Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

Pursuing Privacy and Personalization in Recommender Systems: The Role of Federated Learning

This is a PhD Thesis

Original Citation:
Pursuing Privacy and Personalization in Recommender Systems: The Role of Federated Learning / Ferrara, Antonio. - ELETTRONICO. - (2022). [10.60576/poliba/iris/ferrara-antonio_phd2022]

Availability:
This version is available at http://hdl.handle.net/11589/232721 since: 2021-12-28

Published version
DOI:10.60576/poliba/iris/ferrara-antonio_phd2022

Publisher: Politecnico di Bari

Terms of use:

(Article begins on next page)

07 May 2024

Politecnico
di Bari

Department of Electrical and Information Engineering
Electrical and Information Engineering Ph.D. Program
SSD: ING-INF/05 - Information Processing Systems

Final Dissertation

Pursuing Privacy and Personalization in Recommender Systems: The Role of Federated Learning

by

Antonio Ferrara

Supervisor
Prof. Tommaso Di Noia

Coordinator of the Ph.D. Program
Prof. Mario Carpentieri

Course n° 34, 01/11/2018 - 31/10/2021

Dedicated to all my *families*.

Abstract

In the era of big data, the massive data availability has created a tough world to navigate for users, who are often overwhelmed by alternatives of products, services, music, movies. Big data is also the fuel of machine learning applications and powerful tools like recommender systems that help users choose among the myriad of alternatives. Regrettably, this big data often contains sensitive information (e.g., socio-demographic attributes, social relations, browsing history, patterns of visited location or played music) that reflects their personal behavior. Although the performance of machine learning-powered services is strictly related to the amount of collected data, today, people pay more and more attention to their privacy, and international jurisdictions have legislated new laws to limit and control data collection. In this context, federated learning is a recent paradigm for performing on-device machine learning model training without requiring personal data collection. Today, federated learning is considered, together with privacy-preserving techniques like differential privacy and encryption, the best candidate to face the data privacy challenges in machine learning. This thesis summarizes our effort to bring the opportunities offered by federated learning in recommender systems. We propose a model that puts users in control of their data and grants high-quality recommendations even when users decide not to disclose part of their sensitive preferences. Moreover, we propose an entirely new federated recommender model based on a sparse entropy-weighted combination of feature embeddings. This work is motivated by our significant findings on the relation between federated data distribution and training efficiency and makes us realize that the on-device training of federated learning can lead not only to data privacy and control but also to highly personalized user-targeted recommender systems. All our findings are supported by extensive experiments and analyses on a wide range of recommendation dimensions. Finally, this dissertation is enriched by an extensive literature review on recommender systems, privacy-preserving paradigms and techniques, and a survey on the increasingly pivotal branch of privacy-preserving recommender systems.

Publications

Some ideas and figures have appeared previously in other publications. A complete list of my publications is available in the following (the symbol * denotes papers where I contributed as main author).

[1] *Vito Walter Anelli, Luca Belli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, Fedelucio Narducci, and Claudio Pomo. "Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. ACM, 2021, pp. 838–841.

[2] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation." In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. ACM, 2021, pp. 2405–2414.

[3] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "V-Elliot: Design, Evaluate and Tune Visual Recommender Systems." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. ACM, 2021, pp. 768–771.

[4] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation." In: *Proceedings of the 11th Italian Information Retrieval Workshop 2021, Bari, Italy, September 13-15, 2021*. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[5] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, and Antonio Ferrara. "Towards Effective Device-Aware Federated Learning." In: *AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence,*

Rende, Italy, November 19-22, 2019, Proceedings. Vol. 11946. Lecture Notes in Computer Science. Springer, 2019, pp. 477–491.

[6] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, and Antonio Ferrara. "Prioritized multi-criteria federated learning." In: *Intelligenza Artificiale* 14.2 (2020), pp. 183–200.

[7] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "FedeRank: User Controlled Feedback with Federated Recommender Systems." In: *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*. Vol. 12656. Lecture Notes in Computer Science. Springer, 2021, pp. 32–47.

[8] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "Federated Recommender Systems with Learning to Rank." In: *SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy*. 2021.

[9] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "How to put users in control of their data in federated top-N recommendation with learning to rank." In: *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*. ACM, 2021, pp. 1359–1362.

[10] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "User-controlled federated matrix factorization for recommender systems." In: *Journal of Intelligent Information Systems* (2022).

[11] *Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, and Alberto Carlo Maria Mancino. "Sparse Embeddings for Recommender Systems with Knowledge Graphs." In: *Proceedings of the 11th Italian Information Retrieval Workshop 2021, Bari, Italy, September 13-15, 2021*. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[12] *Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, and Alberto Carlo Maria Mancino. "Sparse Feature Factorization for Recommender Systems with Knowledge Graphs." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. ACM, 2021, pp. 154–165.

Contents

List of Figures

List of Tables

Part I

PRELIMINARIES

That is, what can you expect from this thesis, what I have studied and what I have worked for, what you need to know before going on. Preliminaries is my "*Where are you and where do you want to go? Which are your shoes?*".

1 Introduction

The massive information available in the World Wide Web, with its explosion in the last three decades, has made the users closer to each other as well as to unexplored and unknown products, services, music, movies. As a drawback, the larger the information availability, the more challenging it is to retrieve the most acceptable content in this gold mine. Users are often overwhelmed by the alternatives and faced with the difficult task of choosing among a wide range of contents using some limited budgets or a set of constraints [171]. For example, by typing *computer science* in a book search engine, the system could still return thousands of results.

In such situations, recommender systems can significantly help users in deciding what to choose, given that they point them towards not yet experienced items that are most likely relevant and of interest to the user, for instance, returning a ranked list of items [172]. Modern recommender systems usually provide personalized and tailored recommendations to different users or user groups. Based on their formulation, they make suggestions based on different assumptions (e.g., users can make decisions based on their tastes; users' decisions may rely on the behavior of similar users). As a consequence, recommender systems make online services more and more competitive against traditional stores, increasing at the same time the user experience, the provider revenue, and the diversity of the items sold.

As a common factor, all these systems collect information from users regarding their preferences. User ratings and evaluations for movies, books, services, and products, constitute a set of explicit preferences about items enjoyed in the past. Alternatively, the service providers may infer user preferences by interpreting the actions of the user (e.g., clicks, navigation history, listening playlist, search patterns, or mouse movements), which indirectly reflect opinions. Even socio-demographic attributes (e.g., age, gender, job, education), multimedia content, social relations, and contextual information are used to mine the user needs and feed personalized recommender systems exploiting user models [68].

Although the quality of recommendation is strictly related to the amount, richness, and freshness of the data used for user modeling [30,

104], its collection could be a severe flaw under the lens of users' privacy. More in general, almost all the industries providing services powered by machine learning systems benefit from big data, due to its potential to solve many of the challenging problems in several sectors. These systems have can understand the patterns and provide insights by learning from the data. To extract meaningful information and capture patterns of data, traditional machine learning systems need to collect the data to train statistical models centrally.

In this context, traditional machine learning systems are facing, in recent years, several challenges. In fact, the public awareness of privacy issues has been steadily increasing after large-scale data breaches such as Cambridge Analytica in 2018, which shared and harvested data from a massive number of users for political campaigning without their consent [49]. At the same time, these incidents made international headlines and have spurred the European Union, US Congress, and other jurisdictions to legislate new disclosure laws. As an example, in 2018 the European Union proposed GDPR [80], which removes the default option for collecting, storing, and harnessing individuals' data and requires explicit authorization from the users to use their data. Other representative examples are the CCPA in California [52] and the Cybersecurity Law in China [187]. Despite these laws' fundamental role in protecting users' privacy, the consequent data scarcity can thereby jeopardize the training of high-quality models.

In recent years, various solutions have been proposed to preserve the users' privacy in machine learning applications, ranging from barriers for untrusted parties to algorithmic solutions like anonymization, differential privacy, and cryptography-based approaches. In parallel, Google has recently proposed federated learning as a privacy-oriented training paradigm to tackle data isolation while meeting the need for big data [117, 146]. This paradigm embraces the idea of minimization of data collection with a decentralized architecture. Federated learning trains a machine learning model by keeping data on the devices they were generated on (e.g., smartphones, tablets, etc.) without sharing it with a central server. Today, federated learning is often implemented in conjunction with privacy-preserving techniques for providing privacy guarantees and is considered the best candidate to face the data privacy, control, and property challenges posed by the data regulations.

The work at hand, whose investigation started in 2018, takes up the goals and the challenges of federated learning and explores, for the first time, its applications and implications in recommender systems. In detail, this thesis guides the reader towards three core contributions:

- the introduction of a federated pair-wise recommender system, with an extensive study on the impact of limited information availability due to users exercising the right of property and control on their data;
- the proposal of a client-aware strategy for federated learning, attempting to improve the model by favoring the impact of users providing a higher quality contribution to the global model;
- the presentation of a knowledge-aware recommendation model that takes the best from latent factor approaches and content-based approaches and ensures that specific parts of the federated model are updated only by those users able to provide a high quality contribution in terms of expertise.

In the next section, we deeply analyze the research questions that guided our work towards its contributions and form the content of this dissertation.

1.1 Research Contributions

The four concepts that more often will occur in this dissertation are *recommendation*, *data*, *privacy*, and *federated learning*. Their connection is extremely tight since recommender systems, as already mentioned, could not exist and provide suggestions without data. Despite their usefulness for users, content creators, and content providers, data harvesting may often be a synonym of privacy threat, and federated learning is the way we believe users can place their trust in a recommender system and obtain from it an incoming in terms of utility and personalization.

In the following, we present the research questions that guided this three-year investigation and led to the contributions that outline the main part of this dissertation (see The Showcase, Part II).

1.1.1 *Federated Learning for Data Property and Control in Recommender Systems*

Research Questions A

Is it possible to build a federated recommender system in which users are completely in control of their data? Which is the effect of feedback deprivation when users exercise the right of control on their data? Are

we able to replicate the performance of a centralized recommender system? What is the optimal trade-off between communication costs and recommendation utility? What is the impact of federated computation parameters on the quality of the recommendation?

Federated learning has opened the doors to a new machine learning model class that does not need to collect training data in a centralized location. Nevertheless, when our investigation started, most of the work implementing the federated learning paradigm was concerned with image classification or natural language understanding tasks.

Chapter 4 presents our contribution to move a step in the literature of recommender systems towards architectural and algorithmic solutions accounting for data property and control. We propose a novel factorization model that embraces the federated learning paradigm. Users participating in the federation process can decide if and how they are willing to disclose their private sensitive preferences and fully control their data. The proposed model pushes recommender systems towards the European GDPR specifications, which remove the default option for collecting, storing, and harnessing individual data and require explicit authorization from the users to use their data. An extensive analysis of the system lets us know the behavior of federated recommendation when users decide to share only a small amount of sensitive information and how incomplete data impacts the system performance from a wide range of perspectives. Leveraging non-sensitive information, we show that our model avoids the risk of preserving privacy at the cost of a worse tailored recommendation, which may frustrate users and reduce the acceptance of the recommender system.

1.1.2 *User-Level Knowledge for Improved Aggregation in Federated Learning*

Research Questions B

What should be the impact of the contribution of each user on a federated model? Does the distribution of the local datasets affect the convergence of the model? Can we define some criteria based on data distribution or user expertise to speed up the training of the model?

The pioneering work about federated learning introduced an algorithm based on federated stochastic gradient descent with contributions com-

ing from a large number of users. Such contributions, in the form of model updates, were usually aggregated with naive methods.

In Chapter 5, we study whether other criteria can be used during the aggregation process to improve the effectiveness of the whole global model. Specifically, we formalize a method for assigning a score to each client based on a suite of criteria and their priority. With the support of an extensive set of experiments, we show how important is that each user in the federation updates the model based on her *portion of knowledge* of the ideal non-federated data distribution.

1.1.3 *Federated Learning for Personalization in Recommender Systems*

Research Questions C

Is federated learning also suitable for providing higher personalization in recommender systems? How should the users collaborate to train the federated model based on their expertise? Can a hybrid collaborative/content-based model be used in this scenario and to this aim?

Although federated learning was conceived for preserving the property of personal data, we assert that the recommender models trained in a federated fashion, if properly designed, can provide users with a higher degree of personalization.

Chapter 6 of this thesis introduces a knowledge-aware federated recommender system that couples the advantages of collaborative filtering and content-based filtering recommendation. The proposed model associates items and users with a set of features, each of them weighted by an entropy measure representing the impact it has on the user decision process. Each feature is then embedded into a manifold representation, accounting for each user's perspective and feeling on it. Following the findings of Chapter 5, we apply a principle of expertise, facilitating the training process by letting users update only those relevant features on which they base their decisions. The resulting recommender system shows a high degree of expressiveness, which positively affects recommendation performance in terms of accuracy, diversity, bias, and semantics preservation.

1.2 Organization of the Thesis

The chapters of this thesis are as self-contained as possible and present the notions of specific problems, architectures, paradigms, data structures, and metrics related with their content. Moreover, each chapter independently surveys the state-of-the-art of the specific problem it deals with, showing the most interesting solutions and their limitations in the literature of that field.

However, in the next two chapters, we propose a brief but comprehensive introduction to the most important recurring topics of this dissertation. In detail, we analyze the recommendation problem with its models, solutions, and evaluation techniques (Chapter 2), and the problem of privacy in machine learning (Chapter 3), with an extensive view on *privacy-oriented* paradigm like federated learning, and *privacy-preserving* algorithms, like differential privacy, secure multi-party computation, and homomorphic encryption.

All the chapters in Part II have their specific research questions, and they present the experimentation of the proposed approach that we have designed to answer the research questions with an extensive discussion of the results. However, the three chapters together constitute the unique path that guided this three-year work and helped to shed light on some of the privacy and personalization challenges we wanted to address.

2 Recommender Systems

OUTLINE

Recommender systems can provide users with recommendations when faced with choosing among an extensive catalog of items or whenever they want to receive suggestions. Recommender systems model the users by learning from their past behavioral data (e.g., movies they watched, ratings they gave, items they bought, webpages they visited) and point them towards not yet experienced items that are most likely of interest based on different assumptions.

In this chapter, we will first provide a formal definition of the recommendation problem. Then, we will survey the pioneer models and architectures along with their taxonomy. Finally, we will provide an overview on evaluation of performance, with some in-depth considerations on the metrics used in this dissertation.

2.1 Definition

Formally, let $\mathcal{U}$ be the set of users in the system and $\mathcal{I}$ the set of the items (e.g., the catalog of an online shop). Then, let $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ be the user-item preference matrix containing either explicit feedback or implicit feedback. In the former case, we can have, for instance, $r_{ui} \in \{\varnothing\} \cup \{1, \ldots, 5\}$ if the users are asked to leave up to 5 stars for a product, with $\varnothing$ denoting a missing rating. With implicit feedback, we can observe, for instance, data in the form of $r_{ui} \in \{\varnothing\} \cup \{1\}$ with $r_{ui} = 1$ if u interacted with i (e.g., the user visited a webpage, clicked on a video, bought a product) and $r_{ui} = \varnothing$ if not, or in the form of $r_{ui} \in \{\varnothing\} \cup \{like\,(1), dislike\,(0)\}$. Since the items of a catalog often number in the thousands to millions, most users naturally rate or enjoy only a small percentage of them. Usually, the proportion of missing ratings of several well-known datasets in recommendation literature (e.g., MovieLens, Netflix) is higher than 95%. Often, when the user is asked to give explicit ratings, the datasets exhibit a large skew favoring

very high or very low values. Notably, we define $\mathcal{I}_u$ as the set of items user u interacted with, i.e.,

$$\mathcal{I}_u = \{i \in \mathcal{I} \mid r_{ui} \neq \varnothing\}. \tag{2.1}$$

In general, recommendation problems are associated with a wide range of different tasks, from finding all the possible good items for a user to recommending an ordered sequence or a bundle of items. However, the most important and most studied tasks in literature concern with *rating prediction* and *top-k recommendation* [62, 153].

Definition 2.1 (Rating prediction task)**.** *Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I} \setminus \mathcal{I}_u$ not rated by u, the rating prediction task aims to predict the missing rating of u to i. Formally, the goal is to learn a function $f : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$, often defined as a regression or a classification.*

Definition 2.2 (Top-k recommendation task)**.** *Given a user $u \in \mathcal{U}$, the top-k recommendation task aims to provide u with a list containing k items from $\mathcal{I} \setminus \mathcal{I}_u$ most likely to interest u, ordered by a proper utility function $s : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$.*

Often, the top-k recommendation task is solved by ranking the items of $\mathcal{I}$ based on the predictions given by f, which acts as a utility function.

The solution to the above-mentioned recommendation problems heavily depends on the selected utility function s or the prediction function f — usually, but not necessarily, a machine learning model — and on the type of information encoded in the data.While the idea of utility may be often exclusively associated with user satisfaction, it actually includes the revenue of content creators and content providers (e.g., depending on the number and the diversity of the items sold), and the users' loyalty to the service. The prediction of the utility, or at least the comparison of the utility of some items, is the core function of recommender systems for understanding whether an item is worth recommending [172].

In the following, we present a taxonomy of the most common recommendation approaches, along with their standard and simplest formulations, usually aiming to predict and estimate the value of the missing ratings.

2.2 Overview of Recommendation Approaches

Recommender systems solve the recommendation problem previously defined by relying on different assumptions. For instance, they may

assume that users can make decisions about new items based on the similarity with items already consumed. Alternatively, they may assume that users' decisions may rely on the behavior of similar users.

The most common recommendation strategies can be classified into collaborative filtering, content-based filtering, and hybrid models [153]. In the following, we will present a brief but comprehensive overview of recommendation methods along with their taxonomy.

2.2.1 *Collaborative Filtering Approaches*

The core idea of collaborative filtering recommender system is that the rating of a user for a new item is likely to be similar to that of another user who has rated other items in a similar way [120]. Moreover, these systems assume that a user is likely to similarly rate two items if other users have given similar ratings to these two items. The methods using collaborative filtering approaches are able to provide recommendations to users through the feedback of other users and can recommend items with very different content if other users have shown interest in all these different items. Due to their formulation, these algorithms' performance highly relies on the availability of user transactions, but they have the advantage of not needing any other data source (e.g., attributes of the items).

Collaborative filtering methods can be grouped into neighborhood- and model-based methods [120]. While the first class of methods uses the collected ratings to predict the missing ratings, the model-based approaches aim to build a predictive model able to represent latent characteristics of the users and the items in the system.

2.2.1.1 *Neighborhood-based collaborative approaches*

Neighborhood-based recommender systems, also known as memory-based recommenders, assume that, in the user perspective, the opinion of like-minded users is useful for evaluating the value of an item. In principle, similar users prefer similar items, and similar items are preferred by similar users.

The k-nearest-neighbors (k-NN) approach, with its user-based and item-based variants (e.g., [69, 70]), is the most used one in memory-based recommendation.

User-based k-Nearest-Neighbors

In this schema, the k-NN algorithm computes a similarity matrix $\mathbf{W}$

encoding how similar are the users between each other. Then, the k users v with the highest similarity w_{uv} to a user u constitute the k-nearest-neighborhood $\mathcal{N}(u)$ of u. To estimate a missing rating r_{ui}, user-based k-NN considers the subset $\mathcal{N}_i(u) \subseteq \mathcal{N}(u)$ containing the neighbors of u who have rated i. Then, considering one of the simplest formulations of the algorithm, the rating r_{ui} can be estimated as:

$$\tilde{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \tag{2.2}$$

Intuitively, user u asks the like-minded users in $N_i(u)$ their opinion about i, and considers it for making a decision.

Item-based k-Nearest-Neighbors

With item-based k-NN, the core idea is to find similarities between items by looking at the ratings they have received. Once the item similarity matrix $\mathbf{W}$ has been computed, each item can be associated with a set of neighbors $\mathcal{N}(i)$. To estimate a missing rating r_{ui}, only the items in $\mathcal{N}_u(i)$ (i.e., similar to i that have been rated by u) are considered:

$$\tilde{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|} \tag{2.3}$$

The similarity between two users in user- based methods, which determines the neighbors of a user, is normally obtained by comparing the ratings made by these users on the same items. Conversely, in item-based methods, the similarity between two items is obtained by comparing the ratings they have received from different users. The computation of the similarity weights is one of the most critical aspects of building a neighborhood-based recommender system, as it can have a significant impact on both its accuracy and its performance. Among the most used approaches, it is common to find Cosine similarity, Pearson correlation, Jaccard index, and Euclidean distance [120].

There are several factors to be considered when choosing between user- and item-based k-NN. In terms of recommendation performance, a small number of high-confidence neighbors is preferable to a large number of neighbors for which the similarity weights are not trustable. Thus, where the number of users is much greater than the number of items, item-based methods are preferred [84]. Conversely, user-based

methods usually provide more original recommendations, which may lead users to a more satisfying experience [78]. The choice is also affected by memory and computational efficiency requirements since similarity estimation complexity quadratically grows with the number of users in user-based models and with the number of items in item-based models [153].

Overall, neighborhood-based collaborative approaches are intuitive and simple, produce explainable recommendations, and are efficient in prediction (once the similarity matrix has been computed), albeit they can suffer from problems like *limited coverage* of items (when users have no common ratings, they are necessarily classified as non-neighbor users, thus causing some their favorite items not to be recommended to each other) and high-performance sensitivity to the lack of available data.

2.2.1.2 *Model-Based Collaborative Approaches*

Model-based collaborative filtering approaches use the ratings in the matrix $\mathbf{R}$ to learn a predictive machine learning model representing latent characteristics of the users and the items in the system able to explain the ratings.

These approaches are numerous and have shown improved performance with respect to memory-based collaborative filtering algorithms, especially in the context of the million-dollar prize competition opened by Netflix in October 2006 [37], where they gain significant attention. Thanks to that competition, the research community gained access to a large-scale, industrial-strength dataset of 100 million movie ratings that encouraged the rapid development of model-based recommender systems.

Typical model-based approaches include Bayesian Clustering, Latent Semantic Analysis, Latent Dirichlet Allocation, Support Vector Machines, and neural networks [120]. However, the most popular models are the ones induced by factorization of the user-item rating matrix, thanks to their attractive accuracy and scalability. Singular Value Decomposition is one of the most established techniques for decomposing a matrix and identifying latent factors [121]. Nevertheless, the high portion of missing values leads to difficulties that force to train the model on the few available data and to use regularization for averting the risk of overfitting.

Matrix Factorization with Singular Value Decomposition

Matrix factorization aims to represent both items and users as vectors in a latent space $\mathbb{R}^f$. The latent space tries to explain ratings with latent factors, automatically inferred from user feedback, representing properties (e.g., the genre of a movie) that cannot be directly interpreted by a human being observing the decomposed matrix. The value assumed by each latent factor represents, in the user vector, her interest towards that property, while, in the item vector, the extent to which the item possesses the same property. The dot product between a user vector and an item vector will capture the user's estimated interest in that item.

Notably, given the vector $\mathbf{q}_i$ representing the item i in the latent space $\mathbb{R}^f$, and the vector $\mathbf{p}_u$ representing the user u in the same space, we estimate a missing rating as:

$$\tilde{r}_{ui} = \mu + b_u + b_i + \mathbf{q}_i^T \mathbf{p}_u, \tag{2.4}$$

where μ, b_u, and b_i are the overall rating average, the user u's rating average, and item i's rating average, respectively. These three terms constitute the baseline predictor for r_{ui}, which has a detrimental impact on collaborative filtering since feedback data usually exhibit large biases (e.g., some users have systematic tendencies to give very high or low ratings and some items to receive very high or low ratings). The vectors of all the users and items in the system can be learned with techniques like stochastic gradient descent and alternating least squares by exploiting the available ratings.

The prediction in Eq. 2.4 can be extended and used as a building block for more complex models (e.g., to take into account other types of feedback or to build time-aware factor models).

Generally, model-based approaches exhibit prediction accuracy superior to other collaborative filtering techniques [120] with a relatively memory-efficient and easy-to-train compact model. Moreover, these techniques are also convenient thanks to the possibility of injecting multiple forms of user feedback and modeling specific behaviors [118, 157, 177]. Once the rating prediction estimation has been modeled, based on the task and the assumptions, the loss function to minimize can be differently written (e.g., BPR [167], that we describe in Chapter 4). Then, we can also choose among various training strategies for learning the model parameters (e.g., stochastic gradient descent [86] and alternating least squares [102]). We will see more about the flexibility of these models in Chapter 6, where we propose a recommender

system using a revisited latent factor model to represent feelings on the attributes of the items.

2.2.2 *Content-Based Filtering Approaches*

Recommender systems with content-based filtering exploit attributes of items and users to produce recommendations by matching up the characteristics of a target user with the attributes of the items [153]. The characterization of items with *content information* that can effectively feed the recommendation process is usually a non-trivial task. In most cases, the items' attributes are simple keywords that are extracted from their description or metadata, albeit an increasing number of works has moved from a *keyword*-based to a *concept*-based approach by characterizing the items with semantic information extracted from structured knowledge sources, such as Wikipedia, DBpedia, Freebase, and BabelNet.

The steps to perform a content-based recommendation include [87] i) a *content analyzer* that uses information coming from the most diverse sources to represent the items in a specific description space (e.g., vector space model); ii) a *profile learner* that collects all the users' preference data and with a chosen strategy (e.g., probabilistic methods, relevance feedback, and k-nearest neighbors) tries to generalize this data in order to build the user profile; iii) a *filtering component* that suggests relevant items by matching the profile representations against the item descriptions.

Content-based recommender systems provide several advantages, including the possibility to explain the produced recommendation based on the attributes, and the absence of the cold-start problem for the items, since no collaborative information is needed to feed the recommender. Content-based filtering recommenders also present some shortcomings [87]: they include the cold-start problem for new users, because of the dependence of their profile on their rating history. Moreover, often these systems tend to suffer from an *overspecialization* problem (or lack of serendipity) since they tend to suggest merely items homogeneous with the ones experienced in the past. Finally, another known problem is the availability of structured knowledge about the domain, which would consistently improve the recommendation performance.

2.2.3 *Hybrid Approaches*

In some cases, the limitations of collaborative filtering methods and content-based filtering methods can be partially overcome by realizing a hybrid combination of the two approaches. Hybrid recommenders aim to improve the performance of the system by exploiting the best of both techniques. Some ways to implement hybrid recommenders consider implementing both the approaches separately and then using an aggregation function to merge the results, or injecting content information into a collaborative-filtering method, or vice versa, or building a system exploiting both the sources of information [7].

In Chapter 6, we present a hybrid recommender system that exploits both collaborative data and content information coming from knowledge graphs. In particular, the content information helps to effectively combine the embeddings of a latent factor model to obtain higher personalization with a reduced number of operations for training and inference.

2.2.4 *Evaluation*

We have already seen how many algorithms and approaches can be used to build a recommender system. Its choice depends on a large and variable number of factors, including a wide range of properties of the dataset (e.g., its sparsity, the ratio between users and items). Indeed, all these factors jointly impact different dimensions of the performance of the recommender system. All these dimensions, in turn, may enormously impact the user experience and the acceptance of both the recommendation and the whole system. This makes the evaluation of a recommender system a complex task that focuses on both users and items [91].

It is noteworthy that the evaluation of a recommender system should follow an online protocol, proposing recommendations to the users and waiting for their feedback. Practically, this is not feasible for most researchers due to the limited number of users they can involve in an experiment. To perform an offline evaluation, the set of available feedback is divided into a *training set* used for learning the utility function and a *test set* for assessing the quality of the learned function. Generally, the interactions of each user are independently split, thus generating for each of them the train set of items with an interaction $\mathcal{I}_u^{\text{train}} \subset \mathcal{I}_u$ and the test set of items with an interaction $\mathcal{I}_u^{\text{test}} \subset \mathcal{I}_u$. Depending on the strategy, a user can be provided with a recommendation list containing

only items present in the test set (i.e., already liked in the past) or containing only items not present in the training set (also items without ratings in the test set could be recommended) [188]. In the former case, we refer to *rated test-items* protocol, in the latter to *all unrated items* protocol. In this dissertation, we perform offline tests on all unrated items since they reflect much better the situation in many real-world recommendation tasks [188].

In the following, we briefly survey the metrics used along with this dissertation. Hereinafter, we mainly consider the recommendation problem in the form of top-k recommendation task, which is the mainstream of recommender system research. Indeed, our evaluation is oriented to assessing the quality of the ranking rather than evaluating the prediction score. Thus, the metrics in the following are presented with their definition specifically designed for evaluating the top-k/ranking task with the all unrated items protocol.

2.2.4.1 *Accuracy Metrics*

Accuracy metrics in top-k recommendation look for the presence of *relevant* items in the top-k elements of the recommended list, with k usually in $\{1, 5, 10, 50, 100\}$. Datasets with implicit feedback have an inherent definition of relevance (e.g., positive feedback as clicks, purchases, likes). Explicit feedback data like ratings, instead, have to be discretized into binary values according to a relevance threshold. Thus, $\mathcal{I}_u^+ \subseteq \mathcal{I}_u$ is the set of items relevant to user u, and $\mathcal{I}_u^{+,\text{test}} \subseteq \mathcal{I}_u^{\text{test}}$ contains the relevant items in the test set.

Definition 2.3 (Precision)**.** *For each user $u \in \mathcal{U}$, let $\mathcal{L}_u$ be a recommendation list ranked according to Eq. 2.2, and $\mathcal{L}_u^{(1,\dots,k)}$ its top-k. Then, the precision $P@k$ is defined as the average, over all the users, of the proportion of items in the top-k of each user that are also relevant to the user.*

$$P@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,\text{test}}|}{k} \tag{2.5}$$

Intuitively, precision measures the system's ability to reject any non-relevant item in the retrieved set. Recall, instead, aims to measure the system's ability to find all the relevant items.

Definition 2.4 (Recall)**.** *Given the premises in Definition 2.3, the recall R@k is defined as the average of the proportion of relevant items that are retrieved in top-k of each user.*

$$R@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,\text{test}}|}{|\mathcal{I}_u^{+,\text{test}}|} \tag{2.6}$$

To offer an aggregate measure of the accuracy, precision and recall are sometimes combined with each other in the $F1@k$ measure [173] computed as their harmonic mean:

$$F1@k = 2\frac{P@k \cdot R@k}{P@k + R@k} \tag{2.7}$$

Finally, a widely used accuracy metric is the *discounted cumulative gain* [107] from information retrieval, which assesses the quality of a ranking based on the position of the ranked elements.

Definition 2.5 (Normalized Discounted Cumulative Gain)**.** *For each user, let* $l_{u(i)} \in \mathcal{L}_u^{(1,\dots,k)}$ *the recommended item in position* $i \leq k$*, and* $v_{l_{u(i)}} = 1$ *if* $l_{u(i)} \in \mathcal{I}_u^{+,\text{test}}$*, 0 otherwise.*

$$nDCG@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i=1}^{k} \frac{2^{v_{l_{u(i)}}} - 1}{\log_2(i+1)} \tag{2.8}$$

In $nDCG@k$, the numerator represents the gain that u has from being recommended item $l_{u(i)}$, with its relative positions in the list discounted logarithmically.

2.2.4.2 *Beyond-Accuracy Metrics*

The presence of relevant items in the recommended list may not be sufficient for the evaluation of a recommender system. We have already mentioned that, among the other things, recommendations should be engaging for the users, should increase their trust and serendipity, and should generate revenue for content providers.

Some algorithms may provide accurate recommendations, but may polarize and homogenize users' interest towards a small portion of the items (the *short head*), i.e., the ones rated by the majority of users. This is usually caused by popularity bias since recommender systems often exacerbates and perpetuates social biases present in the rating data by reinforcing the popularity of already popular products (*rich-get-richer* effect). This behavior is often referred to as the *long tail* problem, since the unpopular items, which represent the vast majority of the items, are not recommended at all.

As a consequence, there is need for item diversity in recommendation lists to encourage serendipity, and increase users' satisfaction and sales. In the following, we introduce three sales diversity metrics widely employed in recommender systems.

Definition 2.6 (Item Coverage)**.** *Given the premises in Definition 2.3, the item coverage computes the number of items that are shown to at least one user.*

$$IC@k = \frac{|\bigcup_{u \in \mathcal{U}} \mathcal{L}_u^{(1,\dots,k)}|}{|\mathcal{I}|} \tag{2.9}$$

A higher value of item coverage implies high diversity of the recommendation lists, which may suggest a good performance in terms of personalization with respect to the users.

The other diversity measures considered in this dissertation are Gini Index (G) and Shannon Entropy (SE), that measure the distributional inequality [56], i.e., in this context, how unequally different items are recommended to the users. A recommender system equally recommending its items is likely providing diverse recommendations and equally favoring all the elements in the catalog.

Definition 2.7 (Gini Index)**.** *Given the premises in Definition 2.3, let $(\mathcal{L}^k, m)$ be a multiset, with $\mathcal{L}^k = \bigcup_{u \in \mathcal{U}} \mathcal{L}_u^{(1,\dots,k)}$ being the set of all the recommended items and $m : \mathcal{L}^k \to \mathbb{N}$ a function giving the number of times an item is recommended. Suppose that the elements $l \in \mathcal{L}^k$ are in ascending order by the value of $m(l)$ and $l^{(i)}$ denotes the item in the i-th position. Then, the Gini Index is defined as:*

$$\hat{G}@k = \frac{1}{|\mathcal{L}^k| - 1} \frac{\sum_{i=1}^{|\mathcal{L}^k|} (2i - |\mathcal{L}^k| - 1) m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \tag{2.10}$$

A value of $\hat{G}@k$ close to 0 reflects diversity and means that the items are equally recommended (all products have equal sales), whereas 1 represents concentration (a single item is recommended to all users). In the remainder of this dissertation, the values of Gini Index will be always provided in their *higher is better* version $G@k = 1 - \hat{G}@k$.

Definition 2.8 (Shannon Entropy)**.** *Given the premises in Definition 2.7, the Shannon entropy is defined as:*

$$SE@k = -\sum_{i=1}^{|\mathcal{L}^k|} \frac{m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \log \left(\frac{m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \right) \tag{2.11}$$

The entropy is 0 when a single item is always recommended, and $\log(|\mathcal{L}^k|)$ when all the items are recommended equally often.

2.2.4.3 *Algorithmic Bias*

The problem of unfair outputs in machine learning applications is well studied [46, 76] and also it has been extended to recommender systems [142]. Indeed, bias and fairness analysis is gaining momentum in the last years [67, 165], since it unveils several essential aspects of the recommenders' behavior.

One of the possible fairness problems with recommender systems, which has already come up in the previous sections, is connected to the *popularity bias* [36, 44]. Collaborative filtering recommenders, which will be used along with this thesis, typically emphasize popular items and leave the long-tail items underrepresented. Unfortunately, delivering only popular items does not enhance new item discovery and may be unfair to the creators of less popular or newer items since they are rated by a small set of users.

Three bias metrics will be used in this dissertation to evaluate the degree of underrepresentation of items from the long-tail. The first metric, whose abbreviation is ACLT, measures the fraction of the long-tail items the recommender has covered.

Definition 2.9 (Average Percentage of Long Tail Items [3])**.** *Given the premises in Definition 2.7, it is defined as:*

$$ACLT@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} |\mathcal{L}_u^{(1,\dots,k)} \cap \Gamma|, \tag{2.12}$$

where Γ *is the set containing the long-tail items.*

Moreover, we have also evaluated PopREO and PopRSP, which are specific applications of RSP and REO [237]. Notably, PopREO estimates the equal opportunity of items, encouraging the true positive rate of popular and unpopular items to be the same. PopRSP is a measure of statistical parity, assessing whether the ranking probability distributions for popular and unpopular items are the same in recommendation. For both the measures, lower values indicate that the recommendations are less biased.

Another factor that may impact fairness is the *bias disparity*, which represents the degree to which a group's preferences on various item categories fail to be reflected in the recommendations they receive [48]. This disparity among users will degrade users' satisfaction, loyalty, and effectiveness of recommender system. In detail, in this dissertation, we are sometimes interested in checking whether the recommendation list deviate from the users' original preferences towards different item

categories. In order to do that, we measure the bias disparity defined in Mansoury et al. [142] considering a unique group of users.

Definition 2.10 (Bias disparity)**.** *Let $\{C_1, \ldots, C_P\}$ a partition of $\mathcal{I}$ into P categories of items. Then, the bias disparity on a category of items C_i is defined as:*

$$BD(C_i) = \frac{B_R(C_i) - B_T(C_i)}{B_T(C_i)}, \tag{2.13}$$

where $B_T(C_i)$ is the source bias on category C_i, i.e., how much users were biased towards category C_i in the training set, and $B_R(C_i)$ is the bias on C_i in recommendation lists. In detail:

$$B_T(C_i) = \frac{\sum_{u \in \mathcal{U}} |C_i \cap \mathcal{I}_u^{+,\text{train}}|}{\sum_{u \in \mathcal{U}} |\mathcal{I}_u^{+,\text{train}}|} \frac{1}{\frac{|C_i|}{|\mathcal{I}|}}, \tag{2.14}$$

where the first term is the preference ratio of users on category C_i, and the denominator of the second term is the ratio of item category C_i in the dataset. Analogously:

$$B_R(C_i) = \frac{\sum_{u \in \mathcal{U}} |C_i \cap \mathcal{L}_u|}{\sum_{u \in \mathcal{U}} |\mathcal{L}_u|} \frac{1}{\frac{|C_i|}{|\mathcal{I}|}}. \tag{2.15}$$

Bias disparity shows positive or negative values if the recommender systems deviate the users respectively towards or away from a certain category. Thus, the closer to 0, the more the recommendation lists reflect the users' behavior encoded in the data.

3 Privacy-Preserving Machine Learning

OUTLINE

Massive data collection required for machine learning presents obvious privacy issues. Users' sensitive data is kept indefinitely by the companies that collect it, and it may be subject to a wide range of risks. Furthermore, some attackers with a white- or black-box knowledge of the machine learning model may mine sensitive user information.

This chapter analyzes the most crucial privacy risks and challenges in machine learning and presents architectural solutions like federated learning and algorithmic strategies, from differential privacy to encryption. Finally, we thoroughly survey the literature of privacy-preserving recommender systems with a new classification that, based on the architecture, privatization algorithm, and recommendation model, positions the works in an intersection of taxonomies.

Part of the content of this chapter has been presented in the tutorial "*Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications*" held at the 15th ACM Conference on Recommender Systems.

3.1 Introduction

In recent years, privacy has become a dominant concern in big data applications. The public opinion about this topic has been hugely spurred by large-scale data breaches such as Cambridge Analytica in 2018, which shared and harvested data from a massive number of users for political campaigning without their consent [49].

All the data coming from identity, biometrics, health, facial recognition, use of smartphones and Wi-Fi, use of transportation and vehicles, video surveillance are examples of information collected by corporations, organizations, and government and further used for analyses or for feeding machine learning or data mining algorithms. However, most of this data is personally related and contains private or sensitive information.

As we will see in this chapter, privacy violations may occur either in data collection, if the collector is not trustworthy, or when statistics are published, even though the collector is trusted and applies several simple anonymization techniques. For instance, in 2007, Netflix offered a 1,000,000 $ prize for a 10% improvement in its recommendation system and released a training dataset with 500,000 anonymous movie ratings. To protect the users, all personal information was removed and the users' identifiers replaced by randomly assigned numbers. However, by linking it with the International Movie DataBase (IMDB) dataset, Narayanan et al. [151] partly deanonymized the Netflix training dataset. The possibilities of denanomyzation are very large and can be applied to a wide range of contexts, from medical data [178] to national censorship datasets [191]. Even more, De Montjoye et al. [66] showed that in a dataset where the location of an individual is specified hourly, four location points are sufficient to uniquely identify 95% of the individuals.

In this context, statistical analyses, machine learning techniques, and artificial intelligence strategies have to be applied by keeping in mind the new privacy challenges that may limit the uptake of these applications [184]. The data leakage and privacy violation incidents have brought about heightened public awareness of the need for artificial intelligence systems to be able to preserve user privacy and data confidentiality. For all these reasons, the vulnerabilities of the systems should be analyzed in order to define and implement an adequate privacy model.

Recently, European Union, US Congress, and other jurisdictions legislated new laws about privacy. As an example, in 2018 the European Union proposed GDPR [80], which removes the default option for collecting, storing, and harnessing individuals' data and requires explicit authorization from the users to use their data. Other representative examples are the CCPA in California [52] and the Cybersecurity Law in China [187]. The GDPR applies to all uses of European data that could potentially identify a data subject, prohibiting the use of automated decision-making, so long as that decision-making occurs without human intervention and produces significant effects on data subjects. This is the case of any model that makes a decision without a human being involved in the decision (e.g., profiling users). In practice, users should allow data to be used by a model, and they should be able to do that at any time, and the consent management needs to be granular (allowing many different forms of consent), dynamic (allowing consent to be withdrawn), and user friendly. In most cases, this makes the deployment and management of machine learning models and their in-

put data increasingly difficult. Minimization and ephemerality of data collection to what and how long is necessary concerning the purposes of processing make data collection harder and harder. The traditional machine learning approaches based on centralization of data collection may be no longer compliant with these strict data protection laws. Solving the problem of data fragmentation and isolation while complying with the new stricter privacy-protection laws is a significant challenge for artificial intelligence researchers. Moreover, effectively preserving users' privacy is not as easy as limiting data collection since a privacy threat may happen at any stage of a data cycle. Indeed, the privacy of training data may not be the only concern, given that the model itself stores precious information able to predict future user preferences and behaviors. Thus, privacy and data protection must be guaranteed at all stages of an artificial intelligence system's life cycle.

In addition, while privacy concerns about disclosing personal information may frighten the users of machine learning systems, they actually want to receive accurate predictions, raising a *personalization-versus-privacy paradox*. In recommender systems, this is even more evident if we think of the great value for users in receiving recommendations galvanizing their intended purchases. However, they can be discouraged from sharing a large amount of sensitive data to receive such highly personalized recommendations.

In this chapter, we will first survey the privacy threats we can witness in machine learning. Then, the analysis will more deeply focus on the major risks observable in recommender systems. Finally, we will survey solutions based on privacy-oriented learning paradigms (e.g., federated learning) and privacy-preserving algorithmic strategies, mainly differential privacy and cryptography.

3.2 Why Privacy-Preserving Machine Learning

Privacy-preserving machine learning aims to equip machine learning with defense measures for protecting user privacy and data security. It should be distinguished from secure machine learning, which attempts instead to preserve integrity and availability of a machine learning system from intentional attacks like adversarial or poisoning attacks.

The main target of privacy attacks is the confidentiality of the users' sensitive data. These attacks can be performed by different adversaries and against different parties of the system's life cycle, during any stage of the data cycle, including data publishing, model training, and prediction.

Risks connected with data collection. A server hosting the recommender system usually collects the users' data in a centralized fashion. On the one hand, learning is for sure easier when data is stored in one place. On the other hand, a consequent risk is that online services may attempt to collect even unsolicited user data, either because it might be useful in the future or because it can be monetized. Moreover, centrally collected data can be subject to the risk of being shared with other parties, sold, or accessed by unauthorized actors. Oftentimes, collected data can be used by a malicious server and cross-linked with additional sources to infer sensitive user information such as age, gender, ethnicity, or political orientation [123, 233].

Typical recommender systems rely on a central entity, which accesses personal user data for the purpose of personalizing a service and expose data to the above-mentioned risks. The previous inference attack performed on the anonymized Netflix dataset is an example of this risk. Other examples include possible reconstruction of demographic information such as age, gender, ethnicity, or political orientation [212]. Moreover, by exploiting semantic relations between user interests in music, Abdelberi et al. [2] found demographic similarities between users with similar music tastes, thus inferring gender, age, and country.

Other privacy risks. Even when the data collector is trusted, or data collection step is somehow skipped, some scenarios may lead to unintended data disclosure [135]. In fact, some external entities may threaten user privacy during some steps of the data cycle. For instance, in *model extraction attacks*, an attacker attempts to duplicate or approximate the machine learning model without any prior knowledge about the model parameters or the training data. In detail, the attacker may attempt to query the model, observe the outputs and use equation-solving methods, meta-models and reverse engineering strategies to estimate the model parameters. Once the model has been stolen, the attacker can freely make prediction and inference or try to estimate statistical properties of the data (*feature estimation attack*) or even try to reconstruct some of the original data (*reconstruction attack*). In some cases, the attacker may perform a *membership inference attack* to learn if a specific sample was used for training the model. Usually, the adversary infers whether a sample belongs to the training set or not based on the machine learning model output.

Various works explored the possible privacy risks in machine learning, coming from malicious servers, external attackers, or colluding users [135]. For instance, Song et al. [186] show how service providers

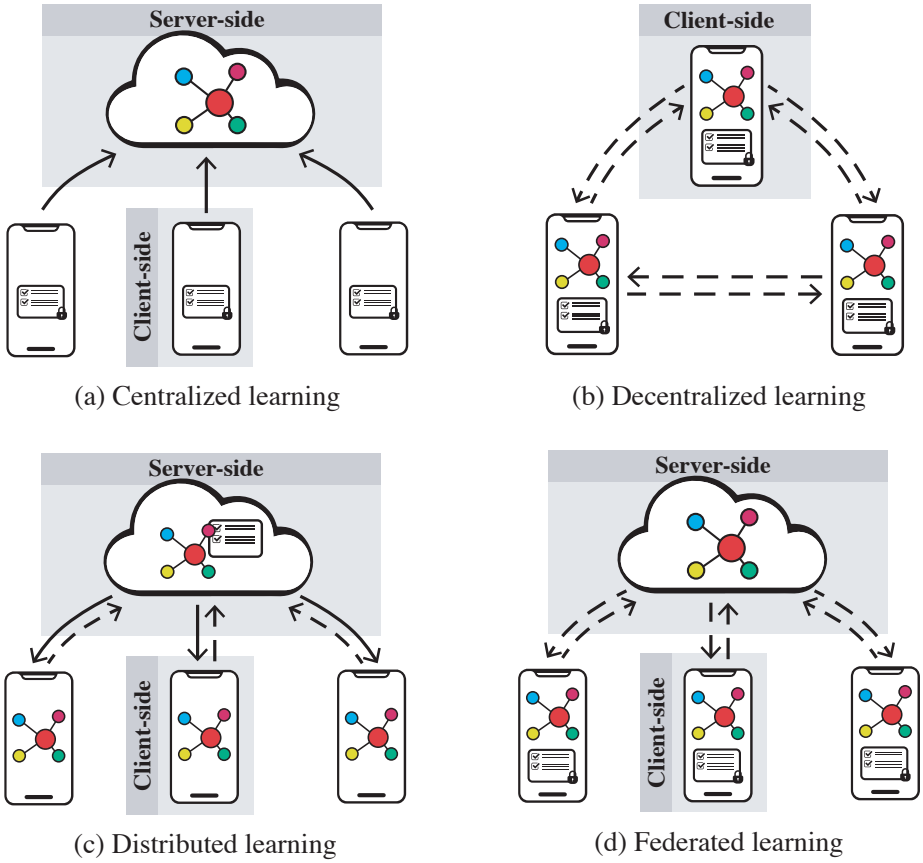
can modify the training algorithm so that the model encodes more information about the users dataset than is strictly necessary. Wang et al. [210] perform a reconstruction attack targeting user-level privacy, Tramèr et al. [194] perform an efficient model extraction. Examples of these risks in recommender systems include the work by Calandrino et al. [50] and will be presented along with possible solutions in Section 3.5.

3.3 Federated Learning for Privacy-by-Design

Privacy protection can be performed at various levels and with substantially different approaches, eventually integrated with each other to limit as much as possible the risks of sensitive data leakage. A prominent category of privacy-oriented practices attempts to mitigate risks connected with data collection by moving machine learning into architectures and training paradigms that minimize data centralization, thus limiting the data leakage threats and the ability of external entities to access user data or to infer new data. This category mainly includes the distributed architectures, which eliminate the single point of failure typical to centralized machine learning systems, including centralized recommenders.

In 2016, Google introduced the notion of *federated learning* [146], a paradigm for collaborative learning that provides an attractive structure for decomposing the overall machine learning workflow into modular units. In federated learning, many clients (e.g., a very large number of mobile or IoT devices) collaboratively train a machine learning model under the orchestration of a central server. The strategy adopted by federated learning is the decentralization of training data supported by an on-device learning approach.

Intuitively, each client of the federation downloads from the central server the current shared global model and improves it by learning from the sensitive data on its device and incorporating the changes in a focused update. Then, all the local updates are collected by the central server and combined in the new shared global model. From a privacy perspective, the federated learning paradigm is structurally built on practical strategies servicing by design the principle of *data minimization*: the collected updates are narrowly scoped to contain the minimum information necessary for the specific learning task (*focused collection*), aggregation is performed as early as possible (*early aggregation*), and both collected and processed data is discarded as soon as possible (*minimal retention*). By moving computation to the node



(a) Centralized learning (b) Decentralized learning

(c) Distributed learning (d) Federated learning

Figure 3.1: Information flow over the network in four machine learning paradigms. Solid lines represent training data flow, dashed lines represent model parameter flow.

where that data resides, federated learning also embraces the principle of *data locality*, thus guaranteeing to users *data ownership*, i.e., possession and control on their own data. All these principles, together with data anonymization, eventually provided by privacy-preserving technologies (see Section 3.4), ensure user privacy and data confidentiality, especially when considering specific goals that can be advanced by computation on privacy-sensitive user data on which users claim their ownership.

In the following, supported by Figure 3.1, we compare federated learning with the other most used learning paradigms. In detail, four architectures are compared, focusing on the location and distribution of data and on the information exchanged over the network: in the figure, the solid lines represent training data flow, the dashed lines represent model parameters flow.

Centralized learning A central server is in charge of data collection, data aggregation, and model training (the data flows from clients

to the server). In terms of privacy, the central server is a single point of failure.

Decentralized learning Also known as a peer-to-peer (P2P) architecture, in this scenario, clients share resources directly with each other without the need for centralized control. Data can remain on the local devices, but some communication and coordination strategies are needed (e.g., blockchain).

Distributed learning A central server holds all the training data and uniformly redistributes them to other computational nodes in order to exploit their computational resources. This learning paradigm is not conceived for privacy concerns, and the data needs to be previously collected.

Federated learning A central server makes use of computational resources of other clients without the need to collect their data: users' data remain on their clients, and only focused parameters are exchanged between the server and the clients.

In recent years, federated learning is being applied in commercial digital products. Google makes extensive use of federated learning in the Gboard mobile keyboard [94], Apple is using cross-device federated learning for applications like the QuickType keyboard, and the vocal classifier for "Hey Siri" [27, 28]. As an innovative modeling mechanism that can build privacy-oriented personalized models on data decentralized among multiple parties, federated learning created a vibrant and dynamic research community aiming to integrate this paradigm in many important fields such as sales, finance, healthcare, education, urban computing, edge computing, blockchain, and recommender systems (see Section 3.5).

In the following, we formalize federated learning and describe its defining characteristics and challenges, highlighting the aspects of major interest for this thesis.

3.3.1 *Formal Definition*

Consider we want to solve a machine learning problem with any finite-sum objective function:

$$\min_{\Theta} f(\Theta), \quad \text{where } f(\Theta) = \frac{1}{|\mathcal{K}|} \sum_{i \in \mathcal{K}} \ell(i; \Theta), \tag{3.1}$$

with $\ell_i(\Theta)$ representing the loss of the i-th training sample made when using the machine learning model Θ, and $\mathcal{K}$ is the set of samples used to solve the problem by training the model Θ. In this context, federated learning assumes the existence of a federation of clients that want to jointly solve the problem above.

Definition 3.1 (Federated learning)**.** *Let Θ be the parameters of a machine learning model, and consider a learning scenario where the objective is to minimize a generic loss function. Federated learning is a paradigm in which the learning problem is solved by training the model across the devices of the users of a federation $\mathcal{U}$, each of them using its private data, under the coordination of a central server S. The training is performed without requiring the users to share or exchange their raw data with any other party in the system.*

Federated learning assumes a synchronous update scheme that proceeds in *rounds of communication*. The federation $\mathcal{U}$ is a fixed set of users, each with a fixed local dataset[1]. Assuming the training data in $\mathcal{K}$ partitioned and distributed over the clients in $\mathcal{U}$ in user-specific training sets $\mathcal{K}_u$ with $u \in \mathcal{U}$, a locally computed loss f_u can be formulated as it follows:

$$f_u(\Theta) = \frac{1}{|\mathcal{K}_u|} \sum_{i \in \mathcal{K}_u} \ell(i; \Theta), \tag{3.2}$$

and the global objective function reformulated as:

$$\min_{\Theta} \tilde{f}(\Theta), \quad \text{where } \tilde{f}(\Theta) = \sum_{u \in \mathcal{U}} \frac{|\mathcal{K}_u|}{|\mathcal{K}|} f_u(\Theta). \tag{3.3}$$

The main problem with the formulation in Equation 3.3 is related to the distribution of the training data. In federated settings, $\mathcal{K}$ is an ideal training set, while $\mathcal{K}_u$, for all $u \in \mathcal{U}$, is a local dataset formed by the private samples of user u. The training data on a given client is local data from end-user devices, typically based on the behavior of a particular user, and hence not representative of the whole population distribution (*non-IID data*). Thus, it does not hold the equation $\mathbb{E}_{\mathcal{K}_u}[f_u(\Theta)] = f(\Theta)$,

1 In federated learning literature, each user corresponds with a client. Thus, hereinafter we will use the terms *user* and *client* interchangeably.

which instead would hold with uniformly distributed data. With these premises, we can observe that:

$$\begin{aligned}\mathbb{E}[\tilde{f}(\Theta)] &= \sum_{u\in\mathcal{U}} \mathbb{E}_{\mathcal{K}_u}\left[\frac{|\mathcal{K}_u|}{|\mathcal{K}|} f_u(\Theta)\right] = \sum_{u\in\mathcal{U}} \mathbb{E}_{\mathcal{K}_u}\left[\frac{|\mathcal{K}_u|}{|\mathcal{K}|}\frac{1}{|\mathcal{K}_u|}\sum_{i\in\mathcal{K}_u}\ell(i;\Theta)\right] \\ &= \frac{1}{|\mathcal{K}|}\sum_{u\in\mathcal{U}} \mathbb{E}_{\mathcal{K}_u}\left[\sum_{i\in\mathcal{K}_u}\ell(i;\Theta)\right] = \frac{1}{|\mathcal{K}|}\sum_{u\in\mathcal{U}} \mathbb{E}_{\mathcal{K}_u}[f_u(\Theta)]\end{aligned} \tag{3.4}$$

Given that, in general, $\mathbb{E}_{\mathcal{K}_u}[f_u(\Theta)] \neq f(\Theta)$, we can only hope for the unbiasedness of the mean in Eq. 3.4, while the individual clients' contributions will be biased towards their local datasets and each of them will tend deviate the model towards a different convergence point [232].

Federated Averaging

From an algorithmic point of view, in McMahan *et al.* [146], authors realize *Federated Averaging* (*FedAvg*), a federated optimization inspired by stochastic gradient descent (SGD) that serves as a template for training in federated settings.

For a certain number of rounds of communication, until the training is stopped, the central server S repeats the following steps:

1. selects a subset of clients $\mathcal{U}^- \subseteq \mathcal{U}$;
2. sends them the current global model parameter Θ;
3. each selected client $u \in \mathcal{U}^-$ locally updates the received model (e.g., with SGD) by using its own private data $\mathcal{K}_u$;
4. the clients in $\mathcal{U}^-$ return to S the gradient $\nabla f_u(\Theta)$;
5. the central server S performs a weighted global aggregation of the received local gradients and updates the model as it follows:

$$\Theta \leftarrow \Theta - \alpha \sum_{u\in\mathcal{U}^-} \frac{|\mathcal{K}_u|}{|\mathcal{K}|}\nabla f_u(\Theta), \tag{3.5}$$

where α is the learning rate. Equivalently, the clients in $\mathcal{U}^-$ can compute $\Theta_u \leftarrow \Theta - \alpha\nabla f_u(\Theta)$, so that the server can build the new Θ as:

$$\Theta \leftarrow \sum_{u\in\mathcal{U}^-} \frac{|\mathcal{K}_u|}{|\mathcal{K}|}\Theta_u. \tag{3.6}$$

In principle, SGD can be applied naively to the federated optimization problem, selecting one client per round of communication performing a single batch gradient calculation. However, this approach

would require a very large number of rounds of training to produce good models. Therefore, FedAvg, with a little cost in wall-clock time, involves more clients per round, each of them locally taking more SGD steps (i.e., more *local computation* is added before sending the update, thus making the algorithm more communication-efficient).

3.3.2 *System Challenges in Federated Learning*

Beyond the statistical challenges presented above, a federated learning setting can encompass a wide range of other problems, mainly coming from the massive number of devices and their limited communication bandwidth [204]. Moreover, only a fraction of clients are available at any one time, and their reliability is very low (e.g., because the device becomes ineligible due to battery, network, or idleness issues). It is interesting to notice how federated learning was originally conceived as a *cross-device* architecture involving up to 10^{10} users' mobile devices and edge device applications, e.g., for Google's Gboard mobile keyboard [94].

Nonetheless, federated learning also extended towards *cross-silo* architectures [110], for multiple organizations willing to train a model in inherently sensitive domains (e.g., financial or medical). For example, in healthcare, regulations prohibit medical institutions from sharing medical data (e.g., to improve medical imaging performance [197]); however, federated learning makes centralization unnecessary by allowing data to remain isolated in the individual organizations while still improving medical AI models [224]. In cross-silo federated learning, some challenges are reduced due to the small number of participating organizations and their computational nodes' relatively high availability and reliability.

To give an idea of the number of challenges and possible solutions raised with federated learning, we can refer to the increasing number of surveys [110, 129, 130, 133, 219] effectively organizing the vast literature about known federated learning problems such as communication costs, resource allocation, systems heterogeneity, statistical heterogeneity, security (e.g., model poisoning [33] and Byzantine adversaries [189]), fairness, and bias.

3.3.3 *Privacy Challenges in Federated Learning*

In a system with perfect privacy, each actor would learn nothing more than the information needed to play its role. For instance, analysts should only observe some quality metrics to decide whether to deploy the model or not. End users, in turn, should only observe their predictions and nothing else. However, providing such perfect privacy features in machine learning systems is generally considered a tricky and daunting activity.

Federated learning provides a strategy for decomposing the data lifecycle in modular units that can be independently studied from the privacy perspective. We have previously mentioned that federated learning embodies the data minimization principle and focused collection, which reduce the risk of data leak and offer significant privacy improvements over centralizing all training data. Nevertheless, the naive definition of federated learning is not able to provide rigorous and formal privacy guarantees with respect to the other privacy risks presented in Section 3.2.

Lyu et al. [140] reviewed some privacy issues regarding federated learning, while a wide range of works in literature focuses on improving the privacy preservation of federated learning on private user data and avoiding privacy leakages [138, 159, 164, 211, 213].

Indeed, depending on the malicious actor, different threats can be presented to the federated architecture. For instance, a malicious client or a malicious server can inspect all the messages exchanged and tamper with the training process. An honest-but-curious server may inspect the received gradient updates and try to infer training examples held by the users. Analogously, engineers and analysts can observe the global model and may potentially learn some information from that, as well as malicious actors, may gain a black-box or white-box access to the deployed model on the thousands or millions of end devices. Techniques from secure computation, including secure multi-party computation, and differential privacy, a rigorous mathematical definition helping to understand how much information about users may be eventually disclosed, are of particular relevance to addressing these concerns. Section 3.4 formally presents these privacy-preserving techniques and surveys their use in privacy-preserving machine learning, with a focus on their applicability in federated learning.

3.3.4 *Personalization with Federated Learning*

Due to its distributed architecture, federated learning may allow scenarios with different users running inference with different model parameters. On-device training and inference allow for the local personalization of global models to better suit the needs of individual users. For instance, a language model for next word prediction can be positively affected by on-device personalization if we locally account for different uses of the language. This outcome can be obtained both by fine-tuning a global model on locally stored data [206] and enriching the model with a variety of other user and context features.

In Chapter 6 of this thesis, we present a new federated recommender system that integrates on the users' devices some local extensions of the model accounting for users' personal feelings on the item characteristics. We experimentally prove that such a on-device extension of the model can provide a high degree of personalization, which contributes to improving the recommender system's performance.

3.4 Algorithms for Privacy Protection

In the following, we present an overview of the privacy-preserving tools and techniques most used in machine learning for protecting models and data against malicious actors.

3.4.1 *Differential Privacy*

Differential privacy [77] represents a formal mathematical definition for quantifying and limiting information disclosure about individuals.

Definition 3.2 ((ϵ, δ)-differential privacy[2])**.** *A randomized mechanism $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathcal{R}$ preserves (ϵ, δ)-differential privacy if given any two adjacent datasets $\mathcal{K}_1$ and $\mathcal{K}_2$ (i.e., they differ by only one record[3]), and for all $\mathcal{S} \subseteq \mathcal{R}$,*

$$\Pr\left[\mathcal{M}(\mathcal{K}_1) \in \mathcal{S}\right] \leq e^{\epsilon} \Pr\left[\mathcal{M}(\mathcal{K}_2) \in \mathcal{S}\right] + \delta \tag{3.7}$$

If $\mathcal{M}$ is $\delta = 0$, we say that $\mathcal{M}$ is ϵ-differentially private.

2 Note that, in the context of federated learning, a dataset/database $\mathcal{K} \in \mathbb{N}^{|\mathcal{X}|}$ is often represented as a collection of records from a universe $\mathcal{X}$, with each entry representing the number of elements of the universe in the database.

3 In other problems, $\mathcal{K}_1$ and $\mathcal{K}_2$ may correspond to datasets such that $\mathcal{K}_2$ can be obtained from $\mathcal{K}_1$ by adding or subtracting all the records of a single user (*user-level differential privacy* [147]).

Notably, differential privacy provides a theoretic security guarantee for the outcome of a function to be similar when changing or removing one record. The degree of similarity depends on the privacy budget ϵ, where a smaller value corresponds to increased privacy. The quantity

$$\ell_{\mathcal{M}(\mathcal{K}_1),\mathcal{M}(\mathcal{K}_2)}(\xi) = \ln\left(\frac{\Pr\left[\mathcal{M}(\mathcal{K}_1) = \xi\right]}{\Pr\left[\mathcal{M}(\mathcal{K}_2) = \xi\right]}\right), \tag{3.8}$$

known as *privacy loss*, corresponds to the actual information gain that the attacker is obtaining about the presence of a target record within the dataset by observing the output ξ. Therefore, differential privacy can be used to resist the membership inference attack. The ϵ-differential privacy ensures that, for every run of the randomized mechanism, the output observed is almost equally likely to be observed on every neighboring database (the privacy loss is limited by ϵ). The (ϵ, δ)-differential privacy, instead, ensures that, for all adjacent datasets $\mathcal{K}_1$ and $\mathcal{K}_2$, the absolute value of the privacy loss will be bounded by ϵ with probability at least $1 - \delta$. However, there is a probability δ to find a database $\mathcal{K}_2$ such that ξ is much more likely to be produced on $\mathcal{K}_2$ than on $\mathcal{K}_1$. Informally, it corresponds to a (very small) probability that something goes wrong with differential privacy.

The practical idea behind differential privacy is to use the randomized mechanism $\mathcal{M}$ to perturb the output of a query randomly. As a consequence, the attacker has limited possibilities to distinguish the presence of an individual from the query result. Typically, the mechanism $\mathcal{M}$ works by adding noise to the output of a function according to its sensitivity or according to an exponential distribution among a set of discrete values. Due to its definition, the smaller the value of ϵ, the higher the privacy, but the more difficult it is to realize a mechanism with high accuracy, i.e., smaller values of ϵ entail accuracy degradation.

Definition 3.3 (Sensitivity)**.** *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two adjacent datasets, and $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^n$ a function interpreting queries that map a database to n real numbers. The sensitivity of f is the maximum change in the output of f using a dataset instead of the other one:*

$$\Delta f = \max \|f(\mathcal{K}_1) - f(\mathcal{K}_2)\|, \tag{3.9}$$

where $\|\cdot\|$ is the norm of the vector. Based on the usage of l_1- or l_2-norm, we refer to l_1-sensitivity $\Delta_1 f$ or l_2-sensitivity $\Delta_2 f$.

Intuitively, the sensitivity captures the magnitude by which an individual in the dataset can change f in the worst case, i.e., the maximum

contribution of a sample to the final value[4]. To effectively evaluate Δf in the worst case, all the possible entries in the dataset should be considered. This value gives the uncertainty in the released response that should be introduced in order to hide the presence of a sample in the dataset, i.e., how much the response should be perturbed.

The Laplace distribution is one of the most effective sources of noise used for providing a mechanism with differential privacy.

Definition 3.4 (Laplace distribution)**.** *The Laplace distribution is the distribution with probability density function:*

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(- \frac{|x|}{b} \right), \tag{3.10}$$

where b is the scale of the random function. Hereinafter, we will simplify the notation with $\text{Lap}(b)$ *to denote a random variable* $X \sim \text{Lap}(b)$.

Definition 3.5 (Laplace mechanism)**.** *Given any function* $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}^n$ *with* l_1*-sensitivity* $\Delta_1 f$*, the Laplace mechanism is defined as:*

$$\mathcal{M}_L(\mathcal{K}, f, \epsilon) = f(\mathcal{K}) + \langle L_1, \ldots, L_n \rangle, \tag{3.11}$$

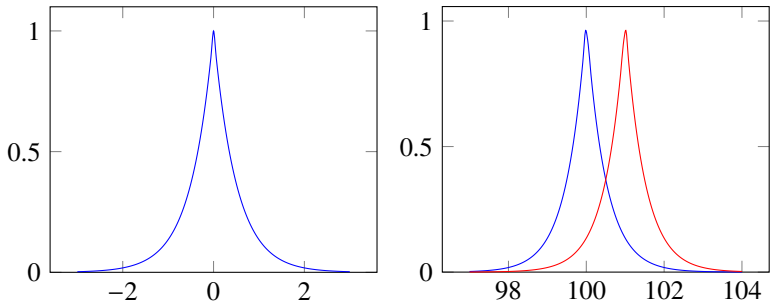
where L_i *are random variables drawn from* $\text{Lap}(\Delta_1 f / \epsilon)$.

Theorem 3.1. *The Laplace mechanism is proven to preserve* ϵ*-differential privacy.*

In the following, we intuitively explain how the Laplace mechanism noise can preserve differential privacy, according to the definition.

Example 3.1. Let us suppose our attacker wants to know about the presence of a single user in the database and is asking how many users in the database are older than 25. Suppose the attacker already knows about the presence of 100 users older than 25 in the database, so the answer 101 would reveal the presence of the target user. The sensitivity of this counting function is 1, given that a single contributes at most with 1 to the final count. Then, to provide $(0.5, 0)$-differential privacy with Laplace mechanism, $\text{Lap}\left(\frac{1}{0.5}\right)$ noise (see Figure 3.2a) should be added to the real count.

4 In general, sensitivity should be measured according to what differential privacy is meant to protect (e.g., the contribution of a sample in sample-level differential privacy or the overall contribution of a user's records in user-level differential privacy). However, the evaluation of sensitivity is a general idea and can be otherwise modeled for peculiar needs.



(a) Laplacian distribution (b) Probability of noisy values

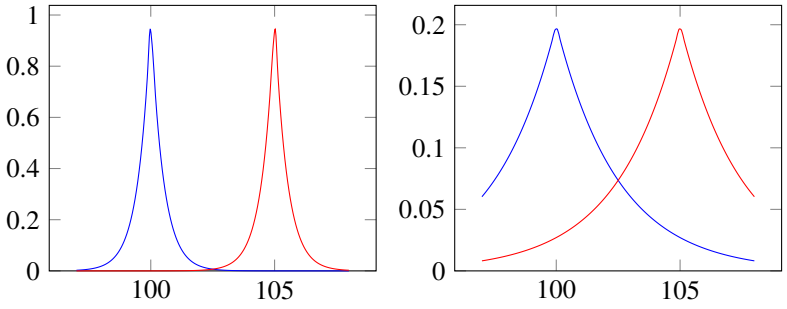
Figure 3.2: Laplacian mechanism on the adjacent counts 100 (red) and 101 (blue). Once a value has been released, an attacker can suppose with higher probability that it comes from the red or the blue line.

In the example above, based on the release value, the attacker could increase or decrease his initial knowledge (see Figure 3.2b). For instance, releasing the value 103, the hypothesis 101 would be more likely. Releasing the value 98, the attacker would think of 100 as a more likely hypothesis. Anyway, the ratio between these likelihoods is proven to be always limited by e^{ϵ}, which corresponds to guarantee $(\epsilon, 0)$-differential privacy.

Example 3.2. Suppose each individual can have a larger contribution on a single count (e.g., the number of ratings left by a user on a movie platform). Suppose that a user left five ratings so that her presence affects the count with 5. If $\epsilon = 0.5$, the distributions with $\text{Lap}(\frac{1}{0.5})$ noise of the possible answers coming from the dataset with or without the target user (see Figure 3.3a) are now very far, and their ratio is limited by $e^{5\epsilon}$. Thus, only $(5\epsilon, 0)$-differential privacy is guaranteed. To provide $(\epsilon, 0)$-differential privacy and confuse the attacker, much more noise is needed. In particular, to provide $(\epsilon, 0)$-differential privacy, the Laplacian noise should be scaled by $5/\epsilon$ (see Figure 3.3b).

In general, the contribution of an individual may be unbounded. In that case, the value of the function f should be clamped to a maximum in order to estimate the sensitivity of the function. Then, the noise can be safely applied.

A fixed but arbitrary list of m counting queries can be viewed as a vector-valued query. As long as a single individual might change the value of only one count, adding noise based on the sensitivity $\Delta_1 f$ is enough to guarantee $(\epsilon, 0)$-differential privacy. In case the individual can affect every count, releasing all the counts degrades the Laplacian mechanism to $(m\epsilon, 0)$-differential privacy. Equivalently, answering m



(a) Mechanism with Lap($\frac{1}{0.5}$) (b) Mechanism with Lap($\frac{5}{0.5}$)

Figure 3.3: Laplacian mechanism on the adjacent counts 100 (red) and 105 (blue). Larger sensitivities require scaling the Laplacian noise.

times to the same query degrades the algorithm to $(m\epsilon, 0)$-differential privacy. This happens because the average of the answer given by each instance of the mechanism will eventually converge to the true value of the statistic. The other side of the coin is that the combination of more differentially private mechanisms still guarantees differential privacy. This behavior is proven by the composition theorem.

Theorem 3.2 (Composition theorem)**.** *Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathcal{R}^i$ be an (ϵ_i, δ_i)-differentially private algorithm, for $i \in 1, \ldots, m$. Then the composed mechanism $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \rightarrow \prod_{i=1}^{m} \mathcal{R}^i$, with $\mathcal{M}(\mathcal{K}) = \langle \mathcal{M}_1(\mathcal{K}), \ldots, \mathcal{M}_m(\mathcal{K}) \rangle$, guarantees $(\sum_{i=1}^{m} \epsilon_i, \sum_{i=1}^{m} \delta_i)$-differential privacy.*

Thus, having a privacy budget ϵ, it can be split between the different answers or mechanisms that compose a more complex algorithm.

An alternative to adding Laplacian noise is to add Gaussian noise. In this case, rather than scaling the noise to the l_1-sensitivity, the l_2-sensitivity is used. This sensitivity grows much more slowly than the l_1-sensitivity with respect to the number of released statistics (i.e., elements of the released vector) that are affected by a single data point, thus reducing the required noise. However, Gaussian mechanism provides (ϵ, δ)-differential privacy with $\delta > 0$. It means that it allows for an (extremely unlikely) distinguishing event that makes the attacker understand whether the dataset $\mathcal{K}_1$ or $\mathcal{K}_2$ has been used. The exact formula that gives the parameters for calibrating the Gaussian noise based on ϵ, δ and $\Delta_2 f$ has been first proposed by Dwork et al. [77], then improved in Balle et al. [34].

Another way for realizing differential privacy is the exponential mechanism. The exponential mechanism was designed for situations in which the best response should be chosen but adding noise directly to the computed quantity can completely destroy its value (e.g., in the

context of the auction theory [77]) or when the output is a categorical value (e.g., choosing the most convenient day for a meeting). Given a quality function q that scores the possible outcomes of a calculation, where higher scores are better, the quality function induces an exponential distribution over the output domain, from which the exponential mechanism samples the outcome to be released, while ensuring ϵ-differential privacy.

Definition 3.6 (Exponential mechanism)**.** *Given an utility function* $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \rightarrow \mathcal{R}$*, which maps database/output pairs to utility scores, the exponential mechanism returns each* $r \in \mathcal{R}$ *with probability proportional to* $\exp\left(\frac{\epsilon u(\mathcal{K},r)}{2\Delta u}\right)$*, where*

$$\Delta u = \max_{r \in \mathcal{R}, \mathcal{K}_1, \mathcal{K}_2} \|u(\mathcal{K}_1, r) - u(\mathcal{K}_2, r)\|_1 , \tag{3.12}$$

for any possible pair of adjacent datasets $\mathcal{K}_1$ *and* $\mathcal{K}_2$*.*

Theorem 3.3. *The exponential mechanism is proven to preserve* ϵ*-differential privacy.*

The applications of differential privacy to machine learning considerably differ according to the potential risks, the system settings, and the mechanism used [234]. The rationale for its application includes limiting privacy issues due to centralization, model-inversion attacks, attackers having the full knowledge of the training mechanism and the model parameters or attackers making malicious inference with the model's inputs and outputs [234]. However, directly applying noise within a machine learning model yields inferior performance due to the high sensitivity of non-convex functions, which require too much noise, and the additivity of privacy budget since iterative training processes correspond to the composition of multiple mechanisms. Indeed, differential privacy trades accuracy for privacy. As a possible solution to these challenges, Abadi et al. [1] consider a centralized setting where they clipped the objective function of a deep learning model to bound its sensitivity and applied a moment accountant method to estimate the consumed privacy loss. On the contrary, Shokri et al. [184] designed a distributed deep learning model with differential privacy applied to the parameter updates shared during training. In general, the methods proposed in literature rely on adding noise to the execution process of an existing optimization algorithm or perturbing the objective functions of the given optimization problem.

It is clear how the system architecture and the trustworthiness of the involved actors play a fundamental role in determining how the

differentially private mechanism should be implemented. For instance, in a centralized scenario with a trusted data collector, each user sends data to the aggregator without noise, while one straightforward way to protect the privacy of training data against external leaks is to add noise to the trained parameters resulting from the learning algorithm or to the final predictions. A trusted aggregator can also be present in a federated scenario so that users do not need to send noisy gradients. This favorable situation, known as *central differential privacy*, allows applying the differentially private mechanism once, with big advantages in terms of accuracy. On the contrary, an untrusted aggregator requires the use of differential mechanisms before the data or the model parameters are submitted to the server. This may also be useful when the network layer is a potential point of failure. This model, already implemented in commercial products [79, 193] and known as *local differential privacy*, has the advantage of no longer requiring trust, but the resulting total noise is much larger.

3.4.2 *Secure Multi-Party Computation*

Secure multi-party computation (SMPC) [221] is a subfield of cryptography considering multiple parties with the objective of jointly computing a function from their private inputs, without revealing them to each other.

Secret sharing is the most important SMPC technique, allowing to distribute the computation across multiple parties and to perform training and inference on encrypted data (property of *homomorphism*).

Definition 3.7 (Secret sharing [180])**.** *Given a secret x, secret sharing splits it into random parts (a.k.a. shares) and distributes them to different parties so that each party has only one share and thus only one piece of the secret. Depending on the specific secret sharing schemes used, all or a known threshold of shares are needed to reconstruct the original value of x.*

The shares contain only a piece of information, which is meaningless without a proper number of other shares. Secure multi-party computation allows to exactly reconstruct the value of x, thus making the process lossless in terms of accuracy. However, the execution of such schemes requires significant computational and communication overheads.

Definition 3.8 (Arithmetic secret sharing)**.** *Consider that a party $u \in \mathcal{U}$ wants to share a secret x among the parties in $\mathcal{U}$ in a finite field*

F_q. To share x, the party u randomly samples $|\mathcal{U}| - 1$ values $\{x_i\}_{i=1}^{|\mathcal{U}|-1}$ from $\mathbb{Z}_q$ and sets

$$x_{|\mathcal{U}|} = x - \sum_{i=1}^{|\mathcal{U}|-1} x_i \mod q. \tag{3.13}$$

The secret can be reconstructed as $x = \sum_{i=1}^{|\mathcal{U}|} x_i \mod q$.

With this scheme, $|\mathcal{U}|$ shares are needed to reconstruct the value of x. Thus, in arithmetic secret sharing, x remains hidden as long as at most $|\mathcal{U}| - 1$ parties collaborate to violate the secret. Other schemes require a smaller number of parties to reconstruct the original value, thus accounting for parties dropping out from the protocol. For instance, Shamir's secret sharing [180] controls the minimum number R of shares needed to reconstruct a secret x. It samples $|\mathcal{U}|$ points from a polynomial f of degree R (with $R < |\mathcal{U}|$) with the condition that $f(0) = x$. Therefore, the polynomial (and hence x) can be retrieved by knowing just R points.

Arithmetic secret sharing is homomorphic with respect to addition with other encrypted values and with respect to addition and multiplication with non-encrypted values. This means that, once the secret has been shared, we can perform addition with other shares and finally retrieve the sum of the secrets.

Example 3.3. Three colleagues, Alice, Bob, and Caroline, want to compute their average salary without revealing to each other how much they earn. Suppose that Alice earns 1,000€, Bob earns 2,000€, and Caroline receives 3,000€. To solve the problem, they share their secret salaries as reported in Table 3.1. Each party can now locally sum (by columns in the table) the three received secret shares, actually obtaining a share of the sum of the three secrets. The three obtained shares can now be reconstructed to obtain the sum of the three initial secrets (6,000€) and their average.

Other protocols extend SMPC to support more operations. For instance, SPDZ [65] introduces multiplication with the support of an honest crypto provider. Moreover, it shifts part of the computation in an offline preprocessing phase, thus speeding up the online phase.

A common aspect of cryptographic solutions is that operations are often done on a finite field, which poses difficulties when representing real numbers. To make machine learning models work with secure multi-party computation, they must operate on normalized quantities and rely on careful quantization.

Table 3.1: Example of additive secret sharing for computing the average of three salaries.

	Alice	**Bob**	**Caroline**
Alice (1,000€)	500 €	300 €	200 €
Bob (2,000€)	-800 €	1,000 €	1,800 €
Caroline (3,000€)	€	3,500 €	-500 €
Local sum	-300 €	4,800 €	1,500 €
Global sum		**6,000 €**	
Average		**2,000 €**	

Most SMPC-based machine learning approaches leverage a two-phase architecture comprising an offline phase and an online phase. The majority of cryptographic operations are conducted in the offline phase. Then the machine learning model is then trained in the online phase. Secure multi-party computation fits well federated learning, for instance, distributing arithmetic shares of the private data [150] or guaranteeing the privacy of each user's model gradients [42].

Nevertheless, these works are typically of practical implementation in cross-silo settings. Porting these protocols to the cross-device setting is not straightforward, as they require a significant amount of communication and are generally sensitive to clients dropping out [42].

3.4.3 *Homomorphic Encryption*

Homomorphic encryption (HE) is a cryptographical scheme allowing certain mathematical operations to be performed directly on ciphertexts without prior decryption. Formally, given two messages x_1 and x_2 in the plaintext space $\mathcal{X}$ and a function Enc mapping them in a space C,

$$\text{Enc}(x_1 \odot_{\mathcal{X}} x_2) \leftarrow \text{Enc}(x_1) \odot_C \text{Enc}(x_2), \tag{3.14}$$

with $\leftarrow$ meaning *can obtained from*. For instance, with the $\text{Enc}_{\mathcal{P}}$ and $\text{Dec}_{\mathcal{P}}$ functions from the Pailler's scheme [156], given two messages $x_1, x_2 \in \mathbb{Z}_q$ and $k \in \mathbb{N}$, the following equalities hold:

$$\text{Dec}_{\mathcal{P}}(\text{Enc}_{\mathcal{P}}(x_1)\text{Enc}_{\mathcal{P}}(x_2) \mod n^2) = x_1 + x_2 \mod n, \tag{3.15}$$

$$\text{Dec}_{\mathcal{P}}(\text{Enc}_{\mathcal{P}}(x_1)^{\text{Enc}_{\mathcal{P}}(x_2)} \mod n^2) = x_1 x_2 \mod n, \tag{3.16}$$

$$\text{Dec}_{\mathcal{P}}(\text{Enc}_{\mathcal{P}}(x_2)^{\text{Enc}_{\mathcal{P}}(x_1)} \mod n^2) = x_1 x_2 \mod n, \tag{3.17}$$

$$\text{Dec}_{\mathcal{P}}(\text{Enc}_{\mathcal{P}}(x_1)^{k} \mod n^2) = k x_1 \mod n. \tag{3.18}$$

Based on the supported operators, homomorphic encryption methods can be divided into three categories [5]:

- *partially homomorphic encryption* (e.g., Paillier [156]), that can reach additive homomorphism or multiplicative homomorphism;
- *somewhat homomorphic encryption* (e.g., Brakerski et al. [47]), where operations can be applied for a limited number of times, since noise is used;
- *fully homomorphic encryption* (e.g., Dijk et al. [74]), that allows unlimited number of additions and multiplications over cyphertexts (it performs a highly costly operation called bootstrap).

Homomorphic encryption schemes have been widely studied in machine learning algorithms. For example, Hardy et al. [95] leverage Paillier's scheme in secure gradient descent to train a two-party logistic regression model. Also use homomorphic encryption during the training of logistic regression with an approximation method, improving computational efficiency. Gilad-Bachrach et al. [88] proposed CryptoNets, a homomorphic encryption-based methodology that allows secure inference of encrypted queries over already trained neural networks on cloud servers, while FedMF [57] uses Paillier's scheme for secure federated matrix factorization.

Homomorphic encryption shows clear advantages in machine learning, including the possibility of performing inference on encrypted data so that the model owner never sees the client's private data and therefore cannot leak it or misuse it. Moreover, homomorphic encryption does require any kind of interactivity between the data and model owners to perform the computation. Finally, it is lossless in terms of accuracy, given its definition shown in Eq. 3.14. Nevertheless, the price to pay for these advantages is a restricted set of calculations, together with a high computational cost.

Homomorphic encryption has been usually studied with simple models like linear regression [83] or logistic regression [26, 185], since it usually requires expensive summation and multiplication protocols and trades efficiency for privacy. However, homomorphic protocols are of great importance for use cases in which the central server is not trusted. Moreover, homomorphic properties remain detrimental for implementing secure multiparty computation when multiple parties want to carry out joint computations on sensitive data.

3.5 Privacy-Preserving Recommender Systems

Researchers have seriously considered the potential for a breach of privacy in the recommendation process, and the wide literature observing how with some background information it is possible to infer the individual's rating or transaction history is evidence of this. With the primary concern of protecting user privacy without compromising the quality of the recommendations, many approaches have been proposed, mainly relying on intersections of solutions ranging from architectural paradigms, such as federated learning, to algorithmic techniques.

Federated learning is considered a promising solution for alleviating privacy problems in recommender systems. Indeed, user's private data are kept in the client device and only model updates are uploaded. For instance, the paper by Ammad-ud-din et al. [9] represents one of the first examples of federated matrix factorization. However, solutions avoiding sharing personal data with a central server had already been proposed before federated learning. For instance, in Shokri et al. [183], both a local offline user profile and a centralized online user profile exists. Users independently synchronize their online profile with the offline profile but inject in the former also ratings coming from other peers with which they have communicated, thus confusing an untrusted server. Moreover, Vallet et al. [196] explored the possibility of decoupling matrix factorization on a local training part and a central training part, but still considering sharing data.

Federated learning also opened the doors to more complex models, such as the one in Lin et al. [134], leveraging the distributed architecture. They consider a collaborative matrix and a meta-recommender stored on the server and a prediction module on the user devices. The collaborative matrix, interacting with a user embedding, generates a collaborative vector that is fed into the meta-recommender, a neural network that returns a private decomposition of the item latent factor matrix. Combining the private user embedding with the reconstructed item latent factor matrix, a local prediction neural network is able to predict the missing ratings. Also the work by Chen et al. [60] tried to overcome the statistical and systematic challenges of federated learning with a shared parameterized algorithm (or meta-learner) instead of a global model, obtaining significant improvements in terms of recommendation accuracy, convergence speed, and communication cost.

Beyond federated learning, also decentralized architectures represented an essential stage for privacy-oriented recommender systems. Chen et al. [59] proposed a decentralized approach for point-of-interest

Table 3.2: Categorization of relevant publications on the topic of privacy-preserving recommender systems.

	Architecture				**Privatization**				**Recomm.**		
	Centr.	Decentr.	Fed.	Other	Centr. DP	Loc. DP	Crypt.	Other	Latent	Neigh.	Other
Ammad-ud-din et al. [9]			✓						✓		
Shokri et al. [183]				✓				✓		✓	
Vallet et al. [196]				✓					✓		
Lin et al. [134]			✓						✓		
Chen et al. [60]			✓						✓		
Chen et al. [59]		✓							✓		
Chai et al. [57]			✓				✓		✓		
Canny [54]		✓	✓				✓		✓		
Jeckmans et al. [108]		✓					✓			✓	
Wang et al. [205]	✓						✓		✓		
Polat et al. [160]	✓							✓		✓	
McSherry et al. [149]	✓					✓					
Friedman et al. [85]	✓		✓		✓	✓			✓		
Hua et al. [103]			✓			✓			✓		
Zhang et al. [225]			✓			✓			✓		
Machanavajjhala et al. [141]	✓				✓						✓
Guo et al. [93]	✓				✓						✓
Kim et al. [114]			✓			✓			✓		✓
Zhang et al. [226]	✓				✓						✓
Zhu et al. [235]	✓				✓					✓	
Qi et al. [161]			✓			✓			✓		✓
Kharitonov [113]			✓			✓					✓

recommendation in which users send each other portions of additively decomposed item embeddings. Moreover, they use a random walk strategy to solve the trade-off between collaboration and costs of communication. Indeed, in this kind of architecture, the further the communication is, the more users can collaborate, but the larger is the communication cost.

In general, architectures that shift computation to the client-side are particularly useful for mitigating privacy risks that follow from data retention on a centralized server. Nevertheless, some works, such as the one by Chai et al. [57] for matrix factorization, attempted to demonstrate that even in these scenarios, data is not safe. For instance, they proved that by observing the gradients of subsequent iterations of federated matrix factorization training, it is possible to reconstruct the

user profile and then infer the ratings. Their solution considers using encryption strategies for securely sending gradients over the network without decreasing the performance. Canny [54] was the very first application of secure multi-party computation to recommender systems, with a federated or decentralized community of users computing a public aggregate of their data. That work considered the collaborative filtering task as an iterative calculation of the aggregate, requiring only the addition of vectors of user data, with homomorphic encryption to allow sums of encrypted vectors without exposing individual data. Also Jeckmans et al. [108] considered multi-party computation for two companies wanting to generate predictions based on a securely computed user-to-user similarity.

The usage of encryption for preserving privacy in recommender systems is also popular in centralized models. For instance, Wang et al. [205] proposed the encryption of the rating vector with a homomorphic scheme to obtain predictions from a pretrained model. They also proposed a fine-tuning strategy for training an encrypted model, together with some sparsification and quantization tricks that increase the possibility of obtaining similar values and then reusing some multiplicative operations, thus reducing the costs of the homomorphic encryption.

Unfortunately, secure computations produce the same recommendations as non-private protocols, but this comes at the cost of computational overhead [54], making these protocols suitable mainly for offline recommendations. On the other side, a significant number of works integrated perturbation mechanisms as a computationally lightweight solution. Polat et al. [160] proposed the first idea of data perturbation in recommender systems in such a way that the central place can only know the range of the data, and such range is broad enough to preserve users' privacy. If the number of users is significantly large, the aggregate information of these users can be estimated with decent accuracy.

The formal idea of differential privacy was introduced for the first time into collaborative filtering recommender systems by McSherry et al. [149], who pioneered a study that randomized each user's rating before submitting it to the system. In detail, they proposed to measure some noisy global effects of the dataset (such as global averages, item and user averages) with carefully calibrated Laplacian noise. Then, their work considers to center and clamp all the ratings, thus reducing the sensitivity of the function and giving the possibility to create a noisy covariance matrix to be used in a non-private recommender algorithm to predict ratings. Later, Friedman et al. [85] used this approach to feed matrix factorization, with noisy ratings clamped to limit the influence

of excessive noise. Moreover, they studied the matrix factorization algorithm with other privatization strategies to maintain differential privacy, such as a perturbed version of stochastic gradient descent with a calibrated Laplace noise applied to the error computed in each iteration of the training algorithm. Alternatively, Hua et al. [103] proposed to inject the noise directly into the objective function of matrix factorization. However, they stated that, an untrusted server could be able to eliminate such noise iteration after iteration performing a difference attack. Therefore, they proposed to add additional noises, which they proved to guarantee differential privacy. Also Zhang et al. [225] injected Laplacian noise into the objective function to satisfy differential privacy. They used the idea of k-coRating that pursues privacy by creating groups of equivalent users, eventually injecting some fake ratings to reach the equivalence.

Machanavajjhala et al. [141] studied privacy-preserving social recommendations on the basis of a graph linking users and items. Given the graph, they derived utility vectors that capture the utility of items for users and wanted to keep the utility vector private. The authors concluded that good recommendations were achievable only under weak privacy parameters since privacy guarantees require modifying social links that could have a negligible effect on the overall accuracy. This work showed that in some settings (e.g., social recommendations), it might be impossible to obtain privacy and accuracy guarantees simultaneously due to the high sensitivity of the involved functions. However, Guo et al. [93] revised the estimation of the sensitivity, thus saving utility also with stricter privacy requirements.

Privacy-preserving techniques have been widely applied in the domain of point-of-interest recommendation. Indeed, it is estimated [66] that only four locations are enough to identify most people. Kim et al. [114], for instance, proposed to apply differential privacy to transition patterns by perturbing them before sending them to the central server. The point-of-interest recommender based on Markov chains and proposed by Zhang et al. [226], instead, uses the relaxed version of (ϵ, δ)-differential privacy by means of a protection module that receives the aggregated statistics from the database and elaborates the queries from the recommender system answering with noisy statistics.

Beyond model-based recommender systems, Zhu et al. [235] proposed differentially private neighborhood-based collaborative recommendations, aiming specifically at the sybil attack presented by Calandrino et al. [50] claiming that if a recommendation algorithm and its parameters are known by an attacker knowing the partial ratings his-

tory of active user, then the attacker can infer user's remaining rating history. Therefore, Zhu et al. [235] considered a differentially-private k-NN algorithm, introducing randomness to the neighbors' selection while ensuring that, with high probability, the selected neighbors have high similarity scores.

Other works integrated federated learning with local differential privacy. For instance, we can mention the news recommendation model by Qi et al. [161], the work by Kharitonov [113] realizing a differentially private federated online learning-to-rank system with evolution strategy optimization.

Part II

THE SHOWCASE

Namely, my efforts, thus the successes and the failures. In a word, my results. The Showcase is here to respond to "*Where have you been to? Which have been your path, your climbs and descents?*".

4 Federated Learning for Data Property and Control in Recommender Systems

OUTLINE

This chapter presents our approach to put users in complete control of their data in a federated recommender system with learning to rank. We move a step in the literature of recommender systems towards solutions embracing the federated learning paradigm and the recent privacy regulations. In particular, we study the behavior of our recommender when users decide to share only a small amount of sensitive information and how incomplete data impacts the system performance from a wide range of perspectives.

The content of this chapter is a summary of the long paper "*FedeRank: User Controlled Feedback with Federated Recommender Systems*", presented at the 43rd European Conference on Information Retrieval, and the short paper "*How to Put Users in Control of their Data in Federated Top-N Recommendation with Learning to Rank*", discussed at the 36th ACM SIGAPP Symposium On Applied Computing. This work has been also accepted for publication in the Journal of Intelligent Information Systems within the paper "*User-Controlled Federated Matrix Factorization for Recommender Systems*". Finally, some of this content also appeared in "*Federated Recommender Systems with Learning to Rank*", presented at 29th Italian Symposium on Advanced Database Systems.

4.1 Introduction

Recommender systems have emerged as a solution to better support users' decision-making and promote business by recommending novel and personalized items. These models are generally hosted on *centralized* servers and train their models by exploiting massive proprietary and sensitive data. For instance, collaborative filtering models, which have been the mainstream research line in the recommender system

community over the last two decades [145, 223], need sufficient in-domain interaction data to discover similar behavioral/preference patterns in a user community. In principle, this could result in a grave threat to users' privacy. Moreover, the European Union, the US Congress, and other jurisdictions legislated new disclosure laws in recent years. As an example, in 2018, GDPR [80] was proposed by the EU that removes the default option for collecting, storing, and harnessing individual data and requires explicit authorization from the users to use their data. Although the fundamental role played by these laws is to protect users' privacy, the consequent data scarcity dilemma can thereby jeopardize the training of high-quality models.

In this context, federated learnign has been proposed by Google in recent years as a means to offer a *privacy-by-design* solution for machine-learned models [116, 117, 146]. Federated learning aims to meet machine learning privacy shortcomings by horizontally distributing the model's training over user devices; thus, clients locally train the global model exploiting private data without sharing it [146]. As we have already analyzed in Section 3.3, federated learning differs from distributed computing, since in the latter we witness a well-balanced computational effort among devices. Instead, with federated learning, the overall data is supposed to be massive in amount and unbalanced between personal devices.

Recently, the benefits of federated learning in recommender systems have led to advantages for the privacy of the users of those systems (see Section 3.5). In this work, we introduce a novel factorization model, called FPL (short for Federated Pair-wise Learning), that embraces the federated learning paradigm. A disruptive effect of employing FPL is that users participating in the federation process can decide if and how they are willing to disclose their private sensitive preferences. Indeed, FPL mainly leverages non-sensitive information (e.g., items the user has not experienced). Here, we show that even only a small amount of sensitive information (i.e., items the user has experienced) lets FPL reach a competitive accuracy. In this work, instead of focusing on how to protect personal information from privacy breaches (that is explored in other active research fields), we investigate how to guarantee the users the control and property of their data as determined by regulations. Nevertheless, for the sake of completeness, we also propose both a privacy analysis and a convergence analysis of FPL. The work's contributions are manifold due to the number of open challenges that still exist with the federated learning paradigm and answer to the following research questions:

RQ1. Is it possible to integrate pair-wise learning with federated learning principles to build a federated version of factorization models? What is the impact of federated parameters (i.e., *computation parallelism*, and *local computation amount*) on the quality of recommendation?

RQ2. The protection of the user's feedback can put the recommendation service in jeopardy. Can users receive a high-quality recommendation while limiting the amount of disclosed sensitive data?

RQ3. The sequentiality of the original pair-wise algorithms can be replicated at the price of increased communication costs. What is the optimal (or sub-optimal) trade-off between communication costs and recommendation utility?

RQ4. With limited training information, the recommendation algorithm might learn differently and unexpectedly. Does the federated recommendation (and the possible reduced information budget) inject additional biases in the final recommendation?

To answer the above questions, we have carried out extensive experiments by considering the accuracy of recommendation and diversity metrics (Item Coverage and Gini Index). Afterward, we analyzed communication cost and accuracy in a multi-objective perspective and fairness (i.e., the Bias Disparity) of FPL recommendations. The experimental evaluation shows that FPL provides high-quality recommendations, putting the user in control of the amount of sensitive data disclosed.

4.2 Related Work

Academia and industry have proposed several competitive recommendation algorithms. Algorithms based on nearest-neighbors, latent factor models and artificial neural networks are undoubtedly the most representative examples of the collaborative filtering systems, that extract user preference patterns in a collaborative fashion. The nearest-neighbors scheme has shown its competitiveness for quite a long time. The user-based scheme and the item-based scheme find the nearest user neighbors and the nearest item neighbors based on a similarity function. It then exploits them to predict a score for each user-item pair. Although they use the same logic behind the scenes, user-based and item-based schemes show their effectiveness in different contexts.

After these models, the most innovative idea to implement collaborative filtering has been decomposing the user-item rating matrix and exploiting the dot product to reconstruct the matrix and compute similarities. This idea led to the matrix factorization technique, which is

probably the most representative of the factorization-based recommendation family. Nevertheless, several generalized/specialized variants have been proposed, such as FM [166], SVD++ [118], PITF [170], FPMC [168]. Unfortunately, rating-prediction-oriented optimization has shown its limits in the recommendation research [148]. Consequently, a new class of *learning to rank* algorithms has been developed in the last decade, mainly ranging from point-wise [122] to pair-wise [167], through list-wise [182] approaches. Among pair-wise methods, BPR [167] is one of the most broadly adopted, thanks to its outstanding capabilities to correctly rank preserving an acceptable computational complexity. It exploits a stochastic gradient descent algorithm to learn the relative order between positive and negative items.

As we have already discussed, recommender systems need to collect user information related to attributes, demands, and preferences to work properly [109]. As a rule of thumb, the accuracy of recommendations is directly proportional to the level of detail of the gathered information [104]. Regrettably, the more detailed the knowledge about users is the more significant the threat to the user's privacy becomes [39]. In contexts like this, federated learning was introduced to learn models from a population while learning as little as possible about individuals. It meets the privacy shortcomings by horizontally distributing the model's training over user devices [146].

In this work, we focus on the application federated learning theoretical and practical principles to a learning to rank recommender system, in order to address the need for user control on data and to meet the privacy regulations mentioned above. A federated implementation of collaborative filtering has been proposed in [9], which uses the SVD-MF method for implicit feedback [102]. Here, the training is a mixture of Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD) for preserving users' privacy. However, its security limits have been analyzed in Chai et al. [57]. Recently, the federated learning paradigm spread to the recommendation tasks, thanks to its capability of dealing with sensitive data. Many examples have been proposed in Section 3.5. Nevertheless, incomprehensibly, at the time of this research, almost no work addressed top-k recommendation exploiting the learning to rank paradigm. In this sense, one rare example is the work by Kharitonov [113], who recently proposed to combine evolution strategy optimization with a privatization procedure based on differential privacy.

4.3 Background

In this section, we introduce the fundamentals of the pair-wise learning to rank approach in the context of the factorization models. In detail, we provide the essential mathematical background, the formal definition of the main concepts, and the notation that is adopted in the following.

A recommendation problem is usually conceived as the activity of finding the items of a catalog a particular user might be interested in. Formally, let $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ be the user-item matrix where each entity x_{ui} represents an explicit or binary implicit feedback (e.g., explicit rating or check-in, respectively) of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. In the work at hand, an implicit feedback scenario is considered — i.e., feedback is, e.g., purchases, visits, clicks, views, check-ins —, with $\mathbf{R}$ containing binary values. Therefore, $r_{ui} = 1$ and $r_{ui} = 0$ denote either user u has consumed or not item i, respectively.

In FPL, the underlying data model is a factorization model, inspired by MF [121], a recommendation model that became popular in the last decade thanks to its state-of-the-art recommendation accuracy [40], that we have already introduced in Section 2.2.1.2.

Definition 4.1 (Matrix Factorization)**.** *Given a set of users $\mathcal{U}$, a set of items $\mathcal{I}$, and a matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$, Matrix Factorization builds a model Θ in which each user u and each item i is represented by the embedding vectors $\mathbf{p}_u$ and $\mathbf{q}_i$, respectively, in the shared latent space $\mathbb{R}^f$. The core of the algorithm relies on the assumption that $\mathbf{R}$ can be factorized such that the dot product between $\mathbf{p}_u$ and $\mathbf{q}_i$ can explain any observed user-item interaction r_{ui}, and that any non-observed interaction can be estimated as:*

$$\tilde{r}_{ui} = b_i + \mathbf{q}_i^T \mathbf{p}_u(\Theta), \tag{4.1}$$

where b_i is a term denoting the bias of the item i.

The global average μ and the user bias b_u are not reported here since their effects will be eliminated in Eq. 4.2.

Among pair-wise approaches for learning to rank the items of a catalog, Bayesian Personalized Ranking (BPR) [167] is one of the most broadly adopted, thanks to its capabilities to correctly rank with *acceptable* computational complexity.

Definition 4.2 (Bayesian Personalized Ranking)**.** *Let $\mathcal{K} : \mathcal{U} \times \mathcal{I} \times \mathcal{I}$ be a training set defined by $\mathcal{K} = \{(u, i^+, i^-) \mid r_{ui^+} = 1 \wedge r_{ui^-} = 0\}$. Bayesian Personalized Ranking is an optimization approach aiming to*

learn a model Θ *that solves the personalized ranking task according to the following optimization criterion:*

$$\max_{\Theta} \sum_{(u,i^+,i^-)\in\mathcal{K}} \ln\, \sigma(\hat{r}_{ui^+i^-}(\Theta)) - \lambda\|\Theta\|^2, \tag{4.2}$$

where $\hat{r}_{ui^+i^-}(\Theta) = \tilde{r}_{ui^+}(\Theta) - \tilde{r}_{ui^-}(\Theta)$ *is a real value modeling the relation between user* u*, item* i^+ *and item* i^-*,* $\sigma(\cdot)$ *is the sigmoid function, and* λ *is a model-specific regularization parameter to prevent overfitting.*

Pair-wise optimization can be applied to a wide range of recommendation models, included factorization. Hereafter, we denote the model $\Theta = \langle \mathbf{P}, \mathbf{Q}, \mathbf{b} \rangle$, where $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}|\times f}$ is a matrix whose u-th row corresponds to the vector $\mathbf{p}_u$, and $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}|\times f}$ is a matrix in which the i-th row corresponds to the vector $\mathbf{q}_i$. Finally, $\mathbf{b} \in \mathbb{R}^{|\mathcal{I}|}$ is a vector whose element i corresponds to the item i.

4.4 Federated Pair-wise Learning

In this section, we introduce the fundamental concepts regarding the collaborative filtering recommendation using a federated learning scheme. Along with the problem definition, the notation we adopt is presented. Hereby, we want to make the reader aware that FPL is a tool for putting users in control of their data. In detail, here we focus on analyzing how different levels of data disclosure affect the recommendation. Providing privacy guarantees, e.g., by incorporating FPL in dedicated frameworks [1, 42, 57], remains out of the scope of this work.

4.4.1 *Architecture*

Following the federated learning principles, let $\mathcal{U}$ be the set of users (clients) with a server S coordinating them. Assume users consume items from a catalog $\mathcal{I}$ and give feedback about them (as in the recommendation problem of Section 4.3). S is aware of the catalog $\mathcal{I}$, while exclusively user u knows her own set of consumed items.

To setup the federation for FPL, a shared global model is built on the server S, while different private local models are built on each user's device.

Definition 4.3 (FPL Global Model)**.** *In FPL, the server* S *builds a global model* $\Theta_S = \langle \mathbf{Q}, \mathbf{b} \rangle$*, where* $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}|\times f}$ *and* $\mathbf{b} \in \mathbb{R}^{|\mathcal{I}|}$ *are the item-factor matrix and the bias vector introduced in Section 4.3.*

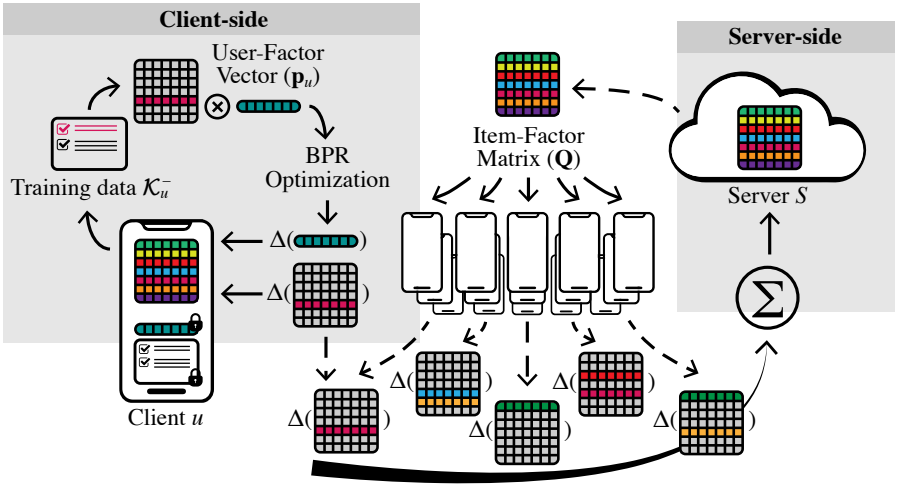


Figure 4.1: Training protocol of FPL. In the middle, item factor Matrix is sent by the server to the federation of devices. On the left, local training phase is represented. The local output, together with the output of the other devices, is sent to the server. On the right, server-side, aggregation of received updates is performed.

Definition 4.4 (FPL Local Model)**.** *On each user u's device FPL builds a model $\Theta_u = \langle \mathbf{p}_u \rangle$, which corresponds to the representation of user u in the latent space of dimensionality f.*

Hence, in FPL, Θ_u and Θ_S are privately combined together. The client produces tailored recommendations by scalar multiplying local $\mathbf{p}_u$ and $\mathbf{q}_i$. Each user u holds her own private dataset $\mathbf{r}_u \in \mathbb{R}^{\mathcal{I}}$, which, analogously to a centralized recommender system, corresponds to the u-th row of matrix $\mathbf{R}$. Each FPL client u hosts a user-specific training set $\mathcal{K}_u : \mathcal{U} \times \mathcal{I} \times \mathcal{I}$ defined by $\mathcal{K}_u = \{(u, i^+, i^-) \mid r_{ui^+} = 1 \wedge r_{ui^-} = 0\}$. Please note that we refer to $R^+ = \sum_{u \in \mathcal{U}} |\{r_{ui} \mid r_{ui} = 1\}|$ as the total number of positive interactions in the system.

4.4.2 *Training Procedure*

The classic BPR-MF learning procedure [167] for model training can not be directly applied to the FPL model, since we have decoupled the representation of users and items respectively on the local devices and the server. In the following, we show the FPL learning procedure that is executed for a number C of rounds of communication and envisages **Distribution to Devices** → **Federated Optimization** → **Transmission to Server** → **Global Aggregation** sequences between the server and the clients. The notation $\{\cdot\}_S^t$ denotes an object computed

by the server S at round t, while $\{\cdot\}_u^t$ indicates an object computed by a specific client u at round t.

(1) **Distribution to Devices.** Let $\{\mathcal{U}^-\}_S^t$ be a subset of $\mathcal{U}$ with cardinality m, containing m clients $u \in \mathcal{U}$. The set $\{\mathcal{U}^-\}_S^t$ can be either defined by S, or the result of a request for availability sent by S to the clients in $\mathcal{U}$. Each client $u \in \{\mathcal{U}^-\}_S^t$ receives from S the latest snapshot of Θ_S, namely $\{\Theta_S\}_S^{t-1}$.

(2) **Federated Optimization.** Each user $u \in \{\mathcal{U}^-\}_S^t$ generates the set $\{\mathcal{K}_u^-\}_u^t$ containing T random triples (u, i^+, i^-) from $\mathcal{K}_u$, with T representing the number of local stochastic updates performed by each client in a round. It is worth underlining that Rendle et al. [167] suggest, for a centralized scenario, to train the recommendation model by randomly choosing the training triples from $\mathcal{K}$, to avoid data is traversed item-wise or user-wise, since this may lead to slow convergence. Conversely, in a federated approach, we are required to train the model user-wise. Indeed, the learning is separately performed on each device u, that only knows the data in $\mathcal{K}_u$. Thanks to the user-wise traversing, FPL can decide who controls (the designer or the user) the number of triples T in the training set $\{\mathcal{K}_u^-\}_u^t$, to tune the degree of local computation, i.e. how much the sampling is user-wise traversing. As a consequence, their updates are independent and computed in parallel, unlike in stochastic centralized learning.

With the local training set, the user u can compute her contribution to the overall optimization of Θ_S with the following update:

$$\{\Delta\Theta_S\}_u^t := \sum_{(u,i^+,i^-)\in\{\mathcal{K}_u^-\}_u^t} \frac{\partial}{\partial\{\Theta_S\}_S^{t-1}} \ln\, \sigma(\hat{r}_{ui^+i^-}(\{\Theta_S\}_S^{t-1}; \{\Theta_u\}_u^{t-1})), \tag{4.3}$$

plus a regularization term. At the same time, the client u updates its local model Θ_u, and incorporates it in the current model by using:

$$\{\Delta\Theta_u\}_u^t := \sum_{(u,i^+,i^-)\in\{\mathcal{K}_u^-\}_u^t} \frac{\partial}{\partial\{\Theta_u\}_u^{t-1}} \ln\, \sigma(\hat{r}_{ui^+i^-}(\{\Theta_S\}_S^{t-1}; \{\Theta_u\}_u^{t-1})), \tag{4.4}$$

plus a regularization term. In detail, given:

$$\begin{aligned} \{\hat{r}_{ui^+i^-}\}_u^t &:= \hat{r}_{ui^+i^-}(\{\Theta_S\}_S^{t-1}; \{\Theta_u\}_u^{t-1}) \\ &= [\{b_{i^+}\}_S^{t-1} + (\{\mathbf{p}_u\}_u^{t-1})^T \cdot \{\mathbf{q}_{i^+}\}_S^{t-1}] \\ &\quad - [\{b_{i^-}\}_S^{t-1} + (\{\mathbf{p}_u\}_u^{t-1})^T \cdot \{\mathbf{q}_{i^-}\}_S^{t-1}], \end{aligned} \tag{4.5}$$

and following Rendle et al. [167], the client u can compute:

$$\frac{\partial}{\partial\theta}\{\hat{r}_{ui^+i^-}\}_u^t = \begin{cases} (\{\mathbf{q}_{i^+}\}_S^{t-1} - \{\mathbf{q}_{i^-}\}_S^{t-1}) & \text{if } \theta = \{\mathbf{p}_u\}_u^{t-1}, \\ \{\mathbf{p}_u\}_u^{t-1} & \text{if } \theta = \{\mathbf{q}_{i^+}\}_S^{t-1}, \\ -\{\mathbf{p}_u\}_u^{t-1} & \text{if } \theta = \{\mathbf{q}_{i^-}\}_S^{t-1}, \\ 1 & \text{if } \theta = \{b_{i^+}\}_S^{t-1}, \\ -1 & \text{if } \theta = \{b_{i^-}\}_S^{t-1}. \end{cases} \tag{4.6}$$

At the end of the federated computation, given a shared learning rate α, each client can update its local model Θ_u — containing the user profile — by aggregating the computed update:

$$\{\Theta_u\}_u^t := \{\Theta_u\}_u^{t-1} + \alpha\{\Delta\Theta_u\}_u^t. \tag{4.7}$$

(3) **Transmission to Server.** In a purely distributed architecture, each user in $\{\mathcal{U}^-\}_S^t$ returns to S the computed update. Here, instead of sending $\{\Delta\Theta_S\}_u^t$, each user transmits a modified version $\{\Delta\Theta_S^\Phi\}_u^t$. To introduce this aspect of FPL, let us define $\mathcal{F} = \{i^+, \forall(u, i^+, i^-) \in \{\mathcal{K}_u^-\}_u^t\}$, and a randomized object $\Phi = \left\langle \mathbf{Q}^\Phi, \mathbf{b}^\Phi \right\rangle$, with $\mathbf{Q}^\Phi \in \mathbb{R}^{|\mathcal{I}|\times f}$, and $\mathbf{b}^\Phi \in \mathbb{R}^{|\mathcal{I}|}$. Each row $\mathbf{q}_i^\Phi$ of $\mathbf{Q}^\Phi$ and each element b_i^Φ of $\mathbf{b}^\Phi$ assume their value according to the probabilities:

$$\begin{aligned} P(\mathbf{q}_i^\Phi = \mathbf{1}, b_i^\Phi = 1 \mid i \in \mathcal{F}) &= \pi, \\ P(\mathbf{q}_i^\Phi = \mathbf{0}, b_i^\Phi = 0 \mid i \in \mathcal{F}) &= 1 - \pi, \\ P(\mathbf{q}_i^\Phi = \mathbf{1}, b_i^\Phi = 1 \mid i \notin \mathcal{F}) &= 1 \end{aligned} \tag{4.8}$$

Based on $\{\mathbf{Q}^\Phi\}_u^t$ and $\{\mathbf{b}^\Phi\}_u^t$, $\Delta\Theta_S^\Phi$ can be computed as it follows:

$$\{\Delta\Theta_S^\Phi\}_u^t = \{\Delta\Theta_S\}_u^t \odot \{\Phi\}_u^t := \left\langle \{\Delta\mathbf{Q}\}_u^t \odot \{\mathbf{Q}^\Phi\}_u^t, \{\Delta\mathbf{b}\}_u^t \odot \{\mathbf{b}^\Phi\}_u^t \right\rangle, \tag{4.9}$$

where the operator $\odot$ denotes the Hadamard product. This transformation is motivated by a possible privacy issue. The update $\Delta\mathbf{Q}$ computed in Eq. 4.3 by user u is a matrix whose rows are non-zero in correspondence of the items i^+ and i^- of all the triples $(u, i^+, i^-) \in \mathcal{K}_u^-$ [167]. An analogous behavior can be observed for the elements of $\Delta\mathbf{b}$. Focusing on the non-zero elements, we observe that, for each triple $(u, i^+, i^-) \in \mathcal{K}_u^-$, the updates $\{\Delta\mathbf{q}_{i^+}\}_u^t$ and $\{\Delta\mathbf{q}_{i^-}\}_u^t$, as well as $\{\Delta b_{i^+}\}_u^t$ and $\{\Delta b_{i^-}\}_u^t$, show the same absolute value with opposite sign [167]. In fact, sharing all the updates may lead to a significant users' private data disclosure that may lead to a privacy issue if the server S is honest-but-curious. On

the one hand, each pair of updates for a consumed item i^+ and for a non-consumed item i^- contains equal but opposite gradients. Thus, if the user u sends all of them to S, they may reveal patterns of like/dislike user tastes. On the other hand, items rated by a user are more likely to be sampled and their corresponding vectors to be updated, thus allowing the server S to reconstruct, after some epochs, part of the user dataset $\mathcal{K}_u$. Section 4.4.4 goes deeper into the details of these and other weaknesses. Since our primary goal is to put users in control of their data, we leave users the possibility to choose a fraction π of positive item updates to send. The remaining positive item updates (a fraction $1-\pi$) are masked by setting them to zero, by means of the transformation in Eq. 4.9. Conversely, the negative updates are always sent to S, since their corresponding rows are always multiplied by a $\mathbf{1}$ vector. Indeed, these updates are related to non-consumed items, which are indistinguishably negative or missing values, assumed to be *non-sensitive* data.

(4) **Global Aggregation.** Once S has received $\{\Delta\Theta_S^\Phi\}_u^t$ from all clients $u \in \{\mathcal{U}^-\}_S^t$, it aggregates the received updates in $\{\mathbf{Q}\}_S^{t-1}$ and $\{\mathbf{b}\}_S^{t-1}$ to build the new global model, with α being the learning rate:

$$\{\Theta_S\}_S^t := \{\Theta_S\}_S^{t-1} + \alpha \sum_{u \in \{\mathcal{U}^-\}_S^t} \{\Delta\Theta_S^\Phi\}_u^t. \tag{4.10}$$

Intuitively, each row of $\{\mathbf{Q}\}_S^t$ and each element of $\{\mathbf{b}\}_S^t$ are obtained by summing up to $\{\mathbf{Q}\}_S^{t-1}$ and $\{\mathbf{b}\}_S^{t-1}$ the contribution of all clients in $\{\mathcal{U}^-\}_S^t$ for the corresponding item.

FPL reshapes the training scheme of centralized BPR-MF. However, it does not affect the computation for the model optimization, thus FPL has the same computational complexity of BPR-MF. Nonetheless, it is important to consider that some hyperparameters, analyzed in Section 4.4.3, can affect the convergence of FPL, increasing/decreasing the computation and communication costs.

4.4.3 *Convergence Analysis of FPL*

Unlike other learning paradigms, in federated learning, the training data is not independent and identically distributed (non-IID). The user's local data is not representative of the overall data distribution. Therefore, one cannot replace them with samples drawn from the overall distribution. In 2020, Li et al. [131] have shown that, given L-smooth and μ-strongly convex local losses like BPR, a federated optimization based on averaging of local parameters converges to the global optimum with

a convergence rate of $O(\frac{1}{CT})$. FPL may converge to a sub-optimal solution at least $\Omega(\alpha(T-1))$ away from the optimal one if weight decay is not considered. The number of rounds needed to reach a target performance is a function of the number of local epochs T, both linearly and inversely dependent on it [131]. Therefore, over-small and over-large values of T may lead to a large number of rounds of communication. In particular, if data is non-IID and T exceeds $O(CT)$, convergence is not guaranteed, since the sum of local minima may not correspond to the global minimum. If sampling probabilities are highly non-uniform across the users, convergence may be slower [232]. However, some novel schemes have been recently proposed to address this issue [131], and we will test them in future investigations. Finally, under the non-IID setting, the convergence rate has a weak dependence on the size of $\mathcal{U}^-$. In practice, the participation ratio can be set small or large, according to the communication requirements and without affecting FPL convergence.

4.4.4 *Privacy Analysis of FPL*

Section 4.4 starts by stating that FPL has not been conceived to be a privacy-preserving framework. Rather, it is a tool to control the trade-off between (potentially) exposed sensitive data and the recommendation quality. Federated learning hides, by design, users' raw data to the server: the updates sent by clients are anonymously aggregated, and only the aggregated information is deployed. Nevertheless, some *malicious* actors might still try to learn sensitive information if they have access to parts of the system, as already discussed in Section 4.4.2. For this reason, federated learning alone is not considered to provide privacy guarantees to users. FPL is a federated recommender system fed by implicit feedback. Consequently, providing privacy guarantees implies that the existence of each transaction in the user's history must be kept secret. With reference to Eq. 4.6, suppose a pair of positive and negative items i^+ and i^-. Simplifying the notation and focusing on a single latent factor h, the values of $\Delta\mathbf{q}^t_{i^+}$ and $\Delta\mathbf{q}^t_{i^-}$ could be rewritten as:

$$\Delta\mathbf{q}^t_{i^+,h} = \mathbf{p}^{t-1}_{u,h}\sigma(\mathbf{p}^{t-1}_{u,h} \cdot (\mathbf{q}^{t-1}_{i^+,h} - \mathbf{q}^{t-1}_{i^-,h})), \tag{4.11}$$

$$\Delta\mathbf{q}^t_{i^-,h} = -\mathbf{p}^{t-1}_{u,h}\sigma(\mathbf{p}^{t-1}_{u,h} \cdot (\mathbf{q}^{t-1}_{i^+,h} - \mathbf{q}^{t-1}_{i^-,h})), \tag{4.12}$$

where $\sigma(\cdot)$ returns values in the range $(0, 1)$. These equations show that the modules of $\Delta\mathbf{q}^t_{i^+,h}$ and $\Delta\mathbf{q}^t_{i^-,h}$ (that have to be sent to the server) are identical, while their signs are opposite. Moreover, the sign of the update depends on both the existence/absence of a transaction for k and

on $sgn(\mathbf{p}_{u,h}^{t-1})$. Therefore, the sign of a gradient does not directly reveal the presence or absence of an item in the user's training set, but the pairs of positive and negative gradients disclose user preference patterns. In a round of communication, all the updates for the consumed items share the same sign, as well as all the updates for the non-consumed items have the same positive or negative sign, depending on $sgn(\mathbf{p}_{u,h}^{t-1})$. Suppose the server S is a honest-but-curious agent, i.e., it may try to inspect the updates to obtain some user information. Let us assume that, as soon as it obtains enough information adequate to identify one or more consumed/non-consumed items, the entire user dataset will be exposed. To avoid this problem, FPL puts users in control of their data. If the users adopt the *privacy-oriented* masking procedure discussed in Section 4.4.2, they can decide the fraction of updates for positive items to send. In the case of exposure of the user transactions, only a fraction (actively decided by the users) is given up. This work studies and analyzes the recommendation performance in this data scarcity scenario. While we do not explicitly define a user-specific protocol for privacy level tuning, the system allows both possibilities: the system designer defines a fixed portion of data users should share, or users actively decide the fraction of data to share. For instance, the users might choose among a set of privacy/accuracy trade-off levels, as already happens with location data in some commercial products. If a user is not satisfied with the accuracy performance, she might modify the privacy/accuracy trade-off level at any moment.

Other possible issues, like active reconstruction of the user profile, are not considered here and are out of the scope of this work. However, federated learning literature already provides privacy protocols like differential privacy and cryptographic methods. They have been proven to guarantee user privacy, FPL architecture has been explicitly designed to work with them.

4.5 Experimental Setup

In this section, we introduce the experimental setting designed to answer the research questions.

4.5.1 *Datasets*

The evaluation of FPL needs to meet some particular constraints: the availability of transaction data to obtain a reliable experimental setting and a domain that guarantees the presence of data the user may prefer to

Table 4.1: Characteristics of the evaluation datasets used in the offline experiment: $|\mathcal{U}|$ is the number of users, $|\mathcal{I}|$ the number of items, R^+ the number of records.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	R^+	$\frac{R^+}{\lvert\mathcal{U}\rvert}$	$\frac{R^+}{\lvert\mathcal{I}\rvert}$	$\frac{R^+}{\lvert\mathcal{L}\rvert\cdot\lvert\mathcal{U}\rvert}\%$
Brazil	17,473	47,270	599,958	34.34	12.69	0.00073%
Canada	1,340	29,518	63,514	47.40	2.15	0.00161%
Italy	1,353	25,522	54,088	39.98	2.20	0.00157%
Amazon DM	1,835	41,488	75,932	41.38	1.83	0.000997%
LibraryThing	7,279	37,232	749,401	102.95	20.13	0.002765%
MovieLens 1M	6,040	3,706	1,000,209	165.60	269.89	0.044684%

protect. Following these constraints, we believe that an optimal domain to test FPL would be that of the Point-of-Interest (PoI), which concerns data that users usually perceive as sensitive. Among the many available datasets, a very good candidate is the *Foursquare* dataset [218]. In fact, it is often considered as a reference for evaluating PoI recommendation models. To mimic a federation of devices in a single country, we have extracted check-ins for three countries, namely Brazil, Canada, and Italy. While selecting the different countries, our only constraint was to obtain datasets with different size/sparsity characteristics. Hence, we choose three countries in three different regions of the world. Furthermore, we have investigated the performance of FPL considering three well-known datasets in recommendation: Amazon Digital Music [144], LibraryThing [230], and MovieLens 1M [96]. The former includes the users' satisfaction explicit feedback for a catalog of music tracks available with Amazon Digital Music service. LibraryThing collects the users' ratings on a book catalog. The latter is MovieLens 1M dataset, which collects users' ratings in the movie domain. To fairly evaluate FPL against the baselines, we have kept users with more than 20 interactions[1]. Moreover, we have split the datasets by adopting a realistic temporal hold-out 80-20 splitting on a per-user basis [25, 91]. The resulting training and test sets have been used with all the methods in comparison, including the state-of-the-art algorithms. Table 4.1 shows the characteristics of the resulting training sets adopted in the experiments.

1 The limitations of collaborative filtering in a cold-start user setting are well-known in literature. However, they are beyond the scope of this work.

4.5.2 *Collaborative Filtering Baselines*

To evaluate the efficacy of FPL, we have conducted the experiments by considering non-personalized methods (random and most popular recommendation), and different recommendation approaches, including the centralized **BPR-MF** implementation [167], **User-kNN** and **Item-kNN** [119], **VAE** [132], and **FCF** [9], which is, to date, the only federated recommendation approach based on MF[2]. Following [64], we considered only **VAE** as representative of the neural approaches.

To evaluate the impact of exploiting only a partial user feedback on recommendation accuracy, we have evaluated different values of π in $[0.0, 1.0]$ with step 0.1, with $\pi = 0.0$ meaning that u is not sharing any positive feedback with the server, and $\pi = 1.0$ meaning that u is sharing the updates on all positive items. Hence, we have considered four different configurations regarding computation and communication:

- ***s*FPL**: it reproduces the centralized stochastic learning, where the central model is updated sequentially; thus, we set $|\{\mathcal{U}^-\}_S^t| = 1$ for all values of t to involve just one random client u in each round t, and it extracts solely one triple (u, i^+, i^-) from its dataset $\mathcal{K}_u$ for the training phase, i.e., $|\{\mathcal{K}_u^-\}_u^t| = T = 1$;
- ***s*FPL+**: we increase client local computation by raising to $\frac{R^+}{|\mathcal{U}|}$ the number of triples T extracted from $\mathcal{K}_u$ by each client involved in the round of communication;
- ***p*FPL**: we enable parallelism by involving all clients in each round of communication ($\{\mathcal{U}^-\}_S^t = \mathcal{U}$ for each t) and we keep $T = 1$;
- ***p*FPL+**: we extend *p*FPL by letting each client sample $T = \frac{R^+}{|\mathcal{U}|}$ triples from $\mathcal{K}_u$; the rationale is that the overall training samples are exactly R^+, as in centralized BPR-MF.

Rendle et al. [167] suggest to set the number of triples used for training in one epoch of BPR to R^+. This corresponds to the number of total number of positive interactions in the system. Therefore, the federated training is comparable to BPR when R^+ optimization steps are performed. To this extent, we introduce the number of rounds of communication per epoch (*rpe*). Consequently, FPL computation after *rpe* rounds is comparable to one epoch of centralized BPR when $|\mathcal{U}^-| \cdot T \cdot rpe = R^+$.

2 Since no source code is available, we implemented it from scratch and considered it in the reader's interest.

This results in $rpe = R^+$ for *s*FPL, $rpe = |\mathcal{U}|$ for *s*FPL+, $rpe = \frac{R^+}{|\mathcal{U}|}$ for *p*FPL, and $rpe = 1$ for *p*FPL+.

4.5.3 *Reproducibility*

For what regards the splitting strategy, we have adopted a **temporal hold-out** 80/20 to separate our datasets in training and test set. Moreover, to find the most promising learning rate α, we have further split the training set, adopting a temporal hold-out 80-20 strategy on a user basis to extract her validation set. **User-kNN** and **Item-kNN** have been experimented for $k \in \{10, 20, ..., 10\}$ considering Cosine Vector Similarity. **VAE** has been trained by considering three autoencoder topologies, with the following number of neurons per layer: 200-100-200, 300-100-300, 600-200-600. We have chosen candidate models by considering the best models after training for 50, 100, and 200 epochs, respectively. For the **factorization models**, we have performed a grid search in BPR-MF for $\alpha \in \{0.005, 0.05, 0.5\}$ varying the number of latent factors in $\{10, 20, 50\}$. Then, to ensure a fair comparison, we have exploited the same learning rate and number of latent factors to train **FPL** and **FCF**, and we explored the models in the range of $\{10, \ldots, 50\}$ iterations. We have set *user*- and *positive item*-regularization parameter to $\frac{1}{20}$ of the learning rate. The *negative item*-regularization parameter is $\frac{1}{200}$ of the learning rate, as suggested in *mymedialite*[3] implementation as well as by Anelli et al. [23]. We made the implementation of FPL publicly available[4]. Moreover, it will be soon integrated into the reproducibility framework Elliot [13].

4.5.4 *Evaluation Metrics*

The RQs (see Section 4.1) cover a broad spectrum of different recommendation dimensions. To this end, we have decided to measure several metrics to evaluate the approaches under the different perspectives.

Accuracy The accuracy of the models is measured by exploiting precision ($P@k$) and recall ($R@k$). They respectively represent, for each user, the proportion of relevant recommended items in the recommendation list, and the fraction of relevant items that have been altogether suggested. Section 2.2.4.1 goes deeper into the details of these metrics. We have assessed the statistical signifi-

3 http://www.mymedialite.net/
4 https://split.to/sisinflab-fpl

cance of results by adopting Student's paired t-test considering p-values < 0.05[5].

Beyond-Accuracy To measure the diversity of recommendations, we have measured the Item Coverage ($IC@k$), and the Gini Index ($G@k$). IC provides the number of diverse items recommended to users. It also conveys the sense of the degree of personalization [6]. Gini (G) is a metric about distributional inequality. It measures how unequally different items a RS provides users with [56]. In the formulation adopted [91], a higher value of G corresponds to higher personalization. See Section 2.2.4.2 for further details.

Fairness The problem of unfair outputs in machine learning applications is well studied [46, 76] and also it has been extended to recommender systems [142]. In detail, in this work, we check whether items belonging to specific groups have equal chance to be shown in the recommended lists. In order to do that, we measure bias disparity (BD) [142] for groups of items. With this metric we quantify, for each category of items, the devation of the proposed recommendations from the initial dataset bias. Section 2.2.4.3 thoroughly presents this metric.

4.6 Results and Discussion

In this Section, we focus on the different experiments conducted to explore the dimensions covered by the Research Questions (see Section 4.1). First, to position FPL with respect to the baselines, we analyze the accuracy, beyond-accuracy, and bias disparity of the recommendations. Once the analysis is completed, we investigate the impact of communication costs, and we study the multi-objective optimization of maximizing the accuracy while minimizing the communication costs. To this extent, we have explored the Pareto frontier, considering the two different dimensions.

4.6.1 *Recommendation Accuracy*

To answer RQ1, we want to assess whether it is possible to obtain a recommendation performance comparable to a centralized pair-wise learning approach while allowing the users to control their data. In this

5 The complete results are available in the implementation repository.

Table 4.2: Results of accuracy metrics for baselines and FPL on the three datasets. For each configuration of FPL and for each dataset, the experiment with the best π is shown (see the bottom part for details). For all metrics, the greater the better. Among federated algorithms, the best performance is in **boldface**.

		Brazil		**Canada**		**Italy**	
		$P@10$	$R@10$	$P@10$	$R@10$	$P@10$	$R@10$
Centralized	**Random**	0.00013	0.00015	0.00030	0.00035	0.00030	0.00029
	Top-Pop	0.01909	0.02375	0.04239	0.04679	0.04634	0.05506
	User-kNN	0.10600	0.13480	0.07639	0.07533	0.06881	0.07833
	Item-kNN	0.07716	0.09607	0.04006	0.03881	0.04663	0.05356
	VAE	0.10320	0.13153	0.06060	0.06317	0.10421	0.21324
	BPR-MF	0.07702	0.09494	0.03694	0.03650	0.04560	0.05458
Federated	**FCF**	0.03089	0.03749	0.03724	0.03836	0.03126	0.03708
	***s*FPL**	0.07757	0.09581	0.04515	0.04550	0.04701	0.05600
	***s*FPL+**	0.08682	0.11004	0.05701	0.05665	**0.05595**	0.06229
	***p*FPL**	0.07771	0.09582	0.04582	0.04637	0.04642	0.05465
	***p*FPL+**	**0.08733**	**0.11085**	**0.05761**	**0.05755**	0.05565	**0.06291**

		Amazon DM		**LibraryThing**		**MovieLens 1M**	
		$P@10$	$R@10$	$P@10$	$R@10$	$P@10$	$R@10$
Centralized	**Random**	0.00005	0.00005	0.00054	0.00028	0.00871	0.00283
	Top-Pop	0.00469	0.00603	0.05013	0.03044	0.10224	0.03924
	User-kNN	0.01940	0.02757	0.14193	0.10115	0.12613	0.06701
	Item-kNN	0.02147	0.03171	0.20214	0.14778	0.08873	0.05475
	VAE	0.01580	0.02289	0.10834	0.07711	0.11735	0.06192
	BPR-MF	0.00921	0.01298	0.07009	0.04303	0.11911	0.05817
Federated	**FCF**	0.00839	0.01222	**0.10760**	0.04392	0.10760	0.04392
	***s*FPL**	0.00610	0.00889	0.06309	0.03738	0.11805	**0.05902**
	***s*FPL+**	**0.01422**	**0.02060**	0.08512	**0.05627**	0.11599	0.05571
	***p*FPL**	0.00812	0.01165	0.06210	0.03643	**0.12275**	0.05806
	***p*FPL+**	0.01351	0.01970	0.07691	0.04965	0.11217	0.05030

respect, Table 4.2 shows the accuracy results of the comparison between the state-of-the-art baselines and the four configurations of FPL presented in Section 4.5. By focusing on accuracy metrics, we may notice that VAE outperforms the other approaches in the three datasets. However, who is familiar with VAE knows that, since it restricts training data by applying k-core, it does not always produce recommendations for all the users. With regards to User-kNN, we notice that it outperforms all the other approaches in all the datasets but *Amazon Digital Music* and *LibraryThing*, where Item-kNN performs better. The performance of Item-kNN and BPR-MF approximately settle in the same

range of values, except for *Amazon Digital Music* and *LibraryThing*, where BPR-MF works significantly worse.

Moreover, it is important to investigate the differences of FPL with respect to BPR-MF, which is a pair-wise centralized approach, being FPL the first federated pair-wise recommender based on a factorization model. The performance of BPR-MF against FPL, in the configuration *s*FPL, shows how precision and recall in *s*FPL are slightly outperforming BPR-MF with *Foursquare* datasets. The consideration that the performance is comparable is surprising since the two methods share the sequential training, but *s*FPL exploits a π reduced to 0.5, 0.1, and 0.4, respectively, for Brazil, Canada, and Italy. This behavior is more evident in Figure 4.2, where the harmonic mean between precision and recall (*F1*) is plotted for different values of π. If we look at the dark blue line with squares, we may observe how the best result does not correspond to $\pi = 1$. We can also note that *s*FPL outperforms BPR-MF with *MovieLens 1M*, but it remains at about 67% and 88% of the centralized algorithm for *Amazon Digital Music* and *LibraryThing*, respectively. Compared to FCF, FPL generally behaves better or similarly and preserves privacy to a greater extent, since sharing gradients of all rated items in FCF can result in a data leak [57].

In the last three rows of Table 4.2, we explore an increasing of the local computation (*s*FPL+), or an increased parallelism (*p*FPL), or a combination of both (*p*FPL+). In detail, we observe that *s*FPL+ takes advantage of the increased local computation, and FPL significantly outperforms BPR-MF for five over six datasets; for instance, for Canada, we observe an interesting increase in precision, while for *Amazon Digital Music* and *LibraryThing* improves accuracy metrics of about 50% and 25% with respect to BPR-MF. It is worth noticing that these results partially contradict Rendle *et al.* [167] since they hypothesize that traversing user-wise the training triples would worsen the recommendation performance. The same accuracy improvements are not visible in *MovieLens 1M*, where we witness results comparable or worse than BPR-MF, probably due to the overfitting caused by the very high ratio between ratings and items. Instead, when comparing *p*FPL with *s*FPL, we observe that the increased parallelism does not affect the performance significantly. Even then, the increased local computation boosts the Precision and Recall performance, up to 24% for precision in the Italy dataset. The results confirm RQ1, since *the proposed system can generate recommendations with a quality that is comparable with the centralized pair-wise learning approach. Moreover, the increased local computation causes a considerable improvement in the accuracy*

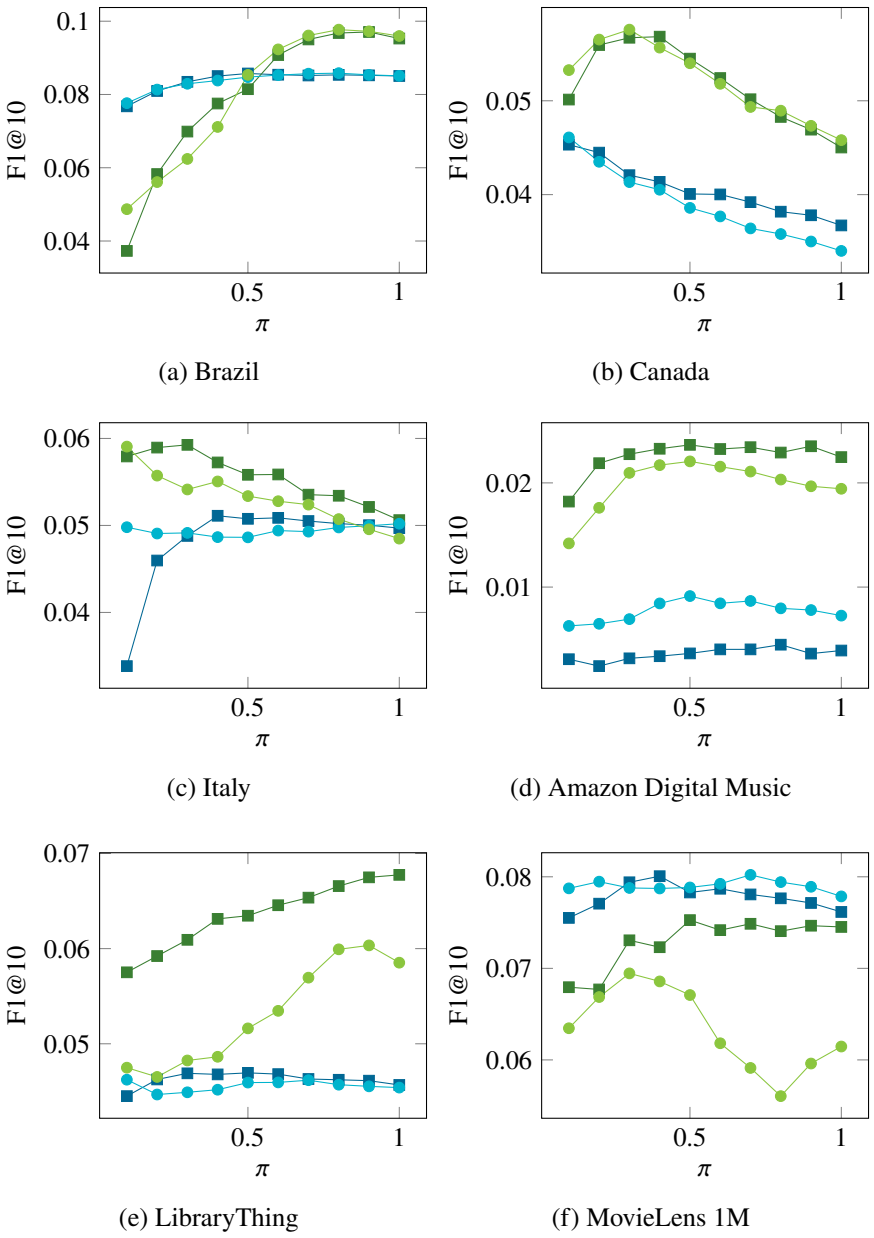


Figure 4.2: F1 performance at different values of π in the range $[0.1, 1]$. The colors represent the four configurations: blue squares refer to *s*FPL, green squares to *s*FPL+, blue circles to *p*FPL, and green circles to *p*FPL+.

of recommendations. On the other side, the training parallelism does not significantly affects results. Finally, when the ***local computation*** *is combined with* ***parallelism****, the results show a further improvement.*

To answer RQ2, we varied π in the range $[0.1, \ldots, 1.0]$ to assess how removal of the updates for consumed items affects the final recommendation accuracy, and we plotted the accuracy performance by

considering $F1$ in Figure 4.2. As previously observed, the best performance rarely corresponds to $\pi = 1$. On the contrary, a general trend can be observed: the training reaches a peak for a certain value of π — depending on the dataset —, and then the system performance decays in accuracy when increasing the value of π. In rare cases, e.g., *s*FPL, and *p*FPL for Brazil and LibraryThing, the decay is absent, but results that are very close for different values of π. The general behavior suggests that the system learning exploits the updates of positive items to absorb information about popularity. This consideration is coherent with the mathematical formulation of the learning procedure, and it is also supported by the observation that for Canada and Italy FPL reaches the peak before with respect to Brazil. Indeed, Canada and Italy datasets are less sparse than Brazil, and the increase of information about positive items may lead to push up too much the popular items (this is a characteristic of pair-wise learning), while the same behavior in Brazil can be observed for values of π very close to 1. The same mathematical background, for *s*FPL+ and *p*FPL+ with Brazil dataset, which is very sparse, explains the higher value of π needed to reach good performance. Here, the lack of positive information with a vast catalog of items, confuses the training that cannot exploit item popularity. A similar behavior is observable in *MovieLens 1M*, where FPL shows accuracy performance extremely close to the best value by sharing only 10% of positive interactions. This behavior may be due to several reasons. Firstly, *MovieLens1M* is a relatively dense dataset in the recommendation scenario (it has a sparsity of 0.955). Secondly, it shows a very high user-item ratio [8] (i.e., 1.63) compared to *Amazon Digital Music* (0.04), and *LibraryThing* (0.20), and it shows high values for the average number of ratings per user (132.87), and ratings per item (216, 56). All these clues suggest that the system learns how to rank items even without the need for the totality of ratings. Now, we can positively answer to RQ2: *user can receive high-quality recommendations also when she decides to disclose a small amount of her sensitive data. However, it should be noted that the more the dataset is sparse, the more the amount of sensitive data should be large*.

4.6.2 *Accuracy or Diversity: Analysis of the Trade-Off*

In Table 4.3, we have depicted the diversity metrics results of each experiment, i.e., item coverage, and Gini Index. What immediately catches our attention is an increase in IC and Gini in accord with the increase of local computation, except for *MovieLens 1M*. In this sense, FPL shows

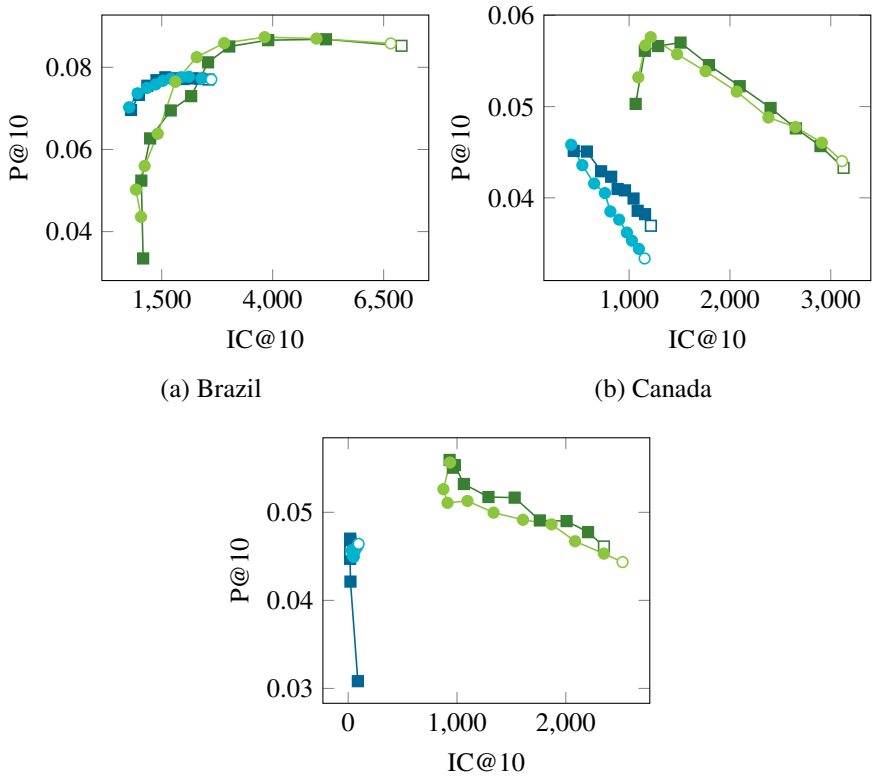
Table 4.3: Results of beyond-accuracy metrics for baselines and FPL on the six datasets. For each configuration of FPL and for each dataset, the experiment with the best π is shown (see the bottom part for details). For all metrics, the greater the better. Among federated algorithms, the best performance is in **boldface**.

		Brazil		**Canada**		**Italy**	
		IC@10	*G*@10	*IC*@10	*G*@10	*IC*@10	*G*@10
Centralized	**Random**	46120	0.70946	10815	0.26809	10478	0.28914
	Top-Pop	19	0.00020	18	0.00030	19	0.00035
	User-kNN	3083	0.01159	609	0.00321	577	0.00282
	Item-kNN	16535	0.07449	4393	0.05404	3241	0.03293
	VAE	5503	0.02117	1044	0.00652	165	0.02336
	BPR-MF	2552	0.00756	1216	0.00998	19	0.00036
Federated	**FCF**	911	0.00095	504	0.00174	403	0.00158
	***s*FPL**	1581	0.00561	451	0.00243	18	0.00036
	***s*FPL+**	**5200**	**0.01449**	**1510**	**0.01259**	932	**0.00789**
	***p*FPL**	2114	0.00638	425	0.00213	96	0.00056
	***p*FPL+**	3820	0.01106	1214	0.00981	**936**	0.00725

		Amazon DM		**LibraryThing**		**MovieLens 1M**	
		IC@10	*G*@10	*IC*@10	*G*@10	*IC*@10	*G*@10
Centralized	**Random**	14186	0.28069	31918	0.60964	3666	0.85426
	Top-Pop	24	0.00023	36	0.00031	118	0.00569
	User-kNN	4809	0.04115	3833	0.01485	737	0.04636
	Item-kNN	4516	0.03801	12737	0.09979	2134	0.19292
	VAE	3919	0.04179	7800	0.04638	1476	0.09259
	BPR-MF	739	0.00415	3082	0.01359	1444	0.08508
Federated	**FCF**	**2655**	0.01861	829	0.01305	829	0.01305
	***s*FPL**	349	0.00136	1650	0.00512	1041	**0.06608**
	***s*FPL+**	2586	0.02153	5404	0.02784	**1326**	0.02513
	***p*FPL**	1052	0.00737	1895	0.00760	901	0.04933
	***p*FPL+**	2614	**0.02163**	**5912**	**0.03133**	792	0.01451

a consistent prominence on BPR-MF. This performance is motivated by mere observation of the algorithm. By increasing local computation, each client compares each positive item with a significantly larger number of negative samples (i.e., wider spread).

For the *Foursquare* datasets, we have also explored the values of IC against the values of precision for each dataset and for each configuration while varying the parameter π. In Figure 4.3, we plot these values by considering increasing π in the direction of the arrows. The plots



(a) Brazil

(b) Canada

(c) Italy

Figure 4.3: Item Coverage (IC@10) versus Precision (P@10) with cutoff 10. The colors represent the four configurations: blue squares refer to *s*FPL, green squares to *s*FPL+, blue circles to *p*FPL, and green circles to *p*FPL+. The white points denote $\pi = 1.0$ to specify the direction of increasing π.

unveil that, for Canada and Italy, by increasing the local computation (*s*FPL+ and *p*FPL+), the plots develop rightwards, i.e., a significant IC increase. Although such an increase may lead to low precision (as in the random recommender), we observe that the same configurations also push up the value of precision, so that the green points are positioned at the top of the plots. We also note that the value of π affects more IC in configurations with high computation than those with low computation. However, while IC seems to increase when increasing π, precision follows the previously described behavior. At first glance, Brazil seems to behave differently from the other datasets. Even here, we may discern a better combination of IC and precision for configurations with high computation. FPL needs to reach a higher value of π to witness a high precision and high IC. This behavior was also evident in the accuracy analysis, considering the different values of π.

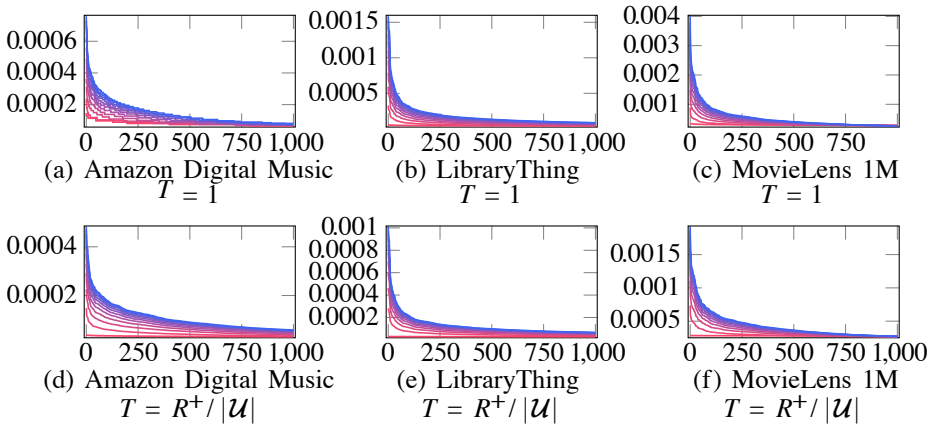


Figure 4.4: Normalized number of item updates during the training: the 1,000 most updated items for different values of π (from $\pi = 0.0$ in red to $\pi = 1.0$ in blue).

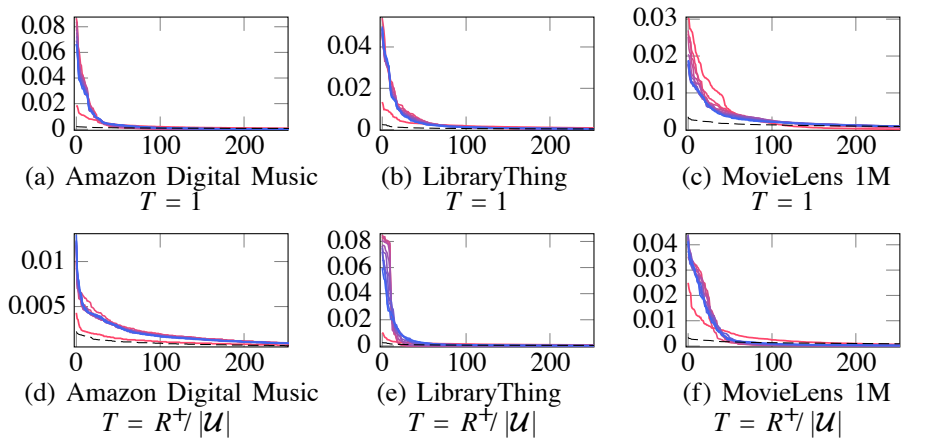


Figure 4.5: Normalized number of recommendations for each item (colored curves from $\pi = 0.0$ in red to $\pi = 1.0$ in blue) vs. normalized amount of positive feedback per item (black dashed curve). The 250 most popular items are shown.

4.6.3 *Impact of the Popularity Bias*

In this section, we study how incomplete transmission of user feedback affects the item popularity in the recommendations and during the learning process. It is essential to discover whether the exploitation of a federated learning approach influences the algorithmic bias, determining popular items to be over-represented [32, 53]. We have re-trained *s*FPL and *s*FPL+ on *Amazon Digital Music*, *LibraryThing* and *MovieLens 1M* with all the values of π in the range $[0.0, \ldots, 1.0]$. For each experiment, we analyzed the data flow between the clients and the server. Afterward, we have extracted the number of updates for

each item. Figure 4.4 illustrates the occurrences for the 1,000 most updated items. In the figure, the curve colors denote the different π, while the values represent the update frequency during the training process for each item on the horizontal axis. Analogously, we considered the final top-10 recommendation list of each user. Following the same strategy, we analyzed the occurrences of the items in the recommendation. Then, we ordered items from the most to the least recommended, and we plotted the occurrences of the first 250 in Figure 4.5. To compare the different datasets, we have normalized the values considering the overall dataset occurrences. Figure 4.4 shows that data disclosure, i.e., the value of π, highly influences the information exchanged during the training process. Additionally, the update frequency curve exhibits a constant behavior for all the datasets, when $\pi = 0.0$. This trend shows that items are randomly updated without taking into account any information about item popularity, suggesting that, when $\pi = 0.0$, FPL acts like a random recommender. The curve for $\pi = 0.1$ shows that the exchanged data is enough to provide the system with information about item popularity. The curves suggest that the information on item popularity is being injected into the system. By increasing the value of π, the trend becomes more evident. Due to the original rating distribution, the system initially exchanges more information about the very popular items. To analyze the algorithmic bias, we can observe Figure 4.5, where the colored curves represent the frequency of item recommendation on the horizontal axis, and the black dashed curve the amount of positive feedback for that item in the dataset. Remarkably, item popularity in recommendation lists does not vary as we may expect based on the previous analysis. The setting $\pi = 0.0$ is an exception, as explained before. Focusing on the curves for $\pi > 0.0$, it is noteworthy that they behave similarly, and they propose the same proportion of popular items. The curves show the model absorbs the initial variation in exchanged item distribution, unveiling an unknown aspect of factorization models, and partially explaining the acceptable (and sometimes outperforming) results with $\pi < 1.0$ observed in Figure 4.2.

4.6.4 *Accuracy and Communication: A Multi-Objective Analysis*

In a federated learning setting, communication rounds between clients and server play a crucial role. In fact, a large amount of information exchanged might hinder the effectiveness of the overall approach as it requires high network costs. This perspective has led us to define a metric, the *Communication Cost per Epoch* (CCE), which calculates

Table 4.4: Total Communication Cost ($\times 10^{-12}$) (*TCC*) versus Precision ($P@10$) on Brazil dataset. *TCC* is the product between the value of *CCE* and the actual number of epochs needed to obtain the best accuracy value. For each configuration, the best precision value is in **boldface** and reported in the summary graph in Figure 4.6.

	***s*FPL**		***s*FPL+**		***p*FPL**		***p*FPL+**	
π	*TCC*	$P@10$	*TCC*	$P@10$	*TCC*	$P@10$	*TCC*	$P@10$
0.1	1.27623	0.06961	1.41913	0.03347	1.27623	0.07026	0.99339	0.04358
0.2	1.27623	0.07327	1.41924	0.05241	1.27623	0.07366	0.99347	0.05022
0.3	1.27624	0.07551	1.41934	0.06269	1.27624	0.07497	0.99354	0.05598
0.4	1.27624	0.07686	1.41944	0.06949	1.27624	0.07582	0.99361	0.06382
0.5	1.27624	**0.07757**	1.41955	0.07298	1.27624	0.07671	0.99368	0.07648
0.6	1.27624	0.07733	1.41965	0.08121	1.27624	0.07723	0.99375	0.08247
0.7	1.27625	0.07714	1.41975	0.08506	1.27625	0.07758	0.99383	0.08590
0.8	1.27625	0.07730	1.41985	0.08660	1.27625	**0.07771**	0.99390	**0.08733**
0.9	1.27625	0.07724	1.41996	**0.08682**	1.27625	0.07726	0.99397	0.08699
1.0	1.27625	0.07702	1.42006	0.08523	1.27625	0.07703	0.99404	0.08582

communication costs that each particular FPL configuration requires as the number of bidirectionally exchanged vectors. Let T be the number of sent updates for non-consumed items and πT the number of sent updates for consumed items. For *rpe* rounds the server establishes a communication with $|\mathcal{U}^-|$ clients, sending to each of them $|\mathcal{I}|$ vectors and receiving from each of them $T(1+\pi)$ update vectors. Therefore, CCE is estimated as $CCE = rpe \cdot |\mathcal{U}^-| \cdot (|\mathcal{I}| + T(1+\pi))$. Given these definitions, in this section, we want to analyze the effects of the different configurations and π values on the communication cost.

For convenience, we focus the analysis on Brazil, the biggest and sparsest dataset. In Table 4.4, we show the values of precision and communication cost for each FPL configuration and each value of π. The Total Communication Cost (*TCC*) is computed as the product between CCE and the number of epochs needed to reach such precision value. At a first glance, it is noteworthy how, within a specific configuration, the value of π does not affect significantly the total communication cost, while it highly impacts on the best precision value. For each configuration, we plot in Figure 4.6 the best values of precision (in boldface in Table 4.4) against their *TCC*. Here, the total communication cost should be minimized, while the precision should be maximized. The optimal solution in terms of multi-objective optimization corresponds to the *p*FPL+ configuration. Instead, in absence of parallelism, we witness a

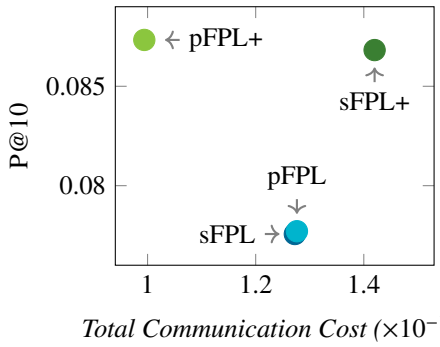


Figure 4.6: Total Communication Cost ($\times 10^{-12}$) compared to precision (P@10) on Brazil dataset. The colors represents the four configurations: dark blue is *s*FPL, dark green is *s*FPL+, light blue is *p*FPL, light green is *p*FPL+. For each configuration, the best accuracy performance is shown. The top-left corner of the plot is the best trade-off between accuracy and communication costs.

much higher communication cost for reaching the best precision. Moreover, it is interesting how *s*FPL and *p*FPL are perfectly overlapping both in terms of accuracy (as also confirmed by the previous analyses) and in terms of communication costs. However, increasing the local computation in a parallel setting make FPL to reach the best performance with the minimum overall communication cost.

The multi-objective analysis between communication cost and accuracy may help the designer in providing the best setup for the federation of clients. Here, the analysis suggests holding high parallelism configurations with high local computation as the set of optimal settings. The experiment shows that in FPL there is no need for sacrificing accuracy for communication costs. Instead, the user can freely choose the value of π without affecting the communication costs. In order to answer the RQ3 we can state that *deciding to limit the communication costs does not particularly affect the recommendation accuracy. Overall, FPL shows its best trade-off between communication costs and accuracy when both parallelism and high local computation are set.*

4.6.5 *Bias Disparity: A Fairness Analysis*

When depriving the recommender of a part of the user's feedback, one of the biggest concerns is the potential bias shift [48]. Bias analysis, and fairness are gaining momentum in the last years [67], they unveil several essential aspects of the recommenders' behavior. To explore what happens the category biases in the different configurations and

Table 4.5: Bias values B_T [142] of population on the different categories in *Foursquare* training data (A&E: Arts & Entertainment, C&U: College & University, NS: Nightlife Spot, O&R: Outdoors & Recreation, P&OP: Professional & Other Places, S&S: Shop & Service, T&T: Travel & Transport).

Dataset	A&E	C&U	Food	NS	O&R	P&OP	Res.	S&S	T&T
Brazil	1.4949	0.6289	1.1024	1.3286	1.1340	0.6424	0.5202	0.8314	1.3699
Canada	1.7224	0.8310	1.0879	1.6594	0.9719	0.6610	0.4134	0.8087	1.2328
Italy	1.4130	0.8221	0.9317	1.3559	1.3292	0.7868	0.4171	0.8482	1.2678

values of π, we measure the bias disparity (BD) in recommendation lists for the categories of the venues. This metric analyzes how much the output of a recommendation algorithm deviates the natural propensity of the users for particular categories of items towards other categories (see Section 2.2.4.3 for further details). Table 4.5 shows the source bias value B_T [142] for the different categories in training data, with a value above 1 denoting a higher susceptibility to choose the category items. Table 4.6 shows the results in terms of Bias Disparity (BD) for FPL and the other baselines. Here, the closer to 0, the closer to the initial bias. As expected, Top-Pop changed the recommendation towards T&T, which is the most popular category in the training set. By focusing on FPL, we may notice that it bias positively and negatively the same categories of the other state-of-the-art algorithms. Notably, it particularly pushes the bias of recommendation towards popular categories (e.g., A&E, Food, T&T), while it emphasizes the unpopularity of specific categories — above all C&U, P&OP, Residence —. This is probably due to the pair-wise nature of the approach, which works by iteratively increasing the difference values between enjoyed items and the others (the same behavior is evident for BPR-MF). The Bias Disparity analysis helps to answer RQ4. Hence, we draw the following consideration: *the proposed system generates recommendations that are biased to the initial user preferences since it emphasizes the differences between consumed and non-consumed items. This behavior is also coherent with the recommendations of the other state-of-the-art algorithms*.

4.7 Conclusion and Future Perspectives

This work proposes Federated Pair-wise Learning (FPL), a novel federated learning framework that exploits pair-wise learning for factoriza-

Table 4.6: Results of recommendation bias disparity for each category in Brazil dataset (see Table 4.1) for baselines and FPL. For each configuration of FPL and for each dataset, the experiment with the best π is shown. The closer to 0 the better. Among federated algorithms, the best performance is in **boldface**.

		A&E	C&U	Food	NS	O&R	P&OP	Res.	S&S	T&T
Centralized	**Random**	-0.325	0.559	-0.080	-0.245	-0.143	0.549	0.911	0.203	-0.273
	Top-Pop	-1.000	-1.000	-0.302	-1.000	-0.999	-1.000	-1.000	-1.000	6.660
	User-kNN	0.445	-0.832	0.261	-0.213	0.162	-0.842	-0.969	-0.431	0.621
	Item-kNN	0.346	0.140	0.068	-0.102	0.153	-0.347	-0.381	-0.227	0.083
	VAE	0.393	-0.723	0.223	-0.315	0.194	-0.776	-0.911	-0.310	0.572
	BPR-MF	0.301	-0.712	0.232	-0.710	0.142	-0.758	-0.992	-0.434	1.165
Federated	**FCF**	-0.464	-0.968	0.687	-0.910	**0.135**	-0.960	-0.994	-0.946	1.239
	***s*FPL**	0.272	-0.738	0.263	-0.756	0.161	-0.814	-0.997	-0.368	1.072
	***s*FPL+**	0.311	-0.675	**0.160**	-0.396	0.278	-0.806	**-0.903**	**-0.291**	0.812
	***p*FPL**	0.253	-0.813	0.218	-0.613	0.143	**-0.739**	-0.992	-0.479	1.239
	***p*FPL+**	**0.154**	**-0.566**	0.190	**-0.351**	0.345	-0.764	-0.913	-0.410	**0.778**

tion models in a recommendation scenario. The model leaves the user-specific information of the original factorization model in the clients' devices so that a user may be entirely in control of her sensitive data and could share no positive feedback with the server. The framework can be envisioned as a general factorization model in which clients can tune the amount of information shared among devices. To analyze the degree of accuracy, the diversity of the recommendation results, we have conducted an extensive experimental evaluation. However, even a vast evaluation is not enough to gain a more in-depth understanding of how FPL operates. Therefore, we have extended the evaluation to investigate the relation between accuracy, popularity bias and amount of shared transactions. Afterwards, the study provides a theoretical analysis of the privacy issues of FPL, the details of computational complexity, and an investigation on communication costs considering the different operational modes. Finally, the work analyzes the shift of the original data bias when the system is fed with partial information. To the best of our knowledge, it is one of the first attempts to understand how a federated learning approach impacts the fairness of the overall system. The proposed model shows performance comparable with several state-of-the-art baselines and the classic centralized factorization model with pair-wise learning. Interestingly, indeed, clients can share a small portion of their data with the server and still receive high-performance

recommendations. We believe that the proposed approach represents the joining link between federated matrix factorization and the modern recommendation systems that optimize the item ranking instead of the prediction error. In the near future, it would be interesting to investigate the behavior of FPL in new privacy settings, examine the effects of each user freely choosing which specific data to keep private, and extend the experimental analysis to other datasets and domains. Finally, we think that federated learning to rank approach, along with a rigorous analysis of the dimensions involved in the recommendation process, may open the doors to a new class of ubiquitous recommendation engines.

5 User-Level Knowledge for Improved Aggregation in Federated Learning

OUTLINE

In the previous chapters, we introduced the federated learning paradigm, able to deal with fundamental issues related to privacy, ownership and locality of data [43] in machine learning. One of the pioneer works about federated learning [146] introduced the *Federated Averaging* (*FedAvg*) algorithm. Despite its potentially disruptive contribution, we argue that FedAvg has several significant shortcomings, related to the fact that it ignores a wealth of qualitative measures about local clients and models that can be impactful for training an effective global model.

In this chapter, we present our approach to push forward the state-of-the-art of the aggregation in federated learning, which we deem crucial for building a high-quality global model, with an approach that takes into account a suite of *client-specific criteria*.

A preliminary version of this work has been accepted as a long paper entitled "*Towards Effective Device-Aware Federated Learning*" at the 18th International Conference of the Italian Association for Artificial Intelligence. An extension of this work, "*Prioritized Multi-Criteria Federated Learning*", has been published in the journal Intelligenza Artificiale.

5.1 Introduction

Federated learning is an approach proposed by Google [116, 117, 146] to train a global machine learning model from a massive amount of data, which is *distributed* on the client devices such as personal mobile phones and IoT devices. Raw data is not shared at all with a central server, thus federated learning respond to concerns about the sensitivity of user data in terms of data privacy and security. However, in this scenario, we have to deal with data that is quantitatively unbalanced and differently distributed over devices, i.e., each device data is not a representative sample of the overall distribution, as we have seen in Section 3.3.2. As a matter of fact, with federated learning, we leverage

users' computing power for training a shared machine learning model while preserving privacy, by actually decoupling the ability to learn a machine learning model from the need to store private data centrally.

In [146], authors introduced the *FederatedAveraging* (FedAvg) algorithm, which combines local stochastic gradient descent on each client via a central server that performs model aggregation by averaging the values of local parameters. Despite its potentially disruptive contribution, we argue that FedAvg has several significant shortcomings. First, the aggregation operation in FedAvg assigns the contribution of each agent proportional to each client's local dataset size. This simplifying assumption ignores a wealth of other qualitative measures that can be impactful for training an effective global model. Examples of such measures include the number of sample classes held by each agent, the divergence of each computed local model from the global model — which may be critical for convergence [175] —, estimations about the agent computing and connection capabilities and finally the amount of client's honesty and trustworthiness.

While FedAvg only uses limited knowledge about local data, we argue that the integration of the aforementioned qualitative measures and the expert's domain knowledge is fundamental for increasing the global model's quality. As a toy example, let us consider a federated scenario with just two users Alice and Bob. The photos in users' mobile phones are the training samples of a machine learning model for classifying clothes. Let us suppose that Alice holds tens of very similar photos with the same outfit, and most of them are blurred. Thus these images do not carry much information. Instead, Bob holds a smaller number of well-labeled photos, but high-quality, and with a lot of different clothes. We argue that in such a situation, the weight of Bob's contribution to the ML model should be higher or comparable to Alice's.

The work at hand considerably extends the FedAvg approach [146] by answering to the following research questions:

RQ1. Is it possible to improve the quality of the global model by incorporating *a set of criteria* measuring some quality properties of the clients, and based on them assigning the contribution of individual update in the final model?

RQ2. What is the effect of introducing a *score function* "summarizing" with a certain priority the introduced criteria?

The proposed system is evaluated through comprehensive experiments. In particular, we compare our approach against FedAvg both on MNIST dataset (with IID and non-IID distribution), used in [146], and on CelebA dataset, where a classification task is performed by also in-

jecting more explicit information about the quality of the local dataset and local model.

5.2 Related Work

In literature, a growing number of works focus on some modifications of FedAvg algorithm, e.g., adding a regularization term in the local objective functions for controlling the convergence of the network with statistical heterogeneity of data [175], changing the optimization algorithm [163]. Some other works focus on some adaptations for controlling the averaging operation on a per-layer basis [203] or for obtaining more personalization on the local models [81]. Finally, the work in [222] studies a protocol for selecting devices based on their dataset quality without any information disclosure.

The approach we propose is configured as a formal tool for incorporating any of the above-mentioned objectives (for example, facing the statistical challenges), when these could be obtained by pushing up or down the contribution of each client, based on some requirements. Such mechanism is completely incorporated in the weight of each client in the global aggregation. To the best of our knowledge, at the time of publishing our proposal, this is the only work attempting to modify the value of the weight of clients in the aggregation step.

5.3 Background

Although federated learning principles can be applied to many machine learning tasks, for the sake of simplicity in the following we focus on a classification task, a fundamental problem in machine learning.

Assume a training set $\mathcal{D}$ composed of n pairs, i.e. $\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \ldots, n\}$, where the elements $x_i \in \mathcal{X}$ are the input samples, and $y_i \in \mathcal{Y}$ the corresponding class labels. It is presumed that the samples of dataset are independently and identically drawn from an unknown distribution $\mathcal{P}$. The problem of classification is often expressed as finding a function $f_\Theta : \mathcal{X} \to \mathcal{Y}$ that can approximate the class labels around the input samples, where $\Theta \in \mathbb{R}^L$ represents the model parameters and L is the dimensionality of the parameters space. Finding the model parameters Θ is achieved by solving a problem of the form:

$$\min_{\Theta} \sum_{(x_i, y_i) \in \mathcal{D}} G(f(x_i; \Theta), y_i) \tag{5.1}$$

where $G(\cdot)$ represents a classification loss function, e.g., cross-entropy.

In the classical federated learning setting, at each round of communication, the server S selects a fraction of clients $\mathcal{U}^- \subseteq \mathcal{U}$ from the federation $\mathcal{U}$ and shares with them the current global model Θ. Every client $u \in \mathcal{U}^-$ uses it own private shard $\mathcal{D}_u$ in order to minimize an *empirical local loss* $G_{\mathcal{D}_u}(\Theta)$. Each client u updates the received global model Θ and calculates a local version Θ_u according to Eq. 5.2. Then, it communicates the updated local model Θ_u (instead of the local gradients) to S:

$$\Theta_u \leftarrow \Theta - \alpha \nabla G_{\mathcal{D}_u}(\Theta). \tag{5.2}$$

Once the end of the communication round has been reached, S computes the updated global model Θ as:

$$\Theta \leftarrow \sum_{u \in \mathcal{U}^-} w_u \Theta_u. \tag{5.3}$$

The federated learning principles prevent a machine learning system from collecting *sensitive* information about users, giving rise to a natural trade-off between users' data privacy and performance of the system. Our assumption is that revealing some *non-sensitive client-related information* and integrating this knowledge in the global aggregation step could lead to learning more effective federated model without harming users' privacy. For instance, such non-sensitive data may carry useful information about specific domain or some marketing objectives that can be leveraged to build more *in-domain* strategies or increase the system *profitability*.

The proposed solution envisioned in this work is a client-aware federated learning strategy based on the following elements:

- we introduce the notion of *criterion*, which measures a specific property about users and their data, and we propose a formal classification for them;
- we propose the use of a *score function* to "summarize" the criterion measurements and compute a score for each client of the federation to be encoded in the aggregation step of each round of communication;
- we explore the benefits of a *prioritized multi-criteria score function* over the identified set of criteria;

5.4 Fundamentals of the Proposed Approach

Let us assume $C = \{C_1, ..., C_m\}$ denotes a set of m measurable properties (a.k.a. criteria) characterizing a local client u or local data $\mathcal{D}_u$. We use the variable $c_{i,u} \in [0, 1]$ to denote, for each client u, the degree of satisfaction of the criterion C_i in a specific round of communication. We assume that the clients compute these values and communicate with the server the correct model updates and the correct property measurements (honest clients). Hence, at the end of the communication rounds, the server S is in charge of collecting not only the model update Θ_u, but also the m-tuple $\mathbf{c}_u = (c_{1,u}, \ldots, c_{m,u})$. To ensure the same scale for each criterion, we assume that the measure $c_{i,u}$ of the i-th criterion on device u is a real value in the interval $[0, 1]$ (with 0 meaning unfulfillment and 1 meaning total adherence to the criterion). Additionally, the measurements of C_i over all devices in $\mathcal{U}^-$ are supposed to be normalized, i.e., $\sum_{u \in \mathcal{U}^-} c_{i,u} = 1$.

The main idea of this work is that we can encode the knowledge about clients in the weights w_u used for aggregating client contributions in Eq. 5.3. Based on that, S can compute the weight w_u for client u according to the following equation:

$$w_u = \frac{f_m(\mathbf{c}_u)}{Z} = \frac{f_m(c_{1,u}, ..., c_{m,u})}{Z}, \tag{5.4}$$

where $f_m : [0, 1]^m \rightarrow \mathbb{R}$ is a score function over the m-tuple of properties (criteria), which evaluates client u's contribution based on the fulfillment of such criteria. Finally, Z is a normalization factor introduced to ensure that $\sum_{u \in \mathcal{U}^-} w_u = 1$ and $w_u \in [0, 1]$; therefore, $Z = \sum_{u \in \mathcal{U}^-} f_m(\mathbf{c}_u)$.

Example 5.1. Let us go back to the toy example we introduced in Section 5.3, with $\mathcal{U}^- = \{Alice, Bob\}$. Let us consider the set C of criteria describing the two clients of the federation, i.e., specific qualities related to their local devices, produced local models and local data; for example we may consider ***D**ataset **S**ize*, ***C**lothes **D**iversity*, and ***I**mage **S**harpness* thus having $C = \{DS, CD, IS\}$. Assume that Alice has received evaluations $c_{DS,\text{Alice}} = 0.9$, $c_{CD,\text{Alice}} = 0.2$, $c_{IS,\text{Alice}} = 0.4$, while Bob has obtained $c_{DS,\text{Bob}} = 0.1$, $c_{CD,\text{Bob}} = 0.8$, $c_{IS,\text{Bob}} = 0.5$. Based on Eq. 5.4, the overall contribution of Alice and Bob will be $f_3(0.9, 0.2, 0.4)$ and $f_3(0.1, 0.8, 0.5)$, respectively, both divided by $Z = f_3(0.9, 0.2, 0.4) + f_3(0.1, 0.8, 0.5)$ to obtain the weights w_{Alice} and w_{Bob}. To get an idea, if we consider the score function f_3 to be a basic mean operation, the final values for w_{Alice} and w_{Bob} would be: $w_{\text{Alice}} = \frac{1.5}{1.5+1.4} = 0.52$, $w_{\text{Bob}} = \frac{1.4}{1.5+1.4} = 0.48$. □

In the following, we detail the main dimensions of the study related to the proposed approach, which include:

- identification of the set C of criteria. These criteria can be related to users (e.g., gender), clients (e.g., number of samples), or local models (e.g., the local model highly diverges from the centralized model). Besides, the criteria can express a boolean fulfillment of the requirement (e.g., Female: True or False), or a quantitative estimation (e.g., the value of the divergence of the model);
- identification of a score function able to "summarize" the computed measurements in a score value representing each client.

To properly study each of the presented dimensions, in the following we start discussing about the identification of criteria, then we go along the study of score functions and, in particular, priority-based score functions.

5.4.1 *Identification of Local Criteria*

In the original FedAvg formulation, the server performs aggregation to compute w_u, without knowing any information about participating clients, except for a pure quantitative measure about local dataset size.

However, a federated setting has to face some key issues [146] related to communication, connectivity and statistics. Among them, if we focus on the statistical aspects about the data, we notice that the training data — which is typically the result of the real user usage of the device — is not IID distributed over the clients. Moreover, the amount of data is also unbalanced across the clients. Such characteristic may affect the accuracy and the efficiency of the resulting aggregated model [232]. In this scenario, a service provider may be interested in defining some statistical criteria such that the rounds of communication needed to reach the desired target accuracy are minimized. This result can be accomplished by enhancing the contribution of clients fulfilling the requirements expressed by the defined criteria.

For instance, useful information could be related, in a classification problem, to the number of classes covered by a local dataset. Moreover, a domain expert could ask users to measure their adherence to some other target properties (e.g. their nationality, gender, age, job, behavioral characteristics, etc.), in order to build a global model emphasizing the contribution of some classes of users; in this way, the domain expert may, in principle, build a model favoring some targeted commercial purposes.

We identified four classes of local criteria, each of them related to different aspects of the local dataset $\mathcal{D}_u$, the local device or the user u:

- criteria describing the quality of the local dataset $\mathcal{D}_u$ (e.g., dataset size, diversity of training samples, etc.);
- criteria describing the quality of the produced local model $\nabla G_{\mathcal{D}_u}$ (e.g., ...);
- criteria describing the trustworthiness of device u;
- criteria capturing the fulfillment of user u with respect to commercial targets (e.g., gender, job, nationality, etc.).

Although we can identify some classes of criteria, the choice for each particular criterion still remain a strictly task- and domain-dependent activity. We provide some insights about how to define such criteria: first, as a rule of thumb, one should start from the objective, e.g., obtaining faster convergence, overshadow unreliable updates, specialize the model with respect to some categories of users and then choose characteristics which can improve the model towards that initial objective. Next, the system designer could play with the identified criteria to test their effectiveness, e.g. with some previous centralized data, or with some synthetic data, or with some validation rounds of federated training. Finally, the empirical evaluations performed in this work suggest that choosing criteria that lead to higher variance in the score obtained across clients results in a better final model. For instance, the criterion dataset size is not an appropriate criterion, if all the clients have the similar number of samples in their private datasets.

For this reason, in this work we do not focus on a particular choice for them, but rather on presenting a formal approach about dealing with them. Moreover, we show in the experimental section — for illustrative purposes only — some examples of how they can be chosen with respect to the task.

5.4.2 *Identification of a Score Function*

A crucial aspect of this work is related to the identification of a score function f_m able to "summarize" the values $c_{i,u}$ for each criterion $C_i \in C, i \in [1, \ldots, m]$ in order to obtain the value w_u as described in Eq. 5.4. The score has to be computed for each client separately, so in the following we will focus on the computation of such value for one client $u \in \mathcal{U}$.

Over the years, a wide range of aggregation functions have been proposed in the field of information retrieval (IR) [143]. Just to mention the most relevant ones, we can refer to the weighted averaging operator, as well as to the ordered weighted averaging (OWA) models [215, 216] — which extend the binary logic of *AND* and *OR* operators, allowing representation of intermediate quantifiers —, to the Choquet-based models [61, 89, 90] — which are able to interpret positive and negative interactions between criteria —, and finally to the priority-based models [63].

Although we have many opportunities, we focus on the priority-based score function proposed in da Costa Pereira et al. [63]. To reformulate such proposal, we first consider a sequence of functions $\{f_m\}_{m\in\mathbb{N}}$, with $f_n : [0,1]^n \to \mathbb{R}$, namely a score function summarizing m data. Then, given a vector $\mathbf{x} \in [0,1]^m$ containing the measurement of m criteria, we can reformulate the score function proposed in [63] as:

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \tag{5.5}$$

This function has the monotonic property

$$f_m(x_1, \ldots, x_i, \ldots, x_m) \leq f_m(x_1, \ldots, x_i', \ldots, x_m) \quad \forall x_i \leq x_i', \quad i = 1, \ldots, m. \tag{5.6}$$

and, at the same time, we have $f_m(\mathbf{0}) = 0$ and $f_m(\mathbf{1}) = m$, i.e., the final score is zero when all values are 0, and it is maximum when all values are 1. One of the most interesting properties of this function is that:

$$f_m(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_m) = f_{j-1}(x_1, \ldots, x_{j-1}), \quad \forall j \in 1, \ldots, m. \tag{5.7}$$

This means that the lack of fulfillment of a higher criterion in the list cannot be compensated with the fulfillment of a lower one [143]. If we adopt a priority order for the criteria in $\mathbf{x}$ from the higher to the lower, it follows that in case a higher criterion is equal to 0 then it cannot be compensated by criteria with a lower priority. Interestingly, we also have that $f_m(0, x_2, \ldots, x_m) = 0$.

The main aim of such function, in a multi-criteria scenario, is to use a priority order over the involved criteria in order to assign to each client u a score based on the measurements $c_{i,u}$, for each $C_i \in C, i \in [1, \ldots, m]$, in a priority-aware fashion. With this function, when a criterion is met, the more it is fulfilled the more the subsequent criteria will be taken

into account; analogously, the less a criterion is fulfilled the less the other criteria will be considered and summed up. Moreover, based on Eq. 5.7, when a criterion is not satisfied at all, all the subsequent criteria will not be considered.

The properties presented so far make this function our main choice for our approach. As an example, we may consider the case where the domain expert may wish to consider extremely important the age of a user rather than its dataset size, so that even a large local dataset would be penalized if the user age criteria is not satisfied.

In the following, we assume, for the sake of clarity of notation, that the set C of criteria can be written in the form $C = \{C_1, \ldots, C_m\}$ where each index represents an identifier for the corresponding criterion, or in the form $C = \{C_{(1)}, \ldots, C_{(m)}\}$, where each index represent the priority of the criterion, from the highest to the lowest one. Analogously, we distinguish between $c_{i,u}$ and $c_{(i),u}$, which represent the measurement on device a of the criterion C_i and the i-th important criterion, respectively.

Example 5.2. ***(continued)*** We carry on with the example about Alice and Bob, where $c_{DS,\text{Alice}} = 0.9$, $c_{CD,\text{Alice}} = 0.2$, $c_{IS,\text{Alice}} = 0.4$, while Bob has obtained $c_{DS,\text{Bob}} = 0.1$, $c_{CD,\text{Bob}} = 0.8$, $c_{IS,\text{Bob}} = 0.5$. We have already shown that with a simple mean function their final scores were $w_{\text{Alice}} = \frac{1.5}{1.5+1.4} = 0.52$, $w_{\text{Bob}} = \frac{1.4}{1.5+1.4} = 0.48$, with Alice obtaining higher score than Bob. Let us consider now the prioritized score function and let us see how score changes based on priority. Suppose that $C_{(1)} = DS$, $C_{(2)} = CD$, $C_{(3)} = IS$. Based on Eq. 5.5,

$$\begin{aligned} f_3(\mathbf{c}_{\text{Alice}}) &= f_3(c_{(1),\text{Alice}}, c_{(2),\text{Alice}}, c_{(3),\text{Alice}}) \\ &= f_3(c_{DS,\text{Alice}}, c_{CD,\text{Alice}}, c_{IS,\text{Alice}}) \\ &= f_3(0.9, 0.2, 0.4) \\ &= 0.9 + (0.9 \cdot 0.2) + (0.9 \cdot 0.2 \cdot 0.4) = 1.152, \end{aligned}$$

while $f_3(\mathbf{c}_{\text{Bob}}) = 0.22$. If we change the priority order to be $C_{(1)} =$ image sharpness, $C_{(2)} =$ clothes diversity, $C_{(3)} =$ dataset size, we would then obtain:

$$\begin{aligned} f_3(\mathbf{c}_{\text{Alice}}) &= f_3(c_{(1),\text{Alice}}, c_{(2),\text{Alice}}, c_{(3),\text{Alice}}) \\ &= f_3(c_{IS,\text{Alice}}, c_{CD,\text{Alice}}, c_{DS,\text{Alice}}) \\ &= f_3(0.4, 0.2, 0.9) \\ &= 0.4 + (0.4 \cdot 0.2) + (0.4 \cdot 0.2 \cdot 0.9) = 0.552, \end{aligned}$$

while $f_3(\mathbf{c}_{\text{Bob}}) = 0.94$. We see that with the second configuration, the score for Alice is lower than the previous one since the most important criterion here is worse fulfilled, conversely for Bob. □

5.5 Experimental Setup

In this section we describe the experimental setup — datasets, tasks and identified local criteria — used to validate the performance of the proposed federated learning system[1]. In detail, we validate our approach on the following datasets and tasks:

- **MNIST** [125]: handwritten digits; we perform a 10-class classification for recognition of digits;
- **CelebA** [137]: face images of celebrities coming with 40 different attributes under varying poses and backgrounds; we perform a binary prediction between smiling and non-smiling faces.

For each dataset we consider different data distributions, as well as a specific model. Moreover, for each of them we define different criteria based on possible useful information we may extract.

5.5.1 *MNIST Experiments*

We run a first cluster of experiments on MNIST dataset [125] for the digit recognition task. This dataset contains 60,000 examples of 10 classes of handwritten digits (plus a test set of 10,000 examples) by 500 writers. The samples are available in the form of 28x28 pixel black and white images.

We use this dataset in a completely user-agnostic way. Therefore, we created the federated dataset by following the two partitioning ways used in [146], in order to simulate both a IID distribution and a non-IID distribution of data over the different clients.

Model. For the sake of comparability, we built the same classifier described in [146], i.e., a CNN with two convolution layers with 5x5 filters — the first layer with 32 channels, the second with 64, each followed with a 2x2 max pooling layer —, a fully connected layer with 512 units and ReLu activation, and a final softmax output layer with 10 neurons, for a total of 1,663,370 total parameters.

Local criteria. For this experimental setting, we aim at both reducing the number of rounds of communication necessary to reach a target accuracy and making the global model not diverging towards local

1 A public implementation of our framework is available at `https://github.com/sisinflab/ClientAware-FL`.

specializations and overfittings. To this aim, we extend the pure quantitative criterion in FedAvg [146] — dataset size — by leveraging two new criteria[2].

The first criterion we consider is the one already used by FedAvg [146], namely the local dataset size (**DS**), given by $c_{1,u} = |\mathcal{D}_u|/|\cup_{i\in\mathcal{U}} \mathcal{D}_i|$. This criterion is a *pure quantitative measure* about the local data, which will serve both as a baseline in empirical validation of the results (i.e., when used in isolation) and as part of the entire identified set of criteria in the developed FL system (i.e., when used in a group).

The second considered criterion is the *diversity of labels* (**LD**) in each local dataset, measuring the diversity of each local dataset in terms of class labels. We assert this criterion to be important since it can provide a clue on how much each device can be useful for learning to predict different labels. To quantify this criterion we use $c_{2,u} = \delta(\mathcal{D}_u)/\sum_{i\in\mathcal{U}} \delta(\mathcal{D}_i)$ where δ measures the number of different labels (classes) present over the samples of that dataset.

With respect to the third criterion, our aim is to reduce the negative effects of a non-IID distribution. Indeed, with non-IID distributions — and this is the case of our dataset — model performance dramatically gets worse [232]. Moreover, a large number of local training epochs may lead each device to move further away from the initial global model, towards the opposite of the global objective [175]. Therefore, a possible solution inspired by [175] is to limit these negative effects, by penalizing higher divergences and highlighting local models that are not very far from the received global model. We evaluate the local model divergence (**MW**) as $c_{3,u} = \varphi_u/\sum_{i\in\mathcal{U}} \varphi_i$ where $\varphi_i = \frac{1}{\sqrt{||\Theta-\Theta_u||_2+1}}$.

5.5.2 *CelebA Experiments*

CelebFaces Attributes Dataset (CelebA) [137] is a large-scale dataset with 202,599 face RGB images of 10,177 celebrities. Each image comes with 40 binary attributes and differs from the other ones for celebrity pose and background. The task is a binary classification between smiling and non-smiling people, which is an information included within the 40 attributes. We chose this task since the smiling attribute has a good balance in the whole dataset between positive and negative outcomes.

2 Please note that we are not stating that the proposed ones are the only possible criteria. We present them just to show how the introduction of new information may lead to a better final model.

We use this dataset in a completely user-aware way. Indeed, we suppose that each celebrity holds her own photos in her mobile phone gallery. This represent a realistic set of local datasets, which could have been generated from the personal device usage. This distribution is inherently non-IID, therefore it is a representative scenario for a federated setting.

We used a subsampled version of the dataset (50% of images). Then, we removed users with less than 5 photos. For each user, we split her private dataset with random hold-out method with a ratio of 80/20. Finally, images have been resized to 64x64 pixels.

Model. For the CelebA experiments, we built a CNN binary classifier with two convolution layers with 3x3 filters — the first layer with 32 channels, the second with 64, each followed with a 2x2 max pooling layer —, a fully connected layer with 512 units and ReLu activation, and a final softmax output layer with 1 neuron, for a total of 8,409,025 total parameters.

Local criteria. Also in these experiments, our aim is to reduce the number of rounds of communication needed to reach a target accuracy. Three criteria have been used to reach such objective.

Also in this case, we keep the dataset size criterion (**DS**). Therefore, $c_{1,u} = |\mathcal{D}_u|/|\cup_{u\in\mathcal{U}^-}\mathcal{D}_i|$. It will serve both as a baseline in empirical validation of the results and as part of the entire identified set of criteria in the developed FL system.

Similarly to what has been done with the MNIST dataset, we want to consider how distributed are the labels of the local samples. In fact, since we are dealing with a binary classification task, we have at most two different classes in each dataset. For this reason, we consider a measure of class balance (**CB**), i.e. how they are similar in number. To this aim, we define a function ζ, which measure the ratio between the number of samples of the two classes, namely:

$$\zeta(\mathcal{D}_u) = \frac{\min\{\#\text{ pos.}, \#\text{ neg.}\}}{\max\{\#\text{ pos.}, \#\text{ neg.}\}} \tag{5.8}$$

Then, the second criterion has been defined as $c_{2,u} = \zeta(\mathcal{D}_u)/\sum_{i\in\mathcal{U}^-}\zeta(\mathcal{D}_i)$.

As mentioned above, this dataset comes with a set of binary attributes describing each image. Among them, we consider the blurriness as a non-sensitive information, so that the percentage of sharp images within the private dataset can be shared with the server. Therefore, we define $c_{3,u} = \xi(\mathcal{D}_u)/\sum_{i\in\mathcal{U}^-}\xi(\mathcal{D}_i)$, where ξ measures the fraction of sharp

images in a specific dataset. This criterion (**IS**) gives us an idea of the quality of the images in the dataset.

5.5.3 *Reproducibility*

We set the hyperparameters for the whole set of our experiments as follows, also guided by the results obtained in [146]. As for the FedAvg client fraction parameter, in each round of communication only 10% of clients are selected to perform the computation, so that $|\mathcal{U}^-| = |\mathcal{U}|/10$. For what concerns the parameters of stochastic gradient decent (SGD), we used full batch approach and we set the number of local epochs equal to 5, i.e., each client takes 5 epochs of gradient descent during training. Moreover, we chose the learning rate based on a grid search by looking for the value which makes it possible to first reach the target accuracy in 50% of devices with the baseline approach (dataset size as the only criterion). Finally, we set the maximum number of rounds of communication per each experiment to 1000 for MNIST and to 100 for CelebA.

5.5.4 *Evaluation*

The classification model we built have been evaluated with respect to classification accuracy, i.e.,

$$\text{accuracy} = \frac{\text{\# correct predictions}}{\text{\# total predictions}}. \tag{5.9}$$

This metric has been computed on each device over the private local test set. Based on LEAF framework [51] — which provides reproducible reference implementations and datasets for FL, as well as system and statistical metrics —, we estimate a global accuracy by averaging local accuracy values and weighting them based on local test set size.

Moreover, we improve the validation of the FL setting by using an approach which offers an overview of the whole training performances, instead of metrics describing a single round of communication. More specifically, we measure the number of round of communication required to allow a certain percentage of devices, which participate to the federation process, to reach a target accuracy (e.g., 75% or 80%), since this measurement is able to fairly show how effective and efficient is the model across the devices.

5.6 Results and Discussion

In this section, we show and discuss the results of the experiments by considering both the evaluation approaches previously mentioned.

Tables 5.1, 5.3, and 5.5 show in the gray cells the number of rounds of communication required to the baseline (only DS) to allow certain fractions of devices to reach a specific percentage of the overall accuracy, for MNIST IID distributed, MNIST non-IID distributed, and CelebA, respectively. In detail, as target accuracies, we have considered 70%, 80%, 90%, and 95% of prediction accuracy for MNIST datasets, and 70%, 80%, and 85% for CelebA dataset. The remaining rows of the tables show the gain in terms of rounds of communication of each experimental setting against the baseline. Therefore, we compute each row as the difference between the results of the dataset size criterion model, and the results of the corresponding model. The higher the positive value, the better is the approach compared to the baseline. Moreover, for each row and target accuracy, we show the average rounds of communication gained by considering all the fractions of devices. For each target accuracy, we show how many training rounds are needed to grant that accuracy for 10% up to 90% of devices. To have a comprehensive view of the effects of each criterion and their combinations, we consider the individual criteria and their combinations separately. Specifically, we first show the results of the experiments realized by exploiting a single criterion (i.e., in Eq. 5.5). Then, we show the results for the six priority permutations of the three criteria.

Moreover, we provide for each dataset the confusion matrices (Tables 5.2, 5.4, and 5.6) which summarize the classification outcomes (correctly classified vs. misclassified) of each experiment against the baseline (only DS criterion). They have been obtained by comparing the best outcome of each experiment in order to highlight the significance of results irrespectively of the accuracy alignment presented in Tables 5.1, 5.3, and 5.5.

Finally, we also show the results in terms of global test accuracy while considering the rounds of communication. In this respect, we have measured global accuracy as the average of the local test accuracy values weighted by the local test size.

In detail, Figures 5.1a, 5.1b, and 5.1c, show these curves for MNIST IID distributed, MNIST non-IID distributed, and CelebA, respectively.

Table 5.1: Results for MNIST IID (the symbol ≻ denotes the priority relation). The row DS provides the number of rounds of communication needed to make the percentage of devices specified in the columns reach the specified target accuracy. The remaining rows show the gain in rounds of communication with respect to the baseline (the higher the better). The column Avg. shows the average gain for all the percentages of devices.

	Target accuracy 70%										**Target accuracy 80%**									
Experiment	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**
DS (baseline)	1	1	1	1	1	4	4	4	5		1	2	5	5	5	5	5	5	5	
LD	0	0	0	-2	-2	1	1	1	0	-0.11	-2	-1	0	0	0	0	0	0	0	-0.33
MW	0	0	0	0	0	3	0	0	0	0.33	0	-2	1	0	0	0	0	0	0	-0.11
DS ≻ LD ≻ MW	0	0	-1	-3	-3	0	0	0	0	-0.78	-3	-2	0	0	0	0	0	0	0	-0.56
DS ≻ MW ≻ LD	0	0	0	0	0	0	0	0	1	0.11	-3	-2	1	1	1	1	1	1	1	0.22
LD ≻ DS ≻ MW	0	0	0	0	0	2	0	0	1	0.33	-1	-2	1	1	1	0	0	0	-1	-0.11
MW ≻ DS ≻ LD	0	0	0	-3	-3	0	0	0	1	-0.56	-3	-2	1	1	1	1	0	0	0	-0.11
LD ≻ MW ≻ DS	0	0	0	0	0	3	3	3	1	1.11	0	1	4	1	1	1	1	1	1	1.22
MW ≻ LD ≻ DS	0	0	0	0	0	1	1	1	1	0.44	-2	-1	1	1	0	0	0	0	0	-0.11

	Target accuracy 90%										**Target accuracy 95%**									
Experiment	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**
DS (baseline)	5	5	5	5	7	7	10	11	14		5	7	10	11	16	21	24	38	56	
LD	0	0	0	0	0	0	1	-1	-1	-0.11	0	0	2	0	-2	1	-3	0	-3	-0.56
MW	0	0	0	0	0	0	0	-1	-1	-0.22	0	0	2	-4	1	-1	-4	1	-3	-0.89
DS ≻ LD ≻ MW	0	0	0	0	2	0	-1	-1	0	0.00	0	0	2	0	-3	0	-5	4	-10	-1.33
DS ≻ MW ≻ LD	1	1	1	-1	0	0	1	-2	-1	0.00	1	0	1	-1	-1	0	-1	3	-14	-1.33
LD ≻ DS ≻ MW	1	0	0	-3	-1	-1	1	0	-2	-0.56	0	-1	2	-2	-1	-2	-2	-2	-3	-1.22
MW ≻ DS ≻ LD	0	0	-1	-1	0	0	2	-1	1	0.00	-1	0	2	-1	2	2	1	4	-4	0.56
LD ≻ MW ≻ DS	1	1	1	1	2	-1	2	1	2	1.11	1	1	2	2	-1	2	-1	0	-26	-2.22
MW ≻ LD ≻ DS	0	0	0	0	2	-1	1	0	-1	0.11	0	0	2	-1	-1	-1	-3	4	-16	-1.78

Table 5.2: Confusion matrices for MNIST IID with the number of samples misclassified (✗) or correctly classified (✓) by DS and our experiments.

		LD		**MW**		**DS≻LD≻MW**		**DS≻MW≻LD**		**LD≻DS≻MW**		**MW≻DS≻LD**		**LD≻MW≻DS**		**MW≻LD≻DS**	
		✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
DS	✗	0	98	0	98	2	96	0	98	2	96	0	98	4	94	0	98
DS	✓	106	9796	107	9795	106	9796	128	9774	118	9784	140	9762	107	9795	122	9780

Table 5.3: Results for MNIST non-IID (the symbol ≻ denotes the priority relation). The row DS provides the number of rounds of communication needed to make the percentage of devices specified in the columns reach the specified target accuracy. The remaining rows show the gain in rounds of communication with respect to the baseline (the higher the better). The column Avg. shows the average gain for all the percentages of devices.

	Target accuracy 70%										**Target accuracy 80%**									
Experiment	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**
DS (baseline)	9	9	9	10	10	10	10	10	10		10	10	10	10	10	10	16	20	20	
LD	3	1	1	2	2	2	2	2	-1	1.56	2	2	2	-1	-3	-3	3	7	2	1.22
MW	4	4	4	5	0	0	0	0	0	1.89	0	0	0	0	-6	-6	0	1	1	-1.11
DS ≻ LD ≻ MW	2	2	2	2	2	2	2	1	-2	1.44	2	1	1	1	-3	-3	1	5	5	1.11
DS ≻ MW ≻ LD	0	0	0	0	0	0	0	0	-4	-0.44	1	0	0	-4	-7	-7	-1	3	2	-1.44
LD ≻ DS ≻ MW	2	2	1	2	2	2	2	2	2	1.89	2	2	2	-1	-1	-1	5	6	5	2.11
MW ≻ DS ≻ LD	1	0	0	0	0	0	0	-1	-1	-0.11	0	0	-1	-1	-1	-1	2	2	2	0.22
LD ≻ MW ≻ DS	0	0	0	1	1	-2	-2	-2	-2	-0.67	1	-2	-2	-2	-3	-5	1	0	0	-1.33
MW ≻ LD ≻ DS	1	1	1	2	1	1	1	-3	-3	0.22	1	1	-3	-3	-3	-5	0	4	3	-0.56

	Target accuracy 90%										**Target accuracy 95%**									
Experiment	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**Avg.**
DS (baseline)	10	16	20	20	23	33	35	44	69		20	23	34	49	66	84	106	137	222	
LD	-1	3	2	2	-2	2	-4	-5	8	0.56	2	-2	-7	-3	3	-5	-1	-11	-23	-5.22
MW	-9	-3	1	-7	-15	-8	-11	-13	-2	-7.44	1	-15	-7	-13	-10	-15	-23	-11	-14	-11.89
DS ≻ LD ≻ MW	-3	1	5	3	4	5	-4	1	4	1.78	5	-3	-5	0	-11	-13	-5	-19	-11	-6.89
DS ≻ MW ≻ LD	-7	-1	2	-5	-7	-6	-9	-12	9	-4.00	2	-7	-5	-5	-14	-14	-6	-16	12	-5.89
LD ≻ DS ≻ MW	2	5	5	-1	-6	0	-3	-4	1	-0.11	5	-6	1	-2	-4	1	12	-2	3	0.89
MW ≻ DS ≻ LD	-1	5	2	-3	-2	3	-4	-4	9	0.56	2	0	3	0	4	0	2	10	-6	1.67
LD ≻ MW ≻ DS	-3	-4	0	-4	-3	1	-1	-4	-1	-2.11	0	-1	-1	-10	-3	-2	-5	5	-6	-2.56
MW ≻ LD ≻ DS	-3	0	3	1	-1	4	3	7	14	3.11	4	3	2	1	0	9	-9	-5	37	4.67

Table 5.4: Confusion matrices for MNIST non-IID with the number of samples misclassified (✗) or correctly classified (✓) by DS and our experiments.

		LD		**MW**		**DS≻LD≻MW**		**DS≻MW≻LD**		**LD≻DS≻MW**		**MW≻DS≻LD**		**LD≻MW≻DS**		**MW≻LD≻DS**	
		✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
DS	✗	1	130	2	129	1	130	2	129	1	130	0	131	2	129	0	131
DS	✓	146	9723	142	9727	119	9750	127	9742	103	9766	104	9765	124	9745	119	9750

Table 5.5: Results for CelebA (the symbol ≻ denotes the priority relation). The row DS provides the number of rounds of communication needed to make the percentage of devices specified in the columns reach the specified target accuracy. The remaining rows show the gain in rounds of communication with respect to the baseline (the higher the better). The column Avg. shows the average gain for all the percentages of devices. Runs that did not reach the target accuracy for the specified percentage of devices in the 100 allowed rounds have been obtained considering such limit value.

	Target accuracy 70%									**Target accuracy 80%**								**Target accuracy 85%**					
Experiment	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**Avg.**	**10%**	**20%**	**30%**	**40%**	**50%**	**Avg.**
DS (baseline)	1	1	1	13	15	21	32	58		1	1	15	21	26	44	—		1	15	21	29	50	
IS	0	0	-2	4	-2	-1	-59	-42	-12.75	0	-2	0	4	-4	-56	—	-9.67	-1	2	4	-1	-50	-9.20
CB	0	0	0	7	9	15	18	20	8.62	0	0	9	15	17	28	—	11.50	0	9	15	15	26	13.00
DS ≻ IS ≻ CB	0	0	0	-15	-13	-7	-12	1	-5.75	0	0	-13	-7	-11	-3	—	-5.67	0	-13	-7	-15	0	-7.00
DS ≻ CB ≻ IS	0	0	0	12	9	15	22	34	11.50	0	0	10	15	18	32	43	16.86	0	10	15	20	30	15.00
IS ≻ DS ≻ CB	0	0	-2	6	6	5	9	-42	-2.25	0	-2	6	10	3	-56	—	-6.50	0	6	8	6	-50	-6.00
CB ≻ DS ≻ IS	0	0	-2	10	6	13	19	18	8.00	0	-2	12	16	14	21	—	10.17	0	12	16	16	26	14.00
IS ≻ CB ≻ DS	0	0	0	9	6	12	-68	-42	-10.38	0	-3	9	12	-21	-56	—	-9.83	0	9	12	-35	-50	-12.80
CB ≻ IS ≻ DS	0	0	-2	0	6	-18	-20	-42	-9.50	0	-2	0	2	-16	-28	—	-7.33	0	0	-10	-13	-33	-11.20

Table 5.6: Confusion matrices for CelebA with the number of samples misclassified (✗) or correctly classified (✓) by DS and our experiments.

		IS		**CB**		**DS≻IS≻CB**		**DS≻CB≻IS**		**IS≻DS≻CB**		**CB≻DS≻IS**		**IS≻CB≻DS**		**CB≻IS≻DS**	
		✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
DS	✗	312	2081	610	1783	275	2118	296	2097	331	2062	618	1775	416	1977	612	1781
DS	✓	2080	14846	3197	13729	1923	15003	2056	14870	2121	14805	3197	13729	2473	14453	3247	13679

5.6.1 *Effects of Individual Criteria*

In the following, we discuss the experiments by investigating the effects of the individual criteria.

MNIST with IID distribution. Table 5.1 and Figure 5.1a show the experimental results for MNIST considering IID distribution. The adoption of criteria that are different from the dataset size seems to have no effects, but some insignificant fluctuations. We expected this behavior since the dataset does not show statistical issues. In detail, the local datasets have been created by randomly picking samples from the original dataset. Consequently, we may represent the divergence of the local models from the original model as a zero-mean Gaussian noise. The same assumption holds for the label diversity. Furthermore, by the design of the dataset, DS is equal for every client, and we also expect that, on average, LD is the same for all clients. As a consequence, MW can not play a significant role, since all clients train the model with the same distribution of data, on average.

In detail, by looking at the differences between the two criteria LD and MW against the baseline DS, we may notice that the gain in terms of rounds of communication for 70%, 80%, and 90% target accuracy, considering all the percentages of clients, has an average very close to 0. Conversely, both LD and MW show slightly worse results than DS for a target accuracy of 95%, notably when we want to grant such accuracy to a very high number of devices (i.e., the last three columns of Table 5.1).

Lastly, the experiment shows that the effectiveness of criteria depends on the data distribution. The dataset shows how datasets built by randomly picking data from the original distribution do not generally need further statistical adjustments during the training phase. Therefore, we do not need to consider, in our approach, any particular statistical-based criteria. On the other hand, it is worth mentioning that the scenario is completely unrealistic in a federated scenario. Indeed, since each user privately generates data, they will not show the same distribution, and they will be significantly different in terms of statistical properties.

MNIST with Non-IID Distribution. Regarding DS, LD, and MW, Table 5.3 shows that DS is the best criterion to reach the best overall system performance in a lower number of rounds. In detail, DS seems to be in trouble in the first rounds to achieve an acceptable degree of accuracy. In fact, if we look at 70% and 80% of the accuracy, LD and

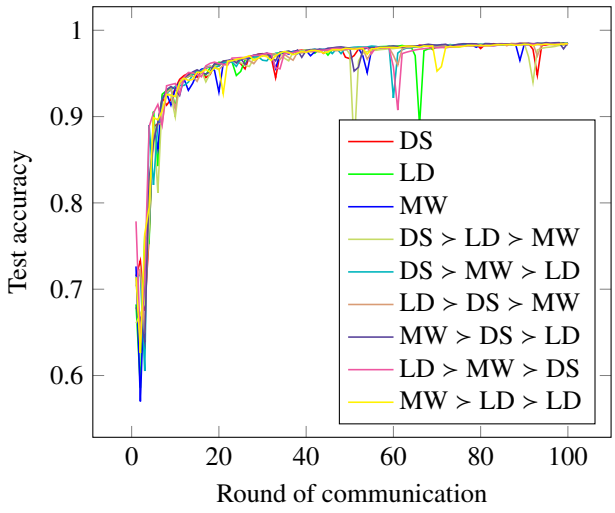
MW reach those goals faster. In this sense, we may notice an average of 1-2 rounds gained by LD and MW. However, if we need a higher degree of accuracy (90% and 95%), DS generally reaches that accuracy from 5 to 12 rounds before, on average. In this respect, we may notice that, without considering the last three columns of 95%, DS and LD show approximately the same performance, with an advantage of 0.5 rounds on average for LD. On the other hand, in the same scenario, MW is 3.6 rounds slower on average. Nonetheless, the remaining three last columns show an average advantage of DS of 12 and 16 rounds for LD and MW, respectively.

CelebA. Results in Table 5.5 show that the three criteria behave in a significantly different way. In this experiment, we note that CB performs much better than DS. To get an impression, we may notice CB reaches 85% of the overall accuracy for the 60% of the devices in less than half of the rounds of DS. Moreover, if we observe the CB row in Table 5.5, we may note that it reaches the same accuracy goals of DS much faster. In detail, CB reaches those goals 7.5 rounds before DS. On the other side, IS is generally slower than DS. Indeed, it generally needs 7.6 rounds more than DS. Another interesting insight is that the advantage of CB over DS progressively increases. In detail, CB shows an advantage of 8.6 rounds for the 70% of accuracy, 11.5 rounds for 80%, and 13 rounds for an accuracy of 85%.

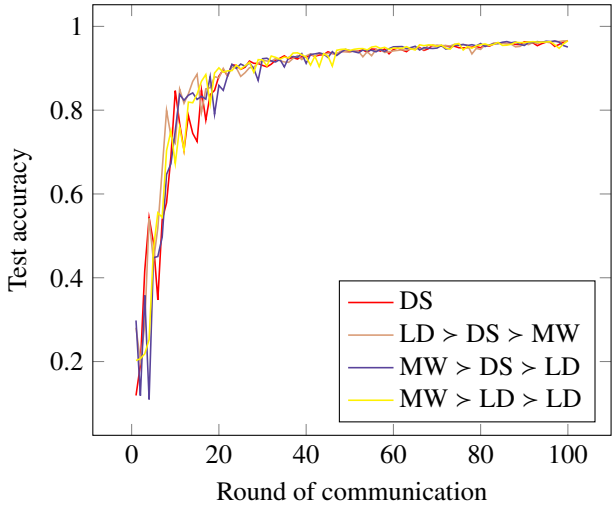
5.6.2 *Effects of Combined Criteria*

In this section, we focus on the effects of the combination of the different criteria. Eventually, in Section 5.6.3, we discuss the different findings of the experiments, and we draw some general remarks.

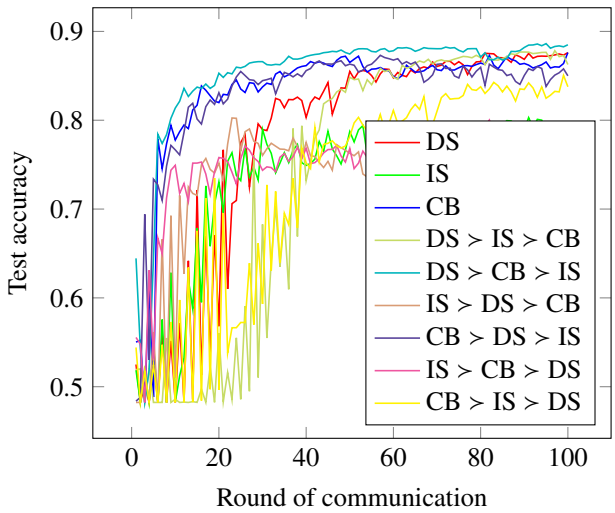
MNIST with IID distribution. In the lower section, Table 5.1 shows the effects of the six priority permutations of the criteria. We may suppose that, since the individual criteria did not give any boost to the training, also their combination should have no effect. As an example, in some situations — and this is very clear in LD $\succ$ MW $\succ$ DS — we can observe a boost for lower target value of accuracy (70%, 80%, and 90%) and a slowdown in reaching the target accuracy of 95% for a high number of devices. Analogously to the individual criteria, we observe that the average gain is quite close to 0, for each target accuracy. The section 95% is an exception, since its last three columns show a slowdown up to 9 rounds of communication, on average. This



(a) IID distribution



(b) Non-IID distribution



(c) CelebA

Figure 5.1: Test accuracy results during training for each round of communication.

behavior is probably due to the adverse effects we already observed for the individual criteria in this dataset.

We can also observe the same behavior in Figure 5.1a, where the curves of the combinations of criteria — and the ones of the individual criteria — are virtually indistinguishable.

MNIST with Non-IID distribution. Once we combine different criteria, we may observe several different behaviors. If we focus on Table 5.3, we can observe two different trends. First, if we analyze the data in the table moving from the left to the right, we may observe an increase of negative gains in the sections related to 90% and 95% of the accuracy. Even here, if we focus on the last three columns, we may notice an average delay that can reach up to 12 rounds. However, even in this case, there are some combinations that achieve an advantage over DS: LD $\succ$ DS $\succ$ MW, MW $\succ$ DS $\succ$ LD, and MW $\succ$ LD $\succ$ DS. LD $\succ$ DS $\succ$ MW shows an interesting trend since it reaches 70%, 80%, 90%, 95% of accuracy 2, 2, 0, 1 rounds sooner than DS, respectively. Here, the advantage is higher for the lower accuracy thresholds, while it is almost 1 for 95% accuracy. Instead, MW $\succ$ DS $\succ$ LD shows a better and incremental trend. Indeed, in this case, the advantage of rounds is -0.1, 0.2, 2.1, and 1.7, respectively. Finally, MW $\succ$ LD $\succ$ DS shows a bad advantage average for 70%, and 80% (0.2, and -0.6), while for 90%, and 95% it shows an interesting performance boost by reaching the goals 3.1, and 4.7 rounds before DS.

CelebA. Results in Table 5.5 show results which are coherent with the effects of the individual criteria. Rows DS $\succ$ IS $\succ$ CB, as well as IS $\succ$ DS $\succ$ CB, IS $\succ$ CB $\succ$ DS, and CB $\succ$ IS $\succ$ DS, denote the negative influence of the IS criterion. In particular, they show a little boost in very few cases for a low percentages of devices, but their averages generally show a slowdown from 2 up to 13 rounds of communication. On the other side, we may observe the effects of BC, which is beneficial even when combined with the other criteria. Even then, this effect is more evident when IS has the last priority (i.e., by minimizing its influence). In this case, we observe a substantial speed-up against the sole DS criterion, and also when compared with the CB individual criterion. In detail, permutation DS $\succ$ CB $\succ$ IS shows the best results for all target values of accuracy and all values of coverage for devices, up to 12 rounds of communications (against the sole DS criterion). Even Figure 5.1c clearly shows these behaviors. Specifically, we may observe the

cyan line (DS $\succ$ CB $\succ$ IS) against the red line (DS only) and the blue line (CB only).

5.6.3 *Discussion*

The analysis in the previous section already shows the fundamental role of the identification of local criteria. By considering the underlying dataset, the different criteria heavily affect the training phase outcome.

As an example, when we create a federated dataset by randomly picking samples from an original dataset (i.e., with an IID distribution), the adoption of statistical criteria is not beneficial, since data, on local devices, shows the same expectation.

On the other hand, even in MNIST non-IID distributed dataset, it is quite disappointing that we can not appreciate the benefits of introducing new criteria like label diversity. However, if we observe the dataset deeper, it contains local shards of the same size, and each client processes, at most, samples of two digits that make the number of classes in local datasets a piece of useless information.

Instead, the criterion on model divergence gives an initial boost to training, but it seems to have adverse effects for higher target values of accuracy. Even here, the behavior is interesting and predictable. Indeed, the penalization of significant divergences is the right choice because it helps to build a robust estimator on the average of the samples. However, at the end of the training, higher precision and fine training could require those differences.

As expected, when we observe the most realistic dataset, CelebA, we may appreciate more the effects of the proposed approach. In fact, it is a realistic non-IID distribution of images where each client holds samples which are different in number and number of classes. Consequently, the attribute about class balancing gives the best results either when considered individually and also combined with the dataset size attribute. This is probably due to the differences both in dataset size and class balancing among the local datasets. Conversely, Sharpness is a theoretically useful attribute, but it results in a slowdown of performance during the training phase. We believe that the reason for this behavior is twofold. First, only a few images are marked as blurry in the dataset. Second, a right combination of sharp and blurry images has widely proven to be beneficial for the generalization of a CNN.

Overall, we can positively answer RQ1, since *the idea of incorporating a set of criteria measuring some quality properties of the clients, and based on them assigning the contribution of individual update in*

the final model has been shown to be promising. In particular, answering RQ2, we can state that, *when dealing with non-IID distributions of data, where differences among clients are more evident, prioritizing criteria corresponding to higher differences is more effective. This leads to a higher skewness of the user contributions, based on how their local datasets can give interesting information about the highest priority criteria*. Then, the lesson learned is that when users of a federation have different levels of "knowledge" about an aspect of the ideal data distribution, giving way to the most "expert" users would be beneficial for the training of the federated model.

5.7 Conclusions and Future Perspectives

In this work, we have introduced a practical Federated Learning (FL) protocol that exploits non-sensitive client information to aggregate the local models. In detail, we have formalized the notion of a local criterion for clients in an FL scenario, and the notion of priority ranked criteria. By considering the ranking of the criteria, we have defined the score functions to weigh the contribution of a client. We have tested our approach on three well known federated learning datasets: IID-distributed MNIST, non-IID-distributed MNIST, and CelebA.

The experiments show that we can substantially improve the standard federated learning approach by exploiting a properly defined set of local criteria. We have observed how some criteria may be useful in some moments of the training, but they also may cause issues in others. In this respect, we propose a self-adaptive model that re-ranks the priority of criteria as partially investigated in [14]. Indeed, in those research fields, local model specialization and privacy are crucial, and this prioritized federated learning approach may be particularly beneficial.

There is still a broad spectrum of criteria that deserves to be explored, and the experiments suggest that that a criterion's efficacy highly depends on the specific dataset/domain. Even though several considerations may lead the researcher, it is not possible to find a unique criterion that meets the needs of all the possible domains. Nevertheless, the study of the individual criteria has revealed some information about their impact on the training phase, which we hope may be useful for other researchers.

The proposed approach is particularly effective when dealing with non-IID distributions of data. This is remarkable since a real federated scenario will show a non-IID distribution of data rather than an IID distribution, where differences among clients are not so perceptible. In

general, substantial differences among devices about a property make the corresponding criterion more effective. In practical terms, we have improved the federated approach by enhancing the individual differences in terms of "knowledge" and "expertise" about a distribution, respecting the true federated learning spirit.

We are particularly interested in this research direction and in the chance of leveraging this federated approach to other machine learning scenarios, like recommender systems [11, 19]. To this aim, we considered extending the idea of expertise based on a property to expertise based on a concept for realizing a knowledge-aware federated recommender system, presented in the next section.

6 Federated Learning for Personalization in Recommender Systems

Outline

Collaborative filtering models have undoubtedly dominated the scene of recommender systems in recent years. However, despite their outstanding performance, these methods require a training time proportional to the size of the embeddings and it further increases when also side information is considered for the computation of the recommendation list. In fact, in these cases we have that with a large number of high-quality features, the resulting models are more complex and difficult to train. This chapter presents KGFlex, a sparse factorization approach with an high power of expressiveness. To achieve this result, KGFlex analyzes the historical data to understand the dimensions the user decisions depend on (e.g., movie direction, musical genre, nationality of book writer) and models user-item interactions as a an entropy-driven combination of embeddings of the item attributes relevant to the user.

Following the findings of Chapter 5, we apply a principle of expertise, facilitating the training process by letting users update only those relevant features on which they base their decisions. Moreover, the user-item prediction is mediated by a user's personal view that considers only relevant features. An extensive experimental evaluation shows the approach's effectiveness, considering the recommendation results' accuracy, diversity, and induced bias.

The preliminary idea of this work has been first presented in the extended abstract "Sparse Embeddings for Recommender Systems with Knowledge Graphs" at the 11th Italian Information Retrieval Workshop. Then, the content of this chapter has been accepted and presented at the 15th ACM Conference on Recommender Systems with the full paper "*Sparse Feature Factorization for Recommender Systems with Knowledge Graphs*".

6.1 Introduction

The history of automated recommendation is closely linked to the evolution of collaborative filtering techniques. Their notable accuracy has unquestionably helped Recommender Systems getting famous. Despite their leading performance, these methods are based on the simple idea to recommend certain items since "similar users have experienced those items", or "other users, who have experienced the same items, have also experienced those items." In the past, matrix factorization [121] and nearest-neighbors were the main algorithms to implement collaborative filtering and, over the last years, Deep Learning models [58] have joined this shortlist. The main limitation of these approaches is the requirement of many parameters that further increase at least proportionally according to the dataset size.

Differently from collaborative approaches, content-based recommendation techniques aim to identify the common characteristics of items that a user liked in the past [158]. They match the user profile against the attributes of the items and recommend new items that share the same features. On the one hand, the use of content features can make the model interpretable [229] while, on the other hand, these techniques suffer from overspecialization since they fail to recommend items that are different from the items enjoyed in the past. In order to get the benefits of the two approaches and mitigate their drawbacks, researchers worked to integrate into collaborative filtering the side information used in content-based approaches such as tags [236], images [17], demographic data [231], structured knowledge [24]. However, even there, the predominant adoption of large and dense models implies that user-item interactions are predicted by taking into account hundreds or thousands of features.

In this work, we introduce KGFlex, a knowledge-aware federated recommendation system, that tackles this issue by exploiting a sparse embedding model with an even greater degree of expressiveness. KGFlex extracts facts and knowledge from publicly available knowledge graphs to describe the catalog items. Then, low-dimensionality embeddings are adopted to represent the semantic item features. KGFlex models the user-item interaction by combining the subset of item features relevant to the user. Moreover, to compute the recommendations, it analyses the user-specific decision-making process of consuming or not consuming an item. According to that process, the system weights feature embeddings using an entropy-based strategy. Therefore, in KGFlex, each user computes a set of features her decisions are based on. According to a

principle of expertise, during training, only the features the specific user is expert about are updated for a given user-item pair. Hence, the user profile itself only contains a personal representation of each relevant feature.

The research questions that guide our work are the following.

RQ1. Is KGFlex able to provide both accurate and diverse recommendation to users?

RQ2. Is KGFlex robust to popularity bias and disinclined to introduce algorithmic bias?

RQ3. What happens if KGFlex is deprived of high-order features? Which are the effects on accuracy and diversity?

RQ4. Do KGFlex recommendations preserve the semantics included in the original features?

To answer to the above research questions, we conduct extensive experiments on three different publicly available datasets. The content-based features have been extracted from data encoded in the DBpedia[1] knowledge graph, thanks to public mappings from the dataset items to DBpedia URIs[2]. We evaluate the accuracy and diversity of recommendation results and analyze whether the algorithm produces biased recommendations. Finally, we study how users' decision-making process differs from KGFlex's one by graphically showing the semantic shift produced in the recommendation. The results show that KGFlex has competitive accuracy performance, and at the same time, generates highly diversified recommendations with a low induced bias.

6.2 Related Work

6.2.1 *Knowledge-Aware Recommender Systems*

Nowadays, modern RSs exploit various side information such as metadata (e.g., tags, reviews) [154], social connections [31], images [17], and users-items contextual data [11] to build more in-domain [92] (i.e., domain-dependent), cross-domain [82], or context-aware [101, 105] recommendation models. Among the diverse information sources, what is, likely, the most relevant source is Knowledge Graphs ($\mathcal{KG}s$). Thanks to the heterogeneous domains that $\mathcal{KG}s$ cover, the design of knowledge-based recommendation systems has arisen as a specific research field of its own in the community of RSs, usually referred to by Knowledge-aware Recommender Systems (KaRS [10, 18]). The

1 http://dbpedia.org
2 https://github.com/sisinflab/LinkedDatasets

adoption of $\mathcal{KG}s$ as a source of side-information has generated several advancements in the tasks of recommendation [24], knowledge completion [98], preference elicitation [21], user modeling [201], and thus produced a vast literature. In recent years, the Knowledge-aware Recommender Systems have been particularly impactful for several recommendation tasks: **hybrid collaborative/content-based recommendation** [24, 128], exploiting the $\mathcal{KG}$ information to suffice the lack of collaborative information and to improve the performance; **knowledge-transfer, cross-domain recommendation** [82, 115, 227], where the $\mathcal{KG}s$ allow to find semantic similarities between different domains; **interpretable/explainable-recommendation** [12, 20, 24, 209, 220], with $\mathcal{KG}$ being a backbone for understanding the recommendation model and providing human-like explanations; **user-modeling** [111, 139, 155, 201], since the resource descriptions can drive the construction of the user profile; **graph-based recommendation** [73, 179, 181, 200, 202, 207], where the topology-based techniques have met the semantics of the edges/relations, and the ontological classification of nodes (classes); **the cold-start problem** [82, 152, 176, 214], since the $\mathcal{KG}s$ can overcome the lack of collaborative information; **the content-based recommendation** [22, 72] that solely relies on $\mathcal{KG}$ and still produces high-quality recommendations. KGFlex could be considered a Knowledge-aware hybrid collaborative/content-based recommendation model. While recent models of the same kind made use of Knowledge graph embeddings or factorization models, KGFlex considerably differs from them since it introduces the sparse factorization approach and reweights the user-feature interactions by exploiting the information gain signal. To the best of our knowledge, it is one of the first approaches to adopt this hybrid solution to obtain a personalized view of the embedding matrix.

6.2.2 *Entropy-Driven Recommender Systems*

Entropy-based measures have been widely employed in recommendation systems. A popular strategy to include entropy into the recommendation algorithm is to exploit it in connection with a similarity measure. In this respect, Wang et al. [208] proposed a new information entropy-driven user similarity-based model. They suggest measuring the relative difference between ratings and develop a Manhattan distance-based model. Yalcin et al. [217] proposes two novel aggregation techniques by hybridizing additive utilitarian and approval voting methods to feature popular items on which group members provided a consensus.

They use entropy to analyze rating distributions and detect items on which group members have reached no or little consensus. Entropy has also been used to model the purchase probability for a given set of recommendations for a specific user [106]. The idea is to exploit the maximum entropy principle by analyzing features in the recommendations and user interests. Another example of the exploitation of entropy is Lee [126], where they improve the previous similarity measures by employing the information entropy of user ratings to reflect the user's global rating behavior on items. Karimi et al. [112] proposed an innovative approach for active learning in recommender systems, aiming to take advantage of additional information. They suggest employing entropy to drive the active learning process and increase the system performance for new users. Entropy has also been applied to evaluate the quality and helpfulness of different product reviews [228]. They propose an information gain-based model to predict the helpfulness of online product reviews to suggest the most suitable products and vendors to consumers. Another interesting study [45] integrates entropy more deeply into the recommendation process. They calculate for every feature its information gain by considering item instances that provide the feature and item instances that do not. Despite superficial similarities with Bouza et al. [45], the two works are fundamentally different. In fact, Bouza et al. uses the class of the features with the highest information gain as a decision tree node, while KGFlex exploits the information gain to weigh the single user-feature interactions. Moreover, user and feature embeddings are combined using a dot-product similarity, showing some similarities with the former works. However, that is where the similarities end, since all the mentioned works propose completely different models from KGFlex (e.g., distance-based or active-learning models, feature selection techniques).

6.3 Background

The Semantic Web was initially conceived to connect documents in the Web and improve data retrieving and access. Over the years, a full stack of semantic technologies emerged, leading to the Linking Open Data initiative [100]. The initiative indicates the remarkable effort of a community of researchers and practitioners to build publicly available knowledge bases of semantically linked machine-understandable data [38]. The resulting knowledge graphs are built on a disarmingly simple idea, but they definitely changed the way structured information is stored. Thanks to the Linked Data initiative, today, we can bene-

fit from 1,483 different knowledge graphs connected in the so-called Linked Open Data Cloud[3]. These knowledge graphs share the same ontology and the same schema across multiple domains, giving access to a wide-spread knowledge at the same development cost required for a single domain.

The most appreciated $\mathcal{KG}s$ of this special class undoubtedly are DBpedia [29, 127], Wikidata [198, 199], Yago [190] (the 4th release [192] also supports RDF* [97]), FreeBase [41], Satori[45] [136, 195], NELL [55], Google's Knowledge Graph[6], Facebook's Entities Graph[7], Knowledge Vault [75], Bio2RDF [35]. This availability of knowledge graphs is a clear advantage for KaRS.

A knowledge graph $\mathcal{KG}$ can be represented as a set of triples where entities are linked to each other by binary relations. Each connection in $\mathcal{KG}$ is then a triple $\sigma \xrightarrow{\rho} \omega$, where σ is a subject entity, ρ is a relation (predicate), and ω is an object entity. Therefore, in $\mathcal{KG}$, the edge ρ connects the entity σ and the entity ω with a directed relation. Hereinafter, we generalize the previous notion to multi-hop predicates (i.e., considering chains of predicates that connect two entities at a higher depth). Let n-hop predicate be defined as $\rho = \langle \rho_1, ..., \rho_n \rangle$ if $\sigma \xrightarrow{\rho_1} \omega_1 \xrightarrow{\rho_2} ... \xrightarrow{\rho_n} \omega_n \in \mathcal{KG}$. For convenience, $h(\rho) = n$ for $\rho : \sigma \xrightarrow{\rho} \omega_n \in \mathcal{KG}$ denotes the depth of the predicate chain. When no confusion arises, from now on we will use $\sigma \xrightarrow{\rho} \omega$ to denote a generic chain with $h(\rho) \in \{1, ..., n\}$.

6.4 Approach

In the following, we introduce KGFlex. It exploits the knowledge encoded in a knowledge graph as side information to compute feature-aware user profiles, which are eventually used to provide personalized recommendation lists.

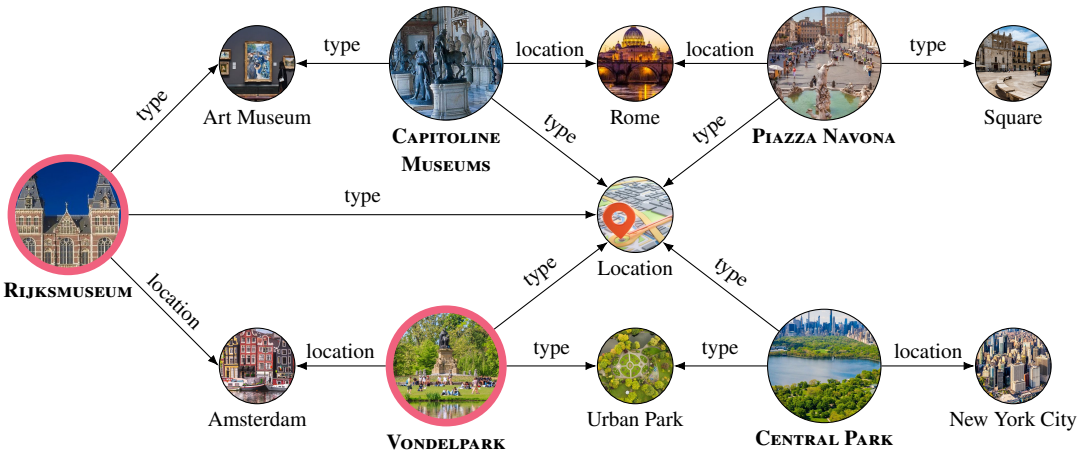


Figure 6.1: An excerpt of a knowledge graph, showing items of a catalog (Rijksmuseum, Vondelpark, Capitoline Museums, Piazza Navona, and Central Park) connected to other entities by predicates. The user Pink has expressed positive feedback for the items highlighted with the colors of their names.

6.4.1 *Item and User Features in KGFlex*

Given a collection of items $\mathcal{I}$ and a knowledge graph $\mathcal{KG}$ we assume each element in $i \in \mathcal{I}$ has a mapping to a corresponding entity in $\mathcal{KG}$. Under this assumption, a server S can explore $\mathcal{KG}$ at depth n starting from an item i to identify the set $\mathcal{F}_i^{(n)}$ of the semantic features describing it:

$$\mathcal{F}_i^{(n)} = \{\langle \rho, \omega \rangle \mid i \xrightarrow{\rho} \omega \in \mathcal{KG}\ , h(\rho) \in \{1, ..., n\}\}. \tag{6.1}$$

Once the features are extracted, KGFlex handles them equally, regardless of their original depth.

Example 6.1. As an example, consider the $\mathcal{KG}$ excerpt in Figure 6.1, where a 1-depth exploration has been performed for each of five items taken from a point-of-interest catalog. The formal *Vondelpark* item description is:

$$\mathcal{F}^{(1)}_{Vondelpark} = \{\langle type,\ Location\rangle,$$
$$\langle location,\ Amsterdam\rangle, \langle type,\ Urban\ Park\rangle\}.$$ □

3 https://lod-cloud.net/datasets
4 https://searchengineland.com/library/bing/bing-satori
5 https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing
6 https://blog.google/products/search/introducing-knowledge-graph-things-not/
7 https://www.facebook.com/notes/facebookengineering/under-the-hood-the-entitiesgraph/10151490531588920/

We consider a federation $\mathcal{U}$ of users, with each user u holding a set $\mathcal{F}_u$ of features representing the items $\mathcal{I}_u^+ \subseteq \mathcal{I}$ she positively enjoyed. We define $\mathcal{F}_u$ as the set of features describing the items that user u has interacted with:

$$\mathcal{F}_u^{(n)} = \bigcup_{i \in \mathcal{I}_u^+} \mathcal{F}_i^{(n)}. \tag{6.2}$$

Example 6.2. ***(continued)*** In Figure 6.1, we represented the items enjoyed by the user Pink by marking them with pink circles. To build the set $\mathcal{F}_{Pink}^{(1)}$, the features of all the items appreciated by Pink have to be considered:

$$\mathcal{F}_{Pink}^{(1)} = \{\langle type,\ Location\rangle, \langle location,\ Amsterdam\rangle, \\ \langle type,\ Urban\ Park\rangle, \langle type,\ Art\ Museum\rangle\}.$$ □

Finally, let $\mathcal{F}^{(n)}$ denote the overall set of the features in the system:

$$\mathcal{F}^{(n)} = \bigcup_{i \in \mathcal{I}} \mathcal{F}_i^{(n)}, \tag{6.3}$$

with $\mathcal{F}_i^{(n)} \subseteq \mathcal{F}^{(n)}$ and $\mathcal{F}_u^{(n)} \subseteq \mathcal{F}^{(n)}$. Depending on the value of n and on the size of $\mathcal{I}$, the size of $\mathcal{F}^{(n)}$ could rapidly increase. Thus, filtering the item features might be a reasonable choice to control the computational and memory load and to improve the system performance. Even though the literature about feature selection is vast, it is worth noticing that with KGFlex also graph pruning and semantic feature selection techniques [71] could apply. In the following, for convenience, the (n) superscript is omitted whenever it is not relevant in the context.

6.4.2 *Entropy of User Features*

The main assumption behind KGFlex is that users make decisions (i.e., items to enjoy) based on a subset of item characteristics. The assumption implies that not all the item features are equally important.

With KGFlex we move a step ahead in this direction by exploring how likely a user considers a feature in her item choice process. Taking a cue from information theory, KGFlex exploits the notion of information gain to measure the relevance of a feature for a user in the process of deciding to consume or not the item. For completeness, information gain is not the only metric used to select the best variable to partition data samples with respect to an outcome variable [174]. Nevertheless, information gain is widely adopted in a myriad of methods since it

works also with non-binary values of each attribute. Moreover, for what regards the decision trees, the various informative metrics are quite consistent with each other and choosing one or another has a limited impact on the performance [162]. For specific reasons, some metrics could be preferred (e.g., Gini impurity is well suited for its low computational cost [162]). However, a discussion about the advantages of the different metrics remains beyond the scope of this work. In information theory, entropy is used to measure the uncertainty of a random variable. The entropy $H(V)$ of a random variable V with k possible values in $\{v_1, ..., v_k\}$ is defined as:

$$H(V) = -\sum_{i=1}^{k} P(V = v_i) \log_2 P(V = v_i). \tag{6.4}$$

It is straightforward to check that a coin that always comes up heads has *zero* entropy, while a fair coin equally likely to come up heads or tails when flipped has entropy 1. Notably, if V is a binary random variable that is true with probability q, we have $H(V) = B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$. Therefore, given a dataset $\mathcal{D}$ of training samples in the form $(\mathbf{x}, y)$, with $\mathbf{x} \in \mathbb{R}^F$ and $y \in \{0, 1\}$, the entropy of the dataset is equal to $H(\mathcal{D}) = B(P(y = 1))$.

In this context, the information gain measures the expected reduction in information entropy from a prior state to a new state that acquires some information. With reference to the dataset $\mathcal{D}$, the new information comes from the observation of one of the attributes x_d in $\mathbf{x}$. The k distinct values $\{x_{d,1}, ...x_{d,k}\}$ that x_d can assume partition the dataset $\mathcal{D}$ into k mutually exclusive subsets, thus inducing a categorical probability distribution on the values of x_d. This gives the possibility to measure the expected entropy of $\mathcal{D}$ conditioned on x_d:

$$H(\mathcal{D}|x_d) = \sum_{i=1}^{k} P(x_d = x_{d,i}) H(\mathcal{D}|x_d = x_{d,i}). \tag{6.5}$$

Then, we define the information gain $IG(\mathcal{D}, x_d)$ obtained from the observation of the attribute x_d as:

$$IG(\mathcal{D}, x_d) = H(\mathcal{D}) - H(\mathcal{D}|x_d). \tag{6.6}$$

The information gain defined in Eq. (6.6) returns a measure of the importance of a single attribute in distinguishing positive from negative examples in a dataset. In KGFlex, we use the notion of information gain to measure how relevant a feature is to a user for deciding to consume or discard an item. In detail, to associate each feature of the system

with an information gain, KGFlex uses the workflow described in the following.

Each user u builds a dataset $\mathcal{D}_u$ with all the positive items (i.e., the items the user has enjoyed) from $\mathcal{I}_u^+$ and the same amount of negative items randomly picked up from $\bigcup_{v \in \mathcal{U}, v \neq u} \mathcal{I}_v \setminus \mathcal{I}_u^+$ (i.e., items not enjoyed by the user u but enjoyed by other users). If this information is not available, due to privacy concerns, also popular items represent a good option. Then, following Eq. (6.4), $H(\mathcal{D}_u) = 1$. Each sample is provided with a set of binary variables corresponding to the features in $\mathcal{F}_u$. Each variable indicates, for each item i in $\mathcal{D}_u$, the presence ($f = 1$) or the absence ($f = 0$) of the corresponding feature in the set $\mathcal{F}_i$.

The information gain for each feature $f \in \mathcal{F}_u$ can be computed using the dataset $\mathcal{D}_u$. Let p_{uf} be the number of positive samples in $\mathcal{D}_u$ for which $f = 1$, n_{uf} the number of negative samples for which the same feature is present, and t_{uf} the short form of $p_{uf} + n_{uf}$. Analogously, we define $p_{u\neg f} = |\mathcal{I}_u^+| - p_{uf}$ as the number of positive samples with $f = 0$, $n_{u\neg f} = |\mathcal{I}_u^+| - n_{uf}$ as the number of negative samples with $f = 0$, and $t_{u\neg f}$ as the short form of $p_{u\neg f} + n_{u\neg f}$. Following Eqs. (6.5) and (6.6):

$$IG(\mathcal{D}_u, f) = 1 - H(\mathcal{D}_u | f = 1) - H(\mathcal{D}_u | f = 0), \tag{6.7}$$

$$H(\mathcal{D}_u | f = 1) = \frac{t_{uf}}{|\mathcal{D}_u|} \left(- \frac{p_{uf}}{t_{uf}} \log_2 \frac{p_{uf}}{t_{uf}} - \frac{n_{uf}}{t_{uf}} \log_2 \frac{n_{uf}}{t_{uf}} \right), \tag{6.8}$$

$$H(\mathcal{D}_u | f = 0) = \frac{t_{u\neg f}}{|\mathcal{D}_u|} \left(- \frac{p_{u\neg f}}{t_{u\neg f}} \log_2 \frac{p_{u\neg f}}{t_{u\neg f}} - \frac{n_{u\neg f}}{t_{u\neg f}} \log_2 \frac{n_{u\neg f}}{t_{u\neg f}} \right), \tag{6.9}$$

where $IG(\mathcal{D}_u, f)$ can be also merely regarded as a function in the values of p_{uf} and n_{uf}.

In KGFlex, we associate a weight $k_{uf} = IG(\mathcal{D}_u, f)$ to each pair of user u and feature f. It represents the influence of a feature —in the view of the user— in the prediction of user-item interactions. To compute the k_{uf} values, the system designer can consider the whole set of features in $\mathcal{F}_u$, or filter them out according to a cutoff value of IG. This user-feature weight is built on a strongly positive or negative interest of the user towards the feature, that suggests an aptitude of the user for declaring something positive or negative about that feature and describing it. This is the rationale for extending the notion of expertise discussed in Chapter 5 to the idea of expertise about a concept (i.e., the feature), that will help KGFlex, by means of a calibrated rating prediction, to learn each part of the federated model from the most interested and expert users (see Section 6.4.3 for further details).

Example 6.3. ***(continued)*** To clarify the use of information gain in KGFlex and show its effects, we consider the example in Figure 6.1. We see that Pink has visited the Rijksmuseum and the Vondelpark, both in Amsterdam. Thus KGFlex supposes she has a preference for the Dutch city. On the other hand, all the items in the catalog share the feature $\langle type, Location \rangle$, thus KGFlex assumes this latter not to be influential in the user decision-making process. To build $\mathcal{D}_{Pink}$, KGFlex combines the set of items experienced by Pink with a set of the same size containing items that Pink did not enjoy, e.g., $\mathcal{D}_{Pink} = \{Rijksmusem, Vondelpark, Piazza\ Navona, Central\ Park\}$. Let us observe the feature $\langle location, Amsterdam \rangle$. According to the previous definitions, it has to be influent for Pink. Given $\mathcal{D}_{Pink}$, KGFlex computes:

$$p_{Pink,\langle location,Amsterdam\rangle} = 2, \quad n_{Pink,\langle location,Amsterdam\rangle} = 0,$$
$$p_{Pink,\neg\langle location,Amsterdam\rangle} = 0, \quad n_{Pink,\neg\langle location,Amsterdam\rangle} = 2.$$

Consequently, according to Eq. (6.7), $k_{Pink,\langle location,Amsterdam\rangle} = 1$, meaning that Pink strongly takes into account if a place to visit is located in Amsterdam or not. Therefore, KGFlex considers this feature to have a high impact on generating the recommendations for Pink. Moreover, it could be observed that $k_{Pink,\langle type,Art\ Museum\rangle} \approx 0.31$. Since it is not completely clear how influential this feature is in Pink's decisions, it will have a smaller influence on the predictions for Pink. Finally, $k_{Pink,\langle type,Location\rangle}$ and $k_{Pink,\langle type,Urban\ Park\rangle}$ have zero information gain and no influence on the predictions. In detail, the former is common to all the items and does not bring additional information. The latter is shared by the same number (i.e., one) of positive and negative samples in $\mathcal{D}_{Pink}$. So, it makes this feature useless in distinguishing positive from negative places for Pink. □

6.4.3 *Model Architecture*

KGFlex does not contain explicit representations for users and items. Instead, it represents the features in $\mathcal{F}$ as embeddings in a latent space $\mathbb{R}^m$. This representation is hosted on the server S and is shared among all the users of the federation. Moreover, KGFlex promotes the idea of having user fine-tuned versions of the same model [14–16]. Therefore, in addition to the global latent representation of each feature in $\mathcal{F}$, it builds, on the client of each user u, a personal view of each feature in $\mathcal{F}_u \subseteq \mathcal{F}$. Each user combines the global embeddings with her personal feature representation to estimate the overall user-item interaction.

Formally, the recommendation model is structured into two distinct parts, both containing a dense representation of dimensionality m for each feature $f \in \mathcal{F}$. On the one hand, KGFlex keeps on the server S a set $\mathcal{G}$ of global trainable embeddings and biases shared among all the users (see Section 6.4.4):

$$\mathcal{G} = \{(\mathbf{g}_f \in \mathbb{R}^m, b_f \in \mathbb{R}) \mid f \in \mathcal{F}\}. \tag{6.10}$$

On the other hand, each user in $\mathcal{U}$ holds her personal representation of the features she interacted with, i.e., the features in $\mathcal{F}_u$. These embeddings are collected within the set $\mathcal{P}^u$, defined as:

$$\mathcal{P}^u = \{\mathbf{p}^u_f \in \mathbb{R}^m \mid f \in \mathcal{F}_u\}. \tag{6.11}$$

KGFlex estimates the possible affinity of a feature f to the user u with the inner product between the personal representation $\mathbf{p}^u_f$ and the global representation $\mathbf{g}_f$, plus the global bias term b_f. To estimate the overall affinity of user u to item i, KGFlex combines the features shared by u and i and then weighs them according to their pre-computed entropy-based values. In detail, being $\mathcal{F}_{ui} = \mathcal{F}_u \cap \mathcal{F}_i$ the set of common features between user u and item i, the interaction $\hat{r}_{ui}$ can be estimated as it follows:

$$\hat{r}_{ui} = \sum_{f \in \mathcal{F}_{ui}} k_{uf}(\mathbf{p}^u_f \mathbf{g}_f + b_f). \tag{6.12}$$

Eq. (6.12) encodes the strategy KGFlex exploits to handle thousands of model features. In fact, it takes advantage of the user profile to involve only a small subset of them in the estimate of the user-item affinity.

Example 6.4. ***(continued)*** From the previous analysis, it is clear how Pink's choices (see Figure 6.1) are influenced by the features $\langle$*location*, *Amsterdam*$\rangle$ and $\langle$*type*, *Art Museum*$\rangle$. Consider that the interaction of Pink with *Capitoline Museum* is to be estimated by KGFlex. The set of the common features is $\mathcal{F}_{Pink,Capitoline\ Museum} = \{\langle type, Art\ Museum\rangle, \langle type, Location\rangle\}$. Accordingly to Eq. (6.12), the interaction is modelled as the summation of contributions of the common features:

$$\begin{aligned}\hat{r}_{Pink,Capitoline\ Museum} = {} & k_{Pink,\langle type,Art\ Museum\rangle} \\ & \cdot (\mathbf{p}^{Pink}_{\langle type,Art\ Museum\rangle} \mathbf{g}_{\langle type,Art\ Museum\rangle} + b_{\langle type,Art\ Museum\rangle}) \\ & + k_{Pink,\langle type,Location\rangle} \\ & \cdot (\mathbf{p}^{Pink}_{\langle type,Location\rangle} \mathbf{g}_{\langle type,Location\rangle} + b_{\langle type,Location\rangle}).\end{aligned}$$

As expected, there is no contribution of the feature $\langle$*type*, *Location*$\rangle$ because of the value of its pre-computed entropy-based weight. Thus, the only contribution to the estimation is given by the embeddings of $\langle$*type*, *Art Museum*$\rangle$. □

In this chapter, we did not center our attention on the federated architecture, rather on the idea behind KGFlex. Nevertheless, the recommendation model described so far has been designed to fit a federated architecture, with a training procedure based on the protocol described in Section 4.4 that does not require sharing any raw data. However, the same KGFlex model can be perfectly integrated into a centralized setting without requiring any significant change.

6.4.4 *Learning to Rank*

To learn the model parameters, KGFlex adopts the well-known Bayesian Personalized Ranking (BPR) optimization criterion, which is a maximum posterior estimator for personalized ranking and the most common pair-wise Learning to Rank strategy. BPR assumes that a user u prefers a consumed item i^+ over a non-consumed item i^-, and optimizes the model by maximizing, for each pair of i^+ and i^-, a function of the difference $r_{ui^+} - r_{ui^-}$. To update the model, KGFlex uses stochastic gradient descent, following Eqs. 4.3 and 4.4, and considering that:

$$\frac{\partial}{\partial\theta}\hat{r}_{ui^+i^-} = \frac{\partial}{\partial\theta}\hat{r}_{ui^+} - \frac{\partial}{\partial\theta}\hat{r}_{ui^-}, \tag{6.13}$$

and that here the partial derivatives of the generic $\hat{r}_{ui}$ with respect to the model parameters are:

$$\frac{\partial}{\partial\theta}\hat{r}_{ui} = \begin{cases} k_{uf}\mathbf{g}_f & \text{if } \theta = \mathbf{p}^u_f, \\ k_{uf}\mathbf{p}^u_f & \text{if } \theta = \mathbf{g}_f, \\ k_{uf} & \text{if } \theta = b_f, \\ 0 & \text{else.} \end{cases} \tag{6.14}$$

6.5 Experimental Setup

This section describes how we have designed the experimental setting, the evaluation protocol, and the baselines to answer the research questions.

6.5.1 *Datasets*

We have evaluated the performance of KGFlex on three datasets from different domains, namely *Yahoo! Movies*, *MovieLens*, and *Facebook*

Books. Each item in these datasets is provided with a *DBpedia* URI, which links to the semantic description of the item as an entity of the *DBpedia* knowledge graph. *Yahoo! Movies* contains 69, 846 movie ratings generated on *Yahoo! Movies* up to November 2003 on a $[1, 5]$ scale. The ratings have been collected from 4, 000 users with respect to 2, 626 items. It provides mappings to *MovieLens* and *EachMovie* datasets. We binarize the explicit data by keeping ratings of 3 or higher and interpret them as positive implicit feedback. The dataset *MovieLens* is a collection of users' ratings in the movie domain: it contains 1, 000, 209 ratings on a $[1, 5]$ scale from 6, 040 users with respect to 3, 706 items. Similar to *Yahoo! Movies*, we binarize the explicit data by keeping ratings of 3 or higher. Finally, *Facebook Books* is a more sparse dataset with 18, 978 positive implicit feedback from 1, 398 users about 2, 933 books. To ensure a fair comparison with the baselines, we applied an iterative 10-core preprocessing on *Yahoo! Movies* and *MovieLens*, and a 5-core preprocessing on *Facebook Books*.

6.5.2 *Feature Extraction*

For a fair comparison, we have used the features resulting from the following workflow for KGFlex and for the baselines that make use of content information, i.e., VSM and kaHFM.

Exploration of Knowledge Graph. The items of the datasets have been described with a set of semantic features retrieved through a knowledge graph exploration at depth 2 in the form of $\langle \rho, \omega \rangle$ pairs (see Eq. 6.1). The semantic information has been retrieved from the *DBpedia* knowledge graph[8], thanks to the item-to-*DBpedia*-URI mapping provided with the datasets. Some features (based on their 1-hop predicate) have not been considered, since they provide auxiliar information not useful for characterizing the content of the item [71]. In detail, we filtered out the predicates *dbo:wikiPageWikiLink*, *owl:sameAs*, *rdf:type*, *gold:hypernym*, *rdfs:seeAlso*, *dbp:wordnet_type*, *dbo:wikiPageExternalLink*, *dbo:thumbnail*, *prov:wasDerivedFrom*, and *dbp:wikiPageUsesTemplate*.

Feature Filtering based on Frequency. Irrelevant features have been removed due to the poor information they bring and to reduce the computational costs. Thus, we have removed the features that are common to less than 10 items. The resulting features constitute the

8 `https://www.dbpedia.org`

common set of features for all the competing models that make use of content features, although some of these models —including KGFlex— may operate some further filtering operations.

Feature Filtering based on Entropy in KGFlex. As mentioned in Section 6.4.2, features in the user personal representation have been filtered as well based on their information gain. For instance, in a 2-hop exploration of the knowledge graph, users with a large number of transactions could reach an impressive number of nodes of the knowledge graph. Thus, we filtered out features with little information gain keeping, as a maximum limit for each user, the 100 most informative features from the 1-hop exploration and the 100 most informative features from the 2-hop exploration.

6.5.3 *Baselines*

To assess the effectiveness of KGFlex, we compare it with various baselines. In particular, we are interested in comparing KGFlex with other latent factor models and other state-of-the-art baselines which helps to position the model with respect to some of the best recommendation approaches in the literature.

Non-competing Algorithms. Random and Most Popular are two non-personalized recommenders used for reference. Among content-based algorithms, we have chosen the Vector Space Model (VSM) [72] where user and item profiles have been generated with TF-IDF and cosine similarities between them have been computed. As collaborative-filtering baselines, we have considered Item-kNN [119] (an item-based implementation of the k-nearest neighbors algorithm) and MultiVAE [132], a non-linear probabilistic model taking advantage of Bayesian inference to estimate the parameters.

Competing Algorithms. We compare KGFlex against factorization-based algorithms, both non-neural and neural. Among them, we consider i) BPR-MF [167], a latent factor model based on the same pairwise optimization criterion used in KGFlex, ii) the neural Matrix Factorization in the version of Rendle et al. [169], iii) NeuMF [99], and iv) kaHFM [24], another factorization-based model making use of knowledge graphs for model building and initialization.

6.5.4 *Reproducibility, Evaluation Protocol and Metrics*

We have chosen the *all unrated items* protocol to compare the different algorithms. In *all unrated items*, for each user we consider as recommendable items all the items not yet rated by that user. We have split the datasets using hold-out 80-20 splitting strategy, retaining for each user the 80% of her ratings in the training set and the remaining 20% in the test set [91]. All the models have been tested in 10 different configurations of hyperparameters, according to the Bayesian hyperparameter optimization algorithm. For the sake of reproducibility, we provide our code and a working configuration file for the framework Elliot [13], with complete and ready-to-use information about the experiments we have run. We have measured the recommendation accuracy by exploiting nDCG [124]. It has been also used for validation and choosing the best hyperparameter configurations. We have also evaluated the diversity of recommendation, adopting Item Coverage [6] and Gini Index [56] (higher is better). The former provides the overall number of diverse recommended items, and it highlights the degree of personalization. The latter measures how unequally a system provides users with different items, with higher values corresponding to more tailored lists. Finally, three bias metrics have been used to evaluate how KGFlex and the baselines behave on the underrepresentation of items from the long-tail. To this aim we have used ACLT (higher is better), which measures the fraction of the long-tail items the recommender has covered [3]. Moreover, we have also evaluated PopREO and PopRSP (smaller is better, in $[0, 1]$), which are specific applications of RSP and REO [237]. Notably, PopREO estimates the equal opportunity of items, encouraging the true positive rate of popular and unpopular items to be same. PopRSP is a measure of statistical parity, assessing whether the ranking probability distributions for popular and unpopular items are the same in recommendation.

6.6 Results and Discussion

In the following, we discuss the main insights coming from the performed experiments, with the aim of answering the research questions posed at the beginning of this chapter.

Table 6.1: Comparison of KGFlex with competing baselines (names in boldface) and other reference baselines on Yahoo! Movies, Facebook Books, and MovieLens 1M. Among the competing baselines, the best result is in boldface, the second-best result is underlined. For the metrics marked with ↑, higher is better; with ↓, lower is better.

a) Yahoo! Movies

	nDCG ↑		**IC ↑**		**Gini Index ↑**		**ACLT ↑**		**PopREO ↓**		**PopRSP ↓**	
	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1
Random	0.0096	0.0084	1050	811	0.8496	0.5659	5.5203	0.5473	0.0985	0.5336	0.0098	0.0034
Most Pop.	0.1585	0.1367	49	11	0.0126	0.0010	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
VSM	0.0478	0.0453	370	93	0.0525	0.0139	3.1177	0.2270	0.4959	0.5443	0.4575	0.6112
Item-kNN	0.3074	0.3472	745	297	0.1583	0.0985	0.9884	0.0862	0.7059	0.7065	0.8347	0.8562
MultiVAE	0.2370	0.2455	399	152	0.0914	0.0419	0.2347	0.0122	0.8543	0.8293	0.9613	0.9799
BPR-MF	0.1857	0.1710	151	35	0.0219	0.0041	0.0006	0.0000	0.9954	1.0000	0.9999	1.0000
MF	0.2897	0.2999	455	177	0.0902	0.0464	0.0823	0.0026	0.8735	0.9353	0.9865	0.9958
NeuMF	0.0918	0.0855	50	13	0.0113	0.0009	0.0006	0.0000	1.0000	1.0000	0.9999	1.0000
kaHFM	**0.3006**	**0.3238**	757	290	0.1659	0.0988	0.4624	0.0334	0.7610	0.7494	0.9234	0.9447
KGFlex	0.2464	0.3122	**851**	**370**	**0.2802**	**0.1361**	**2.1447**	**0.1145**	**0.4477**	**0.6292**	**0.6336**	**0.8080**

b) Facebook Books

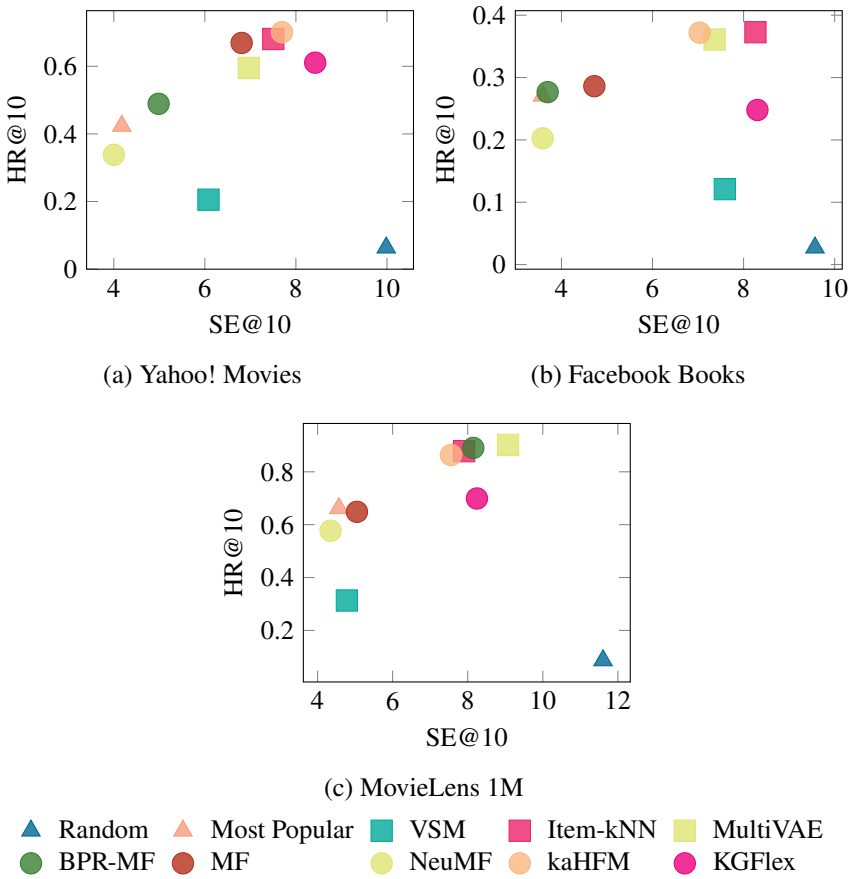
	nDCG ↑		**IC ↑**		**Gini Index ↑**		**ACLT ↑**		**PopREO ↓**		**PopRSP ↓**	
	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1
Random	0.0069	0.0059	782	646	0.8617	0.5878	5.2605	0.5371	0.0980	0.1159	0.0075	0.0088
Most Pop.	0.0939	0.0829	16	4	0.0127	0.0007	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
VSM	0.0362	0.0213	523	203	0.1887	0.0839	3.8056	0.3067	0.2262	0.7618	0.2996	0.4409
Item-kNN	0.1290	0.0924	769	338	0.3752	0.1606	2.2252	0.2106	0.4885	0.4300	0.5986	0.6208
MultiVAE	0.1191	0.0829	620	197	0.1828	0.0634	0.4637	0.0272	0.7770	0.8723	0.9182	0.9522
BPR-MF	0.0947	0.0829	17	4	0.0132	0.0007	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
MF	0.0956	0.0844	87	16	0.0238	0.0012	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
NeuMF	0.0714	0.0756	17	4	0.0125	0.0006	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
kaHFM	**0.1267**	**0.0858**	540	174	0.1387	0.0607	0.3294	0.0249	0.8766	0.9368	0.9420	0.9561
KGFlex	0.0853	0.0653	**606**	**288**	**0.3070**	**0.1588**	**3.0264**	**0.2458**	**0.1521**	**0.1315**	**0.4485**	**0.5554**

c) MovieLens 1M

	nDCG ↑		**IC ↑**		**Gini Index ↑**		**ACLT ↑**		**PopREO ↓**		**PopRSP ↓**	
	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1
Random	0.0096	0.0089	3203	2701	0.8657	0.6036	6.5735	0.6566	0.0715	0.0404	0.0056	0.0020
Most Pop.	0.1984	0.2512	69	19	0.0054	0.0006	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
VSM	0.0476	0.0373	170	34	0.0062	0.0011	1.5333	0.3629	0.8300	0.3413	0.8264	0.5395
Item-kNN	0.3690	0.4803	980	395	0.0584	0.0318	0.0666	0.0041	0.9606	0.9707	0.9930	0.9957
MultiVAE	0.3424	0.4003	1794	856	0.1381	0.0921	0.4455	0.0414	0.7652	0.7605	0.9522	0.9557
BPR-MF	**0.3676**	**0.4679**	1141	**527**	0.0765	**0.0449**	0.0633	0.0018	0.9551	0.9933	0.9933	0.9981
MF	0.1833	0.2079	100	35	0.0075	0.0029	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
NeuMF	0.1374	0.1496	70	18	0.0046	0.0004	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
kaHFM	0.3222	0.4203	955	337	0.0482	0.0219	0.0450	0.0017	0.9720	0.9888	0.9953	0.9983
KGFlex	0.1982	0.2754	1403	492	**0.0844**	0.0327	**0.8613**	**0.0480**	**0.8207**	**0.9384**	**0.9057**	**0.9484**

6.6.1 *Accuracy and Diversity: an Analytical and Qualitative Study*

The first analysis aims to answer RQ1. In fact, the purpose of this evaluation is to assess whether KGFlex is capable to provide accurate and diverse recommendation. Tables 6.1a, 6.1b and 6.1c show an analysis of the accuracy and diversity performance comparing KGFlex with the other baselines in terms of nDCG and Gini Index. The best and the second-best values are highlighted with **boldface** and underline, respectively. Although several baselines are considered (to position the methods), a thorough comparison is mainly made with respect to the latent factor models, highlighted in boldface in the first column of tables. The results in Tables 6.1a, 6.1b and 6.1c have been statistically validated with Student Paired t-test and Wilcoxon test, with a p-value level of 0.05. The complete significance hypothesis test tables are available in the KGFlex repository. The general behavior that can be observed at first glance from Table 6.1a is that KGFlex exhibits a satisfactory performance regarding the accuracy, being outperformed only by kaHFM and MF in the top-10 recommendation. Moreover, it behaves even better in the top-1 task, where its nDCG value becomes comparable to the accuracy result of kaHFM, which shows the best accuracy performance. KGFlex significantly outperforms BPR-MF, although both are learned with a pair-wise BPR optimization, suggesting the useful role of the extracted knowledge. When looking at the diversity performance represented by the item coverage and Gini values, we note the high degree of personalization provided by KGFlex. We link this result to the personalized view of the knowledge granted by the framework. Moreover, in KGFlex the collaborative signal on explicit user interests ensures to recommend diverse items among the ones sharing characteristics of interest for the user. The behavior pointed out so far is not entirely confirmed in Facebook Books (see Table 6.1b). Indeed, here the accuracy results remain below the performance of other factorization-based approaches. However, the diversity results show how BPR-MF, MF and NeuMF may have been completely flooded by popularity signal, which led them to perform poorly regarding the item coverage and Gini metrics. Instead, KGFlex approaches the superior performance of ItemkNN in terms of diversity, and even here it shows an improvement in top-1 recommendation, where the gap is reduced. Furthermore, we analyze Table 6.1c, showing the performance on MovieLens 1M. BPR-MF performs superbly on this dataset, while the other factorization-based approaches —but kaHFM— remain significantly below its capabilities. This downward trend seems to be caused by the strong popularity sig-



(a) Yahoo! Movies

(b) Facebook Books

(c) MovieLens 1M

Random Most Popular VSM Item-kNN MultiVAE
BPR-MF MF NeuMF kaHFM KGFlex

Figure 6.2: Accuracy vs. distributional diversity. The plots show the value of HR@10 against SE@10: the closer to the top-right corner the better.

nal, which is prevailing in MF and NeuMF. Instead, KGFlex does not suffer from this problem and it is the best model in terms of diversity, while providing still meaningful recommendations.

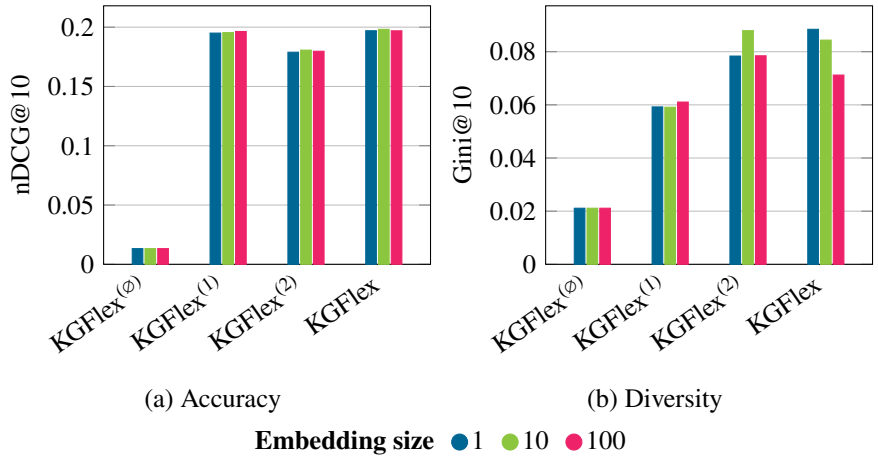
What we have analytically observed is confirmed in Figure 6.2. These graphs show the joint behavior of KGFlex on accuracy and distributional diversity, by analyzing the value of Hit Ratio (HR) on the top-10 recommendation lists with respect to the Shannon Entropy (SE) statistics. Among factorization-based approaches (labelled in the plots), KGFlex approaches the right-top margin to a greater extent. The kaHFM model usually is the second-best model, but, on MovieLens 1M, BPR-MF shows its best performance. The other approaches seem to perform very poorly in at least one dimension or do not have a stable position when varying the dataset. This confirms the previous findings, and *gives KGFlex the merit of providing highly personalized recommendations, thanks to the joint operation of the global and the personal (local) views of the same feature*.

6.6.2 *Induction and Amplification of the Bias*

The behavior of KGFlex led us to analyze the quality of the recommendation in terms of popularity bias, a frequent problem causing popular items to be more and more recommended and less popular ones to remain underrepresented [4]. This algorithmic bias may cause a fairness issue from the item point of view, but also an inappropriate recommendation for users who do not prefer very popular items. Tables 6.1a, 6.1b and 6.1c provide the values of three metrics to measure the bias. KGFlex always outperforms all the other factorization-based approaches and generally outperforms the other approaches. Regarding KGFlex, the Average Coverage of Long-Tail items, measured by ACLT (the higher the better), is comparable with the value obtained by VSM. This result is supported by the values of PopREO and PopRSP (the smaller the better), which encourage the ranking probability distributions and the true positive rates of popular and less popular items to be the same. Indeed, KGFlex and VSM grant the less biased recommendations. Interestingly, while both exploit the same optimization criterion, we notice how KGFlex consistently improves BPR-MF, which is known to be vulnerable to imbalanced data and to produce biased recommendations [237]. To conclude, we can answer to RQ2 asserting that *the personalized representation of content information gives KGFlex the push to provide satisfactory and diverse recommendations without being negatively affected by popularity bias*.

6.6.3 *Impact of Knowledge Graph Exploration*

In the previous experimental setting (see Section 6.5.2), the personal user knowledge was represented by her 100 most informative 1-hop features and her 100 most informative 2-hop features. To give an intuition of how beneficial is the exploitation of these features in KGFlex, we performed an ablation study on MovieLens 1M in which we force KGFlex not to use features from the first hop exploration, or from the second hop exploration, or both. Figure 6.3 shows the accuracy and diversity performance of KGFlex, and its ablated versions KGFlex$^{(1)}$, using only features from the first hop, KGFlex$^{(2)}$, using only features from the second hop, and KGFlex$^{(\varnothing)}$, which eliminates both. Moreover, we also plot how each version performs based on the embedding size, to understand whether it can affect the model variants. As expected, KGFlex$^{(\varnothing)}$ performed in an unsatisfactory way on the recommendation task, since it cannot establish common content between users and items.



(a) Accuracy (b) Diversity

Embedding size 1 10 100

Figure 6.3: Ablation study on MovieLens 1M of KGFlex evaluated with respect of accuracy and diversity, which shows that features from the second hop significantly improves the model. KGFlex$^{(\varnothing)}$ does not use features from knowledge graph, KGFlex$^{(1)}$ only uses 1-hop features, KGFlex$^{(2)}$ only uses 2-hop features. The colors represent different embedding sizes used for training.

With 100 first-hop features per user (KGFlex$^{(1)}$), the system provides accurate recommendation, but its diversity performance remains low. In this configuration, changing the embedding size is not beneficial neither for diversity nor for accuracy. When exploiting only features from the second hop KGFlex$^{(2)}$ the situation changes: these features enable KGFlex to catch more information about the content of the items and their relation, with a beneficial effect for the diversity of the recommendation. The information carried by the second-hop features has more probability of embodying the actual reason why a user decides to enjoy an item (e.g., a user may watch a TV show not strictly for the director himself but rather for his nationality). Finally, with the addition of the first-hop features, the complete version of KGFlex overcomes KGFlex$^{(2)}$ in accuracy, regardless of the embedding size. This latter slightly penalizes the diversity, very likely due to increased awareness regarding item popularity. This aspect requires further investigation and suggests room to increase the KGFlex performance further. The study definitely answers RQ3. As expected, *the lack of a piece of knowledge negatively impacts the system. However, interestingly, the combination of first- and second-hop features positively impacts on accuracy and diversity performance of KGFlex, with second-hop features often helping to discover the hidden attributes that are actually significant in the users' decision process.*

6.6.4 *Preservation of the Original Semantics*

(a) Original dataset

(b) Recommended lists

Figure 6.4: Two word clouds showing the 100 most informative features for the users in Yahoo! Movies on the original dataset and on the recommendation lists. The word clouds suggest that KGFlex is able to preserve the original semantics included in the dataset.

KGFlex makes extensive use of side information and takes advantage of it in various training phases. Nevertheless, there are no guarantees that KGFlex is really able to catch the semantic information encoded and use it to propose coherent recommendations. To investigate this aspect, we have analyzed the features characterizing the recommended items and we have compared those features with the features derived from the user's historical interactions. We have conducted a graphical experiment, showing the information gain of each feature in the system before and after the recommendation. Figure 6.4 depicts the word clouds of the features that could be involved in user decision-making

Table 6.2: Percentage of preserved top-k user features after the recommendation. The first (Q_1), the second (Q_2), and the third (Q_3) quartile (over all the population of users) of per-user percentages are shown.

	$k = \infty$	$k = 100$	$k = 50$	$k = 10$	$k = 5$
Q_1	14.5%	11%	12%	20%	20%
Q_2	21.5%	21%	26%	30%	40%
Q_3	31.5%	37%	48%	60%	60%

gathered from all the users in Yahoo! Movies. In detail, Figure 6.4a represents the prominence of each feature, in terms of information gain, in the original dataset. Instead, in Figure 6.4b, the same analysis is performed on the recommendation lists provided by KGFlex. For the sake of readability, both the word clouds visualize the 100 most informative features. We observe that topics related to science fiction persist at the top, including the interest for the director Steven Spielberg and for John Williams (also by means of his spouse Barbara Ruick) and the interest in fantasy films award-winner movies. More precisely, 79% of the top 100 informative features from the dataset are associated with items in the recommendation list. Additionally, for each user, we have computed the percentage of her k most informative features that have been retained in her recommendation list, with $k \in \{5, 10, 50, 100, \infty\}$. Table 6.2 shows, for each column, the first, the second (median), and the third quartile of such percentages over all the population of users. It is remarkable how KGFlex provides a higher coverage of the original semantic when considering features more important to the users (i.e., lower k). Finally, with the support of Figure 6.4 and Table 6.2, we answer to RQ4. *Overall, it could be easily observed that KGFlex preserves the main users' interests involved in decision-making. This suggests that KGFlex is able to preserve the original semantics, deeply integrating the content-based information into its recommendations. Finally, this evidence suggests that we could be a step closer to providing users with items that they would have chosen autonomously.*

6.6.5 *Limitations of KGFlex*

Section 6.5.1 explicitly refers to datasets linking to a knowledge graph such as *DBpedia*. Nonetheless, the approach behind KGFlex works independently of the type of side information. In fact, a dataset with

each item linked to a generic set of features or attributes would suffice KGFlex. However, the quality of the side information may have a profound impact on the final performance of the method. Moreover, if the side information is structured as a graph, also the depth of the exploration may impact the performance. In this work, we exploited knowledge graphs as side information since they are recognized to provide high-quality information. Nevertheless, their unavailability does not preclude the use of KGFlex.

6.7 Conclusion and Future Perspectives

This paper has introduced KGFlex for producing knowledge-aware recommendations from implicit feedback. KGFlex takes the best from content-based and factorization-based recommendation approaches for building a sparse model, where features extracted from a knowledge graph are embedded in a latent space. The interactions between users and items in KGFlex are combinations of users' feature representations and global feature representations, weighted according to the importance of each feature. KGFlex showed its superior behavior in terms of item diversity on three datasets while being very accurate and resilient to algorithmic bias. We have also shown the role a knowledge graph may play in feeding KGFlex with side information and how the extracted features preserve their semantics in the recommendation lists. This new method seems to be highly flexible and suited to practical applications. As future work, we plan to extend the approach with a finer feature selection, new types of feedback, alternatives to information gain, other types of side information, and other losses. Finally, we will further investigate feature embeddings to achieve an even more precise representation of features.

Part III

CONCLUSION

In other words, an admission of where I have actually come to. Everybody can say you should go faster or go slower. But you have arrived, with your own legs. Conclusion is my "*Look back and ahead: what you have seen and what you see?*".

7 Closing Remarks

This dissertation started with an analysis of the four most mentioned topics of our work: *recommendation*, *data*, *privacy*, and *federated learning*. We ended up synthesizing all of them and adding another concept: *personalization*. The bunch of problems that motivated this three-year work have given us a chance not only to study and propose some potential solutions but also to unveil new opportunities.

In Chapter 4, we have first introduced a pair-wise recommender system giving users the possibility to exercise the right of control on their data. Federated learning turned out to be the right architectural choice to build such a model, also thanks to its wide range of parameters that often allow discovering new room for improving the performance. Our focus has been mainly on the property and control of data according to the recent regulations. We have left the system designer completely free, eventually, to choose the most convenient privacy-preserving methods; that is why these algorithms have been thoroughly presented and discussed in the context of recommender systems in Chapter 3.

Secondly, in Chapter 5, motivated by a general machine learning problem, we have studied ways to make federated training more efficient. We have found that, with data coming from different distributions, identifying the criteria that represent the most relevant discriminating factors when solving the machine learning problem, and enhancing the contribution users with higher expertise on those criteria, can give a significant speed-up to the training of the federated model and an improvement to its performance.

This work, although distant in time, strongly motivated the development of KGFlex, presented in Chapter 6. We have guessed that the importance of an item attribute in deciding to accept or discard that item represent the strength of the knowledge of a user (i.e., her expertise) about that feature. Based on that, we have built a federated recommender system that takes the best from collaborative filtering and content-based filtering models, projects the item attributes in a latent space, and realizes a sparse combination of feature embeddings to efficiently compute a rating prediction that also takes into account the importance of each involved feature. Here, federated learning has

given us the opportunity to provide the users with local personalized embeddings accounting for their personal views on the same concepts, resulting in a model that, among the other advantages, stands out for the impressive personalization it grants to the users.

Not just data property and privacy, not just efficiency, not just personalization. We are pretty sure that there is still much to unveil and that we have just scratched the surface. All the future perspectives we have mentioned at the end of each chapter of Part II are just food for thought, but *recommendation*, *data*, *privacy*, *federated*, and *personalization* can be combined in plenty of ways between them and with a wider .

This is what we have tried to do, or rather I have tried to do with the help, the support, the collaboration, and the encouragement of my colleagues. I have attempted to contribute little to research, conscious that it is only a drop in the ocean. I am aware that all my efforts in studying, developing, experimenting, writing are never free of mistakes. Not for nothing, I should never have learned something new and improved my way if it had not been for rejections and failures. Yes, but also if it had not been for the desire to explore and discover that moves me and us towards great things that are even greater when our minds are set in a stunning peer-to-peer connection.

Bibliography

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep Learning with Differential Privacy." In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM, 2016, pp. 308–318. DOI: 10.1145/2976749.2978318.

[2] Chaabane Abdelberi, Gergely Ács, and Mohamed Ali Kâafar. "You are what you like! Information leakage through users' Interests." In: *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. "Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking." In: *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019*. Ed. by Roman Barták and Keith W. Brawner. AAAI Press, 2019, pp. 413–418.

[4] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. "The Unfairness of Popularity Bias in Recommendation." In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[5] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. "A Survey on Homomorphic Encryption Schemes: Theory and Implementation." In: *ACM Comput. Surv.* 51.4 (2018), 79:1–79:35.

[6] Gediminas Adomavicius and YoungOk Kwon. "Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques." In: *IEEE TKDE* 24.5 (2012), pp. 896–911.

[7] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." In: *IEEE Trans. Knowl. Data Eng.* 17.6 (2005), pp. 734–749. DOI: 10.1109/TKDE.2005.99.

[8] Gediminas Adomavicius and Jingjing Zhang. “Impact of data characteristics on recommender systems performance.” In: *ACM Trans. Management Inf. Syst.* 3.1 (2012), 3:1–3:17. DOI: 10.1145/2151163.2151166.

[9] Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. “Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System.” In: *CoRR* abs/1901.09888 (2019). arXiv: 1901.09888.

[10] Vito Walter Anelli, Pierpaolo Basile, Derek G. Bridge, Tommaso Di Noia, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Markus Zanker. “Knowledge-aware and conversational recommender systems.” In: *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. Ed. by Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O’Donovan. ACM, 2018, pp. 521–522. DOI: 10.1145/3240323.3240338.

[11] Vito Walter Anelli, Vito Bellini, Tommaso Di Noia, Wanda La Bruna, Paolo Tomeo, and Eugenio Di Sciascio. “An Analysis on Time- and Session-aware Diversification in Recommender Systems.” In: *UMAP*. ACM, 2017, pp. 270–274.

[12] Vito Walter Anelli, Vito Bellini, Tommaso Di Noia, and Eugenio Di Sciascio. “Knowledge-Aware Interpretable Recommender Systems.” In: *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. Ed. by Ilaria Tiddi, Freddy Lécué, and Pascal Hitzler. Vol. 47. Studies on the Semantic Web. IOS Press, 2020, pp. 101–124. DOI: 10.3233/SSW200014.

[13] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. “Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation.” In: *SIGIR*. ACM, 2021, pp. 2405–2414.

[14] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, and Antonio Ferrara. “Towards Effective Device-Aware Federated Learning.” In: *AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings*. Ed. by Mario Alviano, Gianluigi Greco, and Francesco Scarcello. Vol. 11946. Lecture Notes in Computer Science. Springer, 2019, pp. 477–491. DOI: 10.1007/978-3-030-35166-3_34.

[15] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "FedeRank: User Controlled Feedback with Federated Recommender Systems." In: *ECIR (1)*. Vol. 12656. Lecture Notes in Computer Science. Springer, 2021, pp. 32–47.

[16] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, and Fedelucio Narducci. "How to put users in control of their data in federated top-N recommendation with learning to rank." In: *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*. ACM, 2021, pp. 1359–1362. DOI: 10.1145/3412841.3442010.

[17] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. "A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images." In: *SIGIR*. ACM, 2021, pp. 1094–1103.

[18] Vito Walter Anelli and Tommaso Di Noia. "2nd Workshop on Knowledge-aware and Conversational Recommender Systems - KaRS." In: *CIKM*. ACM, 2019, pp. 3001–3002.

[19] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Local popularity and time in top-n recommendation." In: *European Conf. on Information Retrieval*. Springer. 2019, pp. 861–868.

[20] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Semantic Interpretation of Top-N Recommendations." In: *IEEE Transactions on Knowledge and Data Engineering* (2020), pp. 1–1. DOI: 10.1109/TKDE.2020.3010215.

[21] Vito Walter Anelli, Renato De Leone, Tommaso Di Noia, Thomas Lukasiewicz, and Jessica Rosati. "Combining RDF and SPARQL with CP-theories to reason about preferences in a Linked Data setting." In: *Semantic Web* 11.3 (2020), pp. 391–419. DOI: 10.3233/SW-180339.

[22] Vito Walter Anelli, Tommaso Di Noia, Pasquale Lops, and Eugenio Di Sciascio. "Feature Factorization for Top-N Recommendation: From Item Rating to Features Relevance." In: *RecSysKTL*. Vol. 1887. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 16–21.

[23] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. "On the discriminative power of hyper-parameters in cross-validation and how to choose them." In: *Proc. of the 13th ACM Conf. on Recommender Systems*. 2019, pp. 447–451.

[24] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "How to Make Latent Factors Interpretable by Feeding Factorization Machines with Knowledge Graphs." In: *ISWC (1)*. Vol. 11778. Lecture Notes in Computer Science. Springer, 2019, pp. 38–56.

[25] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Local Popularity and Time in top-N Recommendation." In: *European Conf. on Information Retrieval*. Vol. 11437. Springer, 2019, pp. 861–868.

[26] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. "Scalable and Secure Logistic Regression via Homomorphic Encryption." In: *CODASPY*. ACM, 2016, pp. 142–144.

[27] Apple. *Designing for privacy (video and slide deck)*. 2019. URL: https://developer.apple.com/videos/play/wwdc2019/708.

[28] Apple. *Private Federated Learning (NeurIPS 2019 Expo Talk Abstract)*. 2019. URL: https://nips.cc/ExpoConferences/2019/schedule?talk_id=40.

[29] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. "DBpedia: A Nucleus for a Web of Open Data." In: *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*. Vol. 4825. Lecture Notes in Computer Science. Springer, 2007, pp. 722–735. DOI: 10.1007/978-3-540-76298-0_52.

[30] Naveen Farag Awad and Mayuram S. Krishnan. "The Personalization Privacy Paradox: An Empirical Evaluation of Information Transparency and the Willingness to be Profiled Online for Personalization." In: *MIS Q.* 30.1 (2006), pp. 13–28.

[31] Lars Backstrom and Jure Leskovec. "Supervised random walks: predicting and recommending links in social networks." In: *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*. 2011, pp. 635–644.

[32] Ricardo Baeza-Yates. "Bias in Search and Recommender Systems." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. 2020, p. 2. DOI: 10.1145/3383313.3418435.

[33] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. "How To Backdoor Federated Learning." In: *CoRR* abs/1807.00459 (2018). arXiv: 1807.00459.

[34] Borja Balle and Yu-Xiang Wang. "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising." In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 403–412.

[35] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. "Bio2RDF: Towards a mashup to build bioinformatics knowledge systems." In: *J. Biomed. Informatics* 41.5 (2008), pp. 706–716. DOI: 10.1016/j.jbi.2008.03.004.

[36] Alejandro Bellogín, Pablo Castells, and Iván Cantador. "Statistical biases in Information Retrieval metrics for recommender systems." In: *Inf. Retr. J.* 20.6 (2017), pp. 606–634.

[37] James Bennett, Stan Lanning, et al. "The netflix prize." In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York, NY, USA. 2007, p. 35.

[38] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web." In: *Scientific American* 284.5 (May 2001), pp. 34–43.

[39] Alper Bilge, Cihan Kaleli, Ibrahim Yakut, Ihsan Gunes, and Huseyin Polat. "A Survey of Privacy-Preserving Collaborative Filtering Schemes." In: *Int. Journal of Software Engineering and Knowledge Eng* 23.8 (2013), pp. 1085–1108.

[40] Dheeraj kumar Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. "Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey." In: *CoRR* abs/1503.07475 (2015). arXiv: 1503.07475.

[41] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge." In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. Ed. by Jason Tsong-Li Wang. ACM, 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746.

[42] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. "Practical Secure Aggregation for Privacy-Preserving Machine Learning." In: *CCS*. ACM, 2017, pp. 1175–1191.

[43] Kallista A. Bonawitz et al. "Towards Federated Learning at Scale: System Design." In: *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. 2019.

[44] Rodrigo Borges and Kostas Stefanidis. “On mitigating popularity bias in recommendations via variational autoencoders.” In: *SAC*. ACM, 2021, pp. 1383–1389.

[45] Amancio Bouza, Gerald Reif, Abraham Bernstein, and Harald C. Gall. “SemTree: Ontology-Based Decision Tree Algorithm for Recommender Systems.” In: *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008*. Ed. by Christian Bizer and Anupam Joshi. Vol. 401. CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[46] Engin Bozdag. “Bias in Algorithmic Filtering and Personalization.” In: *Ethics and Inf. Technol.* 15.3 (Sept. 2013), pp. 209–227. ISSN: 1388-1957. DOI: 10.1007/s10676-013-9321-6.

[47] Zvika Brakerski and Vinod Vaikuntanathan. “Efficient Fully Homomorphic Encryption from (Standard) LWE.” In: *FOCS*. IEEE Computer Society, 2011, pp. 97–106.

[48] Robin Burke, Nasim Sonboli, Masoud Mansoury, and Aldo Ordoñez-Gauger. “Balanced neighborhoods for fairness-aware collaborative recommendation.” In: (2017).

[49] Carole Cadwalladr and Emma Graham-Harrison. “Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach.” In: *The guardian* 17 (2018), p. 22.

[50] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. “"You Might Also Like: " Privacy Risks of Collaborative Filtering.” In: *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*. IEEE Computer Society, 2011, pp. 231–246. DOI: 10.1109/SP.2011.40.

[51] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. “LEAF: A Benchmark for Federated Settings.” In: *arXiv preprint arXiv:1812.01097* (2018).

[52] California State Legislature. *The California Consumer Privacy Act of 2018*. 2018. URL: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

[53] Rocío Cañamares and Pablo Castells. “Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems.” In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. 2018, pp. 415–424. DOI: 10.1145/3209978.3210014.

[54] John F. Canny. "Collaborative Filtering with Privacy." In: *2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 12-15, 2002*. IEEE Computer Society, 2002, pp. 45–57.

[55] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. "Coupled semi-supervised learning for information extraction." In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. ACM, 2010, pp. 101–110. DOI: 10.1145/1718487.1718501.

[56] Pablo Castells, Neil J. Hurley, and Saul Vargas. "Novelty and Diversity in Recommender Systems." In: *Recommender Systems Handbook*. Springer, 2015, pp. 881–918.

[57] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. "Secure Federated Matrix Factorization." In: *IEEE Intell. Syst.* 36.5 (2021), pp. 11–20.

[58] Supriyo Chakraborty et al. "Interpretability of deep learning models: A survey of results." In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017, San Francisco, CA, USA, August 4-8, 2017*. IEEE, 2017, pp. 1–6. DOI: 10.1109/UIC-ATC.2017.8397411.

[59] Chaochao Chen, Ziqi Liu, Peilin Zhao, Jun Zhou, and Xiaolong Li. "Privacy Preserving Point-of-Interest Recommendation Using Decentralized Matrix Factorization." In: *Thirty-Second AAAI Conf. on Artificial Intelligence*. 2018, pp. 257–264.

[60] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. "Federated Meta-Learning for Recommendation." In: *CoRR* abs/1802.07876 (2018).

[61] Gustave Choquet. "Theory of capacities." en. In: *Annales de l'Institut Fourier* 5 (1954), pp. 131–295. DOI: 10.5802/aif.53.

[62] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on top-n recommendation tasks." In: *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. Ed. by Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker. ACM, 2010, pp. 39–46. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721.

[63] Célia da Costa Pereira, Mauro Dragoni, and Gabriella Pasi. "Multidimensional relevance: Prioritized aggregation in a personalized Information Retrieval setting." In: *Inf. Process. Manage.* 48.2 (2012), pp. 340–357. DOI: 10.1016/j.ipm.2011.07.001.

[64] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches." In: *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. Ed. by Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk. ACM, 2019, pp. 101–109. DOI: 10.1145/3298689.3347058.

[65] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption." In: *CRYPTO*. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 643–662.

[66] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. "Unique in the crowd: The privacy bounds of human mobility." In: *Scientific reports* 3.1 (2013), pp. 1–5.

[67] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogín, and Tommaso Di Noia. "A flexible framework for evaluating user and item fairness in recommender systems." In: *User Model. User Adapt. Interact.* 31.3 (2021), pp. 457–511. DOI: 10.1007/s11257-020-09285-1.

[68] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. "Recommender Systems Leveraging Multimedia Content." In: *ACM Comput. Surv.* 53.5 (2020), 106:1–106:38. DOI: 10.1145/3407190.

[69] Joaquin Delgado and Naohiro Ishii. "Memory-based weighted majority prediction." In: *SIGIR Workshop Recomm. Syst. Citeseer*. Citeseer. 1999, p. 85.

[70] Mukund Deshpande and George Karypis. "Item-based top-*N* recommendation algorithms." In: *ACM Trans. Inf. Syst.* 22.1 (2004), pp. 143–177.

[71] Tommaso Di Noia, Corrado Magarelli, Andrea Maurino, Matteo Palmonari, and Anisa Rula. "Using Ontology-Based Data Summarization to Develop Semantics-Aware Recommender Systems." In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Ed. by Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 128–144. DOI: 10.1007/978-3-319-93417-4_9.

[72] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. "Linked open data to support content-based recommender systems." In: *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*. Ed. by Valentina Presutti and Helena Sofia Pinto. ACM, 2012, pp. 1–8. DOI: 10.1145/2362499.2362501.

[73] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. "SPrank: Semantic Path-Based Ranking for Top-*N* Recommendations Using Linked Open Data." In: *ACM TIST* 8.1 (2016), 9:1–9:34. DOI: 10.1145/2899005.

[74] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption over the Integers." In: *EUROCRYPT*. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 24–43.

[75] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. "Knowledge vault: a web-scale approach to probabilistic knowledge fusion." In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani. ACM, 2014, pp. 601–610. DOI: 10.1145/2623330.2623623.

[76] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. "Fairness Through Awareness." In: *CoRR* abs/1104.3913 (2011). arXiv: 1104.3913.

[77] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy." In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), pp. 211–407. DOI: 10.1561/0400000042.

[78] Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, and Joseph A. Konstan. "User perception of differences in recommender algorithms." In: *RecSys*. ACM, 2014, pp. 161–168.

[79] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response." In: *CCS*. ACM, 2014, pp. 1054–1067.

[80] European Commission. *2018 reform of EU data protection rules*. 2018. URL: https://ec.europa.eu/info/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules/eu-data-protection-rules_en.

[81] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. “Personalized Federated Learning: A Meta-Learning Approach.” In: *CoRR* abs/2002.07948 (2020). arXiv: 2002.07948.

[82] Ignacio Fernández-Tobías, Iván Cantador, Paolo Tomeo, Vito Walter Anelli, and Tommaso Di Noia. “Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization.” In: *User Model. User Adapt. Interact.* 29.2 (2019), pp. 443–486. DOI: 10.1007/s11257-018-9217-6.

[83] Stephen E. Fienberg, William J. Fulp, Aleksandra B. Slavkovic, and Tracey A. Wrobel. “"Secure" Log-Linear and Logistic Regression Analysis of Distributed Databases.” In: *Privacy in Statistical Databases*. Vol. 4302. Lecture Notes in Computer Science. Springer, 2006, pp. 277–290.

[84] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. “Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation.” In: *IEEE Trans. Knowl. Data Eng.* 19.3 (2007), pp. 355–369.

[85] Arik Friedman, Shlomo Berkovsky, and Mohamed Ali Kâafar. “A differential privacy framework for matrix factorization recommender systems.” In: *User Model. User Adapt. Interact.* 26.5 (2016), pp. 425–458.

[86] Simon Funk. *Netflix Update: Try This at Home*. 2006. URL: http://sifter.org/simon/journal/20061211.html.

[87] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. “Semantics-Aware Content-Based Recommender Systems.” In: *Recommender Systems Handbook*. Springer, 2015, pp. 119–159.

[88] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. “CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy.” In: *ICML*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 201–210.

[89] Michel Grabisch. “The application of fuzzy integrals in multicriteria decision making.” In: *European Journal of Operational Research* 89.3 (1996), pp. 445–456. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(95)00176-X.

[90] Michel Grabisch and Marc Roubens. “Application of the Choquet integral in multicriteria decision making.” In: *Fuzzy Measures and Integrals* (2000), pp. 348–374.

[91] Asela Gunawardana and Guy Shani. “Evaluating Recommender Systems.” In: *Recommender Systems Handbook*. Springer, 2015, pp. 265–308.

[92] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. “A Survey on Knowledge Graph-Based Recommender Systems.” In: *IEEE Transactions on Knowledge and Data Engineering* (2020), pp. 1–1. DOI: 10.1109/TKDE.2020.3028705.

[93] Taolin Guo, Junzhou Luo, Kai Dong, and Ming Yang. “Differentially private graph-link analysis based social recommendation.” In: *Inf. Sci.* 463-464 (2018), pp. 214–226.

[94] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. “Federated Learning for Mobile Keyboard Prediction.” In: *CoRR* abs/1811.03604 (2018). arXiv: 1811.03604.

[95] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. “Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption.” In: *CoRR* abs/1711.10677 (2017).

[96] F Maxwell Harper and Joseph A Konstan. “The movielens datasets: History and context.” In: *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[97] Olaf Hartig. “Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF).” In: *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*. Ed. by Juan L. Reutter and Divesh Srivastava. Vol. 1912. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[98] Gaole He, Junyi Li, Wayne Xin Zhao, Peiju Liu, and Ji-Rong Wen. “Mining Implicit Entity Preference from User-Item Interaction Data for Knowledge Graph Completion via Adversarial Learning.” In: *WWW ’20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. Ed. by Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen. ACM, 2020, pp. 740–751. DOI: 10.1145/3366423.3380155.

[99] Xiangnan He and Tat-Seng Chua. “Neural Factorization Machines for Sparse Predictive Analytics.” In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen

P. de Vries, and Ryen W. White. ACM, 2017, pp. 355–364. DOI: 10.1145/3077136.3080777.

[100] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011. DOI: 10.2200/S00334ED1V01Y201102WBE001.

[101] Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp. "A Recommender System for Complex Real-World Applications with Nonlinear Dependencies and Knowledge Graph Context." In: *ESWC*. Vol. 11503. Lecture Notes in Computer Science. Springer, 2019, pp. 179–193.

[102] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets." In: *Proc. of the 8th IEEE Int. Conf. on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 2008, pp. 263–272.

[103] Jingyu Hua, Chang Xia, and Sheng Zhong. "Differentially Private Matrix Factorization." In: *IJCAI*. AAAI Press, 2015, pp. 1763–1770.

[104] Zan Huang, Hsinchun Chen, and Daniel Dajun Zeng. "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering." In: *ACM Trans. Inf. Syst.* 22.1 (2004), pp. 116–142.

[105] Yujia Huo, Derek F. Wong, Lionel M. Ni, Lidia S. Chao, and Jing Zhang. "Knowledge modeling via contextualized representations for LSTM-based personalized exercise recommendation." In: *Inf. Sci.* 523 (2020), pp. 266–278. DOI: 10.1016/j.ins.2020.03.014.

[106] Tomoharu Iwata, Kazumi Saito, and Takeshi Yamada. "Modeling user behavior in recommender systems based on maximum entropy." In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 1281–1282. DOI: 10.1145/1242572.1242808.

[107] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 422–446.

[108] Arjan Jeckmans, Qiang Tang, and Pieter H. Hartel. "Privacy-preserving collaborative filtering based on horizontally partitioned dataset." In: *CTS*. IEEE, 2012, pp. 439–446.

[109] Arjan J. P. Jeckmans, Michael Beye, Zekeriya Erkin, Pieter H. Hartel, Reginald L. Lagendijk, and Qiang Tang. "Privacy in Recommender Systems." In: *Social Media Retrieval*. Computer Communications and Networks. Springer, 2013, pp. 263–281.

[110] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning." In: *arXiv preprint arXiv:1912.04977* (2019).

[111] Surya Kallumadi and William H. Hsu. "Interactive Recommendations by Combining User-Item Preferences with Linked Open Data." In: *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*. Ed. by Tanja Mitrovic, Jie Zhang, Li Chen, and David Chin. ACM, 2018, pp. 121–125. DOI: 10.1145/3213586.3226222.

[112] Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. "A supervised active learning framework for recommender systems based on decision trees." In: *User Model. User Adapt. Interact.* 25.1 (2015), pp. 39–64. DOI: 10.1007/s11257-014-9153-z.

[113] Eugene Kharitonov. "Federated Online Learning to Rank with Evolution Strategies." In: *WSDM*. ACM, 2019, pp. 249–257.

[114] Jong Seon Kim, Jong Wook Kim, and Yon Dohn Chung. "Successive Point-of-Interest Recommendation With Local Differential Privacy." In: *IEEE Access* 9 (2021), pp. 66371–66386.

[115] Thomas Köllmer, Emanuel Berndl, Thomas Weißgerber, Patrick Aichroth, and Harald Kosch. "A Workflow for Cross Media Recommendations based on Linked Data Analysis." In: *Joint Proceedings of the 4th International Workshop on Linked Media and the 3rd Developers Hackshop co-located with the 13th Extended Semantic Web Conference ESWC 2016, Heraklion, Crete, Greece, May 30, 2016*. Ed. by Raphaël Troncy, Ruben Verborgh, Lyndon J. B. Nixon, Thomas Kurz, Kai Schlegel, and Miel Vander Sande. Vol. 1615. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[116] Jakub Konecný, Brendan McMahan, and Daniel Ramage. "Federated Optimization: Distributed Optimization Beyond the Datacenter." In: *CoRR* abs/1511.03575 (2015). arXiv: 1511.03575.

[117] Jakub Konecný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. "Federated Optimization: Distributed Machine Learning for On-Device Intelligence." In: *CoRR* abs/1610.02527 (2016). arXiv: 1610.02527.

[118] Yehuda Koren. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. 2008, pp. 426–434. DOI: 10.1145/1401890.1401944.

[119] Yehuda Koren. “Factor in the neighbors: Scalable and accurate collaborative filtering.” In: *TKDD* 4.1 (2010), 1:1–1:24.

[120] Yehuda Koren and Robert M. Bell. “Advances in Collaborative Filtering.” In: *Recommender Systems Handbook*. Springer, 2015, pp. 77–118.

[121] Yehuda Koren, Robert M. Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems.” In: *IEEE Computer* 42.8 (2009), pp. 30–37.

[122] Yehuda Koren and Joe Sill. “OrdRec: an ordinal model for predicting personalized item rating distributions.” In: *Proc. of the 2011 ACM Conf. on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. Ed. by Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius. ACM, 2011, pp. 117–124.

[123] Michal Kosinski, David Stillwell, and Thore Graepel. “Private traits and attributes are predictable from digital records of human behavior.” In: *Proceedings of the National Academy of Sciences* 110.15 (2013), pp. 5802–5805. ISSN: 0027-8424. DOI: 10.1073/pnas.1218772110. eprint: https://www.pnas.org/content/110/15/5802.full.pdf.

[124] Walid Krichene and Steffen Rendle. “On Sampled Metrics for Item Recommendation.” In: *KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 1748–1757. DOI: 10.1145/3394486.3403226.

[125] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[126] Soojung Lee. “Using entropy for similarity measures in collaborative filtering.” In: *J. Ambient Intell. Humaniz. Comput.* 11.1 (2020), pp. 363–374. DOI: 10.1007/s12652-019-01226-0.

[127] Jens Lehmann et al. “DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia.” In: *Semantic Web* 6.2 (2015), pp. 167–195. DOI: 10.3233/SW-140134.

[128] Jian Li, Zhuoming Xu, Yan Tang, Bo Zhao, and Haimei Tian. “Deep Hybrid Knowledge Graph Embedding for Top-N Recommendation.” In: *Web Information Systems and Applications - 17th International Conference, WISA 2020, Guangzhou, China, September 23-25, 2020, Proceedings*. Ed. by Guojun Wang, Xuemin Lin, James A. Hendler, Wei Song, Zhuoming Xu, and Genggeng Liu. Vol. 12432. Lecture

Notes in Computer Science. Springer, 2020, pp. 59–70. DOI: 10.1007/978-3-030-60029-7_6.

[129] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. "A survey on federated learning systems: vision, hype and reality for data privacy and protection." In: *arXiv preprint arXiv:1907.09693* (2019).

[130] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions." In: *IEEE Signal Process. Mag.* 37.3 (2020), pp. 50–60. DOI: 10.1109/MSP.2020.2975749.

[131] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. "On the Convergence of FedAvg on Non-IID Data." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[132] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. "Variational Autoencoders for Collaborative Filtering." In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. Ed. by Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis. ACM, 2018, pp. 689–698. DOI: 10.1145/3178876.3186150.

[133] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. "Federated Learning in Mobile Edge Networks: A Comprehensive Survey." In: *IEEE Commun. Surv. Tutorials* 22.3 (2020), pp. 2031–2063. DOI: 10.1109/COMST.2020.2986024.

[134] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, and Xiuzhen Cheng. "Meta Matrix Factorization for Federated Rating Predictions." In: *SIGIR*. ACM, 2020, pp. 981–990.

[135] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. "When Machine Learning Meets Privacy: A Survey and Outlook." In: *ACM Comput. Surv.* 54.2 (2021), 31:1–31:36.

[136] Danyang Liu, Ting Bai, Jianxun Lian, Xin Zhao, Guangzhong Sun, Ji-Rong Wen, and Xing Xie. "News Graph: An Enhanced Knowledge Graph for News Recommendation." In: *Proceedings of the Second Workshop on Knowledge-aware and Conversational Recommender Systems, co-located with 28th ACM International Conference on Information and Knowledge Management, KaRS@CIKM 2019, Beijing, China, November 7, 2019*. Ed. by Vito Walter Anelli and Tommaso Di Noia. Vol. 2601. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 1–7.

[137] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[138] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. "Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics." In: *IEEE Trans. Ind. Informatics* 16.3 (2020), pp. 2134–2143. DOI: 10.1109/TII.2019.2942179.

[139] Yang Luo, Boyi Xu, Hongming Cai, and Fenglin Bu. "A Hybrid User Profile Model for Personalized Recommender System with Linked Open Data." In: *Enterprise Systems Conference, ES 2014, Shangai, China, August 2-3, 2014*. IEEE, 2014, pp. 243–248. DOI: 10.1109/ES.2014.16.

[140] Lingjuan Lyu, Han Yu, and Qiang Yang. "Threats to Federated Learning: A Survey." In: *CoRR* abs/2003.02133 (2020). arXiv: 2003.02133.

[141] Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. "Personalized Social Recommendations - Accurate or Private?" In: *Proc. VLDB Endow.* 4.7 (2011), pp. 440–450.

[142] Masoud Mansoury, Bamshad Mobasher, Robin Burke, and Mykola Pechenizkiy. "Bias Disparity in Collaborative Recommendation: Algorithmic Evaluation and Comparison." In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[143] Stefania Marrara, Gabriella Pasi, and Marco Viviani. "Aggregation operators in Information Retrieval." In: *Fuzzy Sets and Systems* 324 (2017), pp. 3–19. DOI: 10.1016/j.fss.2016.12.018.

[144] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. "Image-based recommendations on styles and substitutes." In: *Proc. of the 38th Int. ACM SIGIR Conf. on Research and Development in Inf. Retrieval*. 2015, pp. 43–52.

[145] Brian McFee, Luke Barrington, and Gert R. G. Lanckriet. "Learning Content Similarity for Music Recommendation." In: *IEEE Trans. Audio, Speech & Language Processing* 20.8 (2012), pp. 2207–2218.

[146] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. Ed. by Aarti Singh and Xiaojin (Jerry) Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1273–1282.

[147] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning Differentially Private Recurrent Language Models." In: *ICLR (Poster)*. OpenReview.net, 2018.

[148] Sean M McNee, John Riedl, and Joseph A Konstan. "Being accurate is not enough: how accuracy metrics have hurt recommender systems." In: *CHI'06 extended abstracts on Human factors in computing systems*. 2006, pp. 1097–1101.

[149] Frank McSherry and Ilya Mironov. "Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders." In: *KDD*. ACM, 2009, pp. 627–636.

[150] Payman Mohassel and Yupeng Zhang. "SecureML: A System for Scalable Privacy-Preserving Machine Learning." In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017, pp. 19–38.

[151] Arvind Narayanan and Vitaly Shmatikov. "How To Break Anonymity of the Netflix Prize Dataset." In: *CoRR* abs/cs/0610105 (2006). arXiv: cs/0610105.

[152] Senthilselvan Natarajan, Subramaniyaswamy Vairavasundaram, Sivaramakrishnan Natarajan, and Amir H. Gandomi. "Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data." In: *Expert Syst. Appl.* 149 (2020), p. 113248. DOI: 10.1016/j.eswa.2020.113248.

[153] Xia Ning, Christian Desrosiers, and George Karypis. "A Comprehensive Survey of Neighborhood-Based Recommendation Methods." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 37–76. ISBN: 978-1-4899-7636-9. DOI: 10.1007/978-1-4899-7637-6_2.

[154] Xia Ning and George Karypis. "Sparse linear methods with side information for top-n recommendations." In: *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*. Ed. by Padraig Cunningham, Neil J. Hurley, Ido Guy, and Sarabjot Singh Anand. ACM, 2012, pp. 155–162. ISBN: 978-1-4503-1270-7. DOI: 10.1145/2365952.2365983.

[155] Ronald Ojino. "User's profile ontology-based semantic model for personalized hotel room recommendation in the web of things: student research abstract." In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*. Ed. by Chih-Cheng Hung and George A. Papadopoulos. ACM, 2019, pp. 2314–2316. DOI: 10.1145/3297280.3297661.

[156] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes." In: *EUROCRYPT*. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238.

[157] Arkadiusz Paterek. "Improving regularized singular value decomposition for collaborative filtering." In: *Proceedings of KDD cup and workshop*. Vol. 2007. 2007, pp. 5–8.

[158] Michael J. Pazzani and Daniel Billsus. "Content-Based Recommendation Systems." In: *The Adaptive Web, Methods and Strategies of Web Personalization*. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007, pp. 325–341. DOI: 10.1007/978-3-540-72079-9_10.

[159] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption." In: *IEEE Trans. Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[160] Huseyin Polat and Wenliang Du. "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques." In: *Proc. of the 3rd IEEE Int. Conf. on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*. IEEE Computer Society, 2003, pp. 625–628.

[161] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. "Privacy-Preserving News Recommendation Model Learning." In: *EMNLP (Findings)*. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 1423–1432.

[162] Laura Elena Raileanu and Kilian Stoffel. "Theoretical Comparison between the Gini Index and Information Gain Criteria." In: *Ann. Math. Artif. Intell.* 41.1 (2004), pp. 77–93. DOI: 10.1023/B:AMAI.0000018580.96245.c6.

[163] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecný, Sanjiv Kumar, and H. Brendan McMahan. "Adaptive Federated Optimization." In: *CoRR* abs/2003.00295 (2020). arXiv: 2003.00295.

[164] Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, and Anderson C. A. Nascimento. "Privacy-Preserving Classification of Personal Text Messages with Secure Multi-Party Computation." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 3752–3764.

[165] Navid Rekabsaz and Markus Schedl. "Do Neural Ranking Models Intensify Gender Bias?" In: *SIGIR*. ACM, 2020, pp. 2065–2068.

[166] Steffen Rendle. "Factorization Machines." In: *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 2010, pp. 995–1000. DOI: 10.1109/ICDM.2010.127.

[167] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian Personalized Ranking from Implicit Feedback." In: *Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence*. 2009, pp. 452–461.

[168] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized Markov chains for next-basket recommendation." In: *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 2010, pp. 811–820. DOI: 10.1145/1772690.1772773.

[169] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. "Neural Collaborative Filtering vs. Matrix Factorization Revisited." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 240–248. DOI: 10.1145/3383313.3412488.

[170] Steffen Rendle and Lars Schmidt-Thieme. "Pairwise interaction tensor factorization for personalized tag recommendation." In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. 2010, pp. 81–90. DOI: 10.1145/1718487.1718498.

[171] Paul Resnick and Hal R. Varian. "Recommender Systems - Introduction to the Special Section." In: *Commun. ACM* 40.3 (1997), pp. 56–58.

[172] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems: Introduction and Challenges." In: *Recommender Systems Handbook*. Springer, 2015, pp. 1–34.

[173] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[174] Lior Rokach and Oded Maimon. "Top-down induction of decision trees classifiers - a survey." In: *IEEE Trans. Syst. Man Cybern. Part C* 35.4 (2005), pp. 476–487. DOI: 10.1109/TSMCC.2004.843247.

[175] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. "On the Convergence of Federated Optimization in Heterogeneous Networks." In: *arXiv preprint arXiv:1812.06127* (2018).

[176] Ashish Kumar Sahu and Pragya Dwivedi. "Knowledge transfer by domain-independent user latent factor for cross-domain recommender systems." In: *Future Gener. Comput. Syst.* 108 (2020), pp. 320–333. DOI: 10.1016/j.future.2020.02.024.

[177] Ruslan Salakhutdinov and Andriy Mnih. "Probabilistic Matrix Factorization." In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Ed. by John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis. Curran Associates, Inc., 2007, pp. 1257–1264.

[178] Pierangela Samarati and Latanya Sweeney. "Generalizing Data to Provide Anonymity when Disclosing Information (Abstract)." In: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*. Ed. by Alberto O. Mendelzon and Jan Paredaens. ACM Press, 1998, p. 188. DOI: 10.1145/275487.275508.

[179] Lei Sang, Min Xu, Shengsheng Qian, and Xindong Wu. "Knowledge graph enhanced neural collaborative recommendation." In: *Expert Syst. Appl.* 164 (2021), p. 113992. DOI: 10.1016/j.eswa.2020.113992.

[180] Adi Shamir. "How to Share a Secret." In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[181] Daqian Shi, Ting Wang, Hao Xing, and Hao Xu. "A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning." In: *Knowl. Based Syst.* 195 (2020), p. 105618. DOI: 10.1016/j.knosys.2020.105618.

[182] Yue Shi, Martha Larson, and Alan Hanjalic. "List-wise learning to rank with matrix factorization for collaborative filtering." In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 269–272.

[183] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux. "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles." In: *RecSys*. ACM, 2009, pp. 157–164.

[184] Reza Shokri and Vitaly Shmatikov. "Privacy-Preserving Deep Learning." In: *CCS*. ACM, 2015, pp. 1310–1321.

[185] Aleksandra B. Slavkovic, Yuval Nardi, and Matthew M. Tibbits. "Secure Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases." In: *ICDM Workshops*. IEEE Computer Society, 2007, pp. 723–728.

[186] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. "Machine Learning Models that Remember Too Much." In: *CCS*. ACM, 2017, pp. 587–601.

[187] Standing Committee of the National People's Congress of Popular Republic of China. *China Internet Security Law*. 2017. URL: http://www.npc.gov.cn/npc/c1481/201507/82ce4cb5549c4f56be8a6744cf2b3273.shtml.

[188] Harald Steck. "Evaluation of recommendations: rating-prediction and ranking." In: *RecSys*. ACM, 2013, pp. 213–220.

[189] Lili Su and Jiaming Xu. "Securing Distributed Gradient Descent in High Dimensional Statistical Learning." In: *POMACS* 3.1 (2019), 12:1–12:41. DOI: 10.1145/3322205.3311083.

[190] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667.

[191] Latanya Sweeney. "k-Anonymity: A Model for Protecting Privacy." In: *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* 10.5 (2002), pp. 557–570. DOI: 10.1142/S0218488502001648.

[192] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. "YAGO 4: A Reason-able Knowledge Base." In: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*. Ed. by Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez. Vol. 12123. Lecture Notes in Computer Science. Springer, 2020, pp. 583–596. DOI: 10.1007/978-3-030-49461-2_34.

[193] Apple's Differential Privacy Team. *Learning with Privacy at Scale*. URL: https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf.

[194] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. "Stealing Machine Learning Models via Prediction APIs." In: *USENIX Security Symposium*. USENIX Association, 2016, pp. 601–618.

[195] Ahmet Uyar and Farouk Musa Aliyu. "Evaluating search features of Google Knowledge Graph and Bing Satori: Entity types, list searches and query interfaces." In: *Online Inf. Rev.* 39.2 (2015), pp. 197–213. DOI: 10.1108/OIR-10-2014-0257.

[196] David Vallet, Arik Friedman, and Shlomo Berkovsky. "Matrix Factorization without User Data Retention." In: *PAKDD (1)*. Vol. 8443. Lecture Notes in Computer Science. Springer, 2014, pp. 569–580.

[197] Anamaria Vizitiu, Cosmin Ioan Niţă, Andrei Puiu, Constantin Suciu, and Lucian Mihai Itu. "Towards Privacy-Preserving Deep Learning based Medical Imaging Applications." In: *2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE. 2019, pp. 1–6.

[198] Denny Vrandecic. "Wikidata: a new platform for collaborative data collection." In: *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*. Ed. by Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab. ACM, 2012, pp. 1063–1064. DOI: 10.1145/2187980.2188242.

[199] Denny Vrandecic and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." In: *Commun. ACM* 57.10 (2014), pp. 78–85. DOI: 10.1145/2629489.

[200] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. "RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea et al. ACM, 2018, pp. 417–426. DOI: 10.1145/3269206.3271739.

[201] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. "Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems." In: *ACM Trans. Inf. Syst.* 37.3 (2019), 32:1–32:26. DOI: 10.1145/3312738.

[202] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. "Knowledge Graph Convolutional Networks for Recommender Systems." In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. Ed. by Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia. ACM, 2019, pp. 3307–3313. DOI: 10.1145/3308558.3313417.

[203] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. "Federated Learning with Matched Averaging." In: *CoRR* abs/2002.06440 (2020). arXiv: 2002.06440.

[204] Jianyu Wang and Gauri Joshi. "Adaptive Communication Strategies to Achieve the Best Error-Runtime Trade-off in Local-Update SGD." In: *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. Ed. by Ameet Talwalkar, Virginia Smith, and Matei Zaharia. mlsys.org, 2019.

[205] Jun Wang, Qiang Tang, Afonso Arriaga, and Peter Y. A. Ryan. "Novel Collaborative Filtering Recommender Friendly to Privacy Protection." In: *IJCAI*. ijcai.org, 2019, pp. 4809–4815.

[206] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. "Federated Evaluation of On-device Personalization." In: *CoRR* abs/1910.10252 (2019).

[207] Ting Wang, Daqian Shi, Zhaodan Wang, Shuai Xu, and Hao Xu. "MRP2Rec: Exploring Multiple-Step Relation Path Semantics for Knowledge Graph-Based Recommendations." In: *IEEE Access* 8 (2020), pp. 134817–134825. DOI: 10.1109/ACCESS.2020.3011279.

[208] Wei Wang, Guangquan Zhang, and Jie Lu. "Collaborative Filtering with Entropy-Driven User Similarity in Recommender Systems." In: *Int. J. Intell. Syst.* 30.8 (2015), pp. 854–870. DOI: 10.1002/int.21735.

[209] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. "Explainable reasoning over knowledge graphs for recommendation." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 5329–5336.

[210] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. "Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning." In: *INFOCOM*. IEEE, 2019, pp. 2512–2520.

[211] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis." In: *IEEE Trans. Inf. Forensics Secur.* 15 (2020), pp. 3454–3469.

[212] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. "BlurMe: inferring and obfuscating user gender based on ratings." In: *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*. Ed. by Padraig Cunningham, Neil J. Hurley, Ido Guy, and Sarabjot Singh Anand. ACM, 2012, pp. 195–202. DOI: 10.1145/2365952.2365989.

[213] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. "HybridAlpha: An Efficient Approach for Privacy-Preserving Federated Learning." In: *AISec@CCS*. ACM, 2019, pp. 13–23.

[214] Usha Yadav, Neelam Duhan, and Komal Kumar Bhatia. "Dealing with Pure New User Cold-Start Problem in Recommendation System Based on Linked Open Data and Social Network Features." In: *Mob. Inf. Syst.* 2020 (2020), 8912065:1–8912065:20. DOI: 10.1155/2020/8912065.

[215] Ronald R Yager. "Quantifier guided aggregation using OWA operators." In: *International Journal of Intelligent Systems* 11.1 (1996), pp. 49–73.

[216] Ronald R. Yager. "On ordered weighted averaging aggregation operators in multicriteria decisionmaking." In: *IEEE Trans. Systems, Man, and Cybernetics* 18.1 (1988), pp. 183–190. DOI: 10.1109/21.87068.

[217] Emre Yalcin, Firat Ismailoglu, and Alper Bilge. "An entropy empowered hybridized aggregation technique for group recommender systems." In: *Expert Syst. Appl.* 166 (2021), p. 114111. DOI: 10.1016/j.eswa.2020.114111.

[218] Dingqi Yang, Daqing Zhang, and Bingqing Qu. "Participatory Cultural Mapping Based on Collective Behavior Data in Location-Based Social Networks." In: *ACM TIST* 7.3 (2016), 30:1–30:23.

[219] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. *Federated Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019. DOI: 10.2200/S00960ED2V01Y201910AIM043.

[220] Zuoxi Yang and Shoubin Dong. "HAGERec: Hierarchical Attention Graph Convolutional Network Incorporating Knowledge Graph for Explainable Recommendation." In: *Knowl. Based Syst.* 204 (2020), p. 106194. DOI: 10.1016/j.knosys.2020.106194.

[221] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)." In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[222] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. "Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach." In: *IEEE Access* 8 (2020), pp. 23920–23935. DOI: 10.1109/ACCESS.2020.2968399.

[223] Jianbo Yuan, Walid Shalaby, Mohammed Korayem, David Lin, Khalifeh AlJadda, and Jiebo Luo. "Solving cold-start problem in large-scale recommendation engines: A deep learning approach." In: *2016 IEEE Int. Conf. on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*. IEEE Computer Society, 2016, pp. 1901–1910.

[224] Fadila Zerka, Samir Barakat, Sean Walsh, Marta Bogowicz, Ralph TH Leijenaar, Arthur Jochems, Benjamin Miraglio, David Townend, and Philippe Lambin. "Systematic Review of Privacy-Preserving Distributed Machine Learning From Federated Databases in Health Care." In: *JCO Clinical Cancer Informatics* 4 (2020), pp. 184–200.

[225] Feng Zhang, Victor E. Lee, and Kim-Kwang Raymond Choo. "JO-DPMF: Differentially private matrix factorization learning through joint optimization." In: *Inf. Sci.* 467 (2018), pp. 271–281.

[226] Jia-Dong Zhang and Chi-Yin Chow. "Enabling Probabilistic Differential Privacy Protection for Location Recommendations." In: *IEEE Trans. Serv. Comput.* 14.2 (2021), pp. 426–440.

[227] Qian Zhang, Peng Hao, Jie Lu, and Guangquan Zhang. "Cross-domain Recommendation with Semantic Correlation in Tagging Systems." In: *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*. IEEE, 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8852049.

[228] Richong Zhang and Thomas T. Tran. "An information gain-based approach for recommending useful product reviews." In: *Knowl. Inf. Syst.* 26.3 (2011), pp. 419–434. DOI: 10.1007/s10115-010-0287-y.

[229] Yongfeng Zhang and Xu Chen. "Explainable Recommendation: A Survey and New Perspectives." In: *CoRR* abs/1804.11192 (2018). arXiv: 1804.11192.

[230] Tong Zhao, Julian McAuley, and Irwin King. "Improving latent factor models via personalized feature projection for one class recommendation." In: *Proc. of the 24th ACM Int. on Conf. on information and knowledge management*. 2015, pp. 821–830.

[231] Wayne Xin Zhao, Sui Li, Yulan He, Liwei Wang, Ji-Rong Wen, and Xiaoming Li. "Exploring demographic information in social media for product recommendation." In: *Knowl. Inf. Syst.* 49.1 (2016), pp. 61–89. DOI: 10.1007/s10115-015-0897-5.

[232] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. "Federated Learning with Non-IID Data." In: *CoRR* abs/1806.00582 (2018). arXiv: 1806.00582.

[233] Elena Zheleva and Lise Getoor. "To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles." In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl. ACM, 2009, pp. 531–540. DOI: 10.1145/1526709.1526781.

[234] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. *Differential Privacy and Applications*. Vol. 69. Advances in Information Security. Springer, 2017.

[235] Tianqing Zhu, Yongli Ren, Wanlei Zhou, Jia Rong, and Ping Xiong. "An effective privacy preserving algorithm for neighborhood-based collaborative filtering." In: *Future Gener. Comput. Syst.* 36 (2014), pp. 142–155.

[236] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. "Heterogeneous hypergraph embedding for document recommendation." In: *Neurocomputing* 216 (2016), pp. 150–162. DOI: 10.1016/j.neucom.2016.07.030.

[237] Ziwei Zhu, Jianling Wang, and James Caverlee. "Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems." In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 449–458. DOI: 10.1145/3397271.3401177.