Evaluating Federated Session-Based Recommender Systems

Marko Harasic
Dennis Lehmann
Adrian Paschke
marko.harasic@fokus.fraunhofer.de
dennis.lehmann@fokus.fraunhofer.de
adrian.paschke@fokus.fraunhofer.de
Fraunhofer FOKUS
Berlin, Berlin, Germany

Babak Mafakheri
Babak.mafakheri@zii.aero
Safran Passenger Innovations
Weßling, Bavaria, Germany

ABSTRACT

Recommender Systems are essential for enhancing the user experience in various domains. As the next generation of in-flight entertainment (IFE) systems offers a vast selection of multimedia items, onboard recommender systems become necessary. However, due to the unique requirements of anonymous travelers, only session-based recommender systems can effectively operate. Moreover, local models are required, and collaboration among multiple peers with minimal data sharing is essential. Federated Learning, a privacy-aware and scalable approach for training and operating Machine Learning algorithms, is a natural solution to combine with traditional recommendation algorithms, ensuring privacy and delivering high-quality recommendations.

While there are established evaluation standards for centralized recommender systems, there is currently no common agreed benchmark for federated recommender systems, especially for session-based recommendations. This research aims to address this gap by presenting the results of a performance comparison between popular recommendation algorithms. We compare their traditional monolithic implementation with their federated counterparts, considering the unique challenges of the in-flight entertainment setting. By analyzing and comparing these approaches, we provide insights into the effectiveness and efficiency of federated recommender systems for session-based recommendations.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Distributed artificial intelligence*; • **General and reference** → **Evaluation**.

KEYWORDS

Federated Learning, Recommender systems, Federated Recommender Systems, Data sets, Evaluation, Session-based Recommender Systems

⟨KDD '23 OARS Workshop⟩, ⟨7th August 2023⟩, ⟨Long Beach, CA, US⟩

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Marko Harasic, Dennis Lehmann, Adrian Paschke, and Babak Mafakheri. 2023. Evaluating Federated Session-Based Recommender Systems. In *Proceedings of The 3rd International Workshop on Online and Adaptive Recommender Systems (⟨KDD '23 OARS Workshop⟩).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recommender Systems (*RS*s) have become essential to contemporary information portals, streaming platforms, and e-commerce solutions. Originating from the field of *RS* [32], *RS*s now play a crucial role in helping users gather information, plan vacations, consume multimedia content, and make purchases. For instance, popular streaming websites like Netflix boast a staggering catalog of over 15 thousand movies and series, leaving customers with the daunting task of making decisions. In this context, *RS* serve as a valuable tool, assisting users in navigating through the vast array of options and aiding their decision-making process by offering personalized recommendations that align with their preferences and requirements.

*RS*s employ user behavior analysis, social connections, and demographic data to understand individual preferences and generate personalized recommendations. However, such data collection poses a significant threat to user privacy. While participants willingly share personal information to receive enhanced recommendations, there is simultaneous concern that these systems may gather excessive information or exploit it for malicious purposes [41].

Recognizing the sensitivity of users regarding their data, governments have taken steps to implement regulations and policies governing the collection and processing of personal data. In 2017, the European Union introduced the European Commission's General Data Protection Regulation (*GDPR*), establishing privacy as a fundamental right in the digital realm [38]. Compliance with these regulations and laws is mandatory, necessitating incorporating privacy awareness in modern *RS* [40].

In response to these privacy concerns, Google proposed a decentralized machine learning approach known as Federated Learning (*FL*)[25]. *FL*s architecture ensures that training data remains on individual devices, enabling peers to train a shared model collaboratively. Only model updates are transmitted to a central instance, which aggregates the specific results, updates the global model, and then distributes it back to the participants. This iterative process is repeated until the desired level of accuracy is achieved or the system continuously adapts to the user's interactions.

Given the promising privacy-preserving advantages of *FL*, it became apparent that the research community began exploring its integration into *RS*. This combination is known as Federated Recommender System (*FedRec*) [42]. Unlike traditional monolithic *RS*, *FedRec* ensures user privacy by conducting personalization directly on the user's device. Only interpretations of actions are shared among peers, and data security can be further enhanced by applying privacy techniques to the data and communication. With *FedRec*, users can maintain their preferences in a locally refined profile while still benefiting from the collective intelligence of the crowd. So far, the interest on *FedRec* only started but showed a high potential to address scalability, cooperation among operators, and privacy challenges [21]. Several surveys giving an overview of the current state of research were published quite recently to present the different new algorithms [42], their applied domains, and utilized data sets [3].

2 MOTIVATING CASE-STUDY

Millions of air passengers rely on In-Flight Entertainment (*IFE*) systems daily to enhance their travel experience. Studies have shown that a positive multimedia experience can reduce passenger stress levels. From the early days of in-flight movies in 1921 to the personal *IFE* systems available today, the evolution of multimedia content on airplanes has been significant. However, current IFE systems often have limited content, causing frequent travelers to exhaust all available options quickly. To address this, the deployment of next-generation *IFE* systems with a vast library of movies is the logical next step.

Recently, *IFE* system providers have started expanding the range of multimedia content available on airplanes, hosting hundreds to thousands of items. With this abundance of choices, passengers now face a challenge similar to browsing streaming platforms like Netflix. They are overwhelmed by the options and experience the Paradox of Choice while using the small devices installed in front of their seats. To address this, passengers require suitable tools to navigate the extensive content space. *RS*s are the natural choice to enhance the user experience by offering personalized recommendations based on their preferences.

2.1 Problem statement

During a flight, there are usually limited interactions between travelers and the *IFE* system. Given the nature of the content and the rather short time in front of the device, gathering sufficient data to create comprehensive user profiles isn't easy. Additionally, travelers remain anonymous, and no persistent storage of their preferences is established.

Due to the lack of network connectivity during the flight, a central server on the airplane cannot calculate recommendations, and interaction data from the local devices cannot be transmitted to a remote server. The local devices, usually ARM-based, have limited memory and computational capabilities resources. Therefore, they cannot handle the large models typically used by providers of personalized services.

Even if local devices could operate on optimized models, updating them with new information poses additional challenges. Airplanes rely on satellite communication, and it is rare to have connections with enough bandwidth to upload user interactions and update models. Training modern Deep Learnings (*DL*s) models require substantial resources, and there is a possibility that the aircraft won't have a stable connection to the central server when a new model is ready for deployment. Consequently, there may be a delay between the insights obtained from recent interaction data and the implementation of an updated model. This delay can result in lower-quality recommendations generated by outdated models.

Furthermore, both *IFE* system manufacturers and multimedia providers gather valuable customer data, including browsing patterns, interaction behavior, and viewing preferences. This data is highly valuable for strategic decision-making, directly impacting the companies' worth. Consequently, these companies are reluctant to share their data with others. However, an extended user base could benefit all collaborators, including competitors, presenting a dilemma.

2.2 Open questions regarding the case study

Given the environment of the case study, the following main questions were identified:

- How can anonymous users get good recommendations with only minor captured information?
- How to share information between different peers without disclosing private or business data?
- How Wisdom of the Crowd can be incorporated into a distributed recommendation environment?
- How to identify the best algorithmic approach?

The first three questions can be answered with modern state-of-the-art approaches. Section 4 overviews the algorithmic and architectural solutions for the tree questions. The last one is still an open research topic, and this paper's main contribution is one of the first attempts to answer this question.

3 PAPER CONTRIBUTION

Since their first appearance, traditional centralized *RS* have been extensively studied in research. Over the past two decades, a gold standard has emerged for comparing different algorithms [7, 11, 14, 23, 44]. However, when it comes to *FedRec*, there is currently no widely accepted guideline for comparing the capabilities and limitations of different approaches. Many research publications lack a comparative analysis of their proposed algorithms against well-established methods in centralized and distributed settings.

In contrast to centralized *RS* that have access to the complete dataset and operate on a single machine or cluster in a data center, *FedRec* systems consist of potentially thousands of heterogeneous devices, each with a limited view of its local data. This introduces additional challenges and uncertainties in evaluating *FedRec* algorithms, particularly in the context of Session-based Recommender System (*SBRS*). To the best of our knowledge, there is no existing combination of *SBRS* and *FedRec*.

Our work aims to address this gap by creating one of the first comprehensive benchmarks for evaluating the impact of *FL* on the overall quality of *SBRS*. We aim to transfer several popular *SBRS* algorithms into a federated setting and compare their performance against their centralized counterparts. By doing so, we aim to shed

light on the potential benefits and limitations of using *FL* in *SBRS* and provide valuable insights for future research in this area.

4 ALGORITHMIC BACKGROUND

*RS*s in the investigated environment consists of several central core components. First, there is the core recommendation algorithm. Especially in short contextualized interactions, *SBRS*s showed promising results whose quality in accuracy exceeded those of traditional approaches. *FL* allows training, distributing, and especially updating a *RS* model, which is even offline available. The combinations of *RS* and *FL* formed an emerging research field, which is called *FedRec*s. The following will describe the approaches and give background information on each core component.

4.1 Session-based recommender systems

*SBRS*s are the state-of-the-art approach to generate recommendations when only limited knowledge about their preferences is available. Widely deployed in portals, where the captured information consists of interaction sequences with mostly 4-10 events, *SBRS*s treat each interaction stream independently. They are called sessions. A session is an ordered or unordered list of interactions from one user with a clear temporal boundary, which happen in the same context and contain latently the recent user goals [39]. As *SBRS*s treat sessions isolated and allow to discover the user's goal independently. Therefore, they can generate high-quality recommendations even to anonymous users without needing a persistent profile [39].

The algorithms of *SBRS* can be classified into the following three families of methods [39]: Conventional approaches, which rely on sequential rule mining, apply modified traditional *RS* approaches or use Markov Chains to recommend the next items; Latent representation approaches, which first learn latent features of users and items and then use the learned representations to forecast future steps of the user and more recently Deep Learning (*DL*)-based approaches, which learn non-linear functions which capture sequentially dependent relations between different entities and then predict the next interactions. The following presents the most popular *DL* approaches for *SBRS*.

4.1.1 Variational Auto Encoders (VAEs). Auto Encoders (*AE*s) [31] take the users' unordered sessions as input and reconstruct the input vector as an output [33]. *VAE*s have a similar architectural structure to *AE*. Conversely, they reconstruct the probabilistic distribution of the input data [16]. As a side-effect, the preference likelihood of unseen items is generated in that process, and those items with the highest values form then the users' recommendations. Outperforming the traditional *AE*, they are considered today as one of the state-of-the-art approaches in recommendation accuracy [22].

4.1.2 Neural Collaborative Filtering (NCF). With their huge success in the Netflix competition [5], the research community for *RS* applied factorization methods in the recommendation tasks [18]. As the original unordered sessions consisting of user-item interactions form a large but sparse matrix, decomposing the original matrix into two smaller ones creates low-dimensional and dense representations of the items' features and users' preferences in the form of their embeddings. *NCF* replaces then the scalar product of classical Matrix Factorisation (*MF*) with a Multilayer Perceptron (*MLP*) to estimate and function and to learn non-linear dependencies between items and users [13].

4.1.3 DeepFM. DeepFM is a *SBRS* algorithm that combines *DL* and Factorisation Machine (*FM*) [30] techniques to provide personalized recommendations in session-based scenarios. As introduced by Guo et al., DeepFM aims to capture low-order and high-order feature interactions for improved recommendation accuracy [12]. It employs a Deep Neural Network (*DNN*) to learn complex representations of user behavior patterns and combines them with factorization machines to model feature interactions. DeepFM can effectively capture linear and non-linear relationships in session data by incorporating shallow and deep learning components.

4.1.4 Caser. Caser is a *SBRS* algorithm that utilizes Convolutional Neural Networks (*CNN*s) to provide personalized recommendations based on sequential user behavior [35]. By treating user behavior as a sequence of items within a session and employing 1D convolutions over item embeddings, Caser captures temporal and sequential patterns effectively. It consists of session-parallel mini-batches for parallel computation, convolutional operations to capture item dependencies, and a fully connected layer for generating recommendation scores. Caser excels in session-based scenarios by addressing long-term preferences and short-term interests.

4.1.5 GRU4Rec. GRU4Rec is a *SBRS* algorithm that utilizes Gated Recurrent Units (*GRU*s) [8] to provide personalized recommendations based on sequential user behavior. As former interactions directly influence future decisions, the regard of that additional information would lead to higher recommendation accuracy in *SBRS*. GRU4Rec and its extensions are the first recommendation system that incorporated that kind of approach [15]. By employing *GRU*s, Gru4rec effectively models the temporal dynamics and captures the sequential patterns in user behavior. After several Recurrent Neural Network (*RNN*)-layers, a final group of *MLP* layers is used to learn a classifier for predicting the next item of interest.

4.1.6 BERT4Rec. Transformer architectures are state-of-the-art in Natural Language Processing (*NLP*) processing [37]. In our written language, each word carries a semantic that influences the meaning of the other sentences' words. BERT was one of the first successful applications of the transformer model on normal languages [9]. That concept was applied in the context of *SBRS* in its first implementation BERT4Rec [34], as the underlying idea is that each interaction event influences a future one comparable to a word in a sentence. In the training phase of such *RS*, intermediate interactions in ordered sessions are masked in the training phase, and the model tries to fill those gaps.

4.2 Federated Learning

The traditional centralized approach for deep learning *DL* involves uploading and processing data on a central server or data center. However, this approach is no longer sustainable due to privacy concerns associated with user data.

FL addresses these privacy concerns by enabling collaborative and distributed training of machine learning models without exposing sensitive data to training partners [25]. Training models for

Machine Learning (*ML*) require substantial amounts of data. Still, companies have valid reasons to keep their data confidential, and individuals are cautious about sharing personal information like location and browsing behavior. With increasing privacy concerns and government regulations, collecting data has become a significant challenge. Valuable data that could be utilized for advanced recommender systems are often trapped in data silos or personal devices, limiting its use for training. However, *FL* offers a solution to this fundamental challenge by enabling private and secure training of modern recommender system models.

With its decentralized architecture, the training data in *FL* remains on personal devices. Users collaboratively train a shared model by sending updates derived from locally refined models to a central instance. The central instance aggregates these updates and transfers the refined model or updates back to the peers. This iterative process continues until the desired level of precision is achieved, or the system continuously adapts to user interactions over its usage period. As a result, each device has a locally available global model, enabling independent inferences without relying on a central instance or network availability. The setup of a plain *FL* consists in general of three main parts: The *ML* model to be fitted in the training/operation process, a central managing instance, and an arbitrary number of participating peers:

- ***ML* model:** An *ML* model is a set of algorithms and their parameters to calculate predictions based on the collected data. While the model is defined priorly, its weights and biases are adjusted during the iterative training of the model.
- **Central instance:** The central instance manages the coordination of the peers. It selects a subset of peers which will contribute to the upcoming training round. These active peers transmit their updates to the central instance, which merges that information into the new model and distributes it afterward to the network.
- **Participating peers:** If a peer becomes active, it conducts a predefined number of training steps with local data. After finishing that task, the peer updates the central instance. Each peer uses its local model to perform the model inferences with its data.

In *FL*, the model's training is not performed centrally, each peer trains the model with his own collected training data individually. Figure 1 shows the training process of traditional *FL*. It is organized in rounds, where each round has three separate steps:

(1) **Model distribution** (Step 1): The central instance starts each new training round by selecting a set of participating peers from the available ones. Then, the central instance transmits the current model to them. Each selected peer will then perform the training with their locally available data.
(2) **Model training** (Step 2): The training phase of the model consists of two steps for each selected peer: First, the recent model is fitted with the locally available data (Step 2a), and then the updates on the model are transmitted back to the central instance (Step 2b).
(3) **Model aggregation** (Step 3): Once the central instance collects all local updates from the participating peers in the current round, it merges that information into a global update on the model. In the traditional *FL* setting, the central instance updates the global model by averaging the collected model weights into a new model (FedAVG) [25]. This step is called aggregation (Step 3).

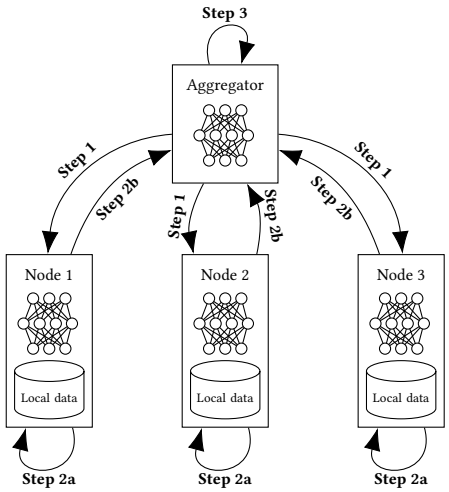

Figure 1: Typical training process of a horizontal Federated Learning system.

4.3 Federated Recommender Systems

Beginning with the first implementations and the promising benefits of *FL* on privacy challenges, it was natural that the research community started to examine the training and operation in *RS*s a *FL* setting. The concept of *FedRec*s was introduced by Yang et al. [42]. *FedRec* combines traditional *RS* methods with the *FL* approach to creating a decentralized and privacy-preserving *RS*. In *FedRec*, peers, whether they are users or companies, retain their data locally, including interaction patterns, ratings, and user profiles. This information is not shared with other peers during the training and operation of the *RS*. Additionally, each party typically has access to publicly available item information, such as descriptions, features, or categories. *FedRec*s can be roughly categorized into three types: Horizontal *FL*, Vertical *FL* and into Transfer *FL* [43]:

- **Horizontal FedRec:** Peers share a common item space. Items on all clients belong to the same domain and have the same feature space. Each peer contains a set of users or even refers to a single one. An example of a Horizontal *FedRec* would be a mobile movie *RS* from a single company, where each user has his user profile on his device, and all users have access to the same collection of movies.
- **Vertical FedRec:** Vertical *FedRec* describes federated model training on heterogeneous data distributed across multiple data silos. The peers have the same user set, but each provides different recommendable items or item features. Its target is aggregating these features in a privacy-assuring process to create a collaborative model with information from various parties. An example of such a system would be a shared *RS* from two companies of different fields, one for books and one for movies. They have the same user base consisting of a shared user identifier.

- **Transfer FedRec:** Transfer *FedRec*s are used in scenarios where the peers' data differ not only in the user space but also in the item space. For example, two companies from different businesses and locations want to build a new shared *RS* without disclosing their private data. The user sets of those two companies only overlap to a small degree, and because of their different domains, only a tiny feature space intersection of their provided items exists. In this case, a common representation between the two spaces is learned with transfer learning techniques [27]. Those few common user/item combinations form a bridge to generate the recommendations with only features taken from one side.

5 EVALUATION SETUP

This section focuses on evaluating algorithms using a real data set provided by an industrial partner. The evaluation was conducted offline, independently of real users, and involved comparing predicted behavior with actual captured information. Figure 2 shows the general process of the applied offline evaluation process according to [7].

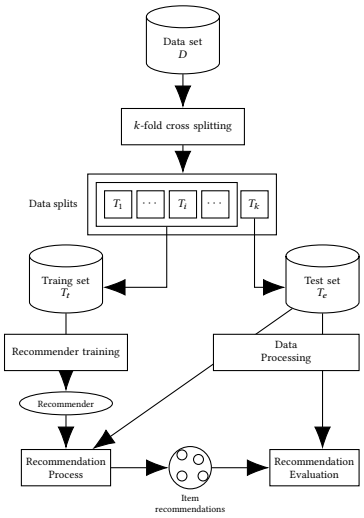

Figure 2: General overview if the offline evaluation process

We measured the systems by comparing the predicted behavior with the real captured information [14]. In this process, "real" user data is preferable over synthetic data sets [2]. Even though an offline evaluation has its particular drawbacks as it relies on heuristics with assumptions, that step gives valuable insights before the roll-out of the systems. Therefore it is a best practice to design and implement several recommendation algorithms and evaluate them before the best-performing algorithms are deployed and evaluated online [17].

The implementation of the algorithms employed PyTorch for recommendation purposes and the IBM FL framework for federated learning. A central system provided a simulated peer environment to simulate a federated environment. Even though with FedScale [20] and LEAF [6] powerful simulation frameworks for *FL* exist, we decided to create our implementation because of the particular requirements, data sets, and evaluation metrics of *FedRec*.

The evaluation criteria encompassed accuracy, measured through Normalized Discounted Cumulative Gain (*nDCG@k*), and diversity, evaluated using Long Tail Coverage (*LCC@k*). By employing these evaluation methods, we aimed to gain valuable insights into the algorithms' performance before deploying the systems. This introductory section provides an overview of the evaluation methodology utilized in the subsequent analysis.

5.1 Software and hardware environment

The recommendation algorithms and their evaluation were implemented in PyTorch [29]. For the *FL*-part, the IBM *FL* framework was used [24]. That framework decouples the *FL* methods, such as peer-aggregator communication, local data loading, and model aggregation from the model development. Furthermore, it contains several state-of-the-art aggregation methods and connects seamlessly with many existing centralized PyTorch implementations of *DL*-algorithms. Nevertheless, several extensions of that library, which regard the particular details of the applied *RS*-algorithms, needed to be applied. See table 1 for further details on versions of the used software stack.

Operating system	Linux Ubuntu 22.04	
Python version	3.6.13	
JupyterLab version	3.0.12	
PyTorch version	1.8.1	
IBM FL version	1.0.7	
CPU Model	Intel Xeon E5-2695	
Cores	18	
CPU Frequency	2.1	GHz
RAM	512	GByte
GPU Model	GTX 1080	
RAM	12	GByte
CUDA version	11.2	

Table 1: Evaluation Environment

As the number of peers in a *FL*-based *RS* easily surpasses hundreds or even thousands of nodes, an evaluation running on real devices on such a scale is not feasible. A central system provided the environment with several simulated peers and gave a good real-world distribution estimation. Table 1 shows the hardware specifications of that server. Each peer, as well as the central aggregating instance, runs locally in Jupyter notebooks. Therefore, the impact of bandwidth and network restrictions can not be measured, and that factor is omitted in the evaluation.

5.2 Data source

Using a data set provided by an industrial partner, the algorithms were evaluated offline independently of real users. SAFRAN, an industrial partner who manufactures *IFE*-systems, created that data set. It contains 900k play-stop-pause events on multimedia content of airplane travelers and identifiers of 1.7k flights and *IFE*-seats from one week. We consider all interactions from one *IFE*-seat in a single flight as an independent session and regard only play-events as

significant interaction type. As there are indications that consumed content correlates with the flight destination, we treat each flight as a single peer.

#Flights	Items	Sessions	∅ Sessions/Peer	∅ Items/Session
8	192	3,149	393.6	5.5
16	320	6,116	382.3	5.3
32	336	11,070	346.0	5.0
64	523	19,659	307.4	4.7
128	1,253	32,658	255.5	4.5

Table 2: Statistics of the fly-media data set after its preprocessing and cleaning

We derived out of the original data set partitions of size $N = [8, 16, 32, 64, 128]$, each consisting of the N peers with the highest session count from the original data, regardless of the particular session length. All items were removed that were only accessed once. Table 2 shows its characteristics after its preparation and cleaning. The data was then partitioned for each particular number of peers into five disjunctive slices of roughly equal size for a 5-fold cross evaluation [19]. It was ensured that all session data of a single airplane was never distributed over different slices. Four slices were used for training the models, and the remaining slice then for evaluating the system. The data set for training was augmented by splitting every single session into sequences of shorter length, always beginning with the first item of the session and its k succeeding items. The last item was then the label to predict. For evaluation, the last item of each test session was removed, and the systems' task was to predict that missing item. This approach allows the simulation of the recommenders' prediction behavior based on the information it would have available when requesting a recommendation at a certain point of time [11].

5.3 Applied evaluation criteria

Evaluating a *RS* using a comprehensive set of metrics is significant. Relying on a single metric for the evaluation would generally result in a biased assessment that fails to capture the overall performance of the *RS* [44]. When optimizing an RS based on one metric, it is crucial to assess whether this optimization negatively impacts performance in other areas [10]. Accuracy is probably the most important dimension when evaluating a *RS*. Nevertheless, recent research trends indicated that diversity is equally important in the perceived users' experience [26]. We apply *nDCG@k* to measure the ranking performance of the investigated algorithms and *LCC@k* to measure the algorithms' coverage in the long tail. For both metrics, $k = 20$ was chosen as the size of the recommendation lists, as this is a common length in recommenders' literature.

5.3.1 Normalized Discounted Cumulative Gain (nDCG@k). as ranking metric regards the position of the items in the result list [36]. The calculated estimation of preference likelihood -denoted as relevance *rel*- orders that list. *nDCG@k* involves discounting the relevance score by dividing it by the log of the corresponding position. All items $i \in I_{e,u}$ taken from the users' u item test set $I_{e,u}$ are considered relevant ($rel = 1$), and all other items of the recommendation list are irrelevant ($rel = 0$). The result is normalized by the ideal order, given by *iDCG@k*. For each user, there is usually exactly one item ranked first. The system's performance is then measured by normalizing the *nDCG@k* scores of the test sets' U_t users.

$$DCG@k = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$iDCG@k = \sum_{i=1}^{|REL_N|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$nDCG@k = \frac{DCG@k}{iDCG@k}$$

5.3.2 Long Tail Coverage LCC@k. First formulated by Park and Tuzhilin as the long-tail problem [28], it is challenging for most *RS* algorithms to access items in the tail. The long tail denotes the distribution of items, where only a few items have high popularity, and most items in the system are rarely accessed [4]. Our data set follows such a typical distribution of interaction count on items (figure 3). The short-head H denotes the set of those 20% of items with the most user interactions, while the set of long-tail items T contains the majority of items (80%) with fewer interactions.

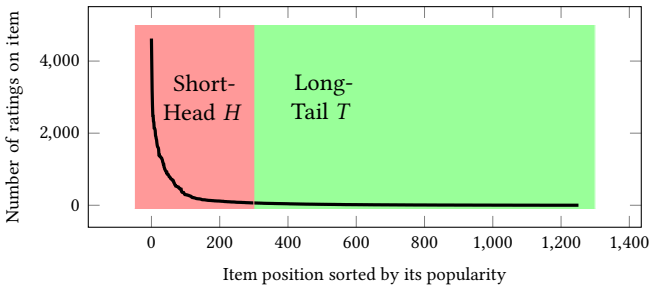

Figure 3: Typical long tail distribution of items and their corresponding ratings in the fly-media data set.

To compare different algorithms in their capability to access the long-tail, coverage metrics were introduced [1]. The most important among those is the Long Tail Coverage *LCC@k*. Unlike other long-tail metrics such *APL@k* or *RP@k*, it measures the diversity of long-tail recommendations by considering the uniqueness of items. A system could even reach high values for *RP@k* and *LCC@k* by recommending only a small set of long-tail items. If measures the fraction of unique long-tail items $i \in T$ from the whole set T over the full recommendation lists $RL@k(u)$ for the users u in the test set U_t. It is formally defined as:

$$LCC@k = \frac{1}{|T|} \cdot \left| \left(\bigcup_{u \in U_t} RL@k(u) \right) \cap T \right|$$

6 EVALUATION

This section describes the training process of the implemented algorithms and their evaluation. We show and compare the behavior

of the algorithms in monolithic and federated settings according to the deployed evaluation criteria. The impact of the number of nodes is shown for the federated setting. Furthermore, investigate the suitability of the algorithms to run on embedded computers with ARM architecture, as similar devices are deployed in the real world *IFE*-systems.

6.1 Training of the models

The algorithms from section 4.1 were implemented following the reference architecture from their corresponding publications. Table 3 shows the sizes for the models and their determined hyperparameters. We applied a grid-search approach to determine the best values for learning rate, batch size, epochs, and in the federated setting for the number of rounds and local epoch. We discovered that the number of *FL* rounds have only a minor impact on the overall performance. The same algorithms were then implemented according to the *FL* paradigm in a horizontal architectural style, as data from the same domain is used as input. Models of identical configuration are used for both the monolithic and federated implementations. That means they have the same number of weights/biases and sizes.

Furthermore, table 3 presents the measured RAM demands for each model during the training phase. Although Bert4Rec demonstrates promising potential, its usage in the federated setting was unfeasible due to its significantly greater memory requirements than the simpler models. Despite having a similar number of weights and biases as the other algorithms, the excessive resource demands of Bert4Rec surpassed the capabilities of our systems, rendering any further advancements in the federated setting impractical.

6.2 Impact of number of nodes

Figure 4 shows the accuracy in the *nDCG@k* metric of federated algorithms regarding the *FL*-nodes of the network. With a growing number of participating peers, more session data is included in the training process. While increasing the node count initially improves accuracy, there is a point beyond which the accuracy starts to decline. Including more parties with a more disjunct set of preferences and a larger corpus of items results in a more sparse data set. That data shifting challenge degrades the overall system's accuracy. This leads to a stronger shift in focus towards the tail of the data set, resulting in a decline in overall performance. It appears that capturing the more complex patterns and details in the tail becomes more challenging, especially for the VAE. Therefore, adjusting the VAE's learning strategy to prioritize the tail of the dataset does not yield improved results but worsens its accuracy. At the same time, the other approaches are more stable over the number of nodes.

6.3 Accuracy

Figure 5 shows the monolithic implementations' accuracy compared to its federated counterpart running on 128 nodes. Our analysis indicates that VAE performs exceptionally well on the data set. This success can be attributed to its ability to learn session-independently and focus on the data set's key aspects, specifically the head. In contrast, the session-regarding approaches suffer from the data sets unbalanced distribution of session-based data, given by the data sets strong bias on the tail. As NCFs accuracy is already

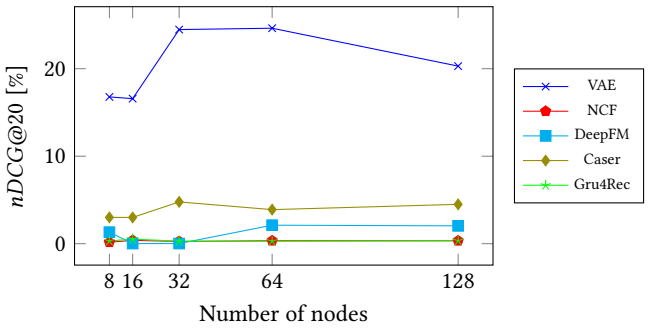

Figure 4: Accuracy of federated algorithms concerning the number of participating FL nodes.

low in its monolithic version, its accuracy loss in the federated implementation is not significant enough to state a negative impact because of the federalization. Casers minor accuracy degradation was expected because of the aggregation process of *FL* but could be mitigated with different aggregation strategies. The isolated examination of individual sessions within the nodes hinders the ability to obtain a reliable mean during aggregation in GRU4Rec. This is due to the model's focus on individual sessions, preventing it from capturing the overall trends and patterns in the data. As a result, the aggregated mean may not accurately represent the underlying data distribution, limiting the effectiveness of GRU4Rec in providing accurate recommendations. On the other hand, the DeepFM model significantly benefits from FL, although the reasons behind this observation remain unclear. Nevertheless, the small monolithic accuracy gives no real statistical evidence for that improvement.

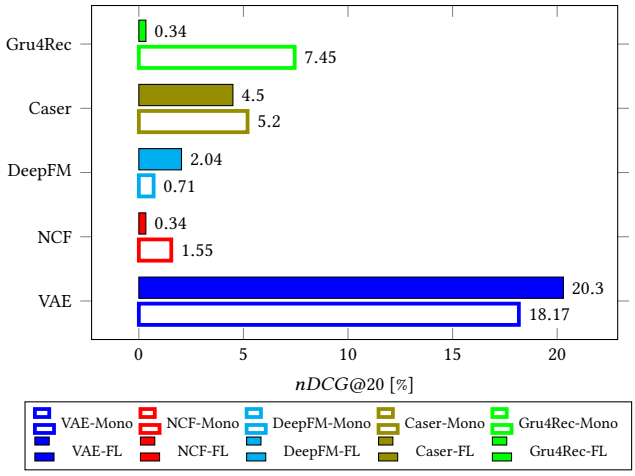

Figure 5: Accuracy of the implemented algorithms in the monolithic and federated setting. The outlined bar represents its monolithic version, and the filled bar represents its federated version.

The experimental results suggest that training *VAE*s using Federated Learning yields better performance than traditional monolithic

Name		Weights and Biases	Model Size	Required RAM	Learn rate	Batch size	Epochs	*FL* rounds
VAE	mono	8,425	36.4 KByte	4,010 MByte	0.00001	500	199	-
	federated				0.00001	126	2	15
NCF	mono	15,404,897	59 MByte	1,250 MByte	0.00001	500	50	-
	federated				0.00001	126	2	8
DeepFM	mono	1,513,803	6.3 MByte	6,700 MByte	0.00001	256	100	-
	federated				0.00001	126	2	10
Caser	mono	2,090,711	8.4 MByte	3,550 MByte	0.001	512	21	-
	federated				0.001	126	2	10
GRU4Rec	mono	1,277,006	5.1 MByte	4,000 MByte	0.0001	100	98	-
	federated				0.0001	126	2	12
BERT4Rec	mono	1,087,507	4.5 MByte	50,000 MByte	-	-	-	-
	federated				-	-	-	-

Table 3: Values of the models' hyperparameters for their monolithic and federated training and their required memory.

training on a data set with a highly imbalanced distribution of data points. The concentration of data points on a small subset of the data set makes reproducing the desired outputs easier for the *VAE*s. With FL, where data is distributed across nodes, the *VAE*s may experience a smaller shift in learning within the nodes. This shift could potentially be further reduced through aggregation, leading to improved results. This observation supports the hypothesis that *FL* can enhance *VAE* training on imbalanced data sets.

6.4 Long-Tail-Coverage

Figure 6 shows the monolithic implementations' long tail coverage *LLC*@20 compared to its federated counterpart running on 128 nodes. The uneven distribution of data, with a significant emphasis on the head, poses a challenge for models aiming to achieve comprehensive coverage. While VAE and Casser excel in capturing patterns within the head, their limited attention to the tail hinders their ability to make accurate predictions in that region. On the other hand, alternative architectures that exhibit higher LLC address this limitation by generating diverse predictions throughout the data distribution. Nevertheless, the broader coverage achieved by these architectures comes at the expense of decreased accuracy, as the models may struggle to precisely capture the more frequent patterns.

6.5 Recommendation time on embedded devices

Figure 7 shows the needed time to calculate a single recommendation for each algorithm. We deployed the pre-trained models each on Raspberry Pi version 4 and version 3 computing modules to simulate the small computing devices in existing *IFE*-systems. Those devices have the same ARM architecture and similar computing capabilities and available memory as the computing modules. To avoid the impacts of parallel running processes on the system, 10000 recommendations were generated, and the average time needed for a single recommendation gives the presented value. DeepFM is not included in this graph, as its default architecture only predicts a single item's usefulness and does not give a likelihood of preference over all available items in parallel. While on a Pi 4, a single recommendation already needed about 13 seconds, the hardware watchdog of the Pi 3 canceled the process because of DeepFMs

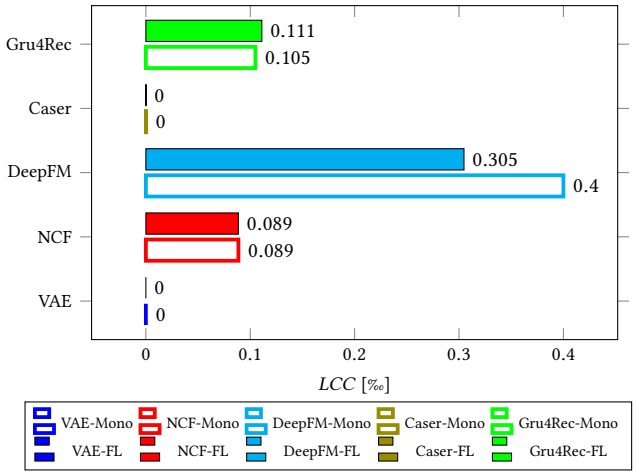

Figure 6: Long-tail coverage in ‰of the implemented algorithms. Hereby denote the outlined colored bar the algorithms' accuracy in its monolithic version and the filled bar for its achieved accuracy as a federated version.

high computing demands. Bert4Rec could not be deployed, as its requirements are a magnitude higher than the available resources. It was expected that the more modern Pi 4 has a shorter calculation time for a single recommendation, and for all investigated algorithms, it outperforms the older model by about 20 % . Naturally, the *RNN* based Gru4Rec needs the highest computation time, while the simple VAE outperforms the other models.

7 CONCLUSION

In conclusion, this study provides insights into the transferability of monolithically designed *DL*-based *RS*s into the federated setting. The results indicate that the success of such a transfer is highly dependent on the characteristics of the data set used.

Unexpectedly, the VAE architecture demonstrated exceptional performance in the federated setting. However, *SBRS*s, particularly the *GRU*-based system, encountered challenges related to data distribution when it came to federated learning. On the other hand,

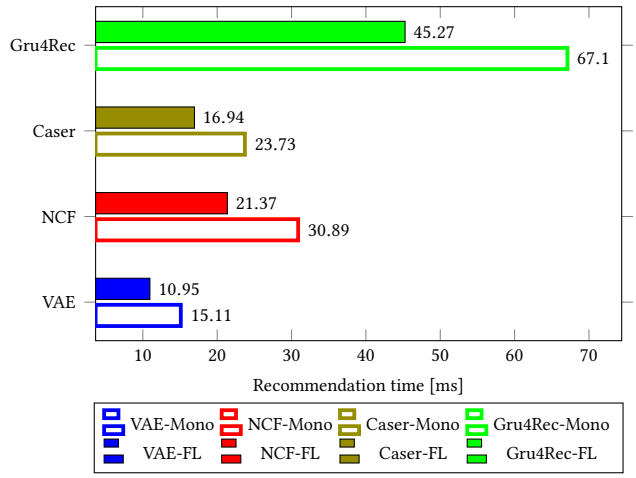

Figure 7: Average time needed to generate a single recommendation of the implemented algorithms. Hereby denote the outlined bar for the algorithms' average time on a RaspberryPi 3 and the filled bar for its time on a RaspberryPi 4.

the Caser architecture experienced only minor difficulties during the federation step.

With its architectural focus on isolating and prioritizing the user-centric distribution of data, *FL* holds the potential for improving the diversity of recommendations, especially for tail-related recommendations. The investigated architectures have also shown the ability to run on small embedded devices, such as those found in *IFE*- solutions while maintaining a satisfactory user experience. Unfortunately, the size of transformers still poses a limitation on their feasibility for federated learning in embedded systems.

The advantages of federated learning, particularly regarding data privacy and scalability, make it a promising approach for recommender systems. However, it is crucial to conduct comprehensive investigations using diverse data sets to ensure unbiased conclusions and mitigate potential biases associated with the specific data set used in this study.

ACKNOWLEDGMENTS

Research for this paper was partially funded by the German Aerospace Center (DLR) project CANARIA under project number 20D1931E. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial or non-financial interest in the subject matter or materials discussed in this manuscript.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2018. Popularity-Aware Item Weighting for Long-Tail Recommendation. https://doi.org/10.48550/arXiv.1802.05382

[2] Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. 1999. Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*. ACM, New York, NY, USA, 201–212. https://doi.org/10.1145/312129.312230

[3] Zareen Alamgir, Farwa K. Khan, and Saira Karim. 2022. Federated recommenders: methods, challenges and future. *Cluster Computing* 25, 6 (June 2022), 4075–4096. https://doi.org/10.1007/s10586-022-03644-w

[4] Chris Anderson. 2006. *The long tail: why the future of business is selling less of more* (1st ed ed.). Hyperion, New York.

[5] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. 2007. KDD Cup and workshop 2007. *ACM SIGKDD Explorations Newsletter* 9, 2 (Dec. 2007), 51–52. https://doi.org/10.1145/1345448.1345459

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. https://doi.org/10.48550/arXiv.1812.01097 arXiv:1812.01097 [cs, stat].

[7] Pedro G. Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware Recommender Systems: A Comprehensive Survey and Analysis of Existing Evaluation Protocols. *User Modeling and User-Adapted Interaction* 24, 1-2 (Feb. 2014), 67–119. https://doi.org/10.1007/s11257-012-9136-x

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. https://doi.org/10.48550/arXiv.1406.1078

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805

[10] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems (RecSys '10)*. Association for Computing Machinery, New York, NY, USA, 257–260. https://doi.org/10.1145/1864708.1864761

[11] Asela Gunawardana, Guy Shani, and Sivan Yogev. 2022. Evaluating Recommender Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). Springer US, New York, NY, 547–601. https://doi.org/10.1007/978-1-0716-2197-4_15

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, Melbourne, Australia, 1725–1731.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[14] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 5–53.

[15] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, Torino Italy, 843–852. https://doi.org/10.1145/3269206.3271761

[16] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. https://doi.org/10.48550/arXiv.1312.6114

[17] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery* 18, 1 (Feb. 2009), 140–181. https://doi.org/10.1007/s10618-008-0114-1

[18] Yehuda Koren and Robert Bell. 2015. Advances in Collaborative Filtering. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). Springer US, Boston, MA, 77–118. https://doi.org/10.1007/978-1-4899-7637-6_3

[19] Max Kuhn and Kjell Johnson. 2013. *Applied Predictive Modeling*. Springer New York, New York, NY.

[20] Fan Lai, Yinwei Dai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. FedScale: Benchmarking Model and System Performance of Federated Learning. In *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*. ACM, Virtual Event Germany, 1–3. https://doi.org/10.1145/3477114.3488760

[21] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2023. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (April 2023), 3347–3366. https://doi.org/10.1109/TKDE.2021.3124599

[22] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[23] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 31, 1 (March 2021), 149–181. https://doi.org/10.1007/s11257-020-09277-1

[24] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, Mark Purcell, Ambrish Rawat, Tran Minh, Naoise Holohan, Supriyo Chakraborty,

Shalisha Whitherspoon, Dean Steuer, Laura Wynter, Hifaz Hassan, Sean Laguna, Mikhail Yurochkin, Mayank Agarwal, Ebube Chuba, and Annie Abay. 2020. IBM Federated Learning: an Enterprise Framework White Paper V0.1. https://doi.org/10.48550/arXiv.2007.10987 arXiv:2007.10987 [cs].

[25] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. https://doi.org/10.48550/arXiv.1602.05629

[26] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. Association for Computing Machinery, New York, NY, USA, 1097–1101. https://doi.org/10.1145/1125451.1125659

[27] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (Oct. 2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[28] Yoon-Joo Park and Alexander Tuzhilin. 2008. The Long Tail of Recommender Systems and How to Leverage It. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)*. ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/1454008.1454012

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

[30] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. Ieee, Sydney, Australia, 995–1000. https://doi.org/10.1109/icdm.2010.127

[31] David E. Rumelhart and James L. McClelland. 1987. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MIT Press, Cambridge, 318–362.

[32] Gerard Salton. 1988. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, Reading, Mass.

[33] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112. https://doi.org/10.1145/2740908.2742726

[34] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450. https://doi.org/10.1145/3357384.3357895

[35] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 565–573. https://doi.org/10.1145/3159652.3159656

[36] Saúl Vargas and Pablo Castells. 2011. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 109–116. https://doi.org/10.1145/2043932.2043955

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. https://doi.org/10.48550/arXiv.1706.03762 arXiv:1706.03762 [cs].

[38] Paul Voigt and Axel von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR)*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-57959-7

[39] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2022. A Survey on Session-based Recommender Systems. *Comput. Surveys* 54, 7 (Sept. 2022), 1–38. https://doi.org/10.1145/3465401

[40] Yang Wang and Alfred Kobsa. 2006. Impacts of privacy laws and regulations on personalized systems. In *PEP06, CHI06 workshop on privacy-enhanced personalization*. Montreal, Canada, 44–46.

[41] Kun Xu, Weidong Zhang, and Zheng Yan. 2018. A privacy-preserving mobile application recommender system based on trust evaluation. *Journal of Computational Science* 26 (May 2018), 87–107. https://doi.org/10.1016/j.jocs.2018.04.001

[42] Liu Yang, Ben Tan, Vincent W. Zheng, Kai Chen, and Qiang Yang. 2020. Federated Recommendation Systems. In *Federated Learning: Privacy and Incentive*, Qiang Yang, Lixin Fan, and Han Yu (Eds.). Springer International Publishing, Cham, 225–239. https://doi.org/10.1007/978-3-030-63076-8_16

[43] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (Jan. 2019), 12:1–12:19. https://doi.org/10.1145/3298981

[44] Eva Zangerle and Christine Bauer. 2022. Evaluating Recommender Systems: Survey and Framework. *Comput. Surveys* 55, 8 (2022), 170:1–170:38. https://doi.org/10.1145/3556536

Received 09 June 2023