we adapt PDGD to be updated in batch too. Instead of updating the ranker after each interaction (batch size 1), we accumulate gradients computed on the same batch size as for FOLtR-ES. Specifically, with 2000 clients for FOLtR-ES, the batch size of each update is 8,000 iterations ($4 \times 2{,}000$). We then compute the updated gradients for PDGD on 8,000 interactions too.

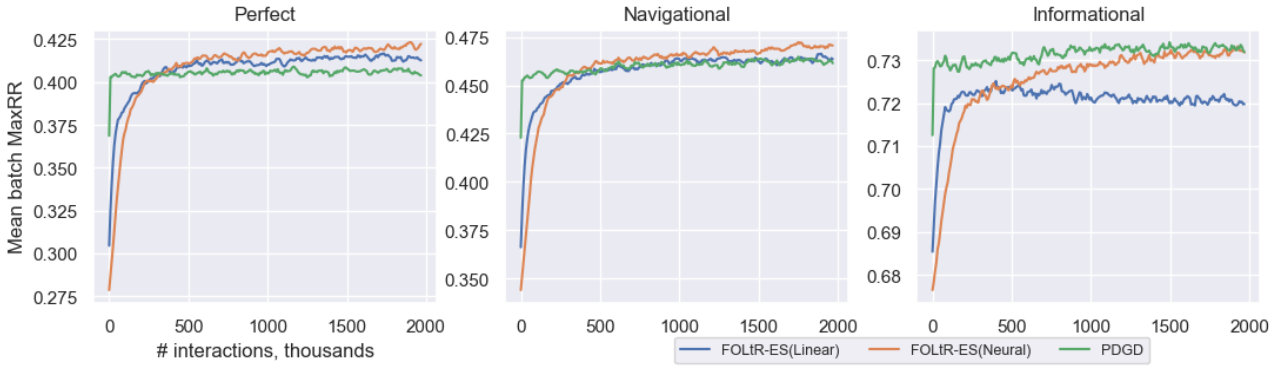
Results are shown in Figure 3.3: for MQ2008 and Yahoo! datasets, PDGD converges faster than FOLtR-ES, which is observed from the higher ranking performance in the early interaction steps. However, the performance gap between PDGD and FOLtR-ES is narrowed after more interactions. For MQ2007 and MSLR-WEB10K datasets, regardless of linear or neural ranker, FOLtR-ES is less effective than PDGD. The gap in performance is greater in larger dataset (i.e., MSLR-WEB10K) than in the smaller MQ2007. This gap becomes even bigger, especially for the first iterations, if the PDGD ranker was updated after each iteration (not shown here), rather than after a batch has been completed. This highlights that FOLtR-ES has the merit of being the first privacy preserving federated OLTR approach available; however, more work is needed to improve the performance of FOLTR based methods so as to close the gap between privacy-oriented approaches and centralise approaches that do not consider user privacy.

RQ1.1.4: Extending FOLtR-ES evaluation to common OLTR practice

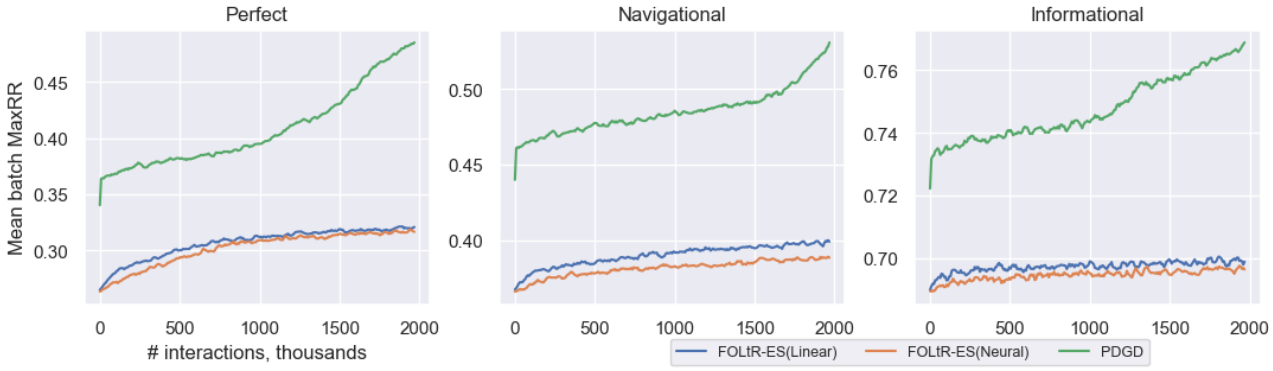
In the original work and in the sections above, FOLtR-ES was evaluated using MaxRR computed with respect to the clicks performed by the simulated users (click models). This is an unusual evaluation for OLTR because: (1) usually nDCG@10 is used in place of MaxRR as metric, (2) nDCG is computed



(a) Mean batch MaxRR for MQ2007.



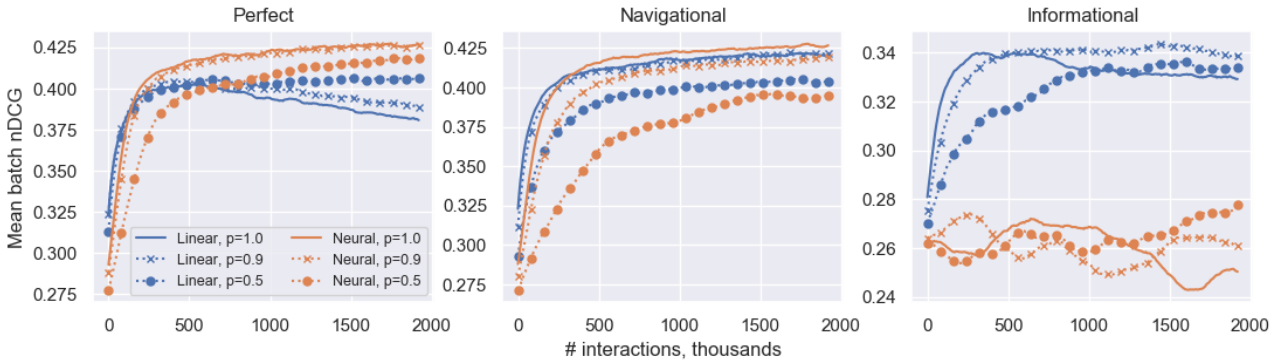
(b) Mean batch MaxRR for MQ2008.



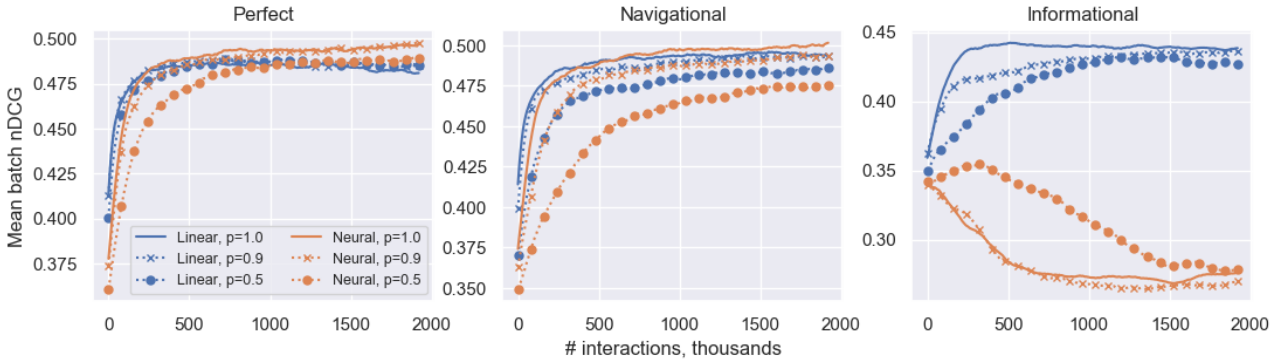
(c) Mean batch MaxRR for MSLR-WEB10K.

(d) Mean batch MaxRR for Yahoo!.

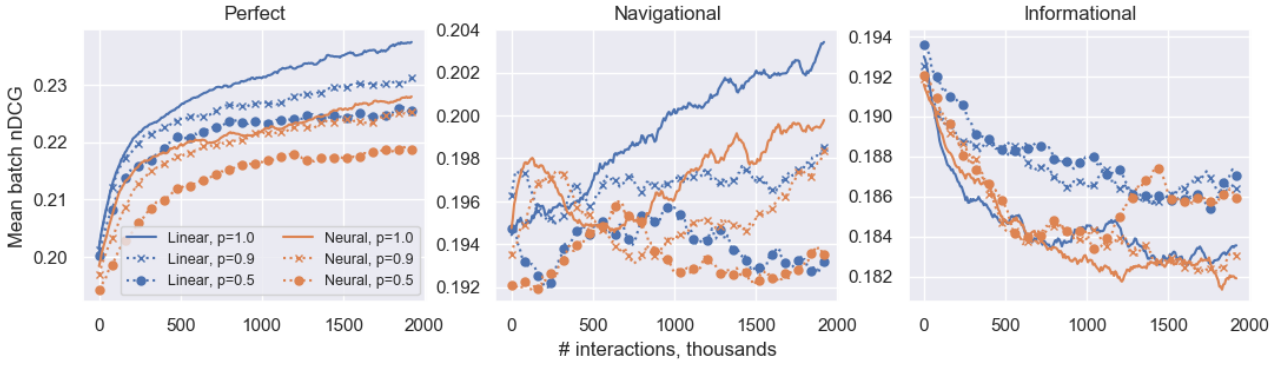
Figure 3.3: Results for RQ1.1.3: performance of FOLtR-ES and PDGD across datasets with privatization parameter $p = 1$ and 2,000 clients (averaged across all dataset splits).



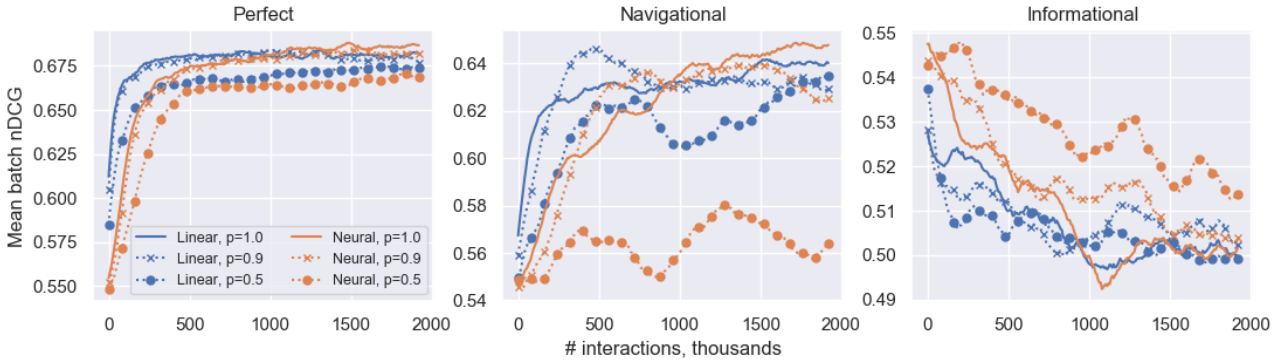
(a) Mean batch nDCG@10 for MQ2007.



(b) Mean batch nDCG@10 for MQ2008.



(c) Mean batch nDCG@10 for MSLR-WEB10K.



(d) Mean batch nDCG@10 for Yahoo!.

Figure 3.4: Results for RQ1.1.4: performance of FOLtR-ES in terms of online nDCG@10 computed using relevance labels and the SERPs used for obtaining user iterations (averaged across all dataset splits).

with respect to relevance labels, and not clicks, and on a withheld portion of the dataset, not on the interactions observed – this is used to produce learning curves and is referred to as offline nDCG, (3) in addition online nDCG is measured from the relevance labels in the SERPs from which clicks are obtained, and either displayed as learning curves or accumulated throughout the sessions – these values represent how OLTR has affected user experience. We then consider this more common evaluation of OLTR next, where we set the number of clients to 2,000 and experiment with $p = \{0.5, 0.9, 1.0\}$.

Results are reported in Figure 3.4. It is interesting to compare these plots with those in Figure 3.1, that relate to the unusual (for OLTR) evaluation setting used in the original FOLtR-ES work. By comparing the figures, we note that for MQ2007/2008, FOLtR-ES can effectively learn rankers for perfect and navigational clicks. However, when the clicks become noisier (informational clicks), then FOLtR-ES learning is effective for the linear ranker but no learning occurs for the neural ranker: this is unlikely in the evaluation settings of the original work (Figure 3.1). We note this finding repeating also for MSLR-WEB10K and Yahoo!, but this time this affects both linear and neural rankers; we also note that the online performance in MSLR-WEB10K on navigational clicks is also quite unstable and exhibits little learning for specific values of p and ranker type. The online performance on MSLR-WEB10K for informational clicks (noisiest clicks) even exhibits a decreasing trend as iterations increase. Similar findings are also observed from results on Yahoo! dataset.

We further investigate the performance of FOLtR-ES with respect to offline nDCG@10. Results are shown in Figure 3.5, and are plotted along with the offline nDCG@10 of PDGD for additional context. Also the offline performance confirm that FOLtR-ES does not provide stable learning across click settings, datasets and ranker types. We also note that the performance of PDGD are sensibly higher than that of FOLtR-ES for MSLR-WEB10K dataset and for all datasets when informational clicks are considered.

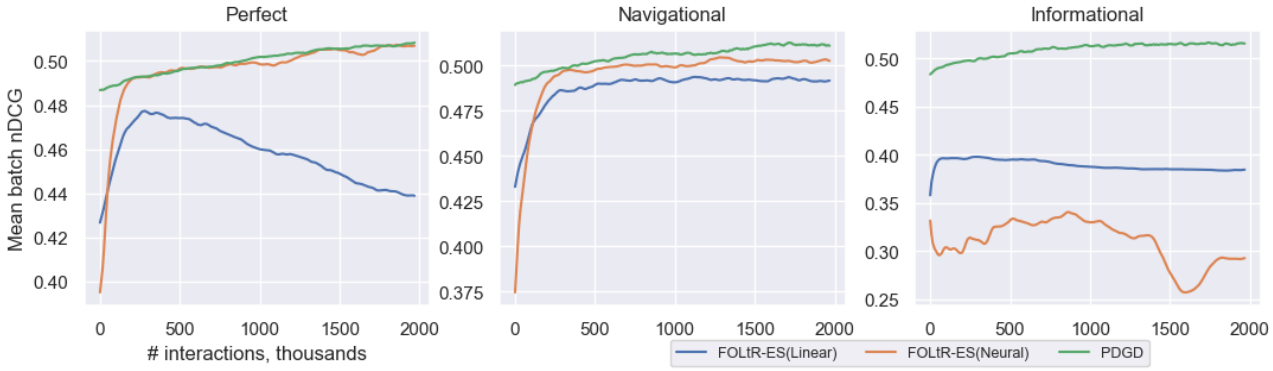
These findings suggest that FOLtR-ES is yet far from being a solution that can be considered for use in practice, and more research is required for devising effective federated, privacy-aware OLTR techniques.

3.1.4 Summary

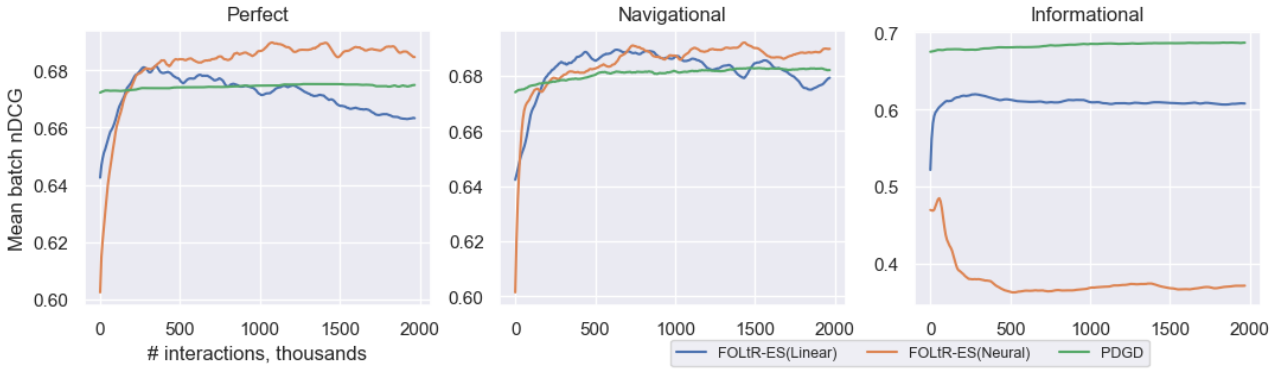
In this section we considered the federated online learning to rank with evolutionary strategies (FOLtR-ES) method recently proposed by Kharitonov [27], which represents the first method addressing privacy requirements in OLTR.

We set to explore four research questions related to FOLtR-ES. RQ1.1.1 aimed to investigate the generalisability of the original results obtained by FOLtR-ES on the MQ2007/2008 dataset to other datasets used in current OLTR practice. Our experiments on MQ2007/2008 show consistent findings with that of Kharitonov [27]. However, when larger LTR datasets are considered, results change. In particular, the neural ranker used in FOLtR-ES is less effective than the linear ranker, especially on MSLR-WEB10K.

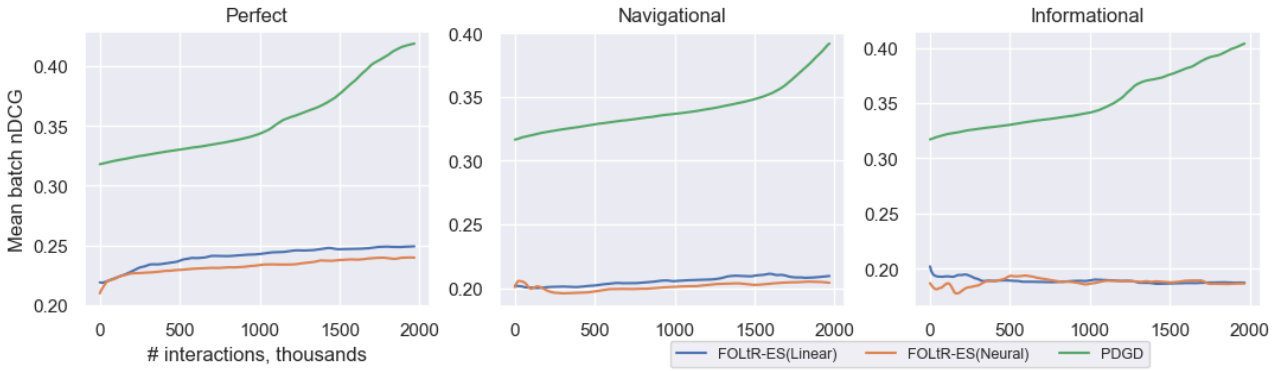
RQ1.1.2 aimed to investigate the effect varying the number of clients involved in FOLtR-ES has



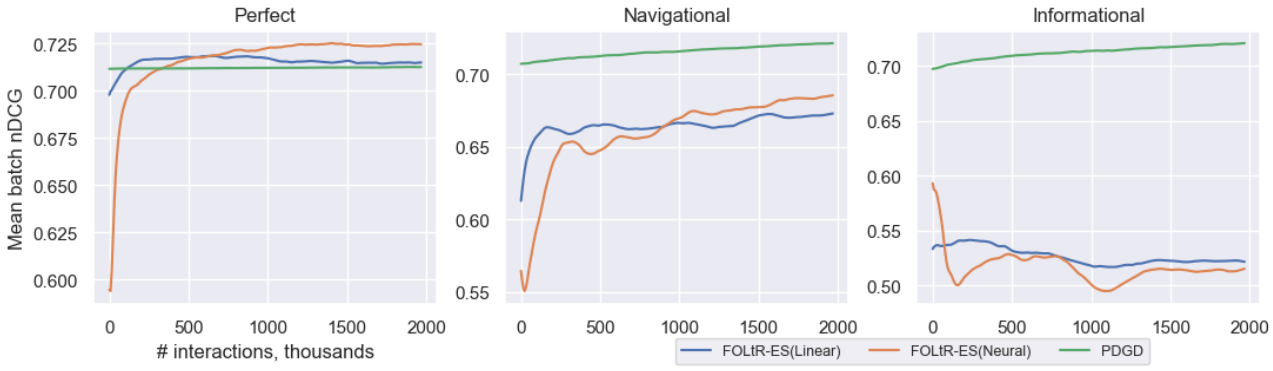
(a) Mean batch nDCG@10 for MQ2007.



(b) Mean batch nDCG@10 for MQ2008.



(c) Mean batch nDCG@10 for MSLR-WEB10K.



(d) Mean batch nDCG@10 for Yahoo!.

Figure 3.5: Results for RQ1.1.4: performance of FOLtR-ES and PDGD in terms of offline nDCG@10 with privatization parameter $p = 1$ and 2,000 clients (averaged across all dataset splits).

on the effectiveness of the method. Our experiments show mixed results with respect to the number of clients: the effect largely varies depending on dataset, ranker type and click settings.

RQ1.1.3 aimed to compare FOLtR-ES with current OLTR state-of-the-art methods to understand the gap required to be paid for maintaining privacy with the help of federated learning. Our experiments show that FOLtR-ES lags behind the current OLTR state-of-the-art in terms of ranking performance: differences become more substantial when noisy clicks or larger datasets are considered.

RQ1.1.4 aimed to investigate the generalisability of the original results obtained for FOLtR-ES to common evaluation practice in OLTR. Our experiments show that if the common evaluation settings used in OLTR are used to evaluate FOLtR-ES, then the method shows high variability in effectiveness across datasets, rankers and clicks types – and overall that FOLtR-ES is unreliable on large datasets and noisy clicks. This finding suggests that more research and improvements are needed before a federated OLTR method, and FOLtR-ES in particular, can be used in practice.

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

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

3.2 Federated Pairwise Differentiable Gradient Descent (FPDGD)

In the previous section, we introduce FOLtR-ES and our further analysis of it. We find that the effectiveness of previous approach for federated OLTR (i.e., FOLtR-ES) exhibits large gaps in performance compared to current state-of-the-art, not federated, OLTR methods. It is this gap that we want to fill in this section. Specifically, in this section, we put forward the following contributions:

- We cast the current state-of-the-art online LTR approach, the Pairwise Differentiable Gradient Descent (PDGD) [39], into the federated learning paradigm, specifically using the federated averaging algorithm. To do so, we devise a federated gradient update for PDGD. Our federated PDGD provides large, significant gains in effectiveness over the current federated OLTR method (FOLtR-ES [1, 27]), and its effectiveness generalises across datasets.
- We apply differential privacy on top of the proposed federated PDGD to secure the gradient updates performed in the federated environment against privacy attacks. We further investigate the trade-off between privacy preservation and ranker effectiveness.

3.2.1 Methodology

In this section we propose a novel federated version of the current state-of-the-art OLTR method, the Pairwise Differentiable Gradient Descent (PDGD) [39], based on the Federated Averaging approach [7].

We then show how the gradient update of our federated OLTR method can be secured using differential privacy. First though, we provide a brief introduction to the PDGD method.

Pairwise Differentiable Gradient Descent

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

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

$$P(d|D) = \frac{e^{f_\theta(d)}}{\sum_{d' \in D} e^{f_\theta(d')}}. \tag{3.7}$$

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

$$P(R|D) = \prod_{i=1}^{k} P(d_i|D \setminus \{d_1, ..., d_{i-1}\}). \tag{3.8}$$

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

$$\nabla f_\theta(\cdot) \approx \sum_{d_k >_c d_l} \rho(d_k, d_l, R, D)[\nabla P(d_k > d_l)]. \tag{3.9}$$

From [114], the probability that the preferred document d_k is sampled before d_l is:

$$P(d_k > d_l) = \frac{P(d_k|D)}{P(d_k|D) + P(d_l|D)} = \frac{e^{f(d_k)}}{e^{f(d_k)} + e^{f(d_l)}}. \tag{3.10}$$

Thus, the gradient estimation equals:

$$\nabla f_\theta(\cdot) = \sum_{d_k >_c d_l} \rho(d_k, d_l, R, D) \frac{e^{f_\theta(d_k)} e^{f_\theta(d_l)}}{(e^{f_\theta(d_k)} + e^{f_\theta(d_l)})^2} (f_\theta{}'(d_k) - f_\theta{}'(d_l)). \tag{3.11}$$

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

$$\rho(d_k, d_l, R, D) = \frac{P(R^*(d_k, d_l, R)|D)}{P(R|D) + P(R^*(d_k, d_l, R)|D)}. \tag{3.12}$$

Algorithm 1 details the PDGD method.

Algorithm 1 Pairwise Differentiable Gradient Descent(PDGD) [39]

Input: initial weights: θ_1; scoring function: f; learning rate η.
for $t \leftarrow 1 \ldots \infty$ **do**
  $q_t \leftarrow receive\_query(t)$ *// obtain a query from a user*
  $D_t \leftarrow preselect\_documents(q_t)$ *// preselect documents for query*
  $\mathbf{R}_t \leftarrow sample\_list(f_{\theta_t}, D_t)$ *// sample list according to Eq. 3.7,3.8*
  $\mathbf{c}_t \leftarrow receive\_clicks(\mathbf{R}_t)$ *// show result list to the user*
  $\nabla f_{\theta_t} \leftarrow \mathbf{0}$ *// initialize gradient*
  for $d_k >_{\mathbf{c}} d_l \in \mathbf{c}_t$ **do**
    $w \leftarrow \rho(d_k, d_l, R, D)$ *// initialize pair weight (Eq. 3.12)*
    $w \leftarrow w \frac{e^{f_{\theta_t}(\mathbf{d}_k)} e^{f_{\theta_t}(\mathbf{d}_l)}}{(e^{f_{\theta_t}(\mathbf{d}_k)} + e^{f_{\theta_t}(\mathbf{d}_l)})^2}$ *// pair gradient (Eq. 3.11)*
    $\nabla f_{\theta_t} \leftarrow \nabla f_{\theta_t} + w(f'_{\theta_t}(\mathbf{d}_k) - f'_{\theta_t}(\mathbf{d}_l))$ *// model gradient(Eq. 3.11)*
  $\theta_{t+1} \leftarrow \theta_t + \eta \nabla f_{\theta_t}$ *// update the ranking model*

Algorithm 2 FederatedAveraging PDGD.
• set of clients participating training: C, each client is indexed by c;
• number of local interactions for client c: n_c ($\sum_{c=1}^{|C|} n_c = n$)
• local interaction set: B, model weights: θ.

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

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

Federated Averaging for PDGD

The Federated Averaging algorithm [7] is a popular approach for Federated Learning. It has the advantages of being able to handle large-scale data spread across numerous devices and reducing communication overhead by only sharing model updates instead of raw data. Next, we provide a working description of the algorithm.

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

$$\min_{\theta \in \mathbb{R}^d} f(\theta) \qquad \text{where} \qquad f(\theta) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\theta). \tag{3.13}$$

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

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla f(\theta_t). \tag{3.14}$$

To perform the training process in federated manner, we assume that $|C|$ clients hold their own training dataset. For each client c holding the local dataset D_c, the volume of the dataset is $n_c = |D_c|$. The loss function Eq. 3.13 can be re-written as:

$$f(\theta) = \sum_{c=1}^{|C|} \frac{n_c}{n} F_c(\theta) \quad \text{where} \quad F_c(\theta) = \frac{1}{n_c} \sum_{i \in D_c} f_i(\theta). \tag{3.15}$$

Based on Eq. 3.15, $\nabla f(\theta_t) = \sum_{c=1}^{|C|} \frac{n_c}{n} \nabla F_c(\theta)$. On each client c, the average gradient on the local data for the current model is denoted as $g_c = \nabla F_c(\theta_t)$. Then, the local model is updated using $\theta_{t+1}^c \leftarrow \theta_t - \eta g_c$ and the global model is updated by weighted averaging all local models:

$$\theta_{t+1} \leftarrow \sum_{c=1}^{|C|} \frac{n_c}{n} \theta_{t+1}^c. \tag{3.16}$$

To do so, each client locally takes one step of gradient descent on the current model using its local data and the server then uses the weighted average of the local models as the global updated model. This process forms the FederatedSGD (FedSGD) algorithm [7]. Federated Averaging (FedAvg) extends FederatedSGD by adding more updating times to each client: the local update $\theta_{t+1}^c \leftarrow \theta_t - \eta g_c$ is performed within a minibatch portion of the local dataset, before the averaging step.

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

Securing the Gradient: Differential Privacy for FPDGD

A key motivation for adopting a federated learning approach is to protect users' private data: this is achieved by avoiding having to share data across clients or with a server – instead, only gradient updates are shared. However, in the federated learning setting, the model parameters can be attacked to reveal sensitive information about the training data [63, 65].

To provide a strong privacy guarantee for the proposed federated PDGD, we introduce a noise-adding clipping technique [62, 115] which is based on the theory of differential privacy [38, 60].

Definition 3.2.1. *(ε-Differential Privacy): A randomized mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ with a domain $\mathcal{D}$ (e.g., possible training datasets) and range $\mathcal{R}$ (e.g., all possible trained models) satisfies ε-differential privacy when there exists $\varepsilon > 0$ such that:*

$$P[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} P[\mathcal{M}(D_2) \in S] \tag{3.17}$$

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

Algorithm 3 FederatedAveraging PDGD with differential privacy.
• set of clients participating training: C, each client is indexed by c;
• number of local interactions for client c: n_c ($\sum_{c=1}^{|C|} n_c = n$)
• local interaction set: B, model weights: θ.

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

ClientUpdate(c, θ): *// Run on client c*
for each local update i from 1 to B **do**
  $\theta \leftarrow \theta + \eta \nabla f_\theta$ *//PDGD update shown in Algorithm. 1*
$\theta \leftarrow \theta \cdot \min(1, \frac{\Delta}{2 \cdot \|\theta\|})$ *//clip the weights*
return $(\theta + \gamma - \gamma', n_c)$ to server *//γ, γ' gamma noise sampled from Eq. 3.22*

Definition 3.2.2. *(Global Sensitivity): For any real-valued query function $t: \mathcal{D} \to \mathcal{R}$, where $\mathcal{D}$ denotes the set of all possible datasets, the global sensitivity Δ of t is defined as:*

$$\Delta = \max_{D_1, D_2} |t(D_1) - t(D_2)| \tag{3.18}$$

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

Since there is no apriori knowledge about the sensitivity of PDGD, we clip the weights outputed by PDGD to the bound $\frac{\Delta}{2}$, as in Eq. 3.19, so that the parameter updating can meet the global sensitivity Δ. It should be noticed that the sensitivity Δ is used as hyper-parameter and will be determined through grid search for a given privacy level ε.

$$\theta = \theta \cdot \min(1, \frac{\Delta}{2 \cdot \| \theta \|}). \tag{3.19}$$

The Laplacian mechanism preserves ε-differential privacy [60] through leveraging random noise X sampled from a symmetric Laplacian distribution. The probability density function $f(x)$ of a zero-mean Laplacian distribution is:

$$f(x) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \tag{3.20}$$

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

In the federated learning setting, each client contributes a portion of differentially private noise so that the sum of all contributions results in a differentially private noise value. Previous work has used Gamma random variables to generated the Laplace Random variable [115]. Specifically, a Laplace Random variable can be generated from the sum of n Gamma random variables:

$$L(\mu, \lambda) = \frac{\mu}{n} + \sum_{p=1}^{n} \gamma_p - \gamma_p' \tag{3.21}$$

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

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

$$f(x) = \frac{(1/s)^{1/n}}{\Gamma(1/n)} x^{1/n-1} e^{-x/s} \tag{3.22}$$

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

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

3.2.2 Experimental Setup

We rely on the typical evaluation setup used by previous OLTR research [39, 49, 59] to empirically investigate the effectiveness of the proposed FPDGD approach, and how it compares to other methods. This consists of use standard learning to rank datasets, simulate user interactions with SERPs, i.e. simulate clicks, and measure both online and offline performance.

Datasets

We use the MQ2007 [40] and MSLR-WEB10K [40] datasets. While other, larger datasets for learning to rank are currently available, our choice considered the trade-off between representativeness and the high computational costs associated with experimenting with the federated OLTR methods on larger datasets. MQ2007 is relatively small and has fewer assessed documents per query, and was chosen to allow direct comparison with previous work on federated OLTR [27], which mainly used MQ2007, and because this dataset, being small, allows for computationally treatable experiments. The MSLR-WEB10K is larger and more recent than MQ2007, and it was chosen as results in Section 3.1.3 suggested findings for federated OLTR observed on MQ2007 may not generalise on MSLR-WEB10K [1].

User simulations

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

For the querying behaviour, for each client participating in the federated OLTR, we sample B queries randomly, in line with previous work on FOLTR [1, 27]. For each query, we use the local ranking model (i.e. that held by the client) to rank documents; we limit SERP to 10 documents.

For the click behaviour, we rely on the Cascade Click Model (CCM) as described in Section 2.2.1.

Federated setup

We simulate the federated OLTR scenario as follows. Each client holds a copy of the current ranker. For each client, we consider $|B|$ user queries along with the respective interactions, during which the local ranker is optimised using the simulated user clicks. After $|B|$ interactions have occurred, the

client sends the local message (updated weights) to the central server. The central server optimises the global ranker by aggregating the local messages and sends the newly-updated ranker back to each client.

In our experiments for MQ2007, unless otherwise specified, we simulate $|C| = 1,000$ clients with each client performing $|B| = 4$ interactions (queries) locally to contribute to each global model update. We restrict the total interaction budget to 4 million queries, which results in 1,000 global updates. For MSLR-WEB10K dataset, unless otherwise specified, we also simulate $|C| = 1,000$ clients but with only $|B| = 2$ local interactions and we set the total interaction budget to 400,000 queries; thus resulting in 200 global updates.

Baselines

To investigate the impact of the federated averaging process on PDGD, we compare FPDGD against the original, centralised, PDGD model [39]. We follow the original paper and set the learning rate $\eta = 0.1$. For both methods, we train a linear model as the ranker. In federated OLTR, global rankers are updated in batches, that is, the central server updates rankers after each client has executed B searches and the local ranker updates have been sent to the server. However, in centralised OLTR settings, rankers are updated after each user interaction (batch size = 1). For fair comparison, we adapt PDGD to also be updated in batch by accumulating gradients updates.

To date, FOLtR-ES is the only federated OLTR algorithm proposed in the literature [27], and it constitutes a natural baseline for comparison with FPDGD, thus allowing to compare different federated OLTR methods. For FOLtR-ES, we train a linear model to make fair comparison with our method trained on linear ranker. We set the learning rate $\eta = 0.001$ and choose the reciprocal rank of the highest clicked result (MaxRR) in an interaction as the optimization metric in FOLtR-ES.

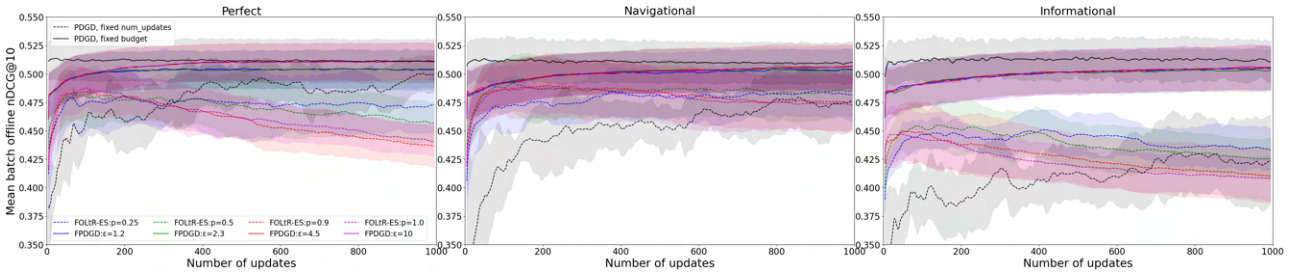
Evaluation measures

To compare the ranking performance, we rely on the evaluation practice from previous OLTR work, which consists of measuring the ranker's offline and online performance using *nDCG@k* as discussed in Section 2.2.1. We consider $k = 10$ as the number of documents in a ranked list for a query. For offline evaluation, we record the offline *nDCG*@10 score of the global ranker during each federated training update. For online evaluation, we take the online *nDCG*@10 score averaged across all clients.

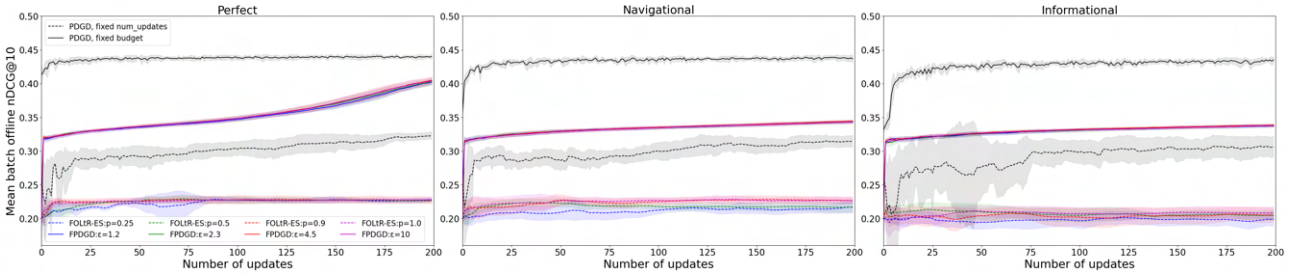
Each experiment is repeated 25 times, spread evenly over all dataset training folds. All evaluation results are averaged and statistical significant differences between system pairs are evaluated using two-tailed Student's t-test with Bonferroni correction.

Choosing privacy parameters

The differential privacy process in FPDGD is controlled by the parameter ε as described in Section 3.2.1. To make comparison with the baseline FOLtR-ES method fair, we utilize the FOLtR-ES's privacy parameter p to fix an upper bound on the value of ε. In FOLtR-ES, the higher the p value, the lower



(a) MQ2007



(b) MSLR-WEB10K

Figure 3.6: Offline performance (nDCG@10) across datasets, under different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

the privacy. In our experiments, we set $p \in \{0.25, 0.5, 0.9, 1.0\}$; these settings correspond to upper bound values of $\varepsilon \in \{1.2, 2.3, 4.5, 10\}$. For a given privacy level ε, we search the best setting of sensitivity Δ through grid search in $\{1, 3, 5, 7, 9\}$. Then, for each $\varepsilon \in \{1.2, 2.3, 4.5, 10\}$, we choose the corresponding $\Delta \in \{3, 3, 5, 5\}$.

3.2.3 Results

Overall ranking performance

Figure 3.6 displays the offline performance (nDCG@10) of the proposed FPDGD method across different settings of differential privacy ε, along with the effectiveness of the non-federated PDGD method and the baseline federated FOLtR-ES method (across different settings of differential privacy p). The online performance of these methods are reported in Table 3.1.

Impact of federation on PDGD. We start the analysis of the results by commenting on the impact of federation on the PDGD method, focusing on the offline performance (Figure 3.6). When PDGD is given the same number of updates than its federated counterpart, its performance are lower. This is not surprising: the non-federated PDGD only uses the click data from one search interaction (query) for each update, and thus it is updated only 200 times in MSLR-WEB10K (1,000 in MQ2007). While FPDGD is also updated 200 times (1,000), when an update is performed this considers 2 interactions per client and 1,000 clients – thus the total interaction budget assigned to FPDGD in MSLR-WEB10K is $200 \times 2 \times 1,000 = 400,000$ (4 million in MQ2007) compared to that of 200 for PDGD (1,000 for MQ2007). However if PDGD is given the same interaction budget as its federated counterpart, then PDGD does perform better than FPDGD. This difference is due to PDGD being able to perform a ranker update at each interaction (query), while FPDGD needs to wait the completion of batch size $\times$

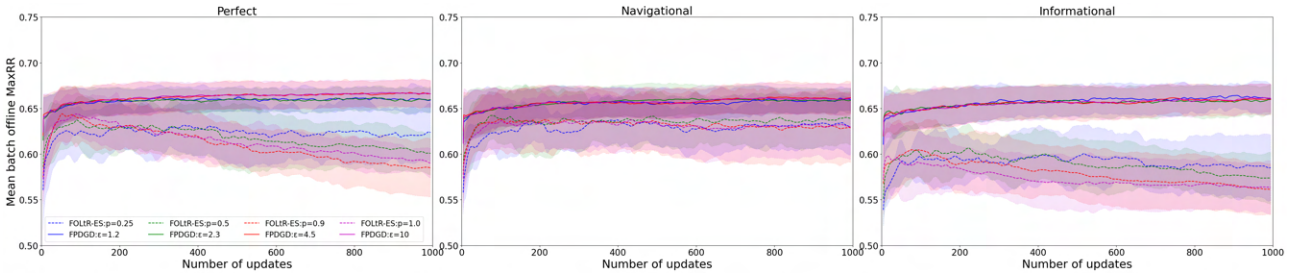
Table 3.1: Online performance (discounted cumulative nDCG@10) for each dataset under different instantiations of CCM. Significant gains and losses of FPDGD over FOLtR-ES (baselines) are indicated by $^{\triangle}$, $^{\triangledown}$ ($p < 0.05$) and $^{\blacktriangle}$, $^{\blacktriangledown}$ ($p < 0.01$), respectively.

		$\varepsilon = 1.2$	$\varepsilon = 2.3$	$\varepsilon = 4.5$	$\varepsilon = 10$
			MQ2007		
perfect	FPDGD	296.03 $^{\blacktriangledown}$	296.09 $^{\blacktriangledown}$	313.28	313.26
	FOLtR-ES	316.66	317.93	313.80	312.29
navigational	FPDGD	293.29 $^{\blacktriangledown}$	293.42 $^{\blacktriangledown}$	303.80 $^{\blacktriangledown}$	303.82 $^{\blacktriangledown}$
	FOLtR-ES	319.90	325.04	324.33	323.05
informational	FPDGD	291.84	292.02	301.45 $^{\blacktriangle}$	301.30 $^{\blacktriangle}$
	FOLtR-ES	292.58	294.78	288.57	285.96
			MSLR-WEB10K		
perfect	FPDGD	54.62 $^{\blacktriangle}$	54.61 $^{\blacktriangle}$	54.64 $^{\blacktriangle}$	54.61 $^{\blacktriangle}$
	FOLtR-ES	39.35	40.50	40.87	41.14
navigational	FPDGD	52.33 $^{\blacktriangle}$	52.30 $^{\blacktriangle}$	52.30 $^{\blacktriangle}$	52.29 $^{\blacktriangle}$
	FOLtR-ES	38.55	39.59	40.32	40.47
informational	FPDGD	51.11 $^{\blacktriangle}$	51.16 $^{\blacktriangle}$	51.14 $^{\blacktriangle}$	51.18 $^{\blacktriangle}$
	FOLtR-ES	37.26	37.18	37.21	37.53

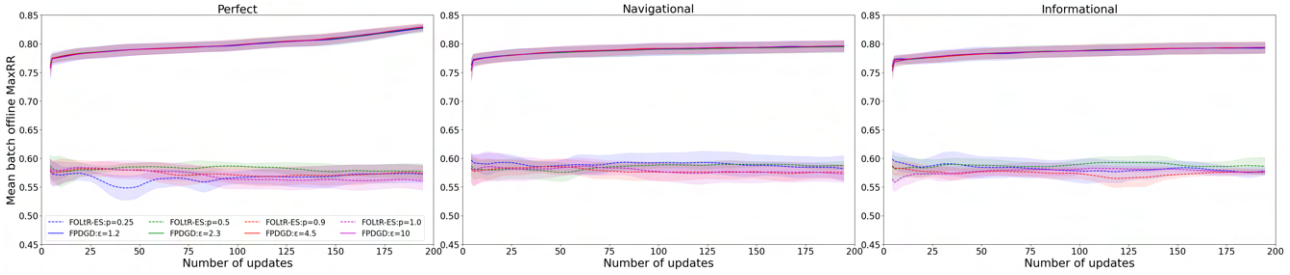
clients $= 2 * 1,000 = 2,000$ interactions in MSLR-WEB10K for doing an update ($4,000$ in MQ2007).

Impact of privacy budget. Next, we analyse the impact of adding differential privacy noise to FPDGD, where lower values of ε correspond to higher privacy guarantees. In theory, with higher privacy guarantee, the performance is worse due to more random noise is added. We find that the privacy budget we select ($\varepsilon \in \{1.2, 2.3, 4.5, 10\}$) has little impact on the performance of FPDGD (both online and offline) for MSLR-WEB10K. For MQ2007, lower differential privacy values tend to provide lower performance, though not significantly worse.

Comparison with FOLtR-ES. When comparing the proposed federated approach, FPDGD, with the currently only other federated OLTR method, FOLtR-ES, we note that FPDGD consistently outperforms FOLtR-ES on offline performance across datasets, click models, and differential privacy settings (Figure 3.6). Gains over FOLtR-ES are specifically notable for all MSLR-WEB10K experiments and for the informational click model for MQ2007. The fact that the performance of FOLtR-ES are particularly poor in MSLR-WEB10K finds confirmation in previous section, which shows FOLtR-ES has poor performance on large datasets 3.1.3. In addition, in MQ2007, FOLtR-ES exhibits a surprisingly decreasing trend as global model updates increase, especially for higher values of differential privacy p ($p = 0.9, 1.0$), when the perfect and informational click models are considered. Similar observations can be made when considering online performance on MSLR-WEB10K dataset (Table 3.1): FPDGD is significantly better than FOLtR-ES under all three click models within different privacy budgets. However, on MQ2007, our method is not consistently better than FOLtR-ES. When the navigational click model is considered, FPDGD is significantly outperformed by FOLtR-ES. We believe the results are caused by the random noise from the sampling strategy of PDGD that occurs in the early training interactions: results on small-scale datasets like MQ2007 tend to be easily affected by this issue.



(a) MQ2007



(b) MSLR-WEB10K

Figure 3.7: Offline performance in terms of MaxRR across datasets, under three different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

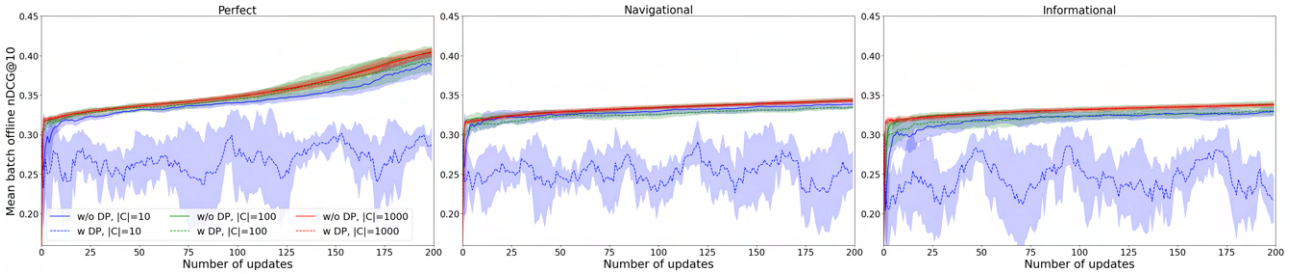


Figure 3.8: Offline performance on MSLR-WEB10K under three different click models, averaged across all dataset splits and experimental runs, for different number of clients $|C|$. We study FPDGD with differential privacy (*w DP*) and without (*w/o DP*).

Evaluation using MaxRR. In our experiments, we used nDCG@10 as evaluation measure and optimisation criteria for the OLTR process, in line with previous work on OLTR. FOLtR-ES however was previously evaluated on MaxRR [27], which is used internally by FOLtR-ES as optimisation criteria. For completeness, in Figure 3.7 we compare FPDGD and FOLtR-ES when MaxRR is used for evaluation. We find that the trends observed when evaluating with nDCG@10 (Figure 3.6) are also found when MaxRR is used.

Influence of number of clients

To study the influence of the number of clients participating in our federated setup, we vary the number of clients in $|C| \in \{10, 100, 1000\}$. We report the offline nDCG@10 results obtained on MSLR-WEB10K in Figure 3.8. We consider two versions of FPDGD: one that uses the introduced differential privacy mechanism with $\Delta = 5$ and $\varepsilon = 4.5$ (labelled *w DP* in Figure 3.8), and one for which differential privacy is turned off (*w/o DP*).

When FPDGD does not rely on differential privacy, the more clients participate to the federated

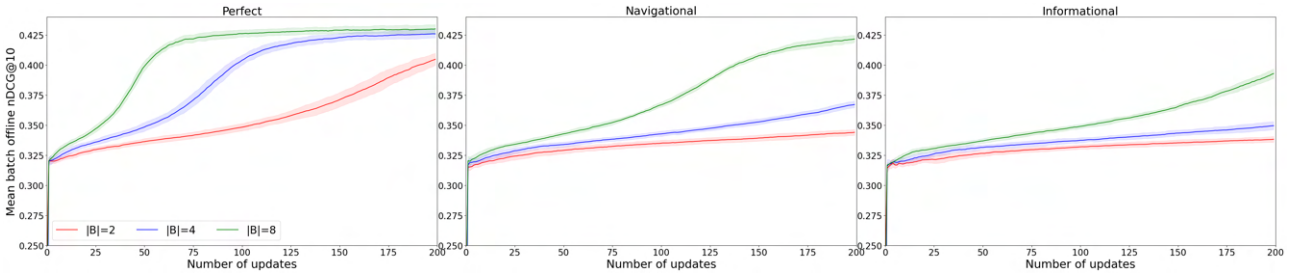


Figure 3.9: Investigation of the influence of batch size $|B|$ on offline performance for MSLR-WEB10K under three different click models, averaged across all dataset splits and experimental runs. Shaded areas indicate the standard deviation.

system the better, although differences are minor and not statistically significant. However, if differential privacy is used, when only a small number of clients participates to the federation, then FPDGD performs poorly. We believe that this is so because the differential privacy process adds noise to the local gradient updates; this noise hinders the convergence of the global model when relying on very little interaction data for updating (i.e., 10 clients $\times$ 2 batch size $=$ 20 queries). In this case, the noise introduced by the differential privacy process has a large impact. Conversely, differential privacy has little to no impact on performance if numerous clients participate ($|C| = 100, 1000$). Across all settings, we note that past a certain point, adding more clients to the system has little impact – this finding is confirmed by prior work in general federated learning, which showed diminishing returns (because of the communication costs) for adding more clients beyond a certain point [7].

Influence of batch size

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

The offline nDCG@10 performance of FPDGD across different B settings are shown in Figure 3.9 for MSLR-WEB10K. The results suggest that the larger the batch size, the better the offline performance. This trend is especially significant on the perfect click model, although models trained on different batch sizes tend to ultimately converge after enough updates have been made. For the informational click models these differences are less remarked (though still significant), and convergence is long delayed. These results should not be surprising: a larger batch size means that more interactions (queries) have been used during a local update, e.g., 200 interactions $\times$ 2 batch size $=$ 400 queries per client vs. 200 interactions $\times$ 8 batch size $= 1,600$ queries per client.

However, when the budget is maintained equal across different batch sizes, and thus smaller batch size means more frequent updates, then the performance obtained across different values of $|B|$ is quite similar (and no significant differences found), although a small batch size guarantees earlier updates, thus resulting in stronger initial performance. This trend is shown in Figure 3.10 for the offline performance on MSLR-WEB10K. The stronger initial performance, which is due to the initial

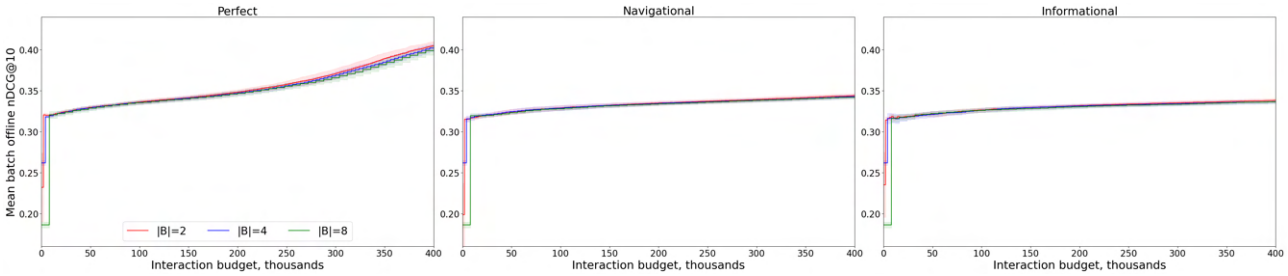


Figure 3.10: Investigation of the influence of batch size $|B|$ on offline performance for MSLR-WEB10K under three different click models, averaged across all dataset splits and experimental runs, under *fixed budget*.

model update occurring earlier when the batch size is small, does result in a slightly higher ranking performance when a small batch size is used.

3.2.4 Summary

In this section, we considered the problem of effective online learning to rank within a privacy-preserving environment. To this aim, we cast the Pairwise Differentiable Gradient Descent method (PDGD) [39], which represents the current state-of-the-art in OLTR, to the Federated Learning framework by using the Federated Averaging algorithm, thus proposing the federated Pairwise Differentiable Gradient Descent method (FPDGD). To further enhance the privacy of the method, we overlaid an ε-differential privacy method to the proposed FPDGD, thus enforcing a level of privacy guarantee on the local gradient updates.

Empirical evaluation shows FPDGD significantly outperforms the only other federated OLTR method (i.e., FOLtR-ES). This resonates with previous section (Section 3.1.3) that showed the performance of FOLtR-ES to be highly variable across datasets and settings, making the method unstable and of risky use in practice. FPDGD instead represents a reliable, stable, and secured alternative to non-federated state-of-the-art OLTR methods.

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

The following publication has been incorporated as Chapter 4.

1. [3] **Shuyi Wang**, and Guido Zuccon, Is Non-IID Data a Threat in Federated Online Learning to Rank?, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2801–2813, 2022

Chapter 4

Federated Online Learning to Rank with non-IID Data

In the previous chapter, we address on the effectiveness of FOLTR paradigm and the factors that affect its performance, with an overview on the proposed and empirically investigated methods [1, 2, 27]. In this chapter, we address on an important issue that affects the performance of federated learning systems but has been fully ignored by research on FOLTR: the presence of bias in how the training data is divided across the clients that join the federation. In other words, the fact that clients may hold non-independent and identically distributed (non-IID) data [6].

Non-IID data can pose severe threat to the effectiveness of a federated learning method. Models trained federatively in the presence of non-IID data across the clients that participate in the federation, in fact, display significantly lower effectiveness, and at times experience difficulties for the model to converge [6]. Effectiveness degradation can mainly be attributed to the weight divergence between the local models resulting from the non-IID distribution of the data across the clients [6]. Local models with the same initial parameters will converge to different models because of the heterogeneity of the local data distributions. This divergence will increase as more communication rounds of the federated learning algorithm are performed. This slows down or even impedes model convergence, worsening the performance of the global model. An illustration of the phenomenon of model divergence for both IID and non-IID data in federated learning is given in Figure 4.1. The ideal global model (θ_t, under centralised learning) and actual global model (θ_t^{avg}, average model created through FedAvg [7]) coincide when data is IID, but diverge when data is non-IID, showing that this is a sizeable problem when the data is non-IID.

This chapter provides a systematic understanding of when non-IID data may occur in the FOLTR setting and the impact of non-IID data in such cases. If not specified otherwise, we only consider horizontal FL [28] and we believe our framework can be applied to both cross-device and cross-silo federated learning [116]. Specifically, we investigate the following research questions:

RQ2: Are FOLTR methods robust to non-IID data among clients?

RQ2.1: What are the potential types of non-IID data in FOLTR and their impact on model performance?

RQ2.2: Do existing methods that deal with non-IID data in FL of general domain generalise to FOLTR?

To answer RQ2.1, we first enumerate possible data distribution settings that may showcase data bias across clients and thus give rise to the non-IID problem. Then, we study the impact of each setting on the performance of the state-of-the-art FOLTR approach, the Federated Pairwise Differentiable Gradient Descent (FPDGD, introduced in Section 3.2), and we highlight which data distributions may pose a problem for FOLTR methods. For RQ2.2, we explore how common approaches proposed in the federated learning literature to deal with non-IID data can be cast in the FOLTR framework and their effectiveness in addressing non-IID data in FOLTR.

4.1 FOLTR Framework and FPDGD

Our study is based on the Federated Pairwise Differentiable Gradient Descent (FPDGD, introduced in Section 3.2) method, which represents the current state-of-the-art in FOLTR and that we use as a representative method in our experiments to investigate the effect of non-IID data on FOLTR. We did not include the *securing gradient* component in FPDGD (introduced in Section 3.2.1) so as not to let the differential privacy noise affect the results of non-IID analysis. We next briefly describe the FOLTR framework used in this chapter.

The federated online learning to rank setting is pictured in Figure 4.2. Searchable data is stored by each client (❶) and not shared with the centralised server or other clients. Different clients may hold

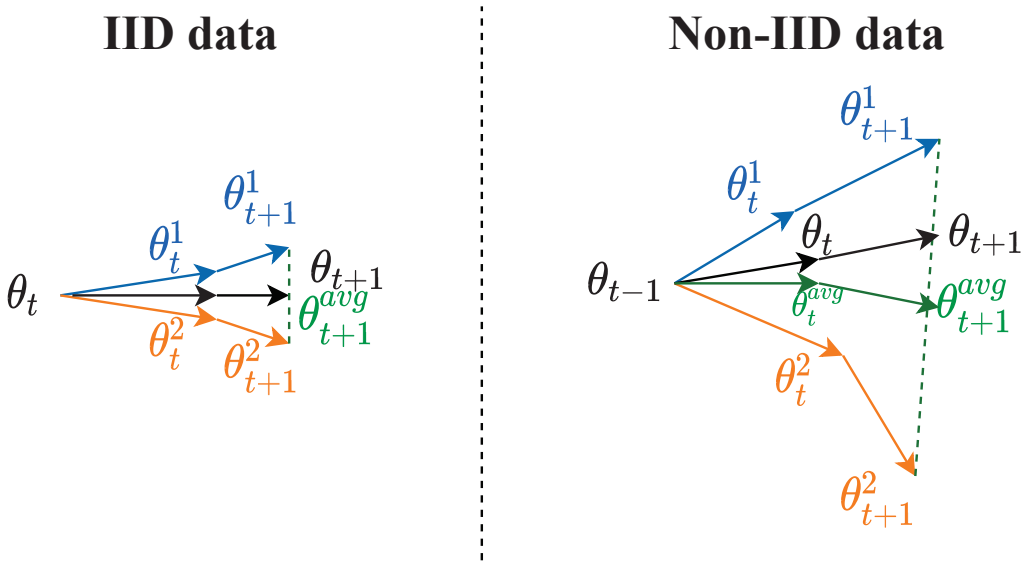


Figure 4.1: Illustration of the model divergence problem in FL, adapted from Zhu et al. [6]. θ_t is the ideal global model under centralised learning, and θ_t^{avg} is the average model created from the local models of Client 1 (θ_t^1) and Client 2 (θ_t^2) through FedAvg [7].

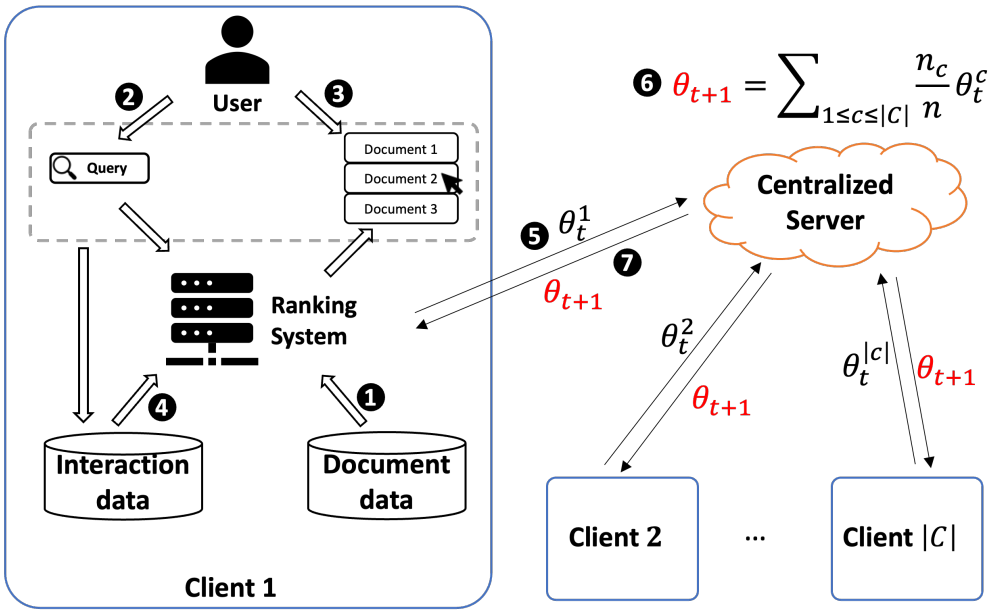


Figure 4.2: Schematic representation of the FOLTR setting.

Algorithm 4 FederatedAveraging PDGD.
- set of clients participating training: C, each client is indexed by c;
- number of local interactions for client c: n_c ($\sum_{c=1}^{|C|} n_c = n$)
- local interaction set: B, model weights: θ.

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

ClientUpdate(c, θ): *// Run on client c*
for each local update i from 1 to B **do**
  $\theta \leftarrow \theta + \eta \nabla f_\theta$ *//PDGD update shown in Algorithm. 5*
return (θ, n_c) to server

all, a portion of, none of the same searchable data. Queries and user's clicks occur at a client side (❷ and ❸) and are not communicated to the centralised server or other clients: search is indeed entirely performed on the user device (❷). Each client exploits search interactions to perform local model updates to the ranker; for FPDGD, the routine executed by the client is shown in Algorithm 4, and the PDGD update is shown in Algorithm 5. Each client considers B interactions before updating the local ranker using the PDGD gradients. These local updates are then shared with the central server (❺), which in turn combines the ranker updates from the clients to produce a revised ranker (❻); for FPDGD, this is achieved according to the server routine in Algorithm 4. The new global model is then distributed to the user's device (❼).

Algorithm 5 Pairwise Differentiable Gradient Descent(PDGD) [39]

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

4.2 Types of non-IID Data in FOLTR

We consider training a ranker for the OLTR system as a supervised learning task in an FL setup, with each client holding a subset of the data. Each data sample is denoted as (x, y), where x is the feature representation of the data and y is the label. The local distribution of the dataset in client i is denoted as $P_i(x, y)$. The presence of non-IID data can be represented as the difference between local data distributions: that is, for different clients i and j, $P_i(x, y) \neq P_j(x, y)$.

In federated learning, data across clients may not be IID due to different reasons: Kairouz et al. [116] and Zhu et al. [6] assert this can be due to how features x and labels y are distributed. However, the translation of these categories to FOLTR is not straightforward. In the following sections, we put forward several situations in which data specific to FOLTR could be distributed in a non-IID manner across clients. Specifically, we consider data in the FOLTR process may not be IID because of biases across clients due to:

- **Type 1:** document preferences skewness (Section 4.3)
- **Type 2:** document label distribution skewness (Section 4.4)
- **Type 3:** click preferences skewness (Section 4.5)
- **Type 4:** data quantity skewness (Section 4.5)

The last data type, Type 4, i.e., the situation in which different clients hold different quantities of data (and in particular interaction data such as queries and clicks), does not necessarily imply that the data is non-IID. However, we note this case is often studied in the FL literature alongside non-IID data [6, 32], and thus we include this situation in our considerations of the non-IID problem. Each data type is defined and investigated in the next sections; in addition we provide a summary overview of the data types in Table 4.1.

We also note that commonly in federated learning, non-IID data occurs because the data is distributed across clients according to its features. In other words, the marginal distribution of

Table 4.1: Summary of non-IID data types in FOLTR.

Data type	Key characteristic	When it happens in FOLTR
Type 1	Document Preferences	Different clients have different preferred candidate documents, although they are searching for the same query.
Type 2	Document Label Distribution	Different clients hold candidate documents with different label distribution while the conditional feature distribution is the shared.
Type 3	Click Preferences	Different clients have various preferred click behaviours when searching for the same query.
Type 4	Data Quantity	Different clients have different frequency on issuing queries and interacting with the searching system.

the features belonging to the data held by each client may vary, i.e. for different clients i and j, $P_i(x) \neq P_j(x)$. This situation may occur in horizontal federated learning settings (also called homogeneous FL) [28], where each client holds different and overlapping datasets. In this case, the non-IID divergence is usually caused by inconsistent data distributions, e.g., feature imbalance of the training data local to each client. However, this case does not seem applicable to FOLTR (thus is not further studied in this thesis). In FOLTR, each data item is represented by the feature vector of a query-document pair and its relevance label. The features often consist of variations of query-dependent features such as TF-IDF scores, BM25 scores, query length, as well as query-independent features such as PageRank, URL lengths, and so on [40]. In this case, bias in the feature distribution across clients would be rare as most features are dependent on the query-document pair.

Next, we describe the non-IID data types we put forward in this chapter and analyse their impact on FOLTR. We empirically find that only Type 1 and partially also Type 2 data have a strong impact on the FOLTR. We thus predominantly focus our attention on these two data types while providing only a definition and a brief account of the remaining two data types in the thesis. For Type 1 and Type 2, we introduce the simulation of non-IID data, further analysis on impact of non-IID data, and implementation of methods on dealing with non-IID data. For Type 3 and Type 4, we only focus on the simulation and analysis on its impact.

Code, experiment scripts and experimental results are provided at `https://github.com/ielab/2022-SIGIR-noniid-foltr`.

4.3 Type 1: Document Preferences

Document preference skewness (Type 1) considers the situation when the conditional distribution $P_i(y|x)$ varies across the clients though $P_i(x)$ remains the same. This happens when different clients have different preferred candidate documents, although they are searching for the same query. As OLTR requires the user's implicit feedback as an optimization objective, which might be highly related to individual preferences, this setting appears to be of very likely occurrence.

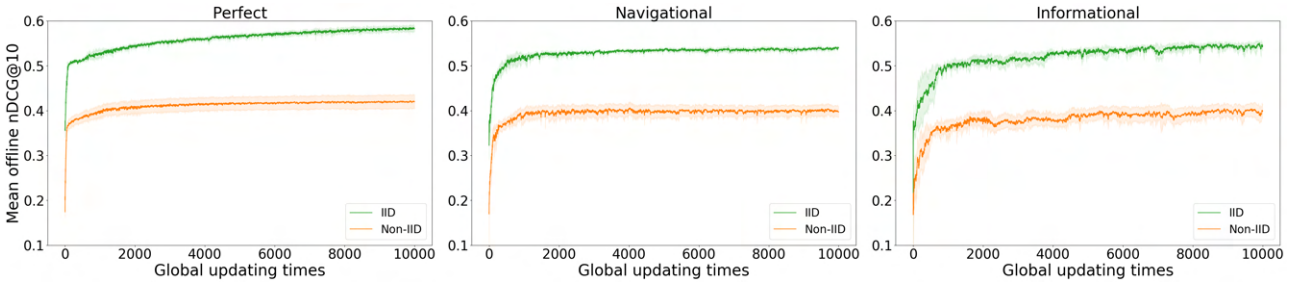
4.3.1 Simulating Type 1 non-IID Data

The mechanism we use to simulate non-IID data of Type 1 and IID data to baseline the FOLTR effectiveness relies on a recent work that empirically studied and demonstrated how OLTR methods adapt when user's search intents change overtime [16]. In particular, Zhuang and Zuccon [16] created a collection for OLTR with several explicit intent types by adapting an existing TREC collection, as no dataset is available for studying this OLTR problem. Derived from ClueWeb09 and the TREC Web Track 2009 to 2012 [117], this intent change collection consists of 200 queries with 4 intents each and, on average, 512 candidate documents per query. Furthermore, query-document pairs' relevance judgements are provided per intent. We believe this is an appropriate collection to adapt to study the effect of Type 1 non-IID data on FOLTR. We can regard each intent as a type of user preference. As the average number of relevant documents per intent varies largely across all intent types, the learning difficulty of optimizing a ranker among different intents also varies. To avoid this bias, we follow Zhuang and Zuccon [16] and we re-label the original intent number for each query through random shuffling: this is possible because all intent types are independent across queries. In our experiments, we repeat this process of re-balancing 5 times, thus giving rise to results averaged across 5 FOLTR experiments. We refer to Zhuang and Zuccon [16] for further details on the dataset creation, and we further highlight that we have made available an implementation of the dataset creation procedure along with the actual dataset at `https://github.com/ielab/2022-SIGIR-noniid-foltr`.

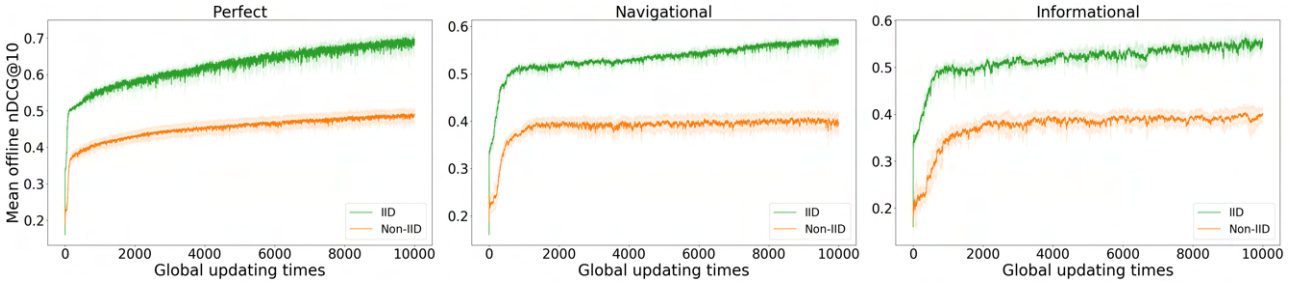
To simulate non-IID data, after randomly shuffling all intents across 4 types, we let each intent represent one client preference. The client preferences differ from each other for the same query-document pair so as the corresponding relevance judgements. The federated setup involves 4 clients (represented by 4 types of intent) and the local updating time $B = 5$ with fixed global communication times $T = 10,000$. These settings are similar to those used in previous work on FOLTR [1, 2, 27][1] For the implicit feedback in FOLTR, same as Chapter 3, we simulate user clicks based on the popular *Cascade Click Model* (CCM). We limit SERP to 10 documents and use *nDCG*@10 for offline evaluation, cumulative discounted *nDCG*@10 [39] for online evaluation, as discussed in Section 2.2.1. We train a linear ranker and a neural ranker on the intent-change dataset. For the linear model, we set the learning rate $\eta = 0.1$ and zero initialization was used. As in the original PDGD studies [39], the neural ranker is optimized using a single hidden-layer neural network with 64 hidden nodes, along with $\eta = 0.1$.

As in Zhuang and Zuccon [16], given that no held-out test set is available, we evaluate both online and offline performance on the original training set across all 4 intent types and average all results. For the IID setting, we merge all intents and mark a document as relevant as long as it is judged relevant for at least one of the intent types. Each client randomly picks a query from the training set and clicks documents based on the same preferences during the federated training with IID data. Other settings remain the same as the non-IID experiments.

[1] In Section 3.1, we have a detailed analysis on the relationships between number of clients, number of local updates B, and FOLTR effectiveness.



(a) intent-change (linear ranker)



(b) intent-change (neural ranker)

Figure 4.3: Offline performance (nDCG@10) on Type 1 data; results averaged across dataset splits and experimental runs.

Table 4.2: Online performance (discounted cumulative nDCG@10) on Type 1 data, averaged across dataset splits and experimental runs. Significant differences between IID and non-IID are indicated by ▲ ($p < 0.05$).

ranker	data types	*perfect*	*navigational*	*informational*
linear	IID	1002.36 ▲	872.12 ▲	894.95 ▲
	non-IID	648.71	546.25	566.23
neural	IID	1061.57 ▲	834.08 ▲	842.87 ▲
	non-IID	668.38	505.64	490.29

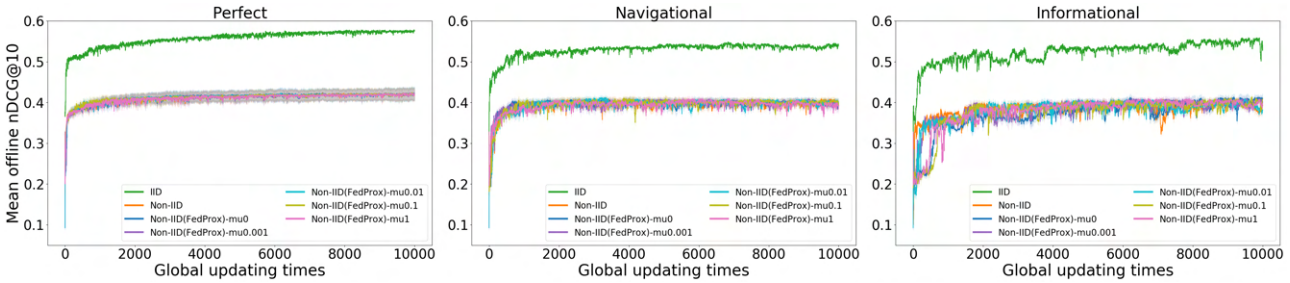
4.3.2 Impact of Type 1 non-IID Data

The offline performance related to Type 1 data is shown in Figure 4.3; the corresponding online performance is shown in Table 4.2. From the offline performance, it is clear that the presence of non-IID data negatively impacts the performance of the learnt ranker, compared to those obtained when data is IID. In terms of online performance, rankers obtained in the presence of non-IID data are also worse than when trained with IID data. This can be explained as follows. Since each client has its preference (intent), the relevant documents are judged in different ways; this leads to the divergence of each client's local ranker update, as exemplified in Figure 4.1.

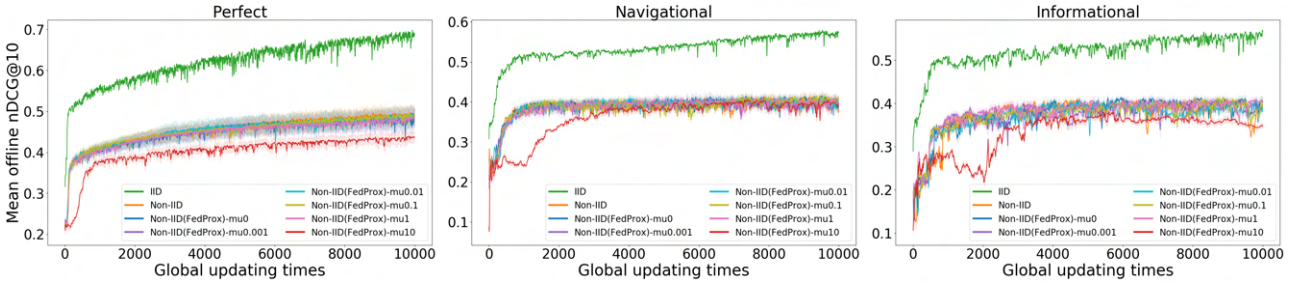
In summary, we find that if data is distributed in a non-IID manner across clients according to Type 1, the effectiveness of FOLTR (and specifically of FPDGD) is seriously affected.

4.3.3 Dealing with Type 1 non-IID Data

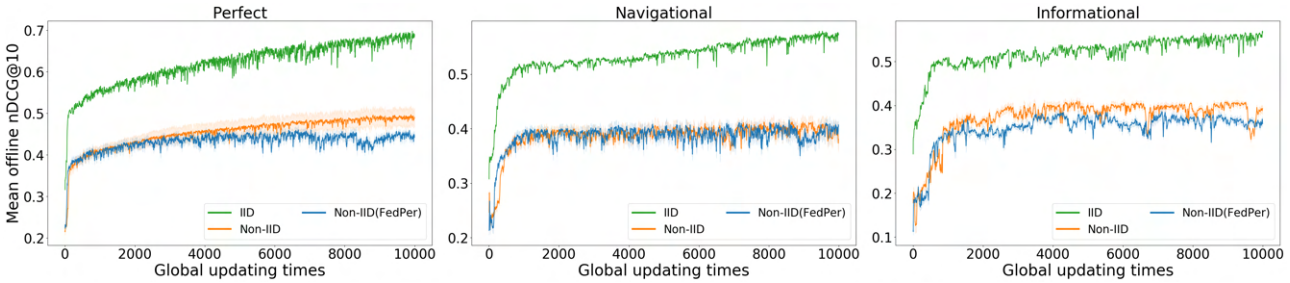
The employed state-of-the-art FPDGD method is based on the FedAvg algorithm. The fact that FPDGD is affected by non-IID data may be due to the underlying federation algorithm, i.e. FedAvg itself. In



(a) intent-change (linear ranker) - FedProx



(b) intent-change (neural ranker) - FedProx



(c) intent-change (neural ranker) - FedPer

Figure 4.4: Offline performance on Type 1 data for FedProx and FedPer; results averaged across dataset splits and experimental runs.

federated learning literature, variations of this federation algorithm have been proposed to tackle the non-IID data problem directly. We select two of such methods, FedProx [118] and FedPer [119], and adapt them to the FPDGD method.

FedProx [118] improves the local objective of FedAvg. Specifically, it introduces an additional L_2 regularisation term (weighted according to a hyper-parameter μ) in the local objective function to limit the distance between the local model and the global model. Thus the use of FedProx adds little computational overhead. However, the main drawback is that the hyper-parameter μ needs to be carefully tuned: a large μ may slow the convergence by forcing the updates to get close to the initial point, while a small μ may not make much difference compared to the use of FedAvg.

FedPer [119] tackles the presence of non-IID exclusively for deep neural networks by separating them into base layers and personalisation layers. The base layers are trained collaboratively through FedAvg, where all clients share the same base layers. Instead, the personalisation layers are trained locally using the clients' local data with stochastic gradient descent (SGD). This procedure works as follows: after initialisation, each client merges and updates its base and personalised layers locally using an SGD style algorithm. Each client only sends its base layers to the global server. The server updates the globally-shared base layers using FedAvg and sends back again the updated ones to each

client. Intuitively, the base layers are updated globally to learn common high-level representations. In contrast, the distinct personalisation layers never leave the local device and capture the personalisation aspects required by the clients. Except for the training and the maintenance of the local personalisation layers, FedPer is quite similar to FedAvg. FedPer, however, reduces the communication costs as only part of the whole model is transferred and has shown enhanced learning performance under highly skewed non-IID data [119].

Our experimental results on FedProx and FedPer are shown in Figure 4.4; for FedProx we explored $\mu \in \{0.001, 0.01, 0.1, 1, 10\}$. The results clearly show that these federated learning methods, which successfully deal with non-IID data in general machine learning tasks, are not effective in the FOLTR context. In fact, not only do these methods not overcome the gap in effectiveness between IID and non-IID setups, but they even only provide limited improvements, if any, compared to FPDGD with FedAvg. This is an important finding because: (1) it shows a realistic case in which non-IID data largely affects FOLTR effectiveness, and (2) it shows that current methods developed in general FL for non-IID data do not work in FOLTR. Thus, a strong need for new methods specialised in the FOLTR settings emerges from these findings.

4.4 TYPE 2: Document Label Distribution

Document label distribution skewness (Type 2) is a widely recognised type of non-IID data type in federated learning. In this setting, the label distributions $P_i(y)$ in each client are different while the conditional feature distribution $P_i(x|y)$ is shared across the clients. In terms of FOLTR, this is equivalent to the following situation. Assume a document is evaluated across the r-level relevance grades, from *not relevant* (0) to *perfectly relevant* $(r-1)$; then the label distribution on each client is such that, for client i, the probability of holding documents with relevance label k is $P_i(R=k) = p_k$, where $\sum_{k=0}^{r-1} p_k = 1, \forall k, p_k \in [0,1]$. For different clients i and j, $P_i(R=k) \neq P_j(R=k)$.

In practice, this may be represented by a situation like the following. Several hospitals are collaboratively creating a FOLTR ranker for clinical-decision-support [120, 121]. Certain hospitals hold a significantly larger portion of highly relevant health records for a certain disease, while some only a small fraction. In this circumstance, the document label distribution is skewed. Under the context of email search [122], different clients might have unique strategies for managing personal emails [123]. Some clients frequently clean up their inboxes and use folders to organise emails. In contrast, some hardly use folders or delete irrelevant messages, resulting in different label distribution when following a learning-to-rank approach.

4.4.1 Simulating Type 2 non-IID Data

In this section, we discuss how we synthetically simulate Type 2 non-IID data and IID data to baseline in the FOLTR effectiveness on the MSLR-WEB10K dataset [40]. We simulate $|C|$ clients with each client performing B interactions (queries) locally to contribute to each global model update and restrict

the global communication times $T = 10,000$. For simulating querying behaviour, for each client participating in the federated OLTR, we sample B queries randomly, in line with Chapter 3. For each query, we use the local ranking model (i.e. that held by the client) to rank documents; we limit SERP to 10 documents. For the click behaviour, we rely on the same CCM click models as Section 4.3. We train both a linear ranker and a neural ranker with same configuration as Section 4.3, following previous OLTR work [39].

We specifically consider two types of non-IID data for Type 2: non-IID subtype 1 and non-IID subtype 2. The main difference between the two types is the number of different labels (i.e. the graded relevance assessments) in each client's local dataset. Following similar partitioning strategies by Li et al. [32], suppose each client only has data samples for k different labels. We first generate all possible k-combinations of the relevance set R and randomly assign to $\binom{R}{k}$ clients. Then, for the query-document pairs of each label, we randomly and equally divide them into the clients who own the label. In this way, the number of labels in each client is fixed, and there is no overlap between the samples of different clients.

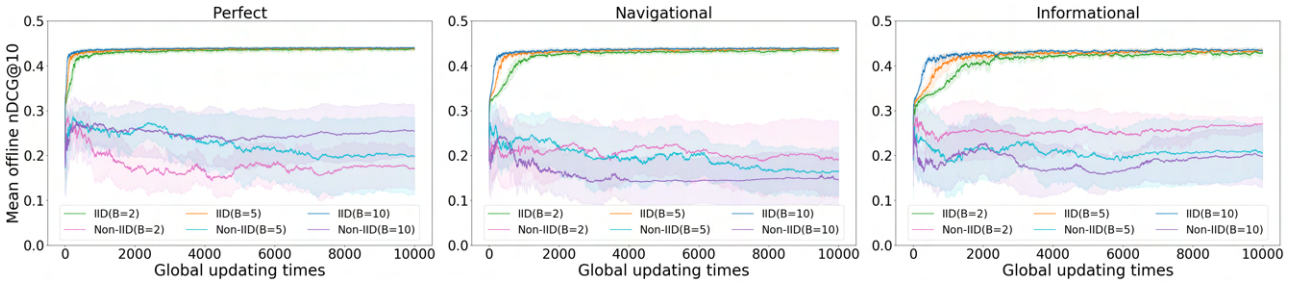
In non-IID subtype 1, each client only holds query-document pairs from one specific value of relevance label. We use $\#R = 1$ to denote this partitioning strategy. This federated setup involves $|C| = 5$ clients, which is in line with the 5-level relevance judgements of MSLR-WEB10K. We also vary the local updating time $B \in \{2,5,10\}$ to investigate the impact of local updating with a fixed global communication time $T = 10,000$. For non-IID subtype 2, each client holds data samples from two relevance labels – we denote this as $\#R = 2$. We simulate $|C| = 10$ clients and for fair comparison between the two non-IID subtypes, we also simulate $|C| = 10$ clients for $\#R = 1$ (with each label distributed on two different clients).

The IID experimental setting is the same as the non-IID in terms of federation, ranker parameters and evaluation procedure, except that each client now randomly picks a query from the whole training set with all graded judgements during the training period.

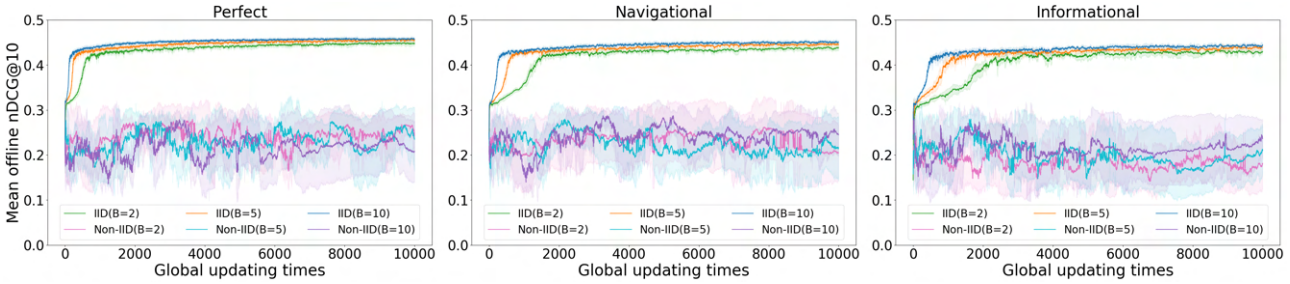
4.4.2 Impact of Type 2 non-IID Data

The offline performance for $\#R = 1$ on MSLR-WEB10K is shown in Figure 4.5 and the corresponding online performance is shown in Table 4.3. From the offline results, it is clear that the rankers learned with non-IID data under-fit the generalized held-out test set under all three settings of local updating times (B). For the linear ranker, a larger number of B achieves better test performance under the *perfect* click model. However, when it comes to noisier clicks (*navigational* or *informational*), the trend is reversed, although differences are minimal and the model performance fluctuates. For the neural ranker, no significant difference is observed from different size of local updates under non-IID settings, which are all outperformed by its IID counterparts. For the online results, all non-IID settings appear to over-fit the maximum value ($online\_ndcg = 1589.23$) as each client's local data only contains data from one relevance label.

Offline results for $\#R = 2$ are shown in Figure 4.6. In this case, the effectiveness of the learnt

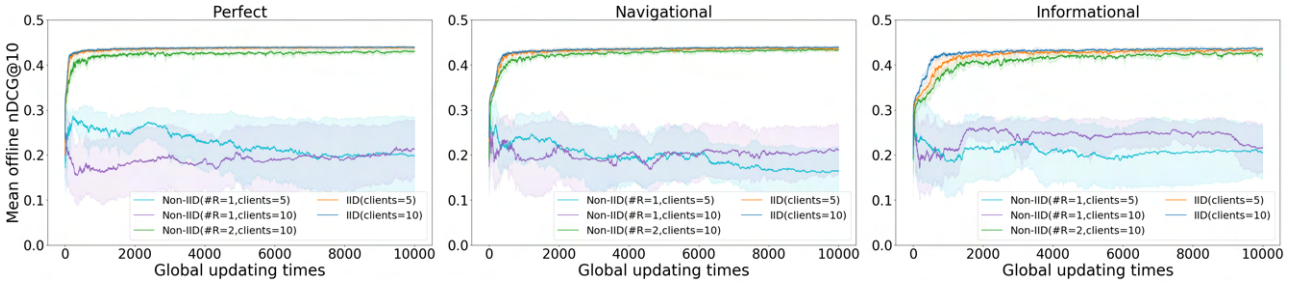


(a) MSLR-WEB10K (linear ranker)

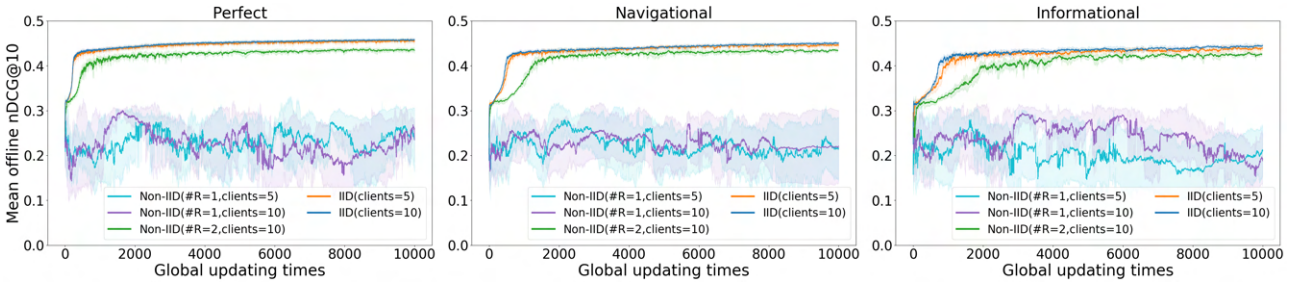


(b) MSLR-WEB10K (neural ranker)

Figure 4.5: Offline performance (nDCG@10) on MSLR-WEB10K for Type 2 ($\#R = 1$), under three instantiations of CCM click model and three local updates setting ($B \in \{2, 5, 10\}$); results averaged across all dataset splits and experimental runs.



(a) MSLR-WEB10K (linear ranker)



(b) MSLR-WEB10K (neural ranker)

Figure 4.6: Offline performance (nDCG@10) on MSLR-WEB10K for Type 2 ($\#R = 2$), under three instantiations of CCM click model with local updates setting ($B = 5$); results averaged across all dataset splits and experimental runs.

rankers is much higher than for $\#R = 1$: a diversity in labels held by a client prevents major losses in FOLTR effectiveness. This result is also consistent with previous results in general federated learning with non-IID data: Li et al. [32] found that the most challenging setting is when each client only has data samples from a single class (label). We further note that another reason for the performance gap is the pairwise loss used in FPDGD [2]: when each client only has one relevance label, it is hard to infer

Table 4.3: Online performance (discounted cumulative nDCG@10) on MSLR-WEB10K for Type 2 ($\#R = 1$), averaged across dataset splits and runs.

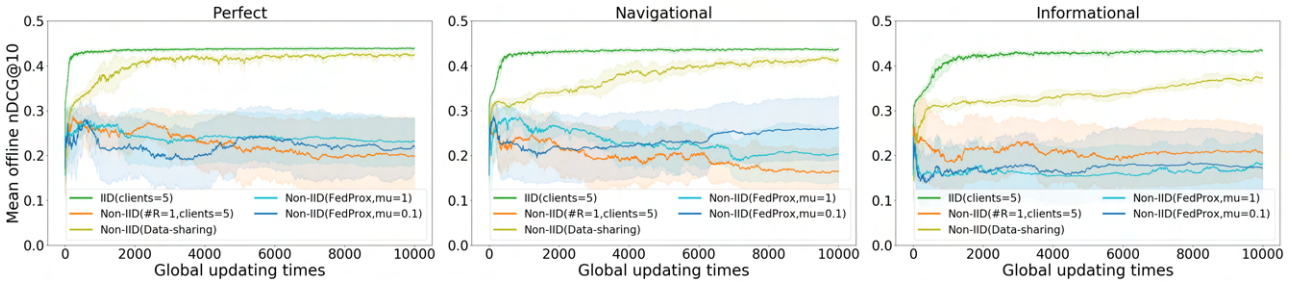
		linear ranker			*neural ranker*		
	click	$B = 2$	$B = 5$	$B = 10$	$B = 2$	$B = 5$	$B = 10$
IID	*perfect*	742.10	778.56	798.05	716.55	781.18	815.8
	navigational	698.25	743.35	771.04	649.83	728.96	775.87
	informational	672.23	722.23	757.10	612.76	693.35	748.64
non-IID	*perfect*	1589.23	1589.23	1589.23	1589.23	1589.23	1589.23
	navigational	1589.23	1589.23	1589.23	1589.23	1589.23	1589.23
	informational	1589.23	1589.23	1589.23	1589.23	1589.23	1589.23

preferences between document pairs (as they both have the same label). However, given labels from two levels of relevance ($\#R = 2$), pairwise differences can be effectively inferred. This suggests that the results obtained here for Type 2 data may not generalise to other FOLTR methods beyond FPDGD if they do not rely on the pairwise preference mechanism. We further note, however, that FPDGD is the current state-of-the-art method and that the only available alternative [27] displays highly variable and sensibly worse performance compared to FPDGD [1, 69]. Therefore, new methods of FOLTR also need to be validated in the presence of Type 2 data.

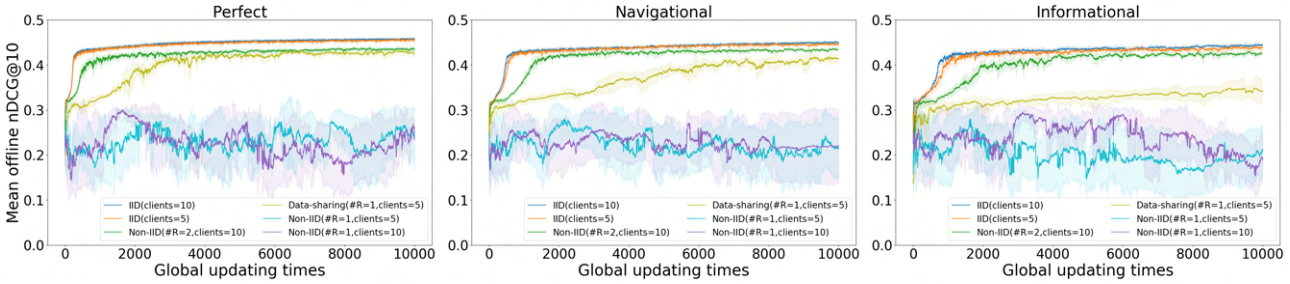
In summary, we find that if data is distributed in a non-IID manner across clients according to Type 2, the effectiveness of FOLTR (and specifically of FPDGD) is seriously affected in the case of $\#R = 1$; however, if $\#R = 2$ then gaps in effectiveness compared to IID settings are minimal.

4.4.3 Dealing with Type 2 non-IID Data

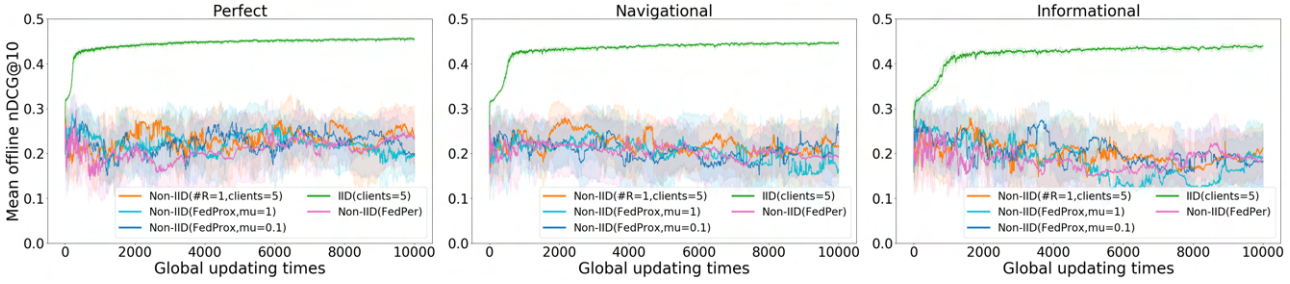
To mitigate the effect of Type 2 non-IID data, we investigate three existing methods from the federated learning literature: Data-sharing, FedProx and FedPer. FedProx and FedPer have been described in Section 4.3.3. Data-sharing was first proposed by Zhao et al. [30]. They attribute the performance reduction observed on non-IID data to the weight divergence, which is further affected by the divergence between the local data distribution and the overall distribution. They then introduce a straightforward idea to improve FedAvg: slightly reduce the divergence that causes the global model to underperform. This can be achieved as follows. A globally shared dataset G characterised by the overall data distribution is centralised on the server, and a warm-up global model is trained from G. Then, a random α proportion of G is sent to all clients to update the local model by both local training data and the shared data from G. Lastly, the server aggregates the local models from the clients and updates the global model with FedAvg. Experimental results on other machine learning tasks show that data sharing can significantly enhance the global model performance in the presence of non-IID data. However, the shortcomings are also pronounced. It is challenging to collect uniformly distributed global datasets in real-world scenarios because either the global server needs some prior knowledge about the local data distributions or each client needs to share parts of the local data (violating the privacy requirement underlying FL).



(a) MSLR-WEB10K (linear ranker)



(b) MSLR-WEB10K (neural ranker) - data sharing



(c) MSLR-WEB10K (neural ranker) - FedProx & FedPer

Figure 4.7: Offline performance on MSLR-WEB10K when using Data-sharing, FedProx and FedPer on Type 2 non-IID data (with $\#R = 1$); results averaged across dataset splits and experimental runs.

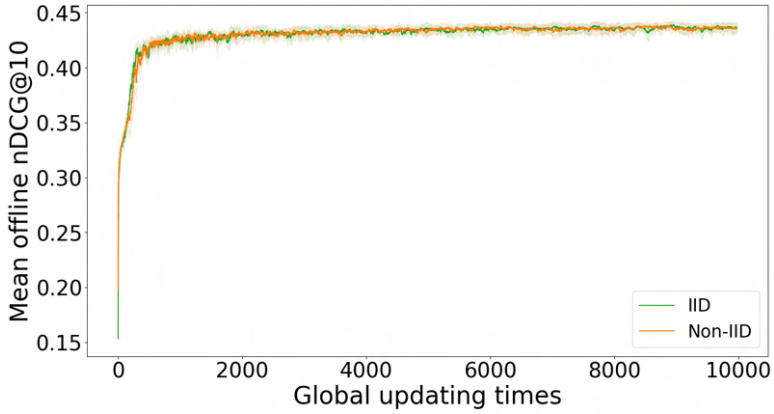
Figure 4.7 reports the results for Data-sharing, FedProx and FedPer on MSLR-WEB10K dataset under label distribution skewness $\#R = 1$. We randomly select 10% of the entire dataset as the globally shared data and simulate $|C| = 5$ clients with $B = 5$ local updates before each global update. Results show that the global performance can be significantly enhanced with data-sharing for both linear and neural rankers. On the other hand, neither FedProx nor FedPer provides statistically significant gains over the basic FPDGD on Type 2 non-IID data (with $\#R = 1$).

4.5 Other Data Types

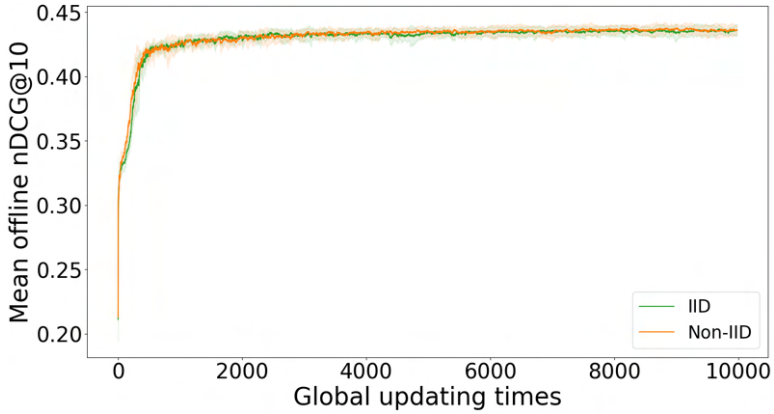
4.5.1 Type 3: Click Preferences

Next, we consider as a source of non-IID data the noise and biases caused by the different click preferences arising from different clients that participate in the FOLTR training; we term this type of non-IID data as *click preference skewness* (Type 3).

The mechanism to emulate non-IID data of Type 3 and IID data baseline in our FOLTR experiments is as follows. We study two widely used click models: the *Cascade Click Model* (CCM) [58], and



(a) MSLR-WEB10K (linear ranker) under CCM clicks



(b) MSLR-WEB10K (linear ranker) under PBM clicks

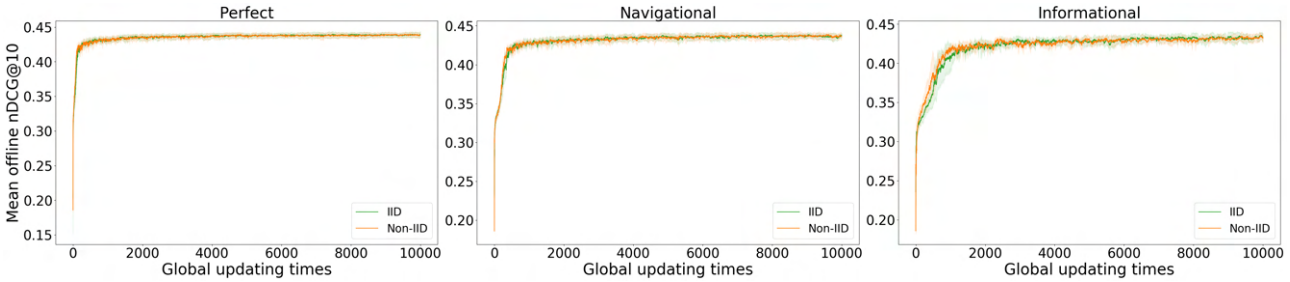
Figure 4.8: Offline performance (nDCG@10) on MSLR-WEB10K for Type 3, separately under CCM and PBM click model; results averaged across all dataset splits and experimental runs.

the *Position-Based Model* (PBM) [124]. For non-IID settings in CCM, each client chooses one of three widely-used instantiations of CCM, namely *perfect, navigational, informational*. For the non-IID settings with PBM, we generate 5 instantiations based on varying the $\eta \in \{0, 0.5, 1, 1.5, 2\}$ parameter: each client is represented by one click type. Thus the federated setup involves 3 clients for CCM clicks and 5 for PBM. We set the local updating time $B = 5$ with fixed global communication times $T = 10,000$. In the IID setting, at every time, each client is simulated based on a click model randomly picked from all click models instantiations detailed above and used in the non-IID setting; this provides a fair comparison between the IID and non-IID settings. Our experiment on MSLR-WEB10K is shown in Figure 4.8: the difference between non-IID and IID data for Type 3 is not significant. This might be because different click instantiations only differ from the level of noise and biases they bring, which impact the speed of convergence of the ranker rather than the actual converging direction in FOLTR.

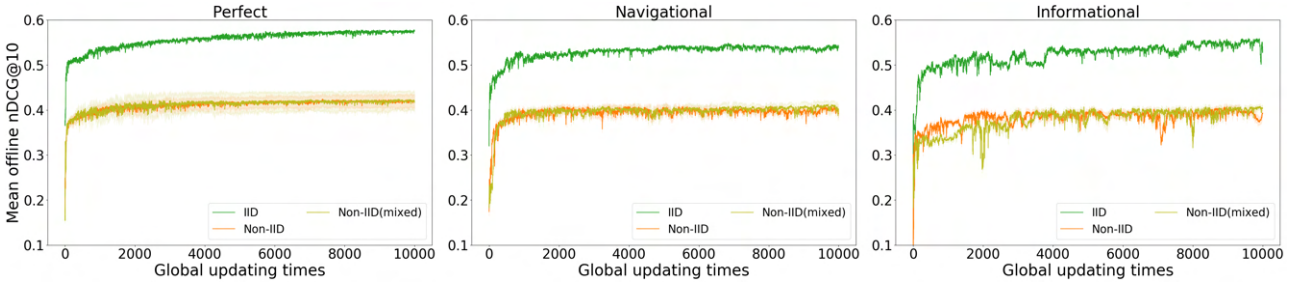
In summary, we find that if data is distributed in a non-IID manner across clients according to Type 3, the effectiveness of FOLTR (and specifically of FPDGD) is not impacted.

4.5.2 Type 4: Data Quantity

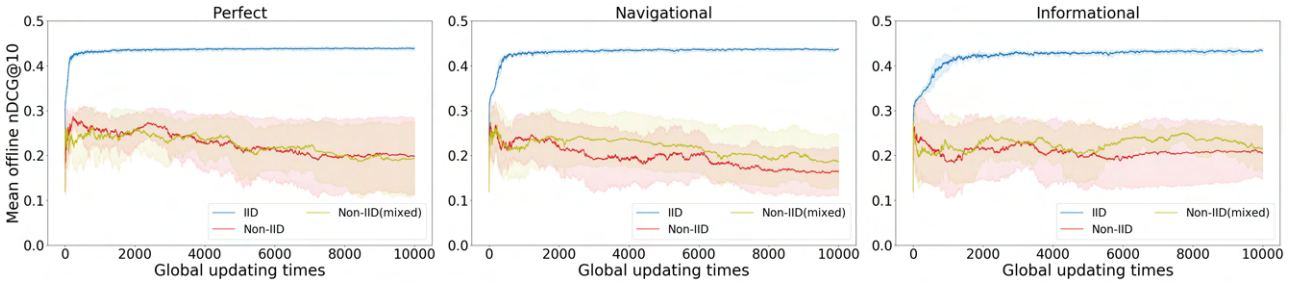
Finally, we consider the case of *data quantity skewness* (Type 4); this occurs when the number of training data varies across different clients. It is a common scenario in real-world applications. For



(a) MSLR-WEB10K (linear ranker)



(b) intent-change (linear ranker) mixed with type 1



(c) MSLR-WEB10K (linear ranker) mixed with type 2 ($\#R = 1$)

Figure 4.9: Offline performance (nDCG@10) on MSLR-WEB10K and intent-change for Type 4, under three instantiations of CCM click model; results averaged across all dataset splits and experimental runs.

example, in FOLTR, some clients tend to issue more queries and interact more with the searching system than others. Thus, they have more data for training than others. The situation represented by Type 4 may occur in combination with the other data types. In our empirical experiments, we have studied Type 4 data both on its own and combined with the document preferences skew (Type 1) and the document label distribution skew with $\#R = 1$ (Type 2).

Type 4 data is simulated by assigning different numbers of queries (Q) to each client during the same local updating period, thus leading to different local updating times for each client. The number of queries varies in $\{1, 3, 5, 7, 9\}$ and we simulate $|C| = 5$ clients in total with fixed global communication times $T = 10,000$.

When mixing other non-IID types with Type 4, we follow the same experimental settings of previous non-IID types, and we also assign different numbers of queries to each client (from $\{1, 3, 5, 7, 9\}$) during the same local updating period. Instead, in the IID simulation, each client has 5 iterations of searching for different queries. For both IID and non-IID, we use CCM click models for click simulation and train a linear ranker using FPDGD on MSLR-WEB10K for Type 1 with $\#R = 1$, and the dataset from Zhuang and Zuccon [16] (intent-change) for Type 2.

Empirical results on MSLR-WEB10K and intent-change show that if data is distributed in a non-IID manner across clients according to Type 4, the effectiveness of FPDGD is not impacted. We stress that this result may be specific to FPDGD because it uses the FedAvg paradigm and does not generalise to other FOLTR methods.

4.6 Chapter Summary

Despite federated learning receiving substantial attention, research in FOLTR is still in its early stages, with only two methods available at the time of writing [2, 27]. Importantly, studies that have proposed FOLTR methods have ignored an important issue that has been shown to affect the performance of federated learning systems [6]: that of the data not being distributed across the federated clients in an identical and independent manner (non-IID data). This chapter provides the first analysis of the impact of non-IID data on FOLTR thus charts directions for future research. Specifically, we develop a benchmark and introduce four types of non-IID data in FOLTR system. Furthermore, we conduce comprehensive experiments on simulating non-IID data and implementing existing remedy methods. We found that certain types of non-IID data present a threat to the performance of FOLTR and existing solutions employed in general federated learning to mitigate the non-IID data problem do not apply to the FOLTR setting. This study sheds light on future directions in addressing the awareness of non-IID issues and improving robustness to non-IID data for FOLTR.

The following publication has been incorporated as Chapter 5.

1. [4] **Shuyi Wang**, and Guido Zuccon, An Analysis of Untargeted Poisoning Attack and Defense Methods for Federated Online Learning to Rank Systems, *In Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 215–224, 2023

Chapter 5

Federated Online Learning to Rank under Poisoning Attacks

In the previous chapters, we address the effectiveness and robustness under non-IID data in FOLTR. Existing FOLTR systems however are not necessarily secure: the federation mechanism provides malicious clients with opportunities for attacking the effectiveness of the global ranker. For example, malicious clients can send arbitrary weights to the server so that the convergence of the global ranker can be perturbed after aggregation. This kind of attack is termed as *untargeted poisoning attack* and aim to compromise the integrity of the global model trained federatively [33]. This issue is critical for federated learning systems, but it has not yet been studied for FOLTR. In this chapter, we initiate the investigation of poisoning attacks and corresponding defense methods in the context of FOLTR systems. Specifically, we investigate the following research questions:

RQ3: Are FOLTR methods secure in the face of potential risks brought by malicious participants?

RQ3.1: Can data poisoning and model poisoning strategies undermine ranking performance on FOLTR?

RQ3.2: Can defense strategies mitigate the impact of malicious attacks in FOLTR?

Outside of FOLTR systems, poisoning attacks on federated learning systems has been shown successful in compromising model integrity across several federated machine learning tasks [82, 88, 125–127], including in natural language processing and recommender systems. To mitigate or remove the threat posed by poisoning attacks, defense strategies have been designed and optimised [84, 85, 128]. Defense strategies typically act upon the aggregation rules used in the global model updating phase. The vulnerability of existing FOLTR methods to these attacks and the effectiveness of the related defense mechanism is unknown. As shown in the previous chapter (Chapter 4 [1]), findings obtained

[1] We find that methods for dealing with non identical and independently distributed data in federated learning systems do not generalise to the context of FOLTR.

with respect to federated learning in other areas of Machine Learning or Deep Learning do not directly translate to the online learning to rank context, and therefore the study of these techniques in the context of FOLTR is important. Specifically, the performance of poisoning attacks and defense methods proposed in general domain cannot be guaranteed when applied to FOLTR: we address this limitation by adapting and investigating these methods to the setting of FOLTR and establish baselines for future studies. Within this scope, to answer RQ3.1, we experiment with and analyse data and model poisoning attack methods to showcase their impact on FOLTR search effectiveness. We further explore the effectiveness of defense methods designed to counteract attacks on FOLTR systems, with the aim of answering RQ3.2. We thus contribute an understanding of the effect of attack and defense methods for FOLTR systems, as well as identifying the key factors influencing their effectiveness.

5.1 PRELIMINARIES

5.1.1 Online Learning to Rank (OLTR)

In OLTR, the ranker is learned directly from user interactions (clicks in our study), rather than editorial labels. In this context, each client performs searches on several queries during each local training phase. For each query q, the candidate documents set is D_q and the local training data held by each client is $\{(x_i, y_i), i = 1...|D_q|\}_q$ with feature representation (x_i) and user's click signal (y_i) for each (q, d_i)-pair (where $d_i \in D_q$). The value of the click feedback y_i is either 0 (unclicked) or 1 (clicked). In practice, the *click* is dependent on the relevance degree of the candidate document d_i to the query q, the rank position of d_i, and other noise or randomness factors.

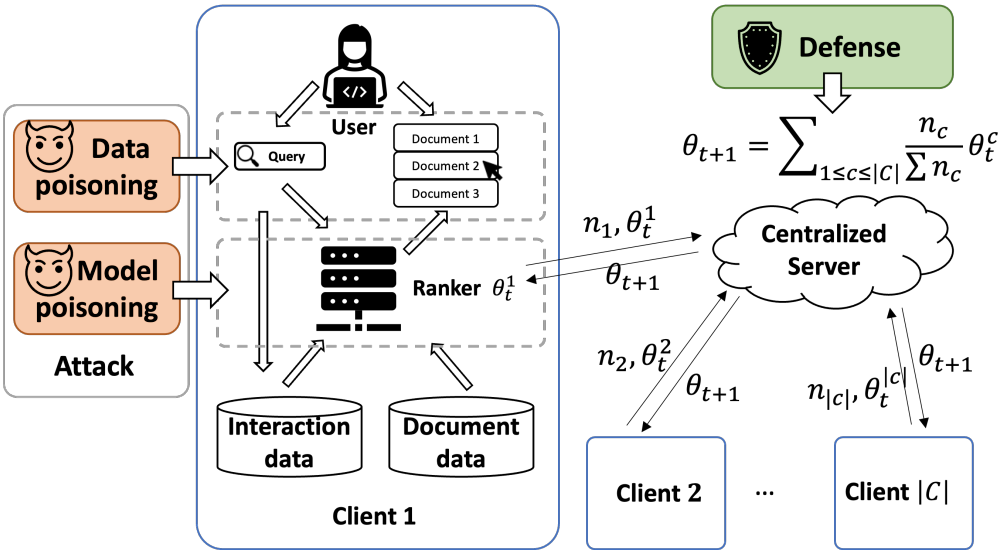


Figure 5.1: Overview of a FOLTR system with attack and defense modules (the arrows point to where these modules will be applied to).

Algorithm 6 FederatedAveraging PDGD.
• set of clients participating training: C, each client is indexed by c;
• number of local interactions for client c: n_c;
• local interaction set: B, model weights: θ.

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

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

5.1.2 Federated Pairwise Differentiable Gradient Descent (FPDGD)

We add our attacking and defense modules to the current state-of-the-art FOLTR system, the Federated Pairwise Differentiable Gradient Descent (FPDGD) [2], which is outlined in Algorithm 6 and illustrated in Figure 5.1. Within each iteration t, each client c considers n_c interactions and updates the local ranker using Pairwise Differentiable Gradient Descent (PDGD) [39]. After the local update is finished, each client sends the trained weights θ_t^c to the server. The server then leverages the widely-used Federated Averaging [7] to aggregate the local model updates. Afterwards, the new global weights θ_{t+1} are sent back to the clients as their new local rankers. We refer the reader to Section 3.2.1 for more details on FPDGD [2].

5.2 Attacks to FOLTR Systems

5.2.1 Problem Definition and Threat Model

We briefly introduce the attacker's capability and background knowledge in our study.

Attacker's capability: Poisoning attacks can come from both members (insiders) and non-members (outsiders) of the FOLTR system. Insiders include both the central server and the clients, while outsiders include eavesdroppers on communication channels and users of the final ranker (this is similar to adversarial attacks during inference). In this study, we focus on insider attacks by malicious participants in the FOLTR system since insider attacks are generally more effective than outsider attacks [126]. We assume the attacker has control over m collusive clients, which means that the training data and local model updates can be exchanged among the malicious clients. We restrict the percentage of collusive clients to less than 50%: higher amounts would make it trivial to manipulate the global model.

Attacker's background knowledge: We assume that the attacker has only access to the compro-

mised clients: the training data and local rankers of all remaining clients remain not accessible to the attacker. Thus, the attacker has limited prior knowledge: the training data and the locally updated models from the poisoned clients, and the shared global model. The exception of having full prior knowledge[2] will only be for the purpose of analysis and will be clarified in place.

Problem Formulation: Assume n clients are involved in the FOLTR system. Among them, m clients are malicious. Without loss of generality, we assume the first m participants are compromised. Be $\theta_{\mathbf{i}}$ the local model that the i-th client sends to the central server. The global ranking model is updated through aggregating all $\theta_{\mathbf{i}}$:

$$\theta_{\mathbf{g}} = agg(\theta_{\mathbf{1}}, ..., \theta_{\mathbf{m}}, \theta_{\mathbf{m+1}}, ..., \theta_{\mathbf{n}}) \tag{5.1}$$

5.2.2 Data Poisoning

Data poisoning methods aim to corrupt the training data in order to degrade the model's effectiveness. This can be done by adding malicious instances or altering existing instances in an adversarial manner.

Our data poisoning attack to FOLTR is inspired by the label flipping strategy [88, 129], in which the labels of honest training samples from one class are flipped to another class, while the features of the flipped samples are kept unchanged. In our case, we want to change the label of irrelevant documents into "high-relevant" and vice versa, without any changes to the feature representation of the corresponding query-document pairs. To achieve so, the attacker needs to intentionally flip the feedback by clicking on irrelevant documents to bring arbitrary noise thus poison the training.

In our experiments, as no click data is available with the considered datasets, we follow the common practice from previous literature in OLTR and FOLTR of simulating click behaviour based on the extensively-used *Cascade Click Model* (CCM) with three instantiations (*perfect*, *navigational*, *informational*) as described in Section 2.2.1. This click model has been shown to produce reasonable predictions of real-world user click behaviour.

Inspired by the three instantiations, we manipulate one *poison* instantiation to simulate malicious clicking behaviour. The click probability of *poison* instantiation is the reverse version of the *perfect* click behaviour: the highest probability of clicking is associated with the least relevance label. All stop probabilities in *poison* instantiation are set to zero as we assume the attacker wants to poison as many clicks as possible. The values we adopt for the four instantiations of CCM are reported in Table 5.1.

5.2.3 Model Poisoning

Unlike data poisoning, model poisoning directly modifies the local model updates (through poisoning gradients or model parameter updates) before sending them to the server. Some literature shows that model poisoning is more effective than data poisoning [33, 82] while it also requires sophisticated technical capabilities and high computational resources than solely poisoning data. In this section, we investigate two existing model poisoning methods.

[2] i.e. the attacker can also access information (training data, model updates) of non-poisoned clients.

Table 5.1: Instantiations of CCM click model for simulating user behaviour in experiments. $rel(d)$ denotes the relevance label for document d. Note that in the MQ2007 dataset, only three-levels of relevance are used. We demonstrate the values for MQ2007 in bracket.

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

Little Is Enough (LIE)

Baruch et al. [89] find that if the variance between local updates is sufficiently high, the attacks can make use of this by adding small amounts of noise to the compromised local models and bypass the detection of defense methods. They provide a perturbation range in which the attackers can successfully poison the learning process. To conduct the attack, the adversaries first compute the average μ and standard deviation σ of the before-attack benign local model updates of all collusive attackers $(\theta_{\mathbf{1}}, ..., \theta_{\mathbf{m}})$. A coefficient z is used and computed based on the number of benign and malicious clients. Finally, the local model of attackers is manipulated as $\theta_{\mathbf{i}}^{\mathbf{m}} = \mu - z\sigma$ for $i \in \{1, ..., m\}$ and sent to the central server who aggregates updates from all participants under certain rules in Equation 5.1. Baruch et al. [89] observe that, for image classification tasks, the small noises sufficiently compromise the global model while being sufficiently small to evade detection from defense strategies.

Fang's Attack

Fang et al. [86] proposed an optimization-based model poisoning attack tailored to specific robust aggregation rules (Krum, Multi-Krum, Trimmed Mean and Median), as will be explained in Section 5.3.

Fang's attack is conducted separately under two assumptions: (1) full knowledge, and (2) partial knowledge. Under full knowledge, the attacker has full access to local model updates of all benign clients. This is a strong and impractical assumption, and it is often not the case in real attacks on federated learning systems. In the partial knowledge scenario, the attacker only knows the local training data and models of the compromised clients.

In their attack to the robust aggregation rules Krum and Multi-Krum, the attacker computes the average μ of the benign updates in their possession, computes a perturbation $\mathbf{s} = -\text{sign}(\mu - \theta_{\mathbf{g}})$, and finally computes a malicious update as $\theta_{\mathbf{i}}^{\mathbf{m}} = (\theta_{\mathbf{g}} + \lambda \cdot \mathbf{s})$ by solving for the coefficient λ, where $\theta_{\mathbf{g}}$ is the before-attack global model during each federated training step. Thus, under the full knowledge

assumption, the average μ and perturbation signal $\mathbf{s}$ are computed based on all benign updates $(\theta_\mathbf{1}, ..., \theta_\mathbf{m}, \theta_{\mathbf{m+1}}, ..., \theta_\mathbf{n})$. For updates of the malicious clients $(\theta_\mathbf{1}, ..., \theta_\mathbf{m})$, the before-attack benign updates are leveraged. Under the partial knowledge scenario, only the before-attack benign updates $(\theta_\mathbf{1}, ..., \theta_\mathbf{m})$ are used to estimate the real values for average μ and the reversed deviation vector $\mathbf{s}$.

When attacking Trimmed Mean and Median, the goal is to craft the compromised local models based on the maximum $\theta_{max,j}$ or minimum $\theta_{min,j}$ benign parameters for each dimension j of the local model (this is one of the key features used by Trimmed Mean and Median for defending). The choice of $\theta_{max,j}$ or $\theta_{min,j}$ depends on which one deviates the global model towards the inverse of its update direction without attacks. Similar to when attacking Krum, the reversed deviation vector $\mathbf{s}$ is computed with full knowledge or estimated under partial knowledge with only before-attack updates from all attackers, so as the estimation of $\theta_{max,j}$ and $\theta_{min,j}$. After getting the j-th value of vector $\mathbf{s}$, the j-th dimension of the compromised local model is randomly sampled from the range built based on $\theta_{max,j}$ (if $s_j = 1$) or $\theta_{min,j}$ (if $s_j = -1$).

5.3 Defense for FOLTR Systems

The current state-of-the-art defense methods against untargeted poisoning attacks focus on enhancing the robustness of the aggregation rules (Equation 5.1) used during the global update phase, to counteract attempts by malicious clients to corrupt the training.

Next, we describe four robust aggregation rules that have been shown effective in general federated learning, but have not been evaluated for FOLTR.

5.3.1 Krum and Multi-Krum

The intuition behind the Krum method for robust aggregation [84] is that the malicious local model updates need to be far from the benign ones in order for the success of poisoning the global model. To evaluate how far a model update $\theta_\mathbf{i}$ is from the others, Krum computes the Euclidean distances between $\theta_\mathbf{i}$ and $\theta_\mathbf{j}$ for $i \neq j$. We denote $i \to j$ if $\theta_\mathbf{j}$ belongs to the set of $n-m-2$ closest local models of $\theta_\mathbf{i}$. Then the sum of $n-m-2$ shortest distances to $\theta_\mathbf{i}$ is computed and denoted as $s(i) = \sum_{i \to j} Euc\_dist(\theta_\mathbf{i}, \theta_\mathbf{j})$. After computing the distance score $s(i)$ for all local updates, Krum selects the local model with the smallest $s(i)$ as the global model $\theta_\mathbf{g}$:

$$\theta_\mathbf{g} = Krum(\theta_\mathbf{1}, ..., \theta_\mathbf{m}, \theta_{\mathbf{m+1}}, ..., \theta_\mathbf{n}) = \arg\min_{\theta_\mathbf{i}} s(i) \tag{5.2}$$

Multi-Krum is a variation of the Krum method. Multi-Krum, like Krum, calculates the distance score $s(i)$ for each $\theta_\mathbf{i}$. However, instead of choosing the local model with the lowest distance score as the global model (as Krum does), Multi-Krum selects the top f local models with the lowest scores and computes the average of these f models ($\theta'_\mathbf{i}$, where $i \in \{1, ..., f\}$) to be the global model.

$$\theta_{\mathbf{g}} = \textit{Multi-Krum}(\theta_{\mathbf{1}}, ..., \theta_{\mathbf{m}}, \theta_{\mathbf{m+1}}, ..., \theta_{\mathbf{n}}) = \frac{1}{f}\sum_{i=1}^{f} \theta'_{\mathbf{i}} \tag{5.3}$$

In our empirical investigation, we set the Multi-Krum parameter $f = n - m$, as in previous work [84].

5.3.2 Trimmed Mean and Median

Assume that θ_{ij} is the j-th parameter of the i-th local model. For each j-th model parameter, the Trimmed Mean method [85] aggregates them separately across all local models. After removing the β largest and smallest among $\theta_{1j}, ..., \theta_{nj}$, the Trimmed Mean method computes the mean of the remaining $n - 2\beta$ parameters as the j-th parameter of the global model. We denote $U_j = \{\theta_{1j}, ..., \theta_{(n-2\beta)j}\}$ as the subset of $\{\theta_{1j}, ..., \theta_{nj}\}$ obtained by removing the largest and smallest β fraction of its elements. That is, the j-th parameter of the global model updated by Trimmed Mean is:

$$\theta_j = \textit{Trimmed Mean}(\theta_{1j}, ..., \theta_{nj}) = \frac{1}{n-2\beta}\sum_{\theta_{ij} \in U_j} \theta_{ij} \tag{5.4}$$

In our implementation, as in previous work on general federated learning [86, 87, 130], we set β to be the number of compromised clients m.

The Median method, like the Trimmed Mean method, sorts the j-th parameter of n local models. Instead of discarding the β largest and smallest values (as in Trimmed Mean), the Median uses the median of $\theta_{1j}, ..., \theta_{nj}$ as the j-th parameter of the global model:

$$\theta_j = \textit{Median}(\theta_{1j}, ..., \theta_{nj}) \tag{5.5}$$

In case n is an even number, the median is calculated as the average of the middle two values.

5.4 Experimental setup

We next describe our experimental setup to evaluate the considered attack and defense mechanism in the context of a FOLTR system.

Datasets. Our experiments are performed on four commonly-used LTR datasets: MQ2007 [40], MSLR-WEB10K [40], Yahoo [45], and Istella-S [56]. Detailed descriptions about each dataset are discussed in Section 2.2.1.

Federated setup. We consider 10 participants ($n = 10$) in our experiments, among which m clients are attackers. This setup is representative of a cross-silo FOLTR system, typical of a federation of a few institutions or organisations, e.g. hospitals creating a ranker for cohort identification from electronic health records [131]. In this chapter we will not consider the setup of a cross-device FOLTR system, where many clients are involved in the federation: this is representative of a web-scale federation.

We assume that the malicious clients can collude with each other to exchange their local data and model updates to enhance the impact of attacks. In the federated setting, each client holds a

copy of the current ranker and updates the local ranker through issuing $B = 5$ queries along with the respective interactions. The attackers can only compromise the local updating phase through poisoning the training data or model updates of the controlled malicious clients. After the local updating finishes, the central server will receive the updated ranker from each client and aggregate all local messages to update the global ranker. In our experiments, we consider the following aggregation rules: (1) FedAvg, (2) other robust aggregation rules introduced in Section 5.3. Unless otherwise specified, we train the global ranker through $T = 10,000$ global updating times.

User simulations. We follow the standard setup for user simulations in OLTR [2, 3, 39, 50]. We randomly sample from the set of queries in the static dataset to determine the query that the user issues each time. After that, the pre-selected documents for the query are ranked by the current local ranking model to generate a ranking result. For every query, we limit the SERP to 10 documents. User interactions (clicks) on the displayed ranking list are simulated through the CCM click models introduced in Sec 2.2.1 and Sec 5.2.2. For the user simulation in model poisoning, we simulate three types of users using the three click instantiations: *perfect*, *navigational*, and *informational*. We experiment on the three types of users separately in order to show the impact of attacking on different types of users. For data poisoning, we simulate the poisoned click based on the *poison* click combining with benign users on the aforementioned three types of click models separately to show the impact of our data poisoning strategies on different types of benign clicks.

Ranking models. We experiment on a linear and neural model as the ranking model when training with FPDGD. For the linear model, we set the learning rate $\eta = 0.1$ and zero initialization was used. As in the original PDGD and FPDGD studies [2, 39], the neural ranker is optimized using a single hidden-layer neural network with 64 hidden nodes, along with $\eta = 0.1$.

Evaluation. We evaluate the attack methods by comparing the gap in offline performance obtained when a specific attack is performed and when no attack is performed. The higher the performance degradation, the more effective the attack. As we limit each SERP to 10 documents, we use *nDCG*@10 for offline evaluation as discussed in Section 2.2.1. We record the offline *nDCG*@10 score of the global ranker during each federated training update.

5.5 Results for Data Poisoning

We perform data poisoning attack and four defense methods across different settings of user behaviours (i.e. click models) and number of attackers ($\{10\%, 20\%, 30\%, 40\%\}$). Results on MSLR-WEB10K with a linear ranker are shown as solid lines in Figure 5.2 – results for other datasets are similar and can be found in chapter appendix Section 5.9.

5.5.1 Attacks

In the plots of Figure 5.2, the solid lines represent the results of data poisoning when no defense method is deployed. Among them, the black line represents no attacking situation (”honest” baseline).

We can observe that the effect of data poisoning depends on the settings of user behaviors (i.e. click models) and the number of attackers.

Effect of number of attackers. By comparing the solid curves in each plot of Figure 5.2, we can observe that the overall performance of the FOLTR system decreases as the number of attackers increases, compared to the "honest" baseline. Thus, the higher the number of attackers, the more degradation on the FOLTR system is experienced.

Ease of attack under different user behaviours. By comparing the plots within each row, we see the effect of data poisoning is different under different user behaviours. In the navigational and informational settings, attacks carried by as little as 20% of clients can significantly affect the system. However, to successfully attack the perfect click, a higher number of malicious clients is needed. Across all datasets, the informational click model is the most affected by attacks, while the perfect click model only experiences considerable losses when a large number of clients has been compromised.

Neural ranker vs. linear ranker. The findings from results for the neural ranker under data poisoning attack are similar to those for the linear ranker – and this pattern is valid across all remaining experiments we report. Therefore, we only report experiments using the linear ranker in the following sections.

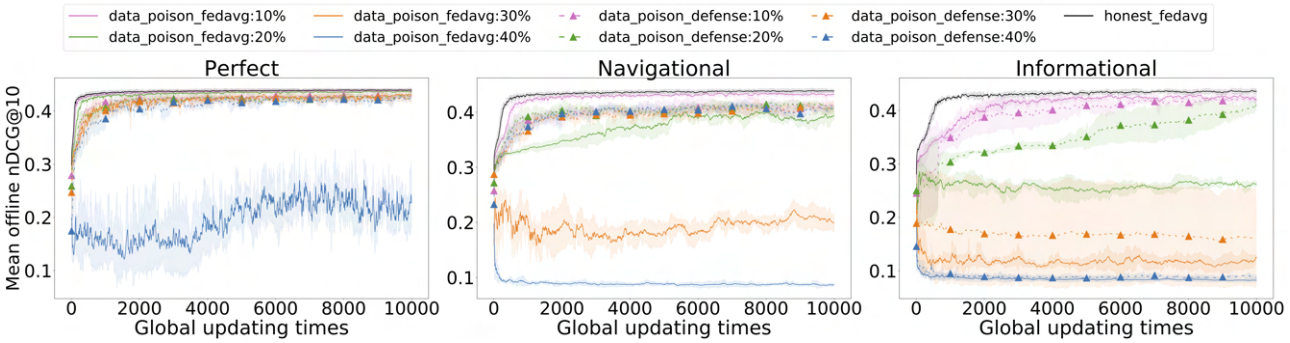
5.5.2 Defense

Next, we demonstrate the effectiveness of our four defense mechanisms against data poisoning attack. The results on MSLR-WEB10K are shown by the dashed curves in Figure 5.2. Each row corresponds to one defense method.

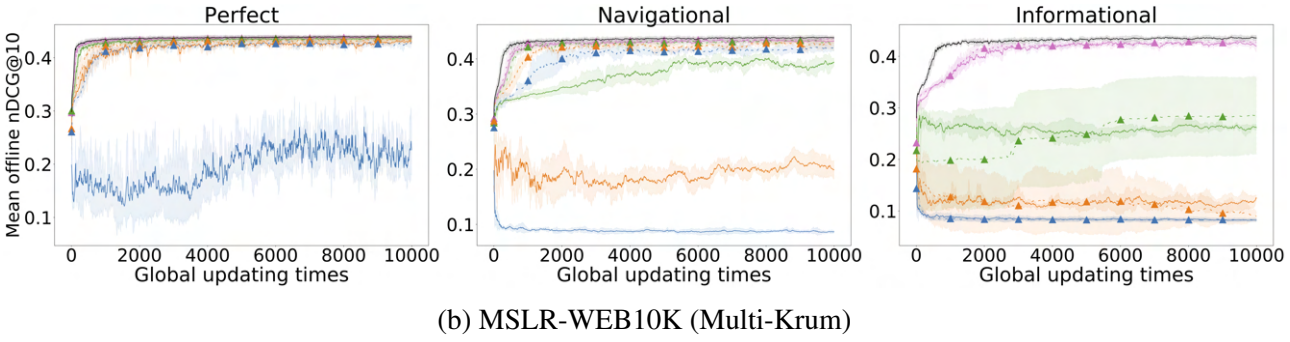
Krum. Overall, Krum performs well across all datasets and for all three types of click models once the percentage of malicious clients reaches 20% or higher, with the exception of MQ2007. However, Krum does not work when defending against 10% of clients, except for Istella-S. The accuracy drop from deploying Krum (as shown in Section 5.7) outweighs its effectiveness in defense, especially when there is a relatively small impact on the effectiveness of the model, as is in the case when 10% of the clients are malicious. Additionally, Krum does not show any improvement in defending certain scenarios under the informational click model, such as for MQ2007 under all percentages of malicious clients, and for MSLR-WEB10K when 40% of clients are malicious.

Multi-Krum. The results obtained for Multi-Krum show similar effectiveness on the perfect click model as Krum. It is important to note that the perfect click model is the hardest to attack among the three types of click models considered. Multi-Krum provides slightly better defense performance on navigational clicks compared to Krum, especially when there are fewer attackers (30% or less). However, for the informational click model, Multi-Krum does not perform as well as Krum. This is because the variance of the local model updates is relatively higher in the noisier informational click model. After averaging the selected local models, the advantage of Multi-Krum is reduced, especially when there are more than 30% malicious clients.

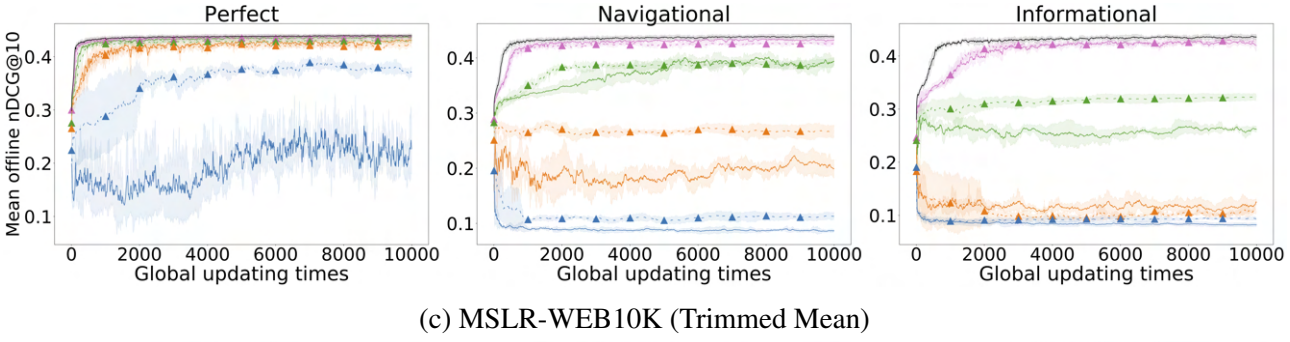
Trimmed Mean. Across all experiments, Trimmed Mean does not perform well on the noisiest



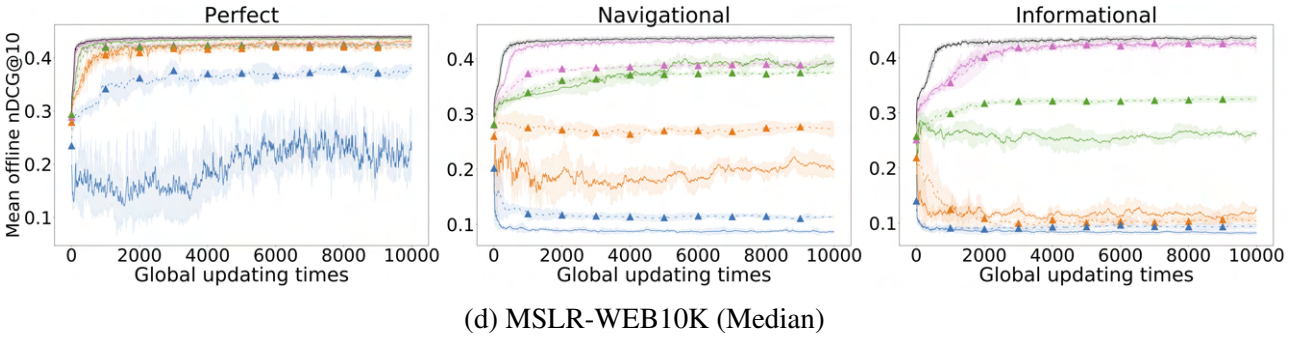
(a) MSLR-WEB10K (Krum)



(b) MSLR-WEB10K (Multi-Krum)



(c) MSLR-WEB10K (Trimmed Mean)

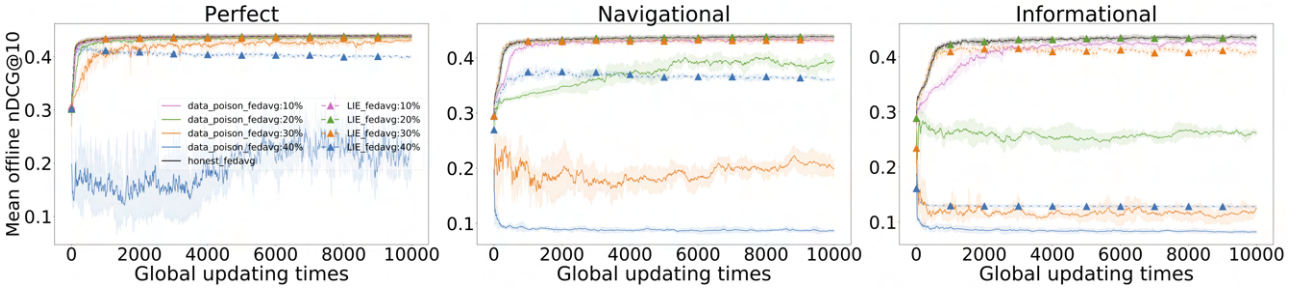


(d) MSLR-WEB10K (Median)

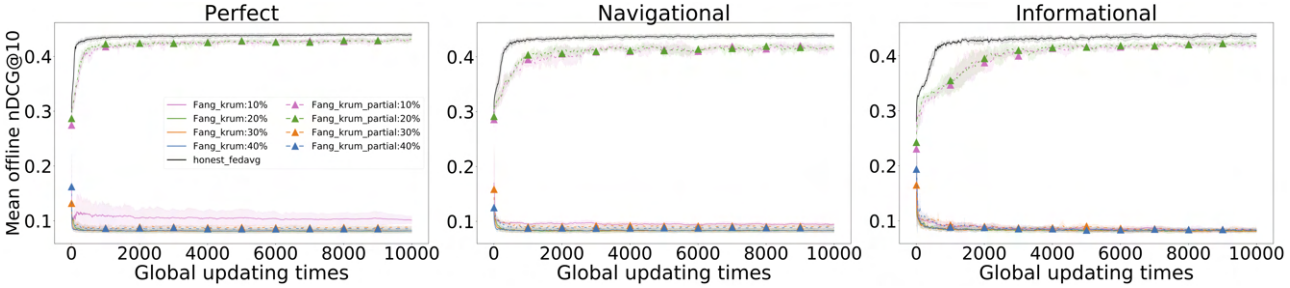
Figure 5.2: Offline performance (nDCG@10) for MSLR-WEB10K under data poisoning attack and defense strategies, simulated with three benign instantiations of click model and different percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$; results averaged across all dataset splits and experimental runs.

click model (informational) when there are more than 30% malicious clients involved. When the malicious clients are 20% or 30%, Trimmed Mean provides lower performance gains compared to Krum, but it performs similarly to Krum when only 10% of the clients are malicious.

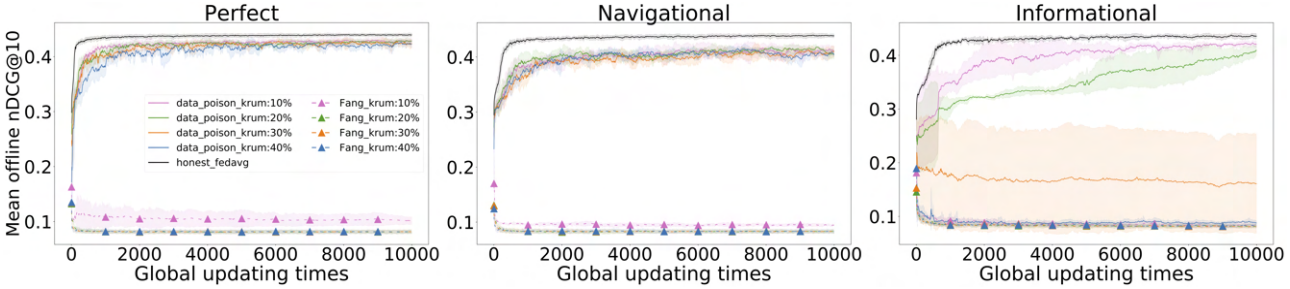
Median. Like Trimmed Mean, Median does not provide improved performance on the noisy informational click model when 30% or 40% of clients are malicious. Similarly, and like other robust aggregation rules, Median does not show significant improvements when only 10% of clients are



(a) MSLR-WEB10K (LIE)



(b) MSLR-WEB10K (Fang's attack on Krum)



(c) MSLR-WEB10K (Fang's attack on Krum - full knowledge)

Figure 5.3: Offline performance (nDCG@10) under model poisoning attacks, simulated using three benign instantiations of click model with percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$. The aggregation rule is FedAvg (for Figure 5.3a) and Krum (for Figure 5.3b- 5.3c).

malicious. In fact, the Median's performance even decreases on the navigational click model for MSLR-WEB10K with 10% of malicious clients. When the malicious clients are 20% and 30% of all clients in the federation, the performance gain provided by Median is similar to that of Trimmed Mean.

Summary. Overall, Krum and Multi-Krum work better than Trimmed Mean or Median when defending against data poisoning attacks, with the exception that Trimmed Mean and Median perform better on the smaller MQ2007 dataset.

5.6 Results for Model Poisoning

We implement the model poisoning strategies specified in Section 5.2.3 and report results on MSLR-WEB10K [3], specifically comparing their poisoning effectiveness with that of data poisoning methods.

[3] Results for other datasets are similar and can be found in chapter appendix Section 5.9.

5.6.1 Little Is Enough (LIE)

The experimental results obtained for LIE are partially shown in Figure 5.3a, along with a comparison with data poisoning.

Ineffectiveness of LIE. The results indicate that LIE is less effective in attacking the performance of the global model compared to data poisoning, with one exception for the perfect click model on the Yahoo dataset when 40% of the clients are malicious. This shows that adding random noise to compromise the local models is less effective for attacking the global ranker performance than compromising the click signals directly. Because of the poor attacking effectiveness of LIE, we do not investigate how it performs when defense strategies are put in place.

5.6.2 Fang's Attack

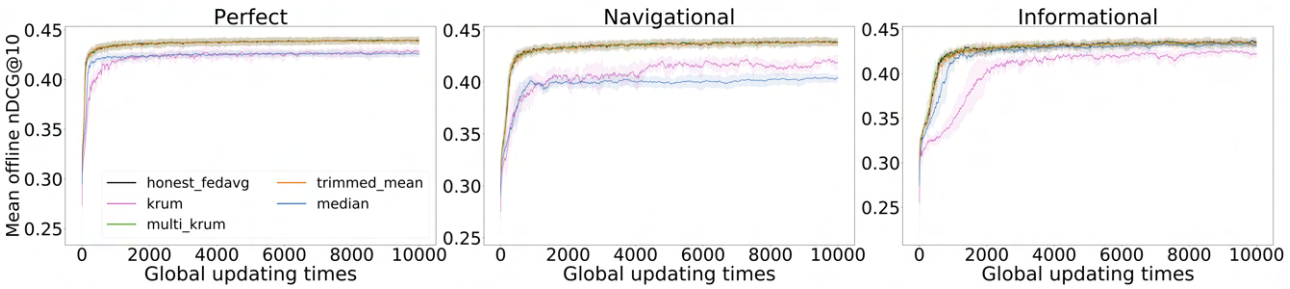
In our experiments, we implement Fang's attacks on four robust aggregation rules, with each attacking strategy tailoring specific defense strategies except that the same attack method is shared for Trimmed Mean and Median.

Full knowledge vs. partial knowledge. First, we compare the attacking performance under both full knowledge and partial knowledge assumptions. According to previous findings in general federated learning [86], attacking with full knowledge performs consistently better than with partial knowledge as the tailored attack can be optimised with auxiliary information about benign clients. From our results (results on MSLR-WEB10K under Krum are shown in Figure 5.3b), we observe that full knowledge performs better with fewer malicious clients (10% and 20%), but the gap in effectiveness obtained between full and partial knowledge decreases as the number of malicious clients increases (30% and 40%), thus leading to differences compared to the general results in federated learning. This is because with more malicious clients, partial knowledge (knowledge of before-attack local model updates for compromised clients) provides enough information to effectively poison the global model while avoiding detection by robust defense strategies.

Fang's Attack vs. data poisoning. Next, we compare Fang's attack under the full knowledge assumption against the data poisoning method under the same robust-aggregation rule (results on MSLR-WEB10K under Krum are shown in Figure 5.3c). We find that Fang's attack can successfully poison FOLTR and mitigate the impact of defense methods compared to data poisoning. This finding aligns with the original results from Fang et al. [86].

5.7 Impact of Defense under No-Attack

Robust aggregation rules exhibit improvements in defending against poisoning attacks under some circumstances. But in real-world settings, the administrator of the FOLTR system has no knowledge of whether an attack is taking place. Thus, if the system administrator wishes to ensure protection against attacks, they may be required to deploy defense strategies irrespective of an attack ever taking place, or not. However, is there a price to pay, in terms of search effectiveness, if a defense strategy is



(a) MSLR-WEB10K (benign clients)

Figure 5.4: Offline performance (nDCG@10) of FOLTR system when no attack is present but defense strategies are deployed; results averaged across all dataset splits and experimental runs.

deployed on a FOLTR system that is not exposed to an attack? We investigate this next, by comparing the effectiveness of a FOLTR system with no malicious clients and with different defense strategies implemented against the effectiveness of the same system with no defense.

The experimental results on MSLR-WEB10K reported in Figure 5.4 show that using Krum and Median leads to a decrease in performance compared to the FedAvg baseline when no attacks are present. Results for other datasets are similar and are put in chapter appendix Section 5.9 for space reasons. This finding has also been reported before in general federated learning literature [132–134], especially when each client's local training data is non independent and identically distributed (non-IID). This is because those Byzantine-robust FL methods exclude some local model updates when aggregating them as the global model update [132, 133]. This decrease raises questions about the use of these methods in FOLTR systems when no malicious client is present – and it suggests that if reliable methods for attack detection were available, then defense mechanisms may better be deployed only once the attack takes place.

5.8 Conclusion

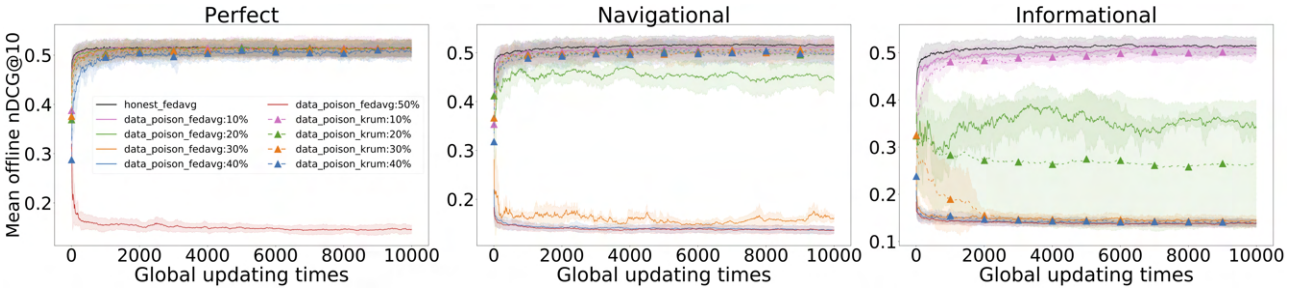
In this chapter we explore attacks and defense mechanisms for federated online learning to rank (FOLTR) systems, focusing on the potential degradation of ranking performance caused by untargeted poisoning attacks. We investigate both data and model poisoning strategies and evaluate the effectiveness of various state-of-the-art robust aggregation rules for federated learning in countering these attacks. Our findings indicate that sophisticated model poisoning strategies outperform data poisoning methods, even when defense mechanisms are in place. We also reveal that deploying defense mechanisms without an ongoing attack can lead to ranker performance degradation. This finding recommends care in the deployment of such mechanisms and suggests that future research should explore defense strategies that do not deteriorate FOLTR ranker performance if no attack is underway.

Code, experiment scripts and experimental results are provided at `https://github.com/ielab/foltr-attacks`.

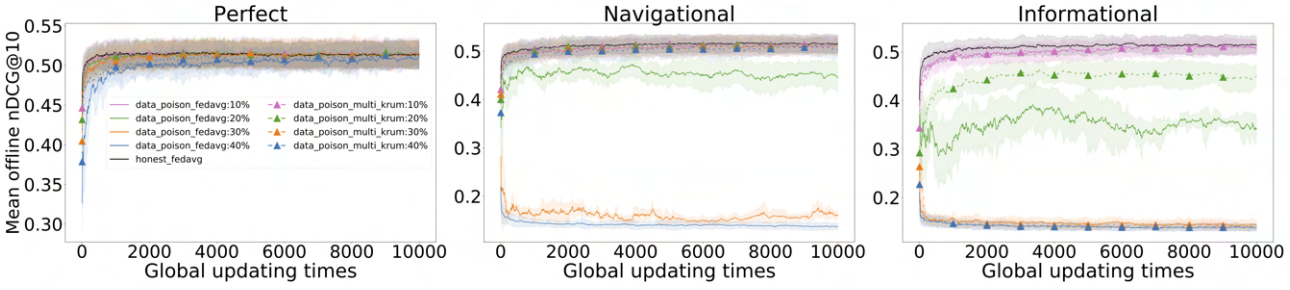
5.9 Chapter Appendix

5.9.1 Results for Data Poisoning

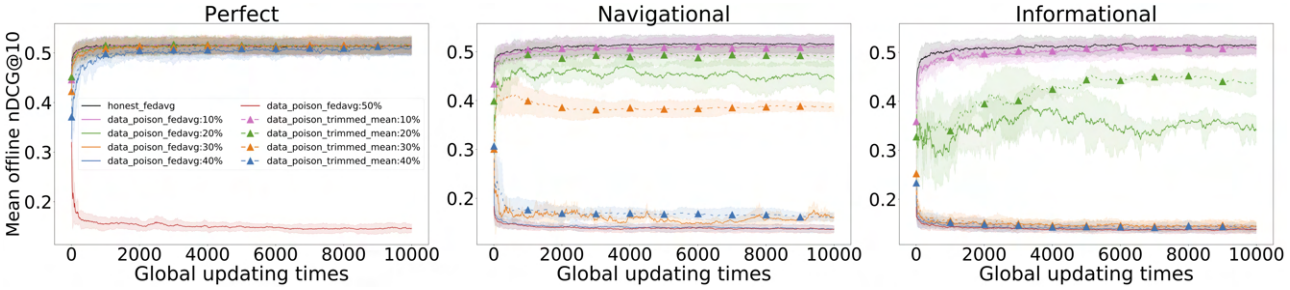
In this section, we report the results of data poisoning attack and four defense methods across different settings of user behaviours (i.e. click models) and number of attackers ($\{10\%, 20\%, 30\%, 40\%\}$) on the remaining datasets. Results on MQ2007 is shown in Figure 5.5. Results on Yahoo is shown in Figure 5.6. Results on Istella-S is shown in Figure 5.7. Further analysis on results is detailed in Section 5.5



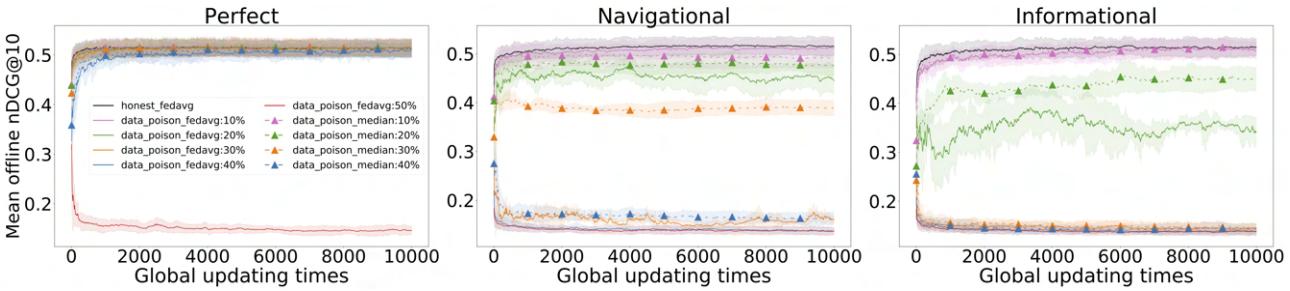
(a) MQ2007 (Krum)



(b) MQ2007 (Multi-Krum)

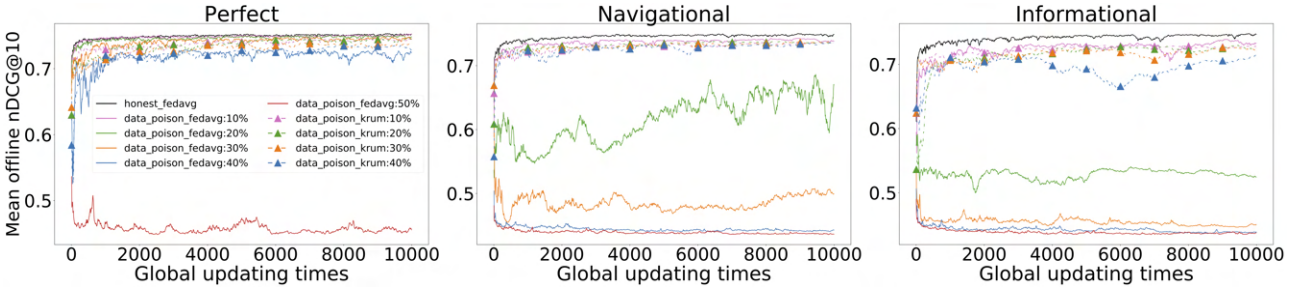


(c) MQ2007 (Trimmed Mean)

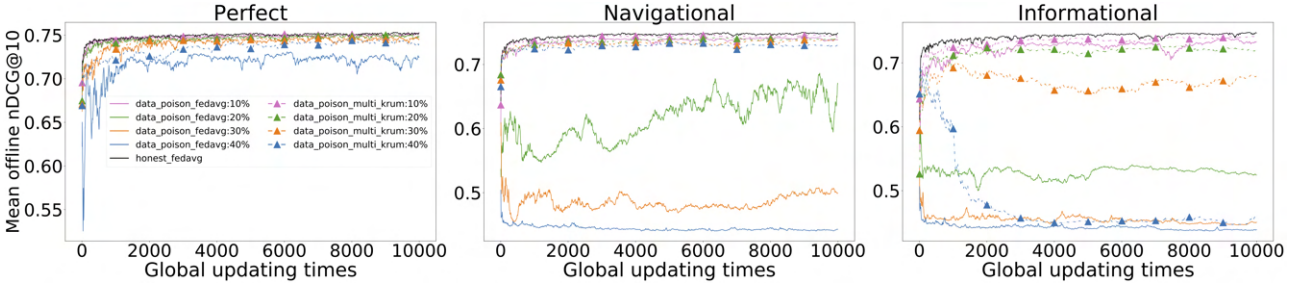


(d) MQ2007 (Median)

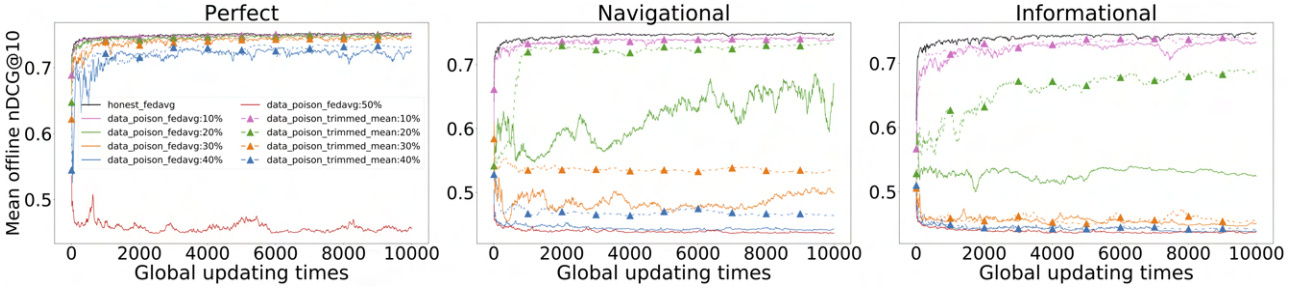
Figure 5.5: Offline performance (nDCG@10) for MQ2007 under data poisoning attack and defense strategies, simulated with three benign instantiations of click model and different percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$; results averaged across all dataset splits and experimental runs.



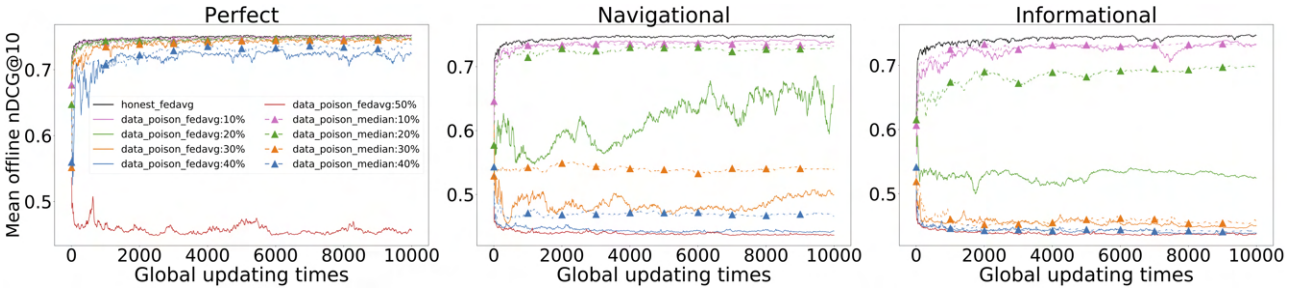
(a) Yahoo (Krum)



(b) Yahoo (Multi-Krum)

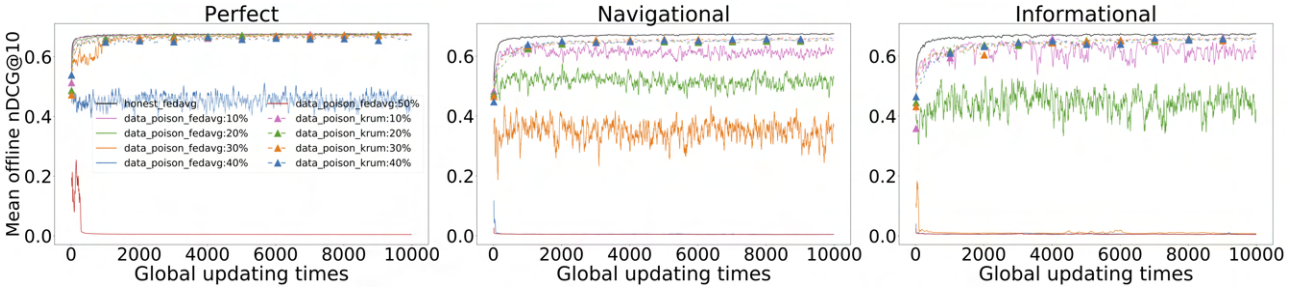


(c) Yahoo (Trimmed Mean)

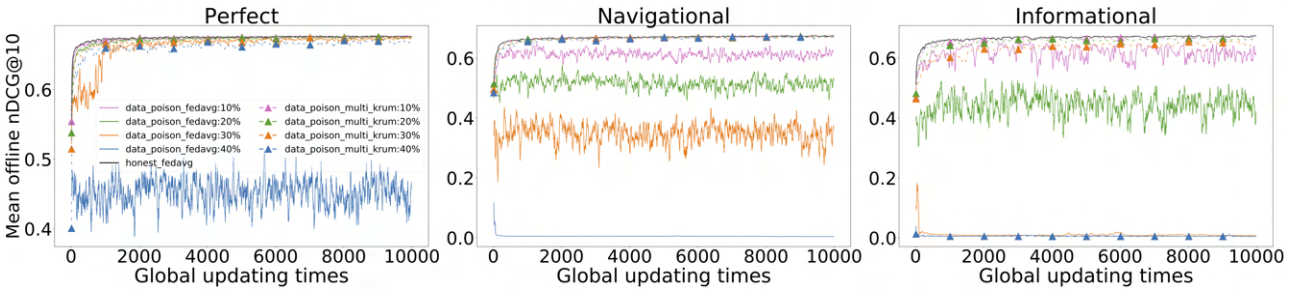


(d) Yahoo (Median)

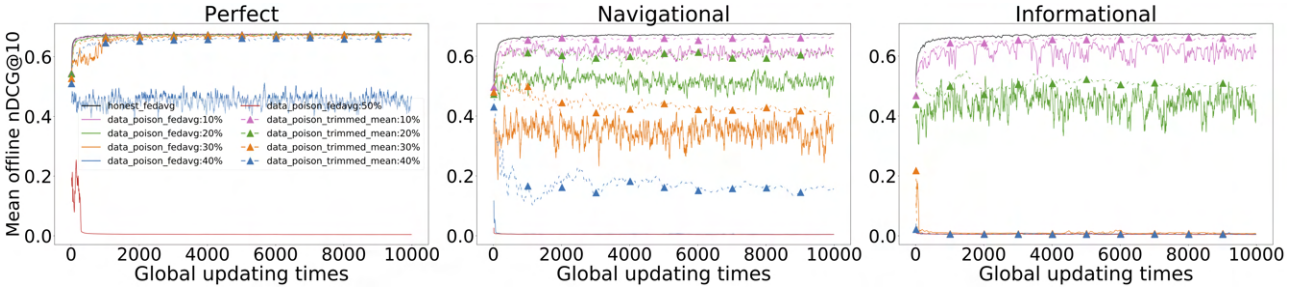
Figure 5.6: Offline performance (nDCG@10) for Yahoo under data poisoning attack and defense strategies, simulated with three benign instantiations of click model and different percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$; results averaged across all dataset splits and experimental runs.



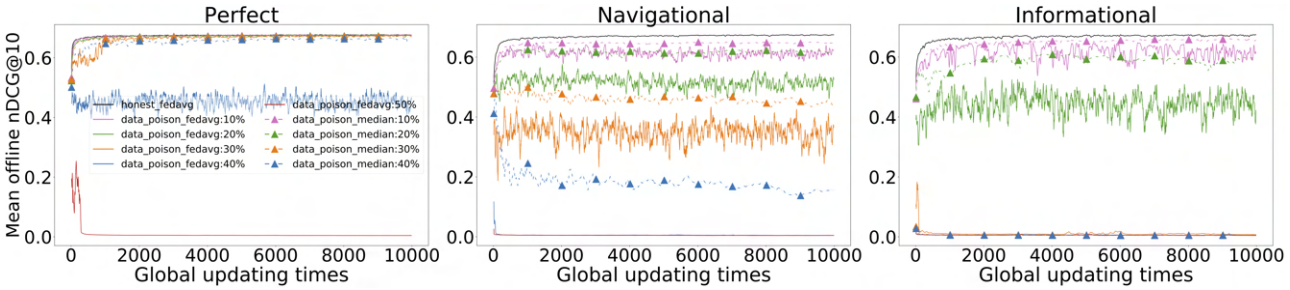
(a) Istella-S (Krum)



(b) Istella-S (Multi-Krum)



(c) Istella-S (Trimmed Mean)

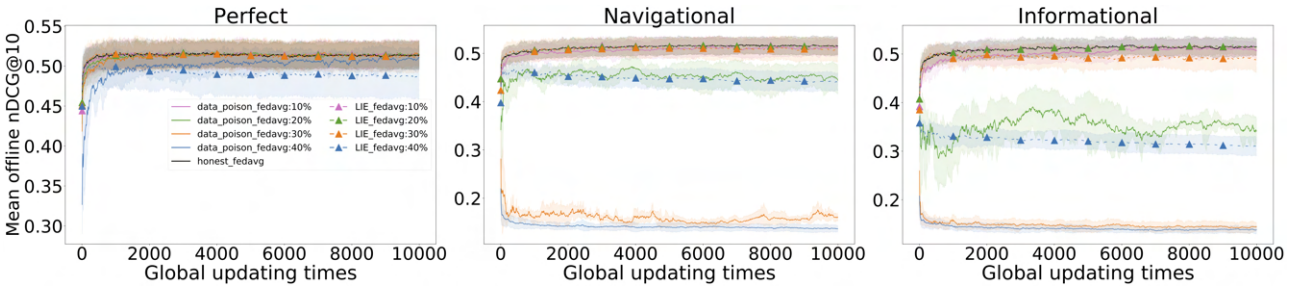


(d) Istella-S (Median)

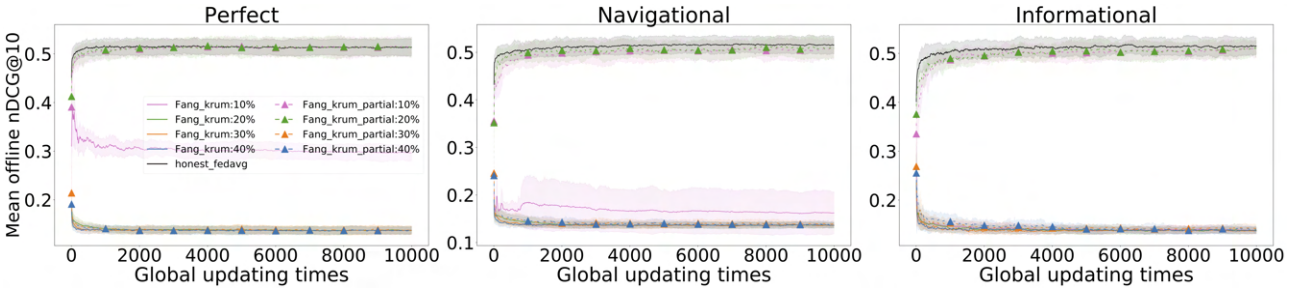
Figure 5.7: Offline performance (nDCG@10) for Istella-S under data poisoning attack and defense strategies, simulated with three benign instantiations of click model and different percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$; results averaged across all dataset splits and experimental runs.

5.9.2 Results for Model Poisoning

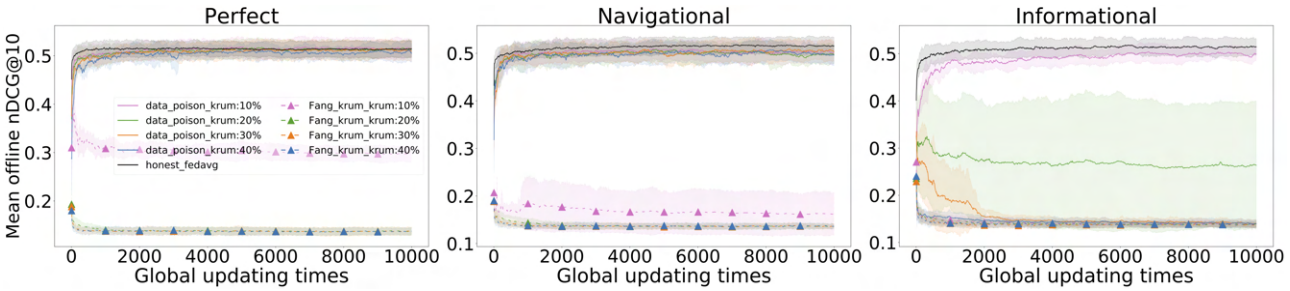
In this section, we report the results for model poisoning attacks on the remaining dataset. Results on MQ2007 is shown in Figure 5.8. Results on Yahoo is shown in Figure 5.9. Results on Istella-S is shown in Figure 5.10. Further analysis on results is detailed in Section 5.6.



(a) MQ2007 (LIE)

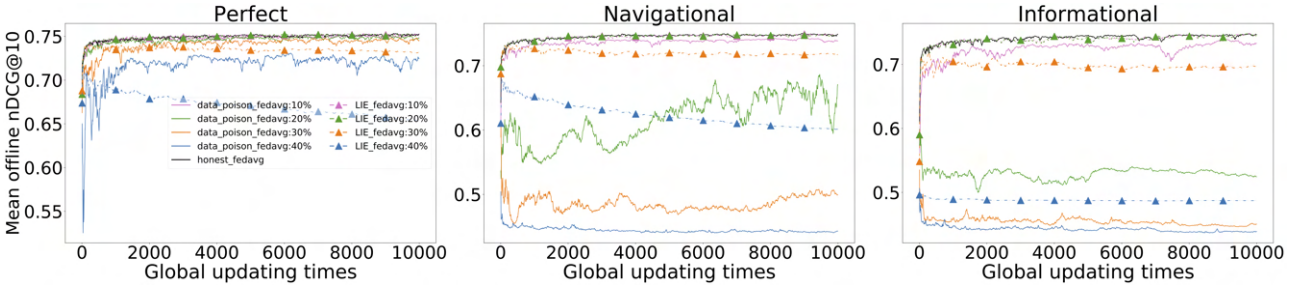


(b) MQ2007 (Fang's attack on Krum)

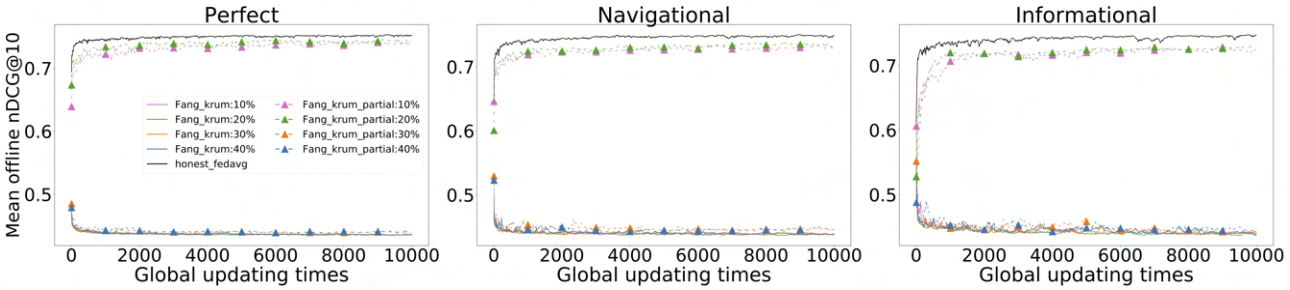


(c) MQ2007 (Fang's attack on Krum - full knowledge)

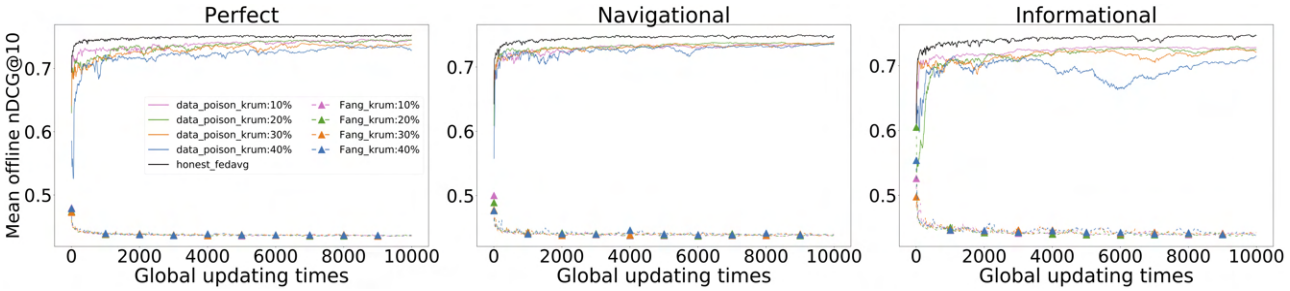
Figure 5.8: Offline performance (nDCG@10) for MQ2007 under model poisoning attacks, simulated using three benign instantiations of click model with percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$. The aggregation rule is FedAvg (for Figure 5.8a) and Krum (for Figure 5.8b-5.8c).



(a) Yahoo (LIE)

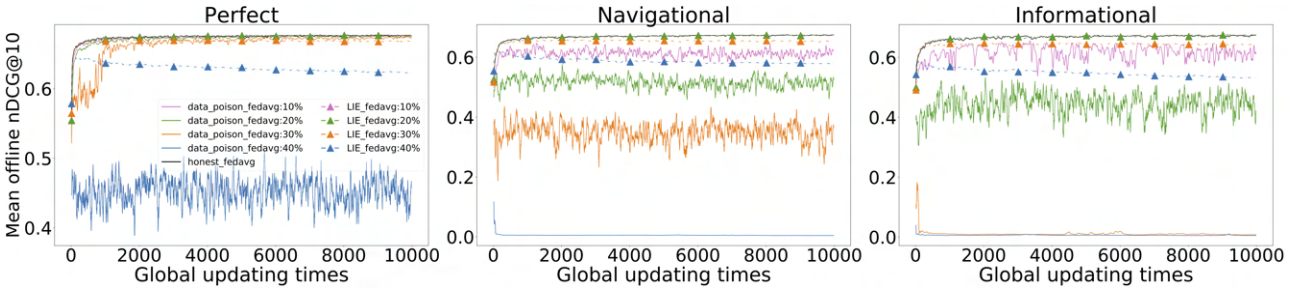


(b) Yahoo (Fang's attack on Krum)

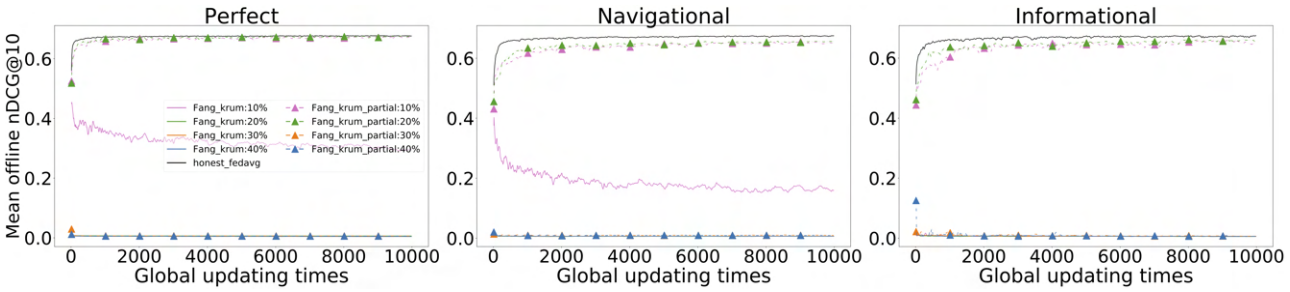


(c) Yahoo (Fang's attack on Krum - full knowledge)

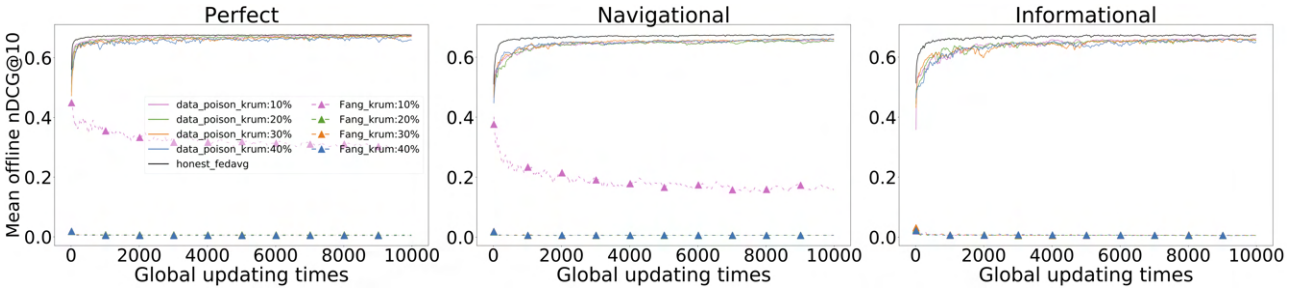
Figure 5.9: Offline performance (nDCG@10) for Yahoo under model poisoning attacks, simulated using three benign instantiations of click model with percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$. The aggregation rule is FedAvg (for Figure 5.9a) and Krum (for Figure 5.9b-5.9c).



(a) Istella-S (LIE)



(b) Istella-S (Fang's attack on Krum)

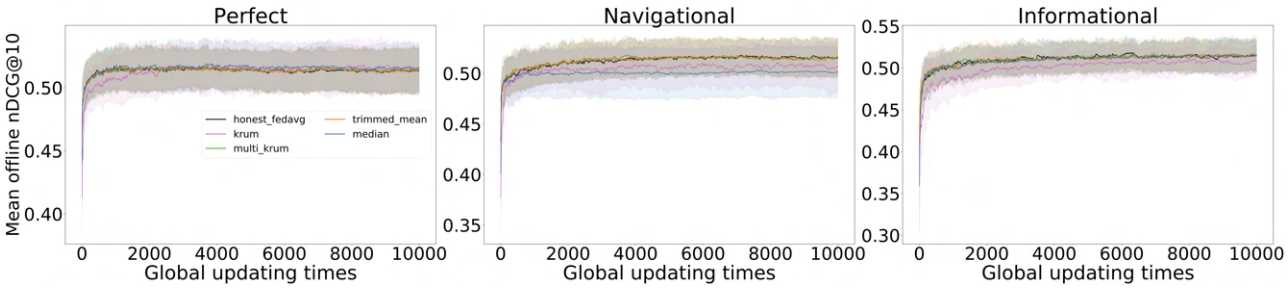


(c) Istella-S (Fang's attack on Krum - full knowledge)

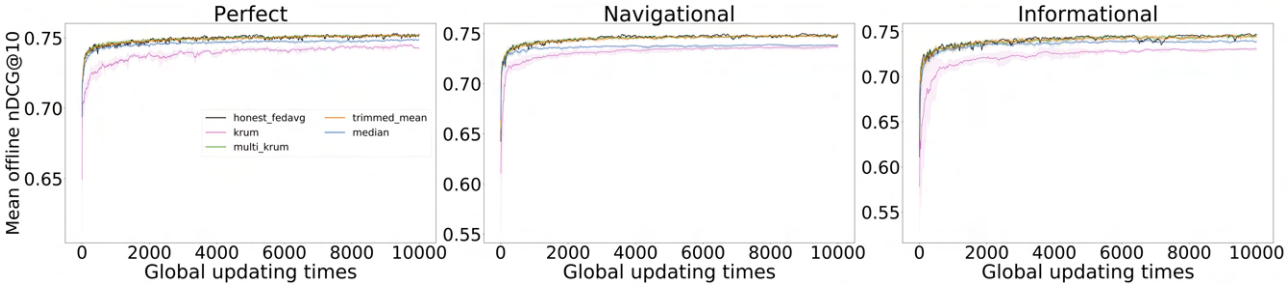
Figure 5.10: Offline performance (nDCG@10) for Istella-S under model poisoning attacks, simulated using three benign instantiations of click model with percentage of attackers equaling to $\{10\%, 20\%, 30\%, 40\%\}$. The aggregation rule is FedAvg (for Figure 5.10a) and Krum (for Figure 5.10b- 5.10c).

5.9.3 Results for Defense under No-Attack

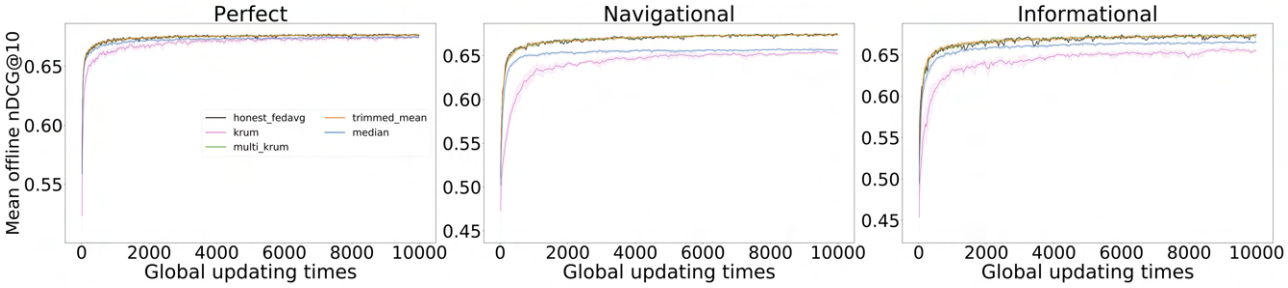
This section reports the results of four Byzantine aggregation rules with no attacker involving. Further analysis on the results is detailed in Section 5.7.



(a) MQ2007 (benign clients)



(b) Yahoo (benign clients)



(c) Istella-S (benign clients)

Figure 5.11: Offline performance (nDCG@10) of FOLTR system when no attack is present but defense strategies are deployed; results averaged across all dataset splits and experimental runs.

The following publication has been incorporated as Chapter 6.

1. [5] **Shuyi Wang**, Bing Liu, and Guido Zuccon, How to Forget Clients in Federated Online Learning to Rank?, *In European Conference on Information Retrieval (ECIR)*, pages 105–121, 2024

Chapter 6

Unlearning for Federated Online Learning to Rank

In this chapter, we study another direction for FOLTR, that is, a considerate federated learning system should consider the possibility for clients to leave the federation and request the contributions of their data erased [37, 102, 103]. This possibility – dubbed the *right to be forgotten* – is contemplated in modern data protection legislation, such as in the General Data Protection Regulation (GDPR) emanated by the European Union [135]. However, the design of existing FOLTR systems is defective as lacking an *unlearning* mechanism to forget certain users' contributions. An effective and efficient unlearning mechanism is not straightforward to design. A naive way is to ask all remaining clients to re-execute the training of ranker from scratch. This carries implications in terms of disruption of service and comes with large computational costs, even if the update was done in an offline manner (counterfactually on log data stored in each client), rather than in an online manner (which in turns is impractical as it needs the users to interact again with the search results). Therefore, a more reasonable unlearning mechanism for FOLTR is necessary, but has not been studied. In this chapter, we aim to fill this gap and provide an initial investigation of unlearning mechanisms for FOLTR. Specifically, we investigate the following research questions:

RQ4: How to effectively and efficiently forget client data from the trained ranker when a client requests to exit the FOLTR system?

RQ4.1: How to evaluate the effectiveness of unlearning?

RQ4.2: How to efficiently forget the client without requiring retraining from scratch?

The research questions align with two main challenges that we strive to overcome. The first challenge is how to *efficiently* unlearn without requiring an unreasonable amount of additional computation. Also, the obtained new ranker is expected to have comparable effectiveness to the one retrained from scratch. The second challenge is how to evaluate the *effectiveness* of an unlearning method. In a FOLTR system with many clients, the effect on ranker's effectiveness of removing a client can be

Table 6.1: Notation used in this chapter.

Symbol	Description
c_i	client i in the FOLTR system
c^*	the unlearned client that requested removal from the FOLTR system
n_i	number of local updates for client i, before unlearning takes place
n_i'	number of local updates for client i, during the unlearning process
T	global update rounds before the unlearning request
$T_{unlearn}$	global update rounds for federated unlearning, also equals to number of stored local updates for unlearning
Δt	interval of time between stored local updates
M_i^{local}	local ranking model of client i before unlearning
ΔM_i	local update of client i in federated learning
$\Delta M_i^{unlearn}$	local update of client i in federated unlearning
ΔM_i^{mal}	compromised local update of the client to be unlearned (client c^*)
z	parameter for poisoning attack (Section 6.1.4)

marginal, or even unnoticeable. A natural question from a user leaving the federation is: how can it be proven that the impact of my data on the ranker has been erased? An adequate evaluation method is then required to verify whether an unlearning process is effective in forgetting.

To address these challenges and facilitate future studies, we build the first benchmark for unlearning in FOLTR. To answer RQ4.2, we adapt an effective unlearning technique emerging from the general federated machine unlearning field (FedEraser [37]) to the context of FOLTR with adaptation into online training. In our method, some historical updates are stored in the local devices and re-used to help retrain a new ranker with much less additional computation cost. To address RQ4.1, for evaluation, we adopt a poisoning attack method [4] to magnify and control the effect of the client leaving the federation. Through extensive empirical experimentation across four learning-to-rank datasets, we study the effectiveness and efficiency of the unlearning method and the factors influencing its performance. The utility of our evaluation method is also verified.

6.1 Methodology

6.1.1 Preliminary

Table 6.1 summarises the main notations used in this chapter. In OLTR, for a certain query q and its candidate documents $\mathbf{D}$, a ranking model M is used to compute a relevance score for each candidate document $d \in \mathbf{D}$. The search engine displays the documents according to their scores in descending order, and collects user's interactions with the result page. In a centralised setting, M is iteratively trained on the server based on the features of each query-document pair and collected interaction data.

On the contrary, in FOLTR, a global ranking model M^{global} is initialised in a central server and distributed to each client. At training step t, each client c_i holds a local ranking model M_i^{local} (received from the central server) and trains the local model with local data (queries, documents,

interactions) for n_i local updating times. After the local training phase[1], each client sends its local update $\Delta M_{i,t} = M_i^{local} - M_t^{global}$ to a central server, that aggregates the updates from all clients to generate an updated global model. The most common aggregation rule is FedAvg [7] which updates the global model using the weighted average of all local updates:

$$M_{t+1}^{global} = M_t^{global} + \sum_i \frac{n_i}{\sum n_i} \Delta M_{i,t} \tag{6.1}$$

The newly-updated global model will be broadcast to each client and replace each local M_i^{local}. The whole process is repeated continuously.

6.1.2 Unlearning in FOLTR

We next illustrate the unlearning process of FedEraser [37], including how we adapted to doing unlearning for a FOLTR system.

Suppose that, during the FOLTR process and after T global update rounds, a client c^* requests to leave the federation and remove all the contributed local updates. Every Δt global update rounds (i.e. at rounds $\{1, 1+\Delta t, 1+2\Delta t, ...\}$ etc.), we instruct each client to store their local updates $\Delta M_{i,t}$. The number of local updates stored by each client should then be $T_{unlearn} = [\frac{T}{\Delta t}]$.

The unlearning process takes place as outlined below, in which steps (2)-(4) are performed iteratively and the iterations correspond to the global update rounds $\{1, 1+\Delta t, 1+2\Delta t, ...\}$ in the original FOLTR process:

1. The global ranking model is initialized in the same way as in the original FOLTR process, and is passed to the clients.

2. Then, each client c_i but c^* (which left the federation), updates the local model by n_i' steps ($n_i' < n_i$, the local step in unlearning n_i' is by design smaller than that before unlearning, i.e. n_i).

3. Each client calibrates the local update $\Delta M_{i,t}'$ using the stored historical local update $\Delta M_{i,t}$ according to:

$$\Delta M_{i,t}^{unlearn} = ||\Delta M_{i,t}|| \frac{\Delta M_{i,t}'}{||\Delta M_{i,t}'||} \tag{6.2}$$

   where $||\Delta M_{i,t}||$ indicates the step size of the global update and $\frac{\Delta M_{i,t}'}{||\Delta M_{i,t}'||}$ indicates the direction of the update ($||\cdot||$ is L_2-norm).

4. Each client c_i sends the calibrated updates $\Delta M_{i,t}^{unlearn}$ to the central server and the global model is updated using Equation 6.1.

After $T_{unlearn}$ times of global aggregation, the unlearned global model is obtained. In theory, the impact of client c^* is still imposed through $\Delta M_{i,t}$ in Eq. 6.2. But this impact is weakened since $\Delta M_{i,t}$ is used in fewer global updating times and the way of using $\Delta M_{i,t}$ is different from the original FOLTR process (Eq. 6.1). The actual impact will be evaluated as in Sec. 6.1.4.

[1] In our empirical study, we adapt FPDGD in which PDGD algorithm is used in the local training phase. Detailed method is specified in Section 3.2.1.

6.1.3 Efficiency Analysis

In total, the above unlearning approach only requires $n'_i \cdot T_{unlearn}$ local updates for each client c_i. Retraining the federated ranker from scratch instead requires $n_i \cdot T$ updates (n'_i is set to be less than n_i to reduce the local computation times). Thus, in terms of training efficiency compared to the baseline condition of retraining from scratch, the unlearning approach provides a reduction of $(n_i \cdot T)/(n'_i \cdot T_{unlearn}) = \frac{n_i}{n'_i} \cdot \Delta t$ local updates for each client c_i. Along with a reduction in local training, the unlearning process also reduces the communications required between each client and the central server by $\frac{T}{T_{unlearn}} = \Delta t$ times. This reduced number of updates comes at the expense of some extra space required to store the $T_{unlearn}$ updates[2]. For each client, the storage space cost is $[\frac{T}{\Delta t}] \cdot ||\Delta M_{i,t}||$, where $||\Delta M_{i,t}||$ indicates the L_2-norm of each local updates and it is determined by ranking model structure and the number of features.

6.1.4 Evaluating Unlearning

The impact of a typical client on the whole FOLTR system can be marginal as later shown in Figure 6.2. For the purpose of evaluating the unlearning approach, we need to magnify the effect of the client that is leaving the FOLTR system and make this effect relatively controllable. In previous research on machine unlearning and federated unlearning, the effectiveness of unlearning is verified by comparing the effectiveness of the model before and after unlearning [101]. This evaluation method comes with a drawback: effective unlearning is associated with a loss in model effectiveness – a situation that is undesirable and that would penalise methods that can unlearn and do not hurt model effectiveness. An alternative direction has been recently proposed: leverage poisoning or backdoor attacks to evaluate unlearning [91, 106, 107, 109, 110, 136].

Inspired by poisoning attacks methods for federated learning [89] and FOLTR systems [4] (introduced in Chapter 5) where malicious clients compromise the effectiveness of the trained models by poisoning the local training data or the model updates, we add noise to the local updates of the client to be unlearned. By doing so, we expect the unlearned clients to be clearly distinguishable from others. Due to the noise being injected, the original (i.e., before unlearning) global model performs worse to some extent compared to the model trained without the poisoned client (i.e., retrained from scratch). However, after unlearning, if the contribution of the poisoned client has been successfully removed, the overall effectiveness of the unlearned global model should improve, achieving similar effectiveness as if it was retrained from scratch. We instantiate this intuition by compromising the unlearned client's (c^*) local model after each local updating phase using:

$$M_{c^*}^{mal} = -z \cdot M_{c^*}^{local} \tag{6.3}$$

where $z > 0$ represents how much we compromise the local model, while the negative coefficient is added to change the local model to its opposite direction. Thus, the compromised local update for the

[2]These local updates would ideally be stored within each client, but they could be stored instead in the central server: this though would required extra communication cost to provide the local updates back to the clients when needed.

client to be unlearned is:

$$\Delta M_{c^*}^{mal} = -z \cdot \Delta M_{c^*}^{local} - (z+1) \cdot M_t^{global} \tag{6.4}$$

In the global updating phase before unlearning (Eq. 6.1), we replace $\Delta M_{c^*}^{local}$ with $\Delta M_{c^*}^{mal}$ for c^*. In our experiments, we set $z = 2$ as we found it sufficient in degrading the effectiveness of the global model. Note here our poisoning method does not have the burden of hiding from detection as in real poisoning attack scenarios and thus it is simple and its parameters can be tuned as per need.

A key tenet of this evaluation is that the ability to remove such a distinguishable client is equivalent to the ability to remove a much less unique client: we are unsure whether this assumption holds true, and we are not aware of relevant literature that clearly support this.

6.2 Experimental Setup

We next describe our experimental setup for unlearning in the context of a FOLTR system.

Datasets. We evaluate using 4 common learning-to-rank (LTR) datasets: MQ2007 [40], MSLR-WEB10K [40], Yahoo [45], and Istella-S [56]. Detailed descriptions about each dataset are discussed in Section 2.2.1. Our experimental results are averaged across all data splits.

User simulations. We follow the standard setup for user simulations in OLTR [2, 3, 39] by randomly selecting queries for a user and relying on the *Cascade Click Model* (CCM) click model for simulating user's clicks. Specifically, for each query, we limit the search engine result page (SERP) to 10 documents. User clicks on the displayed ranking list are generated based on the implementation of CCM click models detailed in Section 2.2.1.

Federated setup. We consider 10 clients participating in the FOLTR process; among these 10, one client requests to be unlearned. The original federated setup (before unlearning) involves 5 local updating steps ($n_i = 5$) among all participants and 10,000 global updating steps (T). We assume the client requests to leave the federation at global step $T = 10,000$. During the original training in our FOLTR experiments (i.e., before the unlearning request is proposed), each client holds a copy of the current ranker and updates the local ranker by issuing $n_i = 5$ queries along with the respective click responses. After the local updating finishes, the central server will receive the updated ranker from each client and aggregate all local messages to update the global one. At each Δt global steps (i.e. at rounds $\{1, 1+\Delta t, 1+2\Delta t, ...\}$ etc.), each client will also keep a copy of their local ranker update to the local device for the calibration purpose in the unlearning process. In our evaluation of unlearning (results in Figure 6.3), we set $\Delta t = 10$ thus the total number of stored local updates is $[\frac{T}{\Delta t}] = 1000$, equaling to the global steps in the unlearning process ($T_{unlearn}$). For further hyper-parameter analysis (results in Table 6.2), we set a wider value ranges of $\Delta t \in \{5, 10, 20\}$. The unlearning process follows the same federated setup, with $T_{unlearn}$ global steps, $n_i' \in \{1, 2, 3, 4\}$ local steps for the remaining 9 clients. We experiment with a linear model as the ranker with learning rate $\eta = 0.1$ and zero initialization, which we train with and rely on the configuration of the state-of-the-art FOLTR method, FPDGD [2].

Evaluation metric. As we limit each SERP to 10 documents, nDCG@10 is used for evaluating the overall offline effectiveness of the global ranker both before and after unlearning. Effectiveness is measured by averaging the nDCG scores of the global ranker on the queries in the held-out test dataset. This is in line with previous work on OLTR and FOLTR. Unlike previous work [2, 39, 53], online evaluation is not considered in this work, as we do not need to monitor user experience during model update. Instead, we focus on measuring the overall impact of the unlearned client (the "attacker" in Figure 6.2) comparing to other baselines, and the dynamic or final performance gain during the unlearning process (results in Figure 6.3 and Table 6.2) following our evaluation process specifically for unlearning (specified in Section 6.1.4).

6.3 Results and Analysis

6.3.1 Validation of Evaluation Methodology

The unlearning evaluation methodology is based on instantiating the unlearned client as a malicious actor that injects noise in the learning process. We start by investigating if our evaluation methodology could be effective in identifying whether the unlearning has happened. For this, we consider three configurations (visualized in Figure 6.1):

1. 9H-1M (green line): effectiveness obtained when one of the 10 clients is set to produce noisy updates (i.e. one client behaves maliciously).

2. 10H-0M (black line): effectiveness obtained when all 10 clients behave in an honest way, (i.e. no client is acting maliciously).

3. 9H-0M (pink l
above, without

Fig. 6.1 clarifies the relationship between these rankers – all rankers share the same common set of 9 honest clients, but they differ in the 10th client considered: a malicious client for 9H-1M, an honest client for 10H-0M, and no 10th client for 9H-0M.

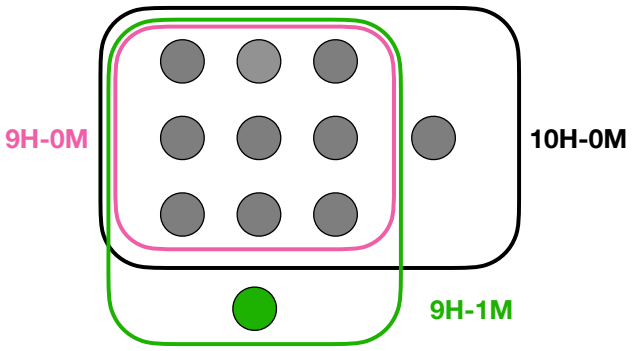
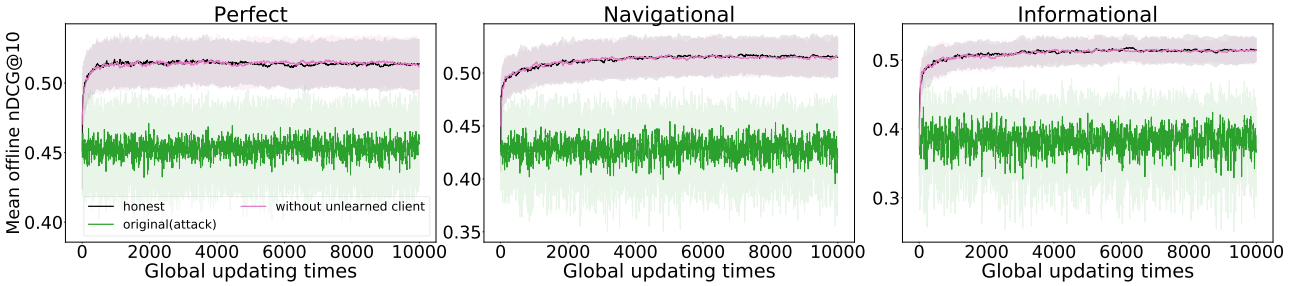
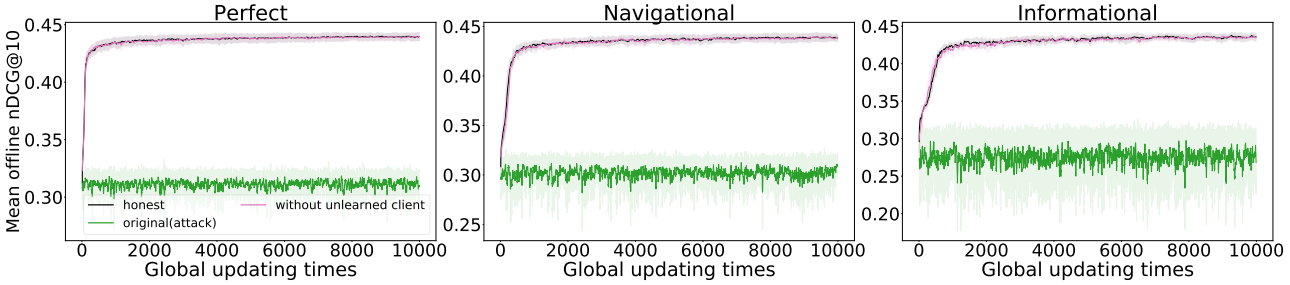


Figure 6.1: Relationships between FOLTR configurations: 9H-1M (green line), 10H-0M (black), 9H-0M (pink). Circles are clients.

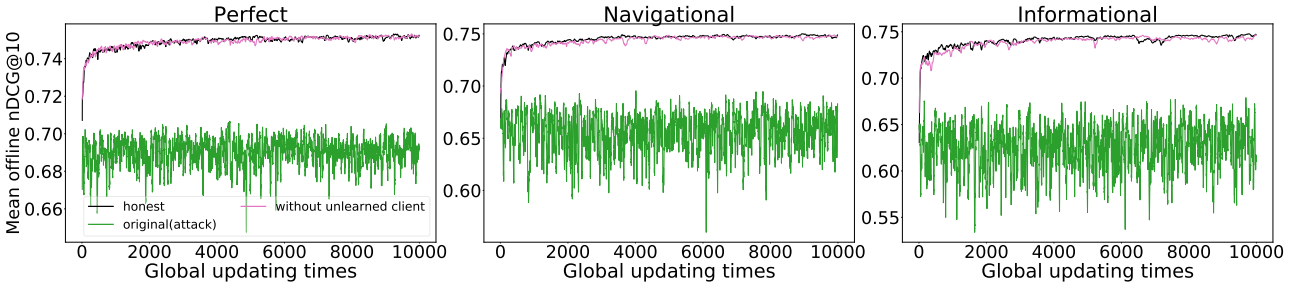
Fig. 6.2 reports the results obtained by these three conditions across three click modes on the four datasets. The results highlight that it is the addition of the malicious client (i.e. client c^* that will be the target of the unlearning) that sensibly reduces the effectiveness of the ranker. Note that the 9H-0M is the ranker one would obtain if the



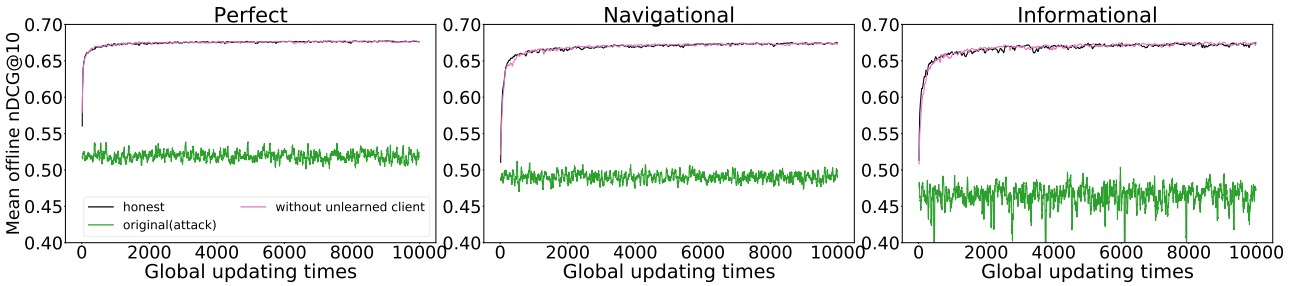
(a) MQ2007



(b) MSLR-WEB10K



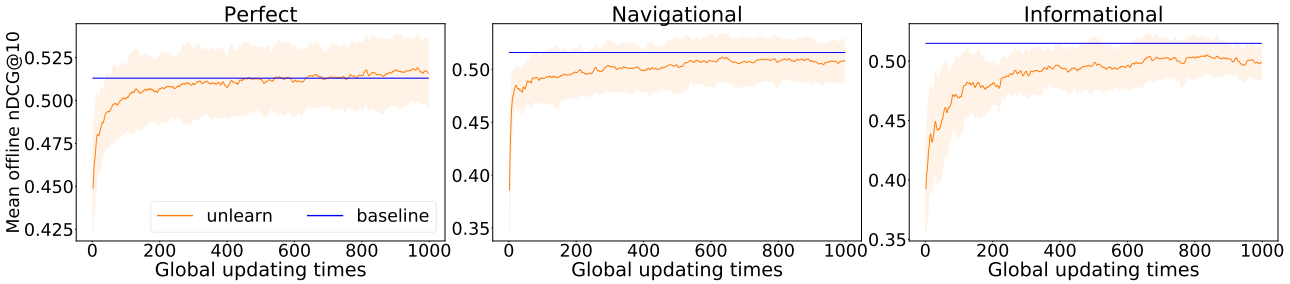
(c) Yahoo



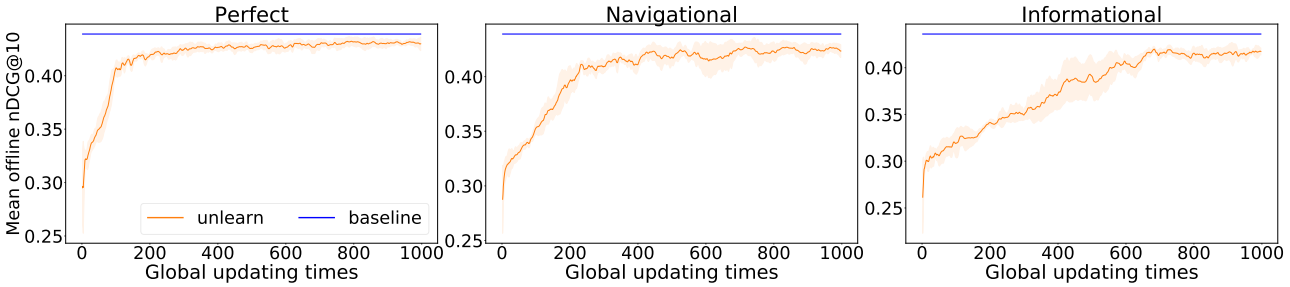
(d) Istella-S

Figure 6.2: Offline effectiveness (nDCG@10) obtained under the 9H-1M (green line), 10H-0M (black line), 9H-0M (pink line) FOLTR configurations with three click modes (*Perfect, Navigational, Informational*). Results are averaged across all dataset splits and experimental runs. These results motivate the use of the evaluation methodology based on the malicious client to evaluate the effectiveness of unlearning.

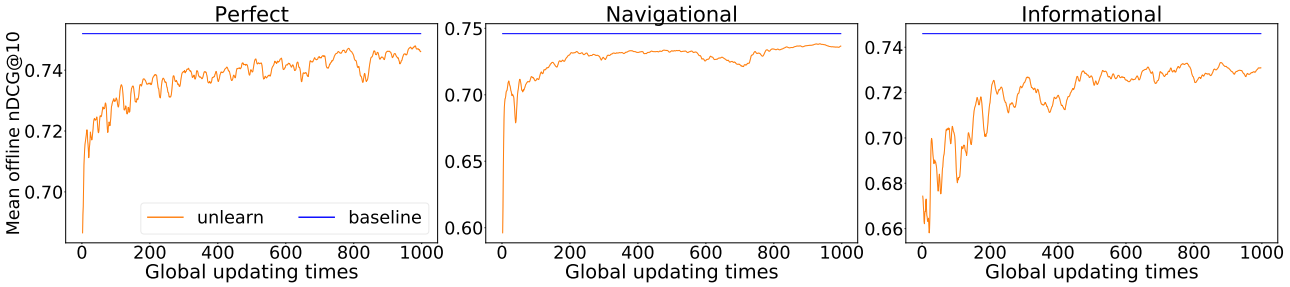
unlearning process was implemented as a federated re-training of the ranker from scratch by only considering the 9 clients remaining after the removal of client c^*. Comparing this ranker with the 10H-0M, we further highlight the need for evaluation based on the malicious client. In fact, 10H-0M and 9H-0M only consider honest clients, but in the 9H-0M one of these clients has been removed – but the effectiveness of two rankers is indistinguishable.



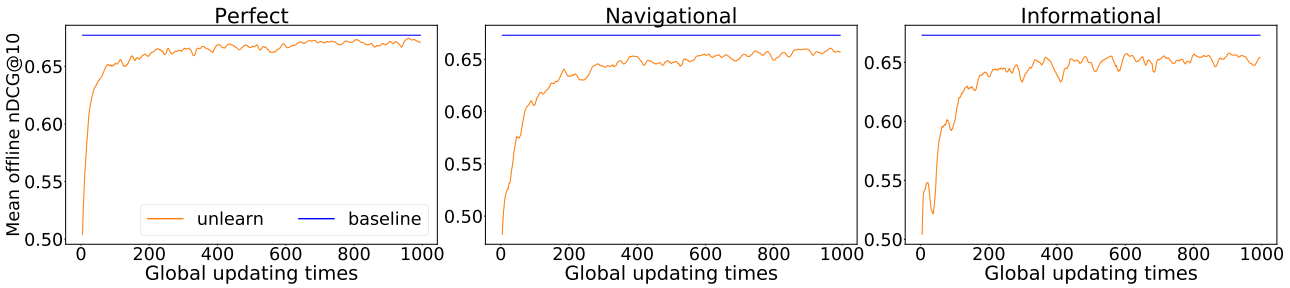
(a) MQ2007



(b) MSLR-WEB10K



(c) Yahoo



(d) Istella-S

Figure 6.3: Comparison between the offline effectiveness (nDCG@10) after the unlearning method is applied (ranker $\mathcal{U}$(9H-1M) denoted as "unlearn") and the ranker is retrained from scratch after client c^* is removed (ranker 9H-0M denoted as "baseline"). For the unlearning process, we set $n'_i = 3$ with $\Delta t = 10$ and show the evaluation values across all global steps. For the baseline setup, we only show the final nDCG@10 score after retraining finishes.

6.3.2 Effectiveness of Unlearning

We now investigate the effectiveness of the unlearning method. For this, we consider ranker 9H-1M and we perform unlearning to remove client c^*, which is the malicious client; this leads to ranker $\mathcal{U}$(9H-1M). Table 6.2 reports the effectiveness of the global model after unlearning has taken place and under different settings of hyper-parameters n'_i and Δt. A more detailed analysis of the impact of n'_i and Δt is presented in Section 6.3.3.

In Fig. 6.3, we report the offline effectiveness on four datasets obtained by the unlearning mechanism during the $[\frac{T}{\Delta t}] = 1,000$ global update times, where we set $\Delta t = 10$ and $n'_i = 3$. Compared to the original model 9H-1M[3], the unlearned model $\mathcal{U}$(9H-1M) achieves better effectiveness and it gradually converges towards the effectiveness of the 9H-0M model, showing that the unlearned model is able to successfully remove the impact of the unlearned client c^*.

6.3.3 Hyper-parameters Analysis

Next, we study the sensitivity of the unlearning method to its two hyper-parameters: the number of local updates for unlearning $n'_i \in \{1,2,3,4\}$ and interval of time between stored updates $\Delta t \in \{5,10,20\}$. We report the results of this analysis in Table 6.2 with the final nDCG@10 score after the unlearning (or retraining). The effectiveness of ranker 9H-0H, i.e. the global model re-trained from scratch for $T = 10,000$ iterations and without client c^*, represents the baseline condition.

Impact of n'_i. By design, lower values of n'_i lead to higher time savings as the time required by each local update is similar. However, lower values of n'_i may mean there is insufficient training data in each iteration, and this can lead to lower ranking effectiveness. Our experimental findings display this trade-off with relative higher effectiveness obtained when more local updates (n'_i) are used during the unlearning process.

Impact of Δt. Larger values of Δt correspond to less global updates needed for unlearning. In fact, the required global updating time is$[\frac{T}{\Delta t}]$, where in our experiments $T = 10,000$. Table 6.2 shows that, in most cases, the model for which unlearning has taken place delivers higher ranking effectiveness for small values of Δt. This higher effectiveness, however, comes at the cost of extra time required by the unlearning process.

[3]The effectiveness of 9H-1M is not shown in Figure 6.3 for clarity. The reader can cross reference Figure 6.3 with Figure 6.2, which instead contains the effectiveness of 9H-1M.

Table 6.2: Offline effectiveness (nDCG@10) of (1) the ranker trained from scratch after client c^* has been removed (9H-0M), and (2) the ranker for which unlearning is performed with our method ($\mathcal{U}$(9H-1M)). Effectiveness is analysed with respect to the hyper-parameters $n'_i \in \{1,2,3,4\}$ and $\Delta t \in \{5,10,20\}$, under three different click models, and averaged across dataset splits. The best results are highlighted in boldface with superscripts denoting results for statistical significance study (paired Student's t-test with $p \leq 0.05$ with Bonferroni correction).

			Click Model: Perfect			Click Model: Navigational			Click Model: Informational		
Dataset	#	Ranker	$\Delta t = 5$	$\Delta t = 10$	$\Delta t = 20$	$\Delta t = 5$	$\Delta t = 10$	$\Delta t = 20$	$\Delta t = 5$	$\Delta t = 10$	$\Delta t = 20$
MQ2007		9H-0M	0.513	**0.513**	**0.513**	**0.516**	**0.516**	**0.516**	**0.515**	**0.515**	**0.515**
	a	$\mathcal{U}$(9H-1M), $n'_i = 4$	0.514	0.513	0.511	0.508	0.507	0.501	0.506	0.502	0.492
	b	$\mathcal{U}$(9H-1M), $n'_i = 3$	0.514	0.513	0.507	0.510	0.504	0.499	0.505	0.498	0.491
	c	$\mathcal{U}$(9H-1M), $n'_i = 2$	**0.515**	0.510	0.507	0.505	0.499	0.499	0.502	0.497	0.483
	d	$\mathcal{U}$(9H-1M), $n'_i = 1$	0.511	0.506	0.501	0.504	0.498	0.496	0.489	0.491	0.484
MSLR-WEB10K		9H-0M	**0.439**	**0.439**^{d}	**0.439**^{cd}	**0.439**^{d}	**0.439**^{cd}	**0.439**^{abcd}	**0.436**^{cd}	**0.436**^{abcd}	**0.436**^{abcd}
	a	$\mathcal{U}$(9H-1M), $n'_i = 4$	0.436	0.433	0.429	0.433	0.428	0.422	0.429	0.413	0.389
	b	$\mathcal{U}$(9H-1M), $n'_i = 3$	0.435	0.432	0.427	0.429	0.428	0.414	0.424	0.412	0.361
	c	$\mathcal{U}$(9H-1M), $n'_i = 2$	0.432	0.431	0.423	0.426	0.421	0.389	0.420	0.386	0.348
	d	$\mathcal{U}$(9H-1M), $n'_i = 1$	0.430	0.422	0.414	0.418	0.382	0.344	0.393	0.343	0.332
Yahoo		9H-0M	**0.752**	**0.752**^{bcd}	**0.752**^{abcd}	**0.746**^{bd}	**0.746**^{bcd}	**0.746**^{abcd}	**0.746**^{abcd}	**0.746**^{abcd}	**0.746**^{abcd}
	a	$\mathcal{U}$(9H-1M), $n'_i = 4$	0.748	0.746	0.741	0.743	0.743	0.734	0.738	0.725	0.726
	b	$\mathcal{U}$(9H-1M), $n'_i = 3$	0.748	0.740	0.742	0.738	0.739	0.735	0.736	0.725	0.721
	c	$\mathcal{U}$(9H-1M), $n'_i = 2$	0.746	0.746	0.742	0.738	0.735	0.733	0.732	0.725	0.715
	d	$\mathcal{U}$(9H-1M), $n'_i = 1$	0.746	0.739	0.732	0.734	0.726	0.725	0.730	0.720	0.713
Istella-s		9H-0M	**0.677**^{cd}	**0.677**^{cd}	**0.677**^{abcd}	**0.673**^{bcd}	**0.673**^{abcd}	**0.673**^{abcd}	**0.673**^{abcd}	**0.673**^{abcd}	**0.673**^{abcd}
	a	$\mathcal{U}$(9H-1M), $n'_i = 4$	0.673	0.672	0.664	0.665	0.659	0.650	0.663	0.655	0.654
	b	$\mathcal{U}$(9H-1M), $n'_i = 3$	0.673	0.670	0.663	0.664	0.662	0.643	0.664	0.658	0.639
	c	$\mathcal{U}$(9H-1M), $n'_i = 2$	0.669	0.670	0.663	0.661	0.652	0.637	0.659	0.653	0.635
	d	$\mathcal{U}$(9H-1M), $n'_i = 1$	0.669	0.662	0.657	0.653	0.637	0.604	0.649	0.641	0.600

6.4 Conclusion

This chapter is the first study that investigates unlearning in Federated Online Learning to Rank. For this, we adapt the FedEraser method [37], developed for general federated learning problems, to the unique context of federated online learning to rank, where rankers are learned in an online manner on implicit interactions (e.g. clicks). We further modify the method to leverage stored historic local updates to guide and accelerate the process of unlearning. To evaluate the effectiveness of the unlearning method, we adapt the idea of poisoning attacks to the context of determining whether the contributions of a client to be unlearned made to the ranker are effectively erased from the ranker itself. Experimental results on four popular LTR datasets show both the effectiveness and the efficiency of the unlearning method. In particular: (1) the search effectiveness of the global model once unlearning takes place converges to the effectiveness of the ranker if the removed client did not take part in the FOLTR system from the start, (2) the contributions made to the ranker by the unlearned client are effectively removed, and (3) less local and global steps are required by unlearning compared to retraining the model from scratch.

Code, experiment scripts and experimental results are provided at `https://github.com/ielab/2024-ECIR-foltr-unlearning`.

Chapter 7

Conclusion

In this thesis, we explored a new training paradigm for Online Learning to Rank that addresses data privacy: Federated Online Learning to Rank (FOLTR). Our study focused on four critical aspects of FOLTR: effectiveness, robustness, security and unlearning. To guide our investigation, four research questions were proposed:

RQ1: How to design an effective and generalisable FOLTR method that can achieve competitive performance compared to state-of-the-art OLTR methods? (*effectiveness*, Chapter 3)

RQ2: Are FOLTR methods robust to non-IID data among clients? (*robustness*, Chapter 4)

RQ3: Are FOLTR methods secure in the face of potential risks brought by malicious participants? (*security*, Chapter 5)

RQ4: How to effectively and efficiently forget client data from the trained ranker when a client requests to exit the FOLTR system? (*unlearning*, Chapter 6)

In this final chapter, we will summarise our findings and contributions for each RQ, along with the limitations and future work. Finally, we outline the broader future directions set by this thesis.

7.1 RQ1: Effective and Privacy-Preserving FOLTR

7.1.1 Overview

Federated learning approaches to search have only recently emerged. In the area of OLTR, FOLtR-ES [27] was the first federated method. FOLtR-ES uses evolutionary strategies similar to those in genetic algorithms to make client rankers explore the feature space, and a parametric privacy preserving mechanism to further anonymise the feedback signal that is shared with the central server by the clients. Although showing promising results, the empirical study of FOLtR-ES did not consider many of the datasets and evaluation metrics commonly used in OLTR studies. To further understand this method, we replicated and extended its study to encompass common OLTR evaluation practices. We also

compared its ranking effectiveness with the state-of-the-art OLTR method in the centralised setting, with the aim of investigating the limitations of this FOLTR method.

We found the performance of FOLtR-ES vary across datasets, lagging far behind the centralised OLTR method. To address this performance gap, we proposed FPDGD, which leverages the state-of-the-art Pairwise Differentiable Gradient Descent (PDGD) and adapts it to the Federated Averaging framework. For a strong privacy guarantee, we further introduced a noise-adding clipping technique based on the theory of differential privacy to be used in combination with FPDGD. To summarise, the main features of our method are: (1) we use PDGD as the core OLTR optimisation algorithm – this method is shown to perform well and converge faster than previous OLTR methods and is unbiased with respect to document pair preferences; (2) the FedAvg framework is easy to implement and it has been shown to work efficiently with a large number of clients [7]; (3) we use differential privacy to protect the communication of the local gradient updates between the clients and the server.

7.1.2 Findings

Our findings question whether FOLtR-ES is a mature method that can be considered in practice: its effectiveness largely varies across datasets, click types, ranker types and settings. Its performance is also far from that of current state-of-the-art OLTR, questioning whether the privacy guarantees offered by FOLtR-ES are not achieved by seriously undermining search effectiveness and user experience. This highlights that more work is needed to improve the performance of FOLTR methods so as to close the gap between privacy-oriented approaches and centralised approaches that do not consider user privacy. Overall, although FOLtR-ES has the merit of being the first FOLTR approach available, our findings suggest that it is far from being a solution that can be considered for use in practice, and more research is required for devising effective, privacy-aware, federated OLTR techniques.

Higher performance is observed in FPDGD. We demonstrate the effectiveness of FPDGD through empirical experiments comparing the method to its non-federated counterpart (PDGD) and the other only federated OLTR method, FOLtR-ES. Compared to the non-federated PDGD, our method converges slower when the same total interaction budget (i.e. number of queries) is considered. In particular, the difference is significant when differential privacy is considered – this is the price to pay for not sharing the clients' data with the central sever (federated setup) and for protecting the local gradient updates (differential privacy). When compared to FOLtR-ES, the proposed method showcases higher performance. In addition, FPDGD is more robust across different privacy guarantee requirements than FOLtR-ES: FPDGD is therefore more reliable for real-life applications.

7.1.3 Contributions

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

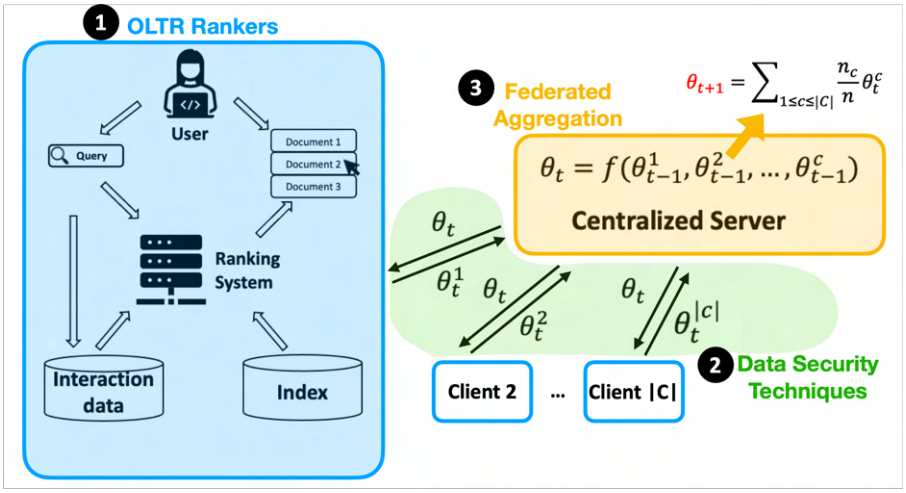


Figure 7.1: A high-level architecture and key components in FOLTR system.

compromising search performance. To this aim, we propose FPDGD which leverages a pairwise loss to address the bias from local implicit feedback thus boosts the effectiveness compared to the baseline. We further contribute an understanding on the performance of FPDGD compared to its centralised counterpart, where we compare them under same interactions budgets and same update budgets. Finally, we investigate the impact of differential privacy noise added to the updates sent to the central server. This privacy preserving mechanism is effective in protecting the gradient updates from privacy attacks [137]. That is, a malicious attacker that is able to observe the ranker model and gradient updates can potentially reconstruct the data that produced such an update [65], thus finally exposing user data.

7.1.4 Limitations and Future Work

To make FOLTR methods more generalisable, future work can look into creating a FOLTR benchmark where different combination of OLTR methods ❶, data security techniques ❷ (such as differential privacy [38], secure multi-party computation [138] and encryption techniques [139]), and federated aggregation algorithms ❸ (federated averaging [7] and its variants) are studied as shown in a high level architecture in Figure 7.1. In future study, methods for reducing computation and communication load can also be investigated through leveraging the decentralised attributes of federated learning while addressing the bias of implicit online feedback. In addition, there is a growing trend of unifying Online LTR and Counterfactual LTR. For example, Zhuang and Zuccon [50] proposed Counterfactual Online Learning to Rank which applies a form of counterfactual evaluation for updating OLTR rankers. Oosterhuis and de Rijke [140] linked OLTR to counterfactual LTR by proposing an intervention-aware estimator that has been shown effective for both counterfactual and online LTR. Thus, future work can investigate novel methods for the unification of Online LTR and Counterfactual LTR under the federated setting.

7.2 RQ2: FOLTR with non-IID Data

7.2.1 Overview

A factor influencing the performance of federated learning systems is the presence of bias in the data distribution across clients. In other words, the fact that clients may hold non-independent and identically distributed (non-IID) data. This non-IID issue poses significant challenges, as it can result in suboptimal model performance and reduced generalisability of the trained models [30]. Addressing this problem is essential for enhancing the robustness and effectiveness of federated learning systems. While FOLTR systems are on their own right a type of federated learning system, the presence and effect of non-IID data in FOLTR has not been studied, so as methods for addressing them. Due to the inherent complexity of users' searching behaviours, the formulation of non-IID data types for FOLTR is not clear. In particular, users may differ in preferences on documents, click behaviours, or frequency on issuing queries, skewing the local learning process towards their local distribution thus hinder the convergence of the global ranker. To this aim, we investigate the effect of non-IID data on FOLTR and explore if the existing methods in handling non-IID can also be adapted to FOLTR, thus chart directions for future work in this new and largely unexplored research area. Specifically, we put forward four different ways data specific to FOLTR could be distributed in a non-IID manner because of biases across clients on: document preferences (Type 1), document label distribution (Type 2), click preferences (Type 3), and data quantity (Type 4). Then, we study the impact of each setting on the performance of current state-of-the-art FOLTR approach.

7.2.2 Findings

Impact of non-IID data. We found that the presence of non-IID characteristics in the distribution of document preferences (Type 1) and specific cases of document labels (Type 2) have severe effects on the effectiveness of FPDGD. Conversely, if data is distributed across clients in a non-IID manner with regard to click preferences (Type 3) or data quantity (Type 4), no significant effects on the quality of FPDGD are observed. These findings contribute an understanding of under which data distributions it is feasible to use FOLTR and when it is not.

Calling for FOLTR methods to address non-IID issues. Our findings chart directions to future work on non-IID data in FOLTR concerning the creation of techniques that provide remedies to Type 1 and 2, while deeming solutions for Type 3 and 4 data less critical. Importantly, we show that existing solutions employed in general federated learning to mitigate the non-IID data problem do not apply to the FOLTR setting, despite some of these non-IID cases (and especially Type 1) being likely to occur across many FOLTR systems.

7.2.3 Contributions

This research sheds light on the factors that need to be considered when devising and deploying FOLTR methods. We detail the experimental conditions for simulating non-IID data in FOLTR, paving the way

for the development and adaptation to OLTR of existing and new methods for dealing with non-IID data. With this regard, we also show how some of the methods proposed in the federated learning literature to deal with non-IID data could be cast in the FOLTR framework and the gaps that still exist in effectively addressing non-IID data in FOLTR. This allows us to unveil new research gaps that, we argue, future research in FOLTR should consider. This is an important contribution to the current state of FOLTR because, for FOLTR systems to be deployed, the factors affecting their performance, including the impact of non-IID data, need to be thoroughly understood and considered. Our work has also inspired a recent study on investigating non-IID data for federated learning to rank with tree-based rankers [141].

7.2.4 Limitations and Future Work

Privacy should be a high priority when dealing with non-IID data. Our analysis found that only the data-sharing technique [30] could address to significant extents Type 2 non-IID data. However, this and similar methods, although performing well, require prior knowledge about the users' local data distributions – and thus require users to share private data, largely defeating the purpose of federated learning. We note that recent work has considered the sharing of synthetic, rather than real, data [76]. In such a setting, real data would be used by each client to generate synthetic data, and the synthetic data only would be shared in the federation. However, we could not find evidence of the loss in effectiveness associated with the use of synthetic rather than real data in the data-sharing scheme. Furthermore, it is unclear what the privacy guarantees are in such a synthetic data sharing scheme. Specifically, we wonder whether the use of synthetic data could jeopardise privacy as this synthetic data is generated from the real data: thus analysis of the synthetic data may reveal key aspects of and information contained in the real data. Thus, how to guarantee user's privacy needs when designing effective FOLTR algorithms on non-IID data is still an open question.

Real-world datasets and benchmarks for FOLTR with non-IID data are needed. The experiments put forward in this study to substantiate our views on the non-IID data problem in FOLTR are based on simulations. While simulations are prevalent in information retrieval and especially in its evaluation [142–146], a key aspect we had to simulate was the nature of the non-IID data, including their distributions. On one hand, this allows us to carefully control the experiments; on the other it limits the generalisability of the findings to real non-IID data that may occur in FOLTR settings. Thus, another future direction is building FOLTR benchmark datasets that provide standard simulations on real-world non-IID scenarios as well as standard hyper-parameter settings so that future FOLTR algorithms can be fairly studied.

Further investigation of online learning to rank algorithm is needed. The local ranking model of this work is updated based on Pairwise Differentiable Gradient Descent (PDGD) algorithm. A further discussion about PDGD by Oosterhuis and de Rijke [140], which links OLTR to counterfactual LTR, found that PDGD is ineffective in some situation where it is not run in a fully online mode. In other words, PDGD can have detrimental performance if the model updates are based on interactions

gathered by a different model than to which they are applied. In this thesis, while the local ranker is updated based on online interactions, the global ranker is updated through federated aggregation and replaces the local ranker. This process dose not follow exactly the same online learning setup as PDGD in a centralised setting [39, 140]. Thus, to fully understand if the cause of this non-IID issue is not running PDGD in an online and centralised setting or from OLTR in general, future work can investigate FOLTR under non-IID setting through considering other OLTR algorithms to enhance the completeness of this study and further explore PDGD running in the federated mode.

7.3 RQ3: FOLTR under Poisoning Attacks

7.3.1 Overview

In this study, we investigated potential risk posed by malicious clients and solutions for defending against them with focus on untargeted poisoning attack. For the untargeted attack, we implement a *data poisoning* method that compromises the local data to impact the training of the model, and two *model poisoning* methods that directly corrupt the local model updates. As for the defense method, we implement four Byzantine-robust aggregation rules (Krum, Multi-Krum, Trimmed Mean and Median) to safeguard against such attacks. These defense mechanisms rely on statistical techniques to identify outliers among the received weights and subsequently exclude them during the aggregation process.

7.3.2 Findings

Based on the empirical results, we identify the following key findings:

It is harder to attack FOLTR systems with clients that exhibit low variance between local updates. Under the *perfect* click type simulation, the users provide the most accurate and lowest noisy feedback. In general, this click type is more difficult to attack compared to the other two click models under both data and model poisoning methods, except in specific instances when employing Fang's attack [86] under the full knowledge assumption. To successfully attack a FOLTR system when *perfect* click feedback is present, a larger number of attackers is required due to the relatively low variance between local updates. As a result, more clients must be compromised by the attacker, otherwise the attack is more likely to be mitigated by robust aggregation rules.

The attack is more successful when more information about the benign users' local updates is known. Among all attacking strategies studied, Fang's attack [86] with full knowledge emerged as the most successful in diminishing the performance of the global model, though some exceptions were observed in the noisy *informational* click scenario. When there were more malicious clients (i.e. 30% or 40% of the total number of clients), Fang's attack with partial knowledge is just as effective as with full knowledge. This indicates that model poisoning is more effective than data poisoning. Furthermore, when defense measures were implemented, Fang's attack demonstrated greater success against Krum and Multi-Krum aggregation rules in comparison to Trimmed Mean and Median.

Defense strategies negatively impact ranking if no attack is in place. Although Krum has proven effective in countering data poisoning and Trimmed Mean in defending against Fang's attack, deploying these aggregation rules should be exercised with caution as they result in an overall decrease in search performance if the system is not under attack. The selection of these defense mechanisms should thus be carefully considered, taking into account the specific context and risk of potential attacks to strike the right balance between security and search effectiveness.

7.3.3 Contributions

This is the first study that systematically analysed the threats brought by untargeted poisoning attacks to FOLTR systems. It demonstrates the effectiveness and associated drawbacks of existing defense methods on mitigating the impact of malicious adversaries under FOLTR system. Through extensive empirical experiments, we (1) investigate the vulnerability of FOLTR systems to untargeted poisoning attacks, and show under which conditions poisoning attacks can represent a real threat to FOLTR systems; and (2) demonstrate the effectiveness of defense strategies, importantly revealing the presence of issues with defense strategies if applied to FOLTR systems for which an attack is not in place.

7.3.4 Limitations and Future Work

We did not consider the setup of a cross-device FOLTR system, where many clients are involved in the federation: this is representative of a web-scale federation. For example, an email service provider aims to improve its email search functionality to deliver more relevant search results. Due to the private attribute of user's emails and search interactions, an on-device FOLTR system can be leveraged, where the retrieval and ranking is done locally and a ranking model is collaboratively learned with the help of local model updates. As more clients are involved, the portion of users that can be controlled by the malicious attacker is relatively low, making the attack more challenging. Under this circumstance, more sophisticated attacking strategies are required to achieve prominent poisoning impact. Future works need to address the unique challenges under the context of FOLTR: (1) as the federated environment limits the adversary's prior knowledge to its local information (i.e., malicious user's data and local models) and the global model, the randomness of user's queries and clicks might obstruct the success of attacking; and (2) the deployed defense methods should ensure the global ranker maintains high ranking performance, avoiding negative impact on user experience. In addition, to make the study more comprehensive thus strengthening the overall depth and impact of the security evaluation, future works can investigate more critical attacks, such as the DYN-OPT [87], and key defense mechanisms, including Bulyan [128] and FLDetector [147], among others.

7.4 RQ4: Unlearning for FOLTR

7.4.1 Overview

Data protection legislation like the European Union's General Data Protection Regulation (GDPR) establishes the *right to be forgotten*: a user (client) can request contributions made using their data to be removed from learned models. We study how to remove the contributions made by a client participating into a FOLTR system, in other words, how to perform unlearning for FOLTR. We conduct our study along two main directions: (1) how to efficiently unlearn with the guidance of historical local updates; and (2) how to evaluate the effectiveness of unlearning.

7.4.2 Findings

Trivial impact on model performance from a single client. In previous studies on machine unlearning and federated unlearning, the effectiveness of unlearning strategies has primarily been verified by comparing model performance against baseline models [36, 110]. Inspired by this, in our empirical simulation, we compare the ranking performance of two models: one trained before the removal of the client wanting to leave the federated system and one trained without this client. Our results indicate that excluding this single client does not significantly affect ranking performance. Although this phenomenon is accentuated due to the synthetic simulation with same configuration of the click model for all participants, it is still reflective of a real-world scenario, particularly in cross-device federated settings. Given the minimal impact on ranking performance from the removal of a single client, a more effective evaluation method for unlearning is required.

A solution for evaluating unlearning inspired from poisoning attacks. For the purpose of evaluating if unlearning takes place, we need to magnify the effect of the client that is leaving the FOLTR system and make this effect relatively controllable. To this aim, we adopt an alternative evaluation which leverages poisoning attacks only on the client requesting to be unlearned but before the unlearning takes place. Thus, mitigating the impact of the poisoned client during unlearning process suggests an effective unlearning strategy.

The effectiveness of unlearning with the help of historical local updates. Under the aforementioned evaluation, we find out that effective and efficient unlearning can be implemented with the help of historical local updates where in particular, we let the historical updates guide the direction for the unlearning process, and set the model update step size based on new interactions gathered during unlearning.

7.4.3 Contributions

This is the first study that investigates unlearning in FOLTR systems, where it is not clear that advances in general federated learning translate to similar improvements. We introduced an unlearning method to effectively and efficiently remove a client's contribution from a global ranking model without the need for complete retraining. The primary challenge tackled is ensuring that the model has genuinely

unlearned the specified client's data. To verify this, we employed a novel approach where the client deliberately introduces noise into its updates (a poisoning attack), and the reduction in the impact of this noise post-unlearning is used as a measure of success in unlearning.

7.4.4 Limitations and Future Work

In this study, we only consider the scenario where only one client requests unlearning to be initiated although we believe the findings might also generalise to the unlearning of multi-clients. However, it is still unclear how to perform multi-unlearning simultaneously and what trade-off between efficiency and effectiveness will be. Additionally, this study utilizes poisoning attacks to evaluate unlearning methods. However, this approach cannot be directly applied in real-world applications due to its disruptive nature. Future research should explore alternative evaluation methods that consider the search experience of remaining clients, ensuring that the unlearning process is both effective and minimally intrusive.

7.5 Future Directions: Federated Learning for Rankers based on Pretrained Language Models

In this thesis, we focused on the learning of feature-based ranking models. Feature-based ranking models are still prevalent in industry [148, 149]. However, in recent years, there is a growing trend in leveraging Pretrained Language Models (PLMs) based on the Transformer architecture [150] for ranking models [151]. One of the most common approaches follow a bi-encoder architecture: the query and documents are separately encoded into dense or sparse vectors by PLMs, and relevance scores are estimated based on the similarity between query and document vectors. By doing so, more semantic meanings for queries and documents can be exploited and vocabulary mismatch problem happened in traditional bag-of-words retrieval methods such as BM25 and TF-IDF [152–154] can be alleviated, demonstrated by the significant effectiveness improvement of the PLMs based rankers [155].

To our best knowledge, leveraging FL for PLMs-based ranker is still unexplored. Due to the need of large amount of labelled data for fine-tuning PLMs towards ranking tasks, privacy concerns on sharing sensitive training data [1] and high cost for editorial labelling remain two critical challenges, especially for domains like health, personal and enterprise search. To this aim, this thesis can inspire the study of a federated online PLMs-based ranker across several directions. First of all, future research can look into the design of a framework where PLM rankers are fine-tuned in a federated and online manner using implicit user interactions, such as clicks. Instead of sharing data, clients fine-tune the PLM ranker on their local data and then communicate to a central server the ranker updates they obtained. As the architecture of rankers based on PLMs is largely different from the feature-based rankers, it is unclear how advances in federated learning and our findings on FOLTR apply to PLM rankers. Importantly, as PLM-based rankers are large in size, fine-tuning the PLM ranker is an expensive

[1] words in textual documents and queries under this context

operation on user devices with low computational power. How to efficiently fine tune them with limited local interactions and limited computation resource remains a challenge. One solution is to use knowledge distillation (KD) rankers [156], where a smaller student ranker is trained by distilling knowledge, such as score distribution and ranking loss, from a larger, more expensive teacher ranker. By reducing the size of the ranking model, both the fine-tuning and the inference costs at the the client side can be reduced. Additionally, PLM rankers typically require a large amount of fine-tuning data to be effective. However, in this FL framework, effective PLM rankers can be trained by using only a small portion of data from each user in the federation. Recent work in information retrieval has explored the promising direction of prompt learning for fine-tuning PLM rankers in the presence of limited labelled data [157] and prompting Large Language Models (LLMs) for zero-shot document ranking [158]. Future work can investigate how to integrate prompt learning into the FL framework to further improve effectiveness of PLM rankers. Learning from the noisy and biased implicit feedback has been largely investigated in the context of counterfactual learning to rank and online learning to rank. However, aspects still unexplored are how this noisy and biased signal affects the learning of PLM rankers and how effective learning can take place in this context. Despite federated learning not involving the sharing of user data, a malicious agent that is able to observe the gradient updates can potentially reconstruct the data that produced such an update [65], leaving the potential of exposing user data. In addition, previous work has found that the PLMs underlying the rankers considered in this context can leak private information [159, 160], and thus this issue plausibly applies to PLM rankers too. Therefore, further studies on privacy issues related to PLM rankers can be conducted, which include (1) the risk caused by the exchange of local ranker updates from the clients, and (2) the inspection of a global or local ranker formed through the aggregation of client models updates; along with guidelines and new methods for controlling such risks.

7.6 Summary

Federated Learning provides a solution for building up effective rankers while respecting user privacy for OLTR. In this thesis, we conducted the first comprehensive study on Federated Online Learning to Rank with focus on the effectiveness, robustness, security and unlearning, tackling a range of unexplored questions for this area. This work will help researchers and practitioners to gain in-depth insights into FOLTR in terms of debiased and effective optimisation, robustness to non-IID data, risks posed by poisoning attacks, and efficient unlearning. As demonstrated in this thesis, there is still much to be done in refining the existing research on FOLTR. We point out the limitations of our work and chart directions for future work.

Our investigation and findings may be valid also in other contexts that consider to create a ranker from feature-based data in a federated manner, e.g., in federated learning to rank [69] and federated counterfactual leaning to rank [161]. Our research questions can inspire the study of federatively learning embedding-based rankers [151] which are founded by PLMs and LLMs.

Bibliography

[1] S. Wang, S. Zhuang, G. Zuccon, Federated online learning to rank with evolution strategies: A reproducibility study, in: European Conference on Information Retrieval, Springer, 2021, pp. 134–149.

[2] S. Wang, B. Liu, S. Zhuang, G. Zuccon, Effective and privacy-preserving federated online learning to rank, in: Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval, 2021, pp. 3–12.

[3] S. Wang, G. Zuccon, Is non-iid data a threat in federated online learning to rank?, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2801–2813.

[4] S. Wang, G. Zuccon, An analysis of untargeted poisoning attack and defense methods for federated online learning to rank systems, in: Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval, 2023, pp. 215–224.

[5] S. Wang, B. Liu, G. Zuccon, How to forget clients in federated online learning to rank?, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part III, Vol. 14610 of Lecture Notes in Computer Science, Springer, 2024, pp. 105–121. doi:10.1007/978-3-031-56063-7_7. URL https://doi.org/10.1007/978-3-031-56063-7_7

[6] H. Zhu, J. Xu, S. Liu, Y. Jin, Federated learning on non-iid data: A survey, Neurocomputing 465 (2021) 371–390.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[8] D. Locke, G. Zuccon, Case law retrieval: problems, methods, challenges and evaluations in the last 20 years, arXiv preprint arXiv:2202.07209 (2022).

[9] J. Y. Kim, N. Craswell, S. Dumais, F. Radlinski, F. Liu, Understanding and modeling success in email search, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 265–274.

[10] R. Datta, D. Joshi, J. Li, J. Z. Wang, Image retrieval: Ideas, influences, and trends of the new age, ACM Computing Surveys (Csur) 40 (2) (2008) 1–60.

[11] N. Orio, et al., Music retrieval: A tutorial and review, Foundations and Trends® in Information Retrieval 1 (1) (2006) 1–90.

[12] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, H.-W. Hon, Adapting ranking svm to document retrieval, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 186–193.

[13] T.-Y. Liu, et al., Learning to rank for information retrieval, Foundations and Trends® in Information Retrieval 3 (3) (2009) 225–331.

[14] M. Sanderson, Test collection based evaluation of information retrieval systems, Now Publishers Inc, 2010.

[15] D. Lefortier, P. Serdyukov, M. De Rijke, Online exploration for detecting shifts in fresh intent, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 589–598.

[16] S. Zhuang, G. Zuccon, How do online learning to rank methods adapt to changes of intent?, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021.

[17] X. Wang, M. Bendersky, D. Metzler, M. Najork, Learning to rank with selection bias in personal search, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 115–124.

[18] T. Joachims, L. Granka, B. Pan, H. Hembrooke, G. Gay, Accurately interpreting clickthrough data as implicit feedback, in: Acm Sigir Forum, Vol. 51, Acm New York, NY, USA, 2017, pp. 4–11.

[19] H. Oosterhuis, R. Jagerman, M. de Rijke, Unbiased learning to rank: counterfactual and online approaches, in: Companion Proceedings of the Web Conference 2020, 2020, pp. 299–300.

[20] Q. Ai, T. Yang, H. Wang, J. Mao, Unbiased learning to rank: online or offline?, ACM Transactions on Information Systems (TOIS) 39 (2) (2021) 1–29.

[21] S. Cohen, C. Domshlak, N. Zwerdling, On ranking techniques for desktop search, ACM Transactions on Information Systems (TOIS) 26 (2) (2008) 1–24.

[22] D. A. Hanauer, Emerse: the electronic medical record search engine, in: AMIA annual symposium proceedings, Vol. 2006, American Medical Informatics Association, 2006, p. 941.

[23] D. Hawking, Challenges in enterprise search, in: K. Schewe, H. E. Williams (Eds.), Database Technologies 2004, Proceedings of the Fifteenth Australasian Database Conference, ADC 2004, Dunedin, New Zealand, 18-22 January 2004, Vol. 27 of CRPIT, Australian Computer Society, 2004, pp. 15–24.
URL `http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV27Hawking1.html`

[24] M. Barbaro, T. Zeller, S. Hansell, A face is exposed for aol searcher no. 4417749, New York Times 9 (2008) (2006) 8.

[25] C. Carpineto, G. Romano, A review of ten year research on query log privacy, in: G. M. D. Nunzio, F. M. Nardini, S. Orlando (Eds.), Proceedings of the 7th Italian Information Retrieval Workshop, Venezia, Italy, May 30-31, 2016, Vol. 1653 of CEUR Workshop Proceedings, CEUR-WS.org, 2016.
URL `https://ceur-ws.org/Vol-1653/paper_25.pdf`

[26] S. Sousa, C. Guetl, R. Kern, Privacy in open search: A review of challenges and solutions, arXiv preprint arXiv:2110.10720 (2021).

[27] E. Kharitonov, Federated online learning to rank with evolution strategies, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 249–257.

[28] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2) (2019) 1–19.

[29] Y. Yue, T. Joachims, Interactively optimizing information retrieval systems as a dueling bandits problem, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1201–1208.

[30] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, arXiv preprint arXiv:1806.00582 (2018).

[31] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-iid data, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–9.

[32] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, in: IEEE International Conference on Data Engineering, 2022.

[33] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.

[34] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing 10 (3152676) (2017) 10–5555.

[35] E. L. Harding, J. J. Vanto, R. Clark, L. Hannah Ji, S. C. Ainsworth, Understanding the scope and impact of the california consumer privacy act of 2018, Journal of Data Protection & Privacy 2 (3) (2019) 234–253.

[36] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, N. Papernot, Machine unlearning, in: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, 2021, pp. 141–159.

[37] G. Liu, X. Ma, Y. Yang, C. Wang, J. Liu, Federaser: Enabling efficient client-level data removal from federated learning models, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), IEEE, 2021, pp. 1–10.

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

[39] H. Oosterhuis, M. de Rijke, Differentiable unbiased online learning to rank, in: Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 1293–1302.

[40] T. Qin, T. Liu, Introducing LETOR 4.0 datasets, CoRR abs/1306.2597 (2013).
URL `http://arxiv.org/abs/1306.2597`

[41] J. Gao, H. Qi, X. Xia, J.-Y. Nie, Linear discriminant model for information retrieval, in: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, 2005, pp. 290–297.

[42] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 89–96.

[43] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to rank: from pairwise approach to listwise approach, in: Proceedings of the 24th international conference on Machine learning, 2007, pp. 129–136.

[44] C. Burges, R. Ragno, Q. Le, Learning to rank with nonsmooth cost functions, Advances in neural information processing systems 19 (2006).

[45] O. Chapelle, Y. Chang, Yahoo! learning to rank challenge overview, in: O. Chapelle, Y. Chang, T. Liu (Eds.), Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010, Vol. 14 of JMLR Proceedings, JMLR.org, 2011, pp. 1–24.

[46] A. Agarwal, K. Takatsu, I. Zaitsev, T. Joachims, A general framework for counterfactual learning-to-rank, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 5–14.

[47] R. Jagerman, H. Oosterhuis, M. de Rijke, To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 15–24.

[48] K. Hofmann, S. Whiteson, M. De Rijke, A probabilistic method for inferring preferences from clicks, in: Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pp. 249–258.

[49] H. Oosterhuis, A. Schuth, M. de Rijke, Probabilistic multileave gradient descent, in: European Conference on Information Retrieval, Springer, 2016, pp. 661–668.

[50] S. Zhuang, G. Zuccon, Counterfactual online learning to rank, in: European Conference on Information Retrieval, Springer, 2020, pp. 415–430.

[51] K. Hofmann, A. Schuth, S. Whiteson, M. De Rijke, Reusing historical interaction data for faster online learning to rank for IR, in: Proceedings of the sixth ACM international conference on Web search and data mining, 2013, pp. 183–192.

[52] Y. Jia, H. Wang, S. Guo, H. Wang, Pairrank: Online pairwise learning to rank by divide-and-conquer, in: Proceedings of the Web Conference 2021, 2021, pp. 146–157.

[53] Y. Jia, H. Wang, Learning neural ranking models online from implicit user feedback, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 431–441.

[54] Y. Jia, H. Wang, Scalable exploration for neural online learning to rank with perturbed feedback, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 533–545.

[55] A. Schuth, H. Oosterhuis, S. Whiteson, M. de Rijke, Multileave gradient descent for fast online learning to rank, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016, ACM, 2016, pp. 457–466.

[56] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, S. Trani, Post-learning optimization of tree ensembles for efficient ranking, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 949–952.

[57] J. Allan, B. Carterette, J. A. Aslam, V. Pavlu, B. Dachev, E. Kanoulas, Million query track 2007 overview, Tech. rep., MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE (2007).

[58] F. Guo, C. Liu, Y. M. Wang, Efficient multiple-click models in web search, in: Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009, 2009, pp. 124–131.

[59] K. Hofmann, Fast and reliable online learning to rank for information retrieval, in: ACM SIGIR Forum, Vol. 47, ACM New York, NY, USA, 2013, pp. 140–140.

[60] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of cryptography conference, Springer, 2006, pp. 265–284.

[61] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.

[62] H. B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, in: International Conference on Learning Representations, 2018.

[63] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1322–1333.

[64] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[65] J. Geiping, H. Bauermeister, H. Dröge, M. Moeller, Inverting gradients - how easy is it to break privacy in federated learning?, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.
URL https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html

[66] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, arXiv preprint arXiv:1703.03864 (2017).

[67] F. Pinelli, G. Tolomei, G. Trappolini, Flirt: Federated learning for information retrieval, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 3472–3475.

[68] L. Zong, Q. Xie, J. Zhou, P. Wu, X. Zhang, B. Xu, Fedcmr: Federated cross-modal retrieval, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1672–1676.

[69] Y. Wang, Y. Tong, D. Shi, K. Xu, An efficient approach for cross-silo federated learning to rank, in: International Conference on Data Engineering, IEEE, 2021, pp. 1128–1139.

[70] F. Hartmann, S. Suh, A. Komarzewski, T. D. Smith, I. Segall, Federated learning for ranking browser history suggestions, arXiv preprint arXiv:1911.11807 (2019).

[71] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, F. Beaufays, Applied federated learning: Improving google keyboard query suggestions, arXiv preprint arXiv:1812.02903 (2018).

[72] M. R. Ghorab, D. Zhou, A. O'connor, V. Wade, Personalised information retrieval: survey and classification, User Modeling and User-Adapted Interaction 23 (4) (2013) 381–443.

[73] J. Yao, Z. Dou, J.-R. Wen, Fedps: A privacy protection enhanced personalized search framework, in: The Web Conference 2021, 2021, pp. 3757–3766.

[74] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, in: 8th International Conference on Learning Representations, 2020.

[75] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, L. Liang, Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications, in: IEEE International Conference on Computer Design, IEEE, 2019, pp. 246–254.

[76] H. Wang, L. Muñoz-González, D. Eklund, S. Raza, Non-iid data re-balancing at iot edge with peer-to-peer federated learning for anomaly detection, in: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2021, pp. 153–163.

[77] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, D. Ramage, Federated evaluation of on-device personalization, arXiv preprint arXiv:1910.10252 (2019).

[78] A. Fallah, A. Mokhtari, A. Ozdaglar, Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach, Advances in Neural Information Processing Systems 33 (2020).

[79] V. Smith, C.-K. Chiang, M. Sanjabi, A. S. Talwalkar, Federated multi-task learning, Advances in neural information processing systems 30 (2017).

[80] F. Sattler, K.-R. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, IEEE transactions on neural networks and learning systems (2020).

[81] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, Advances in Neural Information Processing Systems 33 (2020) 19586–19597.

[82] A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: International Conference on Machine Learning, PMLR, 2019, pp. 634–643.

[83] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, ACM Transactions on Programming Languages and Systems 4 (3) (1982) 382–401.

[84] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, Advances in Neural Information Processing Systems 30 (2017).

[85] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: International Conference on Machine Learning, PMLR, 2018, pp. 5650–5659.

[86] M. Fang, X. Cao, J. Jia, N. Z. Gong, Local model poisoning attacks to byzantine-robust federated learning, in: Proceedings of the 29th USENIX Conference on Security Symposium, 2020, pp. 1623–1640.

[87] V. Shejwalkar, A. Houmansadr, Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning, in: NDSS, 2021.

[88] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: Proceedings of the 29th International Coference on International Conference on Machine Learning, 2012, pp. 1467–1474.

[89] G. Baruch, M. Baruch, Y. Goldberg, A little is enough: Circumventing defenses for distributed learning, Advances in Neural Information Processing Systems 32 (2019).

[90] Y. Cao, J. Yang, Towards making systems forget with machine unlearning, in: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015, IEEE Computer Society, 2015, pp. 463–480. doi:10.1109/SP.2015.35.
URL https://doi.org/10.1109/SP.2015.35

[91] A. Halimi, S. Kadhe, A. Rawat, N. Baracaldo, Federated unlearning: How to efficiently erase a client in fl?, arXiv preprint arXiv:2207.05521 (2022).

[92] J. Wang, S. Guo, X. Xie, H. Qi, Federated unlearning via class-discriminative pruning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 622–632.

[93] T. Che, Y. Zhou, Z. Zhang, L. Lyu, J. Liu, D. Yan, D. Dou, J. Huan, Fast federated machine unlearning with nonlinear functional theory, in: International conference on machine learning, PMLR, 2023, pp. 4241–4268.

[94] T. Baumhauer, P. Schöttle, M. Zeppelzauer, Machine unlearning: linear filtration for logit-based classifiers, Mach. Learn. 111 (9) (2022) 3203–3226. doi:10.1007/s10994-022-06178-9.
URL https://doi.org/10.1007/s10994-022-06178-9

[95] J. Brophy, D. Lowd, Machine unlearning for random forests, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021,

pp. 1092–1104.
URL http://proceedings.mlr.press/v139/brophy21a.html

[96] Y. Chen, J. Xiong, W. Xu, J. Zuo, A novel online incremental and decremental learning algorithm based on variable support vector machine, Clust. Comput. 22 (Supplement) (2019) 7435–7445. doi:10.1007/s10586-018-1772-4.
URL https://doi.org/10.1007/s10586-018-1772-4

[97] A. Ginart, M. Y. Guan, G. Valiant, J. Zou, Making AI forget you: Data deletion in machine learning, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 3513–3526.
URL https://proceedings.neurips.cc/paper/2019/hash/cb79f8fa58b91d3af6c9c991f63962d3-Abstract.html

[98] A. Mahadevan, M. Mathioudakis, Certifiable machine unlearning for linear models, arXiv preprint arXiv:2106.15093 (2021).

[99] N. Aldaghri, H. Mahdavifar, A. Beirami, Coded machine unlearning, IEEE Access 9 (2021) 88137–88150. doi:10.1109/ACCESS.2021.3090019.
URL https://doi.org/10.1109/ACCESS.2021.3090019

[100] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, Y. Zhang, Graph unlearning, in: H. Yin, A. Stavrou, C. Cremers, E. Shi (Eds.), Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, ACM, 2022, pp. 499–513. doi:10.1145/3548606.3559352.
URL https://doi.org/10.1145/3548606.3559352

[101] V. S. Chundawat, A. K. Tarun, M. Mandal, M. Kankanhalli, Zero-shot machine unlearning, IEEE Transactions on Information Forensics and Security (2023).

[102] Y. Liu, Z. Ma, X. Liu, J. Ma, Learn to forget: User-level memorization elimination in federated learning, CoRR abs/2003.10933 (2020). arXiv:2003.10933.
URL https://arxiv.org/abs/2003.10933

[103] Y. Liu, L. Xu, X. Yuan, C. Wang, B. Li, The right to be forgotten in federated learning: An efficient realization with rapid retraining, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, May 2-5, 2022, IEEE, 2022, pp. 1749–1758. doi:10.1109/INFOCOM48880.2022.9796721.
URL https://doi.org/10.1109/INFOCOM48880.2022.9796721

[104] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, Q. V. H. Nguyen, A survey of machine unlearning, arXiv preprint arXiv:2209.02299 (2022).

[105] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, Y. Zhang, When machine unlearning jeopardizes privacy, in: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security, 2021, pp. 896–911.

[106] H. Hu, Z. Salcic, G. Dobbie, J. Chen, L. Sun, X. Zhang, Membership inference via backdooring, in: The 31st International Joint Conference on Artificial Intelligence (IJCAI-22), 2022.

[107] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, H. Wang, Federated unlearning for on-device recommendation, in: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, 2023, pp. 393–401.

[108] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, J. Chen, Verifi: Towards verifiable federated unlearning, IEEE Transactions on Dependable and Secure Computing (2024).

[109] D. M. Sommer, L. Song, S. Wagh, P. Mittal, Athena: Probabilistic verification of machine unlearning, Proceedings on Privacy Enhancing Technologies 3 (2022) 268–290.

[110] C. Wu, S. Zhu, P. Mitra, Federated unlearning with knowledge distillation, arXiv preprint arXiv:2201.09441 (2022).

[111] F. Radlinski, R. Kleinberg, T. Joachims, Learning diverse rankings with multi-armed bandits, in: Proceedings of the 25th international conference on Machine learning, 2008, pp. 784–791.

[112] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8 (3-4) (1992) 229–256.

[113] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[114] B. Szörényi, R. Busa-Fekete, A. Paul, E. Hüllermeier, Online rank elicitation for plackett-luce: A dueling bandits approach, in: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 604–612.

[115] V. Mugunthan, A. Peraire-Bueno, L. Kagal, Privacyfl: A simulator for privacy-preserving and secure federated learning, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 3085–3092.

[116] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (1–2) (2021) 1–210.

[117] C. L. Clarke, N. Craswell, E. M. Voorhees, Overview of the trec 2012 web track, Tech. rep. (2012).

[118] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proceedings of Machine Learning and Systems 2 (2020) 429–450.

[119] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, S. Choudhary, Federated learning with personalization layers, arXiv preprint arXiv:1912.00818 (2019).

[120] K. Roberts, M. S. Simpson, E. M. Voorhees, W. R. Hersh, Overview of the trec 2015 clinical decision support track, in: Text REtrieval Conference, 2015.

[121] K. Roberts, M. Simpson, D. Demner-Fushman, E. Voorhees, W. Hersh, State-of-the-art in biomedical literature retrieval for clinical cases: a survey of the trec 2014 cds track, Information Retrieval Journal 19 (1) (2016) 113–148.

[122] K. Narang, S. T. Dumais, N. Craswell, D. Liebling, Q. Ai, Large-scale analysis of email search and organizational strategies, in: ACM SIGIR Conference on Human Information Interaction and Retrieval, 2017, pp. 215–223.

[123] S. Whittaker, C. Sidner, Email overload: exploring personal information management of email, in: Conference on Human Factors in Computing Systems: Common Ground, 1996, pp. 276–283.

[124] N. Craswell, O. Zoeter, M. Taylor, B. Ramsey, An experimental comparison of click position-bias models, in: International Conference on Web Search and Data Mining, 2008, pp. 87–94.

[125] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, Poison frogs! targeted clean-label poisoning attacks on neural networks, Advances in neural information processing systems 31 (2018).

[126] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, arXiv preprint arXiv:2003.02133 (2020).

[127] Y. Yu, Q. Liu, L. Wu, R. Yu, S. L. Yu, Z. Zhang, Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 4854–4863.

[128] R. Guerraoui, S. Rouault, et al., The hidden vulnerability of distributed learning in byzantium, in: International Conference on Machine Learning, PMLR, 2018, pp. 3521–3530.

[129] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 480–501.

[130] V. Shejwalkar, A. Houmansadr, P. Kairouz, D. Ramage, Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, pp. 1354–1371.

[131] W. Hersh, Information Retrieval: A Biomedical and Health Perspective, Springer Nature, 2020.

[132] Q. Xia, Z. Tao, Z. Hao, Q. Li, Faba: an algorithm for fast aggregation against byzantine attacks in distributed neural networks, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 4824–4830.

[133] X. Cao, M. Fang, J. Liu, N. Z. Gong, Fltrust: Byzantine-robust federated learning via trust bootstrapping, in: 28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021, 2021.

[134] J. Xu, S.-L. Huang, L. Song, T. Lan, Byzantine-robust federated learning through collaborative malicious gradient filtering, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), IEEE, 2022, pp. 1223–1235.

[135] S. T. de Magalhães, The european union's general data protection regulation (gdpr), in: CYBER SECURITY PRACTITIONER'S GUIDE, World Scientific, 2020, pp. 529–558.

[136] D. M. Sommer, L. Song, S. Wagh, P. Mittal, Towards probabilistic verification of machine unlearning, arXiv preprint arXiv:2003.04247 (2020).

[137] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, R. Rogers, Protection against reconstruction and its applications in private federated learning, arXiv preprint arXiv:1812.00984 (2018).

[138] A. C. Yao, Protocols for secure computations, in: 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, 1982, pp. 160–164.

[139] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178.

[140] H. Oosterhuis, M. de Rijke, Unifying online and counterfactual learning to rank: A novel counterfactual estimator that effectively utilizes online interventions, in: International Conference on Web Search and Data Mining, 2021, pp. 463–471.

[141] J. Cecchetti, N. Tonellotto, R. Perego, Learning to rank for non independent and identically distributed datasets, in: Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval, 2024, pp. 71–79.

[142] M. D. Cooper, A simulation model of an information retrieval system, Information Storage and Retrieval 9 (1) (1973) 13–32.

[143] L. Azzopardi, Simulation of interaction: A tutorial on modelling and simulating user interaction and search behaviour, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 1227–1230.

[144] D. Maxwell, L. Azzopardi, Simulating interactive information retrieval: Simiir: A framework for the simulation of interaction, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 1141–1144.

[145] Y. Zhang, X. Liu, C. Zhai, Information retrieval evaluation as search simulation: A general formal framework for ir evaluation, in: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, 2017, pp. 193–200.

[146] K. Balog, D. Maxwell, P. Thomas, S. Zhang, B. London, Report on the 1st simulation for information retrieval workshop (sim4ir 2021) at sigir 2021 (2021).

[147] Z. Zhang, X. Cao, J. Jia, N. Z. Gong, Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2545–2555.

[148] L. Zou, H. Mao, X. Chu, J. Tang, W. Ye, S. Wang, D. Yin, A large scale search dataset for unbiased learning to rank, Advances in Neural Information Processing Systems 35 (2022) 1127–1139.

[149] D. Dato, S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, The istella22 dataset: Bridging traditional and neural learning to rank evaluation, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 3099–3107.

[150] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[151] J. Lin, R. Nogueira, A. Yates, Pretrained transformers for text ranking: Bert and beyond, Springer Nature, 2022.

[152] S. E. Robertson, S. Walker, Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval, in: SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University, Springer, 1994, pp. 232–241.

[153] F. Crestani, M. Lalmas, C. J. Van Rijsbergen, I. Campbell, "is this document relevant?... probably" a survey of probabilistic models in information retrieval, ACM Computing Surveys (CSUR) 30 (4) (1998) 528–552.

[154] S. Robertson, H. Zaragoza, et al., The probabilistic relevance framework: Bm25 and beyond, Foundations and Trends® in Information Retrieval 3 (4) (2009) 333–389.

[155] R. Nogueira, K. Cho, Passage re-ranking with bert, arXiv preprint arXiv:1901.04085 (2019).

[156] S. Hofstätter, S.-C. Lin, J.-H. Yang, J. Lin, A. Hanbury, Efficiently teaching an effective dense retriever with balanced topic aware sampling, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 113–122.

[157] X. Hu, S. Yu, C. Xiong, Z. Liu, Z. Liu, G. Yu, P3 ranker: Mitigating the gaps between pre-training and ranking fine-tuning with prompt-based learning and pre-finetuning, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1956–1962.

[158] S. Zhuang, B. Liu, B. Koopman, G. Zuccon, Open-source large language models are strong zero-shot query likelihood models for document ranking, in: Findings of the Association for Computational Linguistics: EMNLP 2023, 2023, pp. 8807–8817.

[159] X. Pan, M. Zhang, S. Ji, M. Yang, Privacy risks of general-purpose language models, in: 2020 IEEE Symposium on Security and Privacy (SP), IEEE, 2020, pp. 1314–1331.

[160] J. Morris, V. Kuleshov, V. Shmatikov, A. Rush, Text embeddings reveal (almost) as much as text, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 12448–12460. doi:10.18653/v1/2023.emnlp-main.765.
URL https://aclanthology.org/2023.emnlp-main.765

[161] C. Li, H. Ouyang, Federated unbiased learning to rank, arXiv preprint arXiv:2105.04761 (2021).