Federated against the Cold: A Trust-based Federated Learning Approach to Counter the Cold Start Problem in Recommendation Systems

Article *in* Information Sciences · April 2022

DOI: 10.1016/j.ins.2022.04.027

CITATIONS
46

READS
274

4 authors, including:

Omar Abdel Wahab
Polytechnique Montréal
82 PUBLICATIONS **2,067** CITATIONS

SEE PROFILE

Gaith Rjoub
Concordia University Montreal
30 PUBLICATIONS **485** CITATIONS

SEE PROFILE

Jamal Bentahar
Concordia University Montreal
276 PUBLICATIONS **3,824** CITATIONS

SEE PROFILE

Federated against the Cold: A Trust-based Federated Learning Approach to Counter the Cold Start Problem in Recommendation Systems

Omar Abdel Wahab[a,1,*], Gaith Rjoub[b,2], Jamal Bentahar[b,3], Robin Cohen[c,4]

[a]*Department of Computer Science and Engineering, Université du Québec en Outaouais, Gatineau, Canada*
[b]*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada*
[c]*David R. Cheriton School of Computer Science, University of Waterloo, Canada*

Abstract

Recommendation systems are often challenged by the existence of cold-start items for which no previous rating is available. The standard content-based or collaborative-filtering recommendation approaches may address this problem by asking users to share their data with a central (cloud-based) server, which uses machine learning to predict appropriate ratings on such items. But users may be reluctant to have their (confidential) data shared. Federated learning has been lately capitalized on to address the privacy concerns by enabling an on-device distributed training of a single machine learning model. In this work, we propose a federated learning-based approach to address the item cold-start problem in recommendation systems. The originality of our solution compared to existing federated learning-based solutions comes from (1) applying federated learning specifically to the cold-start problem; (2) proposing a trust mechanism to derive trust scores for the potential recommenders, followed by a double deep Q learning scheduling approach that relies on the trust and energy levels of the recommenders to select the best candidates. Simulations on the *MovieLens 1M* and *Epinions* datasets suggest that our solution improves the accuracy of recommending cold-start items and reduces the RMSE, MAE and running time compared to five

*Corresponding author. Tel.: +1 819 595-3900 ext. 1927
[1]Email: omar.abdulwahab@uqo.ca
[2]Email: g_rjoub@encs.concordia.ca
[3]Email: bentahar@ciise.concordia.ca
[4]Email: rcohen@uwaterloo.ca

benchmark approaches.

Keywords: Federated Learning, Client Selection, Recommendation System, Cold-Start Problem, Deep Reinforcement Learning, Trust Establishment

1. Introduction

Recommender systems are proposed to assist users through offering them a set of personalized items/services[5] that match their preferences [4, 44]. The main idea of recommender systems is to analyze users' profiles and items' specifications to find the best matches. To do so, most recommendation approaches capitalize on data acquired from users to recommend suitable items. Such data could be obtained either explicitly through asking the user to provide ratings for the consumed items or implicitly through grouping users into clusters and deducing the user's preferences based on perceived similarity with other users [13]. However, such approaches are challenged by the existence of newly released items for which no rating about their previous performance/behavior is available. This makes it quite difficult to create encodings for such items and hence quite hard to match users' preferences with items' encodings. This problem is referred to as the item cold-start problem.

1.1. Problem Statement

The current literature on addressing the cold-start problem in recommender systems can be classified into two main categories, i.e., content-based and collaborative filtering [32]. Content-based recommendation systems capitalize on data provided by the user itself to offer a personalized list of items [2]. The data can be provided either in explicit manner using ratings or implicitly through analyzing the user's activity (e.g., clicks on links). Based on these data, the system creates a profile for each user, based on which a set of appropriate items are proposed. As more data about the user come to the recommender system, the system constantly becomes more accurate. This type of recommendation systems necessitates the manual creation of encodings for each item,

[5]For simplicity, we use in the rest of the paper the word *item* to refer to both items and services

which makes it quite hard to be applied when we have a large number of cold-start items that need to be recommended. On the other hand, collaborative filtering-based approaches analyze the relationships among items, users and user-item pairs without having to generate profiles for these items or users [30]. The collaborative filtering approach is hence based on collecting and analyzing data from a large number of users. Most existing collaborative filtering approaches assume that users will continuously share their own interactions (e.g., clicks, logs, etc.) to a centralized (cloud) server, which in its turn, trains a machine learning model on all collected interactions to derive appropriate recommendations. The collaborative filtering approach seems to be more appropriate for the cold-start problem than the content-based even in extreme cases wherein the items and users are both cold-start. The reason is that they do not need to create profiles for users nor encodings for items to be able to suggest recommendations.

The collaborative filtering approach is however hindered by essential privacy and network limitations. From the privacy perspective, users feel more and more worried about sending their private data to a third-party, especially with the strict legislations that are being crafter to protect individuals' data privacy such as the US Consumer Privacy Bill of Rights and the European Commission's General Data Protection Regulation (GDPR) [39]. From the networking aspect, sending raw data to cloud servers which are usually located far from the users entails high delays and high bandwidth and energy utilization at the users' device, which discourages these users from engaging in recommendation scenarios.

To overcome these limitations, we adopt in this work the emerging federated learning approach to address the cold-start items problem in recommendation systems [39]. The main idea of federated learning is to enable the execution of a single machine learning model in a distributed fashion over a number of edge devices. In the case of federated learning, the devices are not required to share their own data, but instead they perform an on-device training on their own data using model parameters shared by a coordinating server. Once done with the local training, users only share an updated version of the model parameters, where the sever collects the new versions from all participating users and performs some sort of aggregation to derive a new set of parameters to be used in the next iteration. This process repeats over many iterations until a

certain level of accuracy is attained. More specifically, we consider a scenario in which the recommenders are Internet of Things (IoT) devices (e.g., smart devices, wearable devices) that are asked to use their own resources and collected data to provide recommendations on some newly released items or services (e.g., movies, healtcare) [1, 35].

1.2. Contributions

We consider a scenario in which a set of recommenders are asked to provide recommendations on a newly released item. The considered recommenders are Internet of Things (IoT) devices that use their own resources to run data analytics on their own data to derive appropriate recommendations. The proposed solution consists of three phases: (1) trust establishment mechanism that capitalizes on devices utilization characteristics and recommendation credibility to derive trust scores for the potential recommenders; (2) double deep Q learning approach that considers recommenders' trust scores and their devices' available resources to make intelligent decisions as to which devices should participate in providing recommendations on the cold-start services; and (3) federated learning approach that enables recommenders to perform an on-device training without having to share their own data. Federated learning is well-suited to be used in our case since worker nodes in federated learning are end users' personal devices, meaning that the across workers data is highly heterogeneous. Embracing and capitalizing on this heterogeneity is of particular importance to improve the recommendations on cold-start items through covering a wider set of items of varying specifications and characteristics. The fact that the IoT devices run the machine learning locally on their own data means that the cold-start recommendation prediction task will be enriched with more data from different types of recommenders. This will help the machine learning model better generalize to a wider set of cold-start item scenarios. This is crucial in our environment as usually no (or very) little data are available on the cold-start items. Yet, it is important to mention that the data heterogeneity also entails serious challenges at the computation level as the data in such scenarios are usually non-Independent and Identically Distributed (non-IID). To counter this challenge, we employ in this work the *FedProx* [18] aggregation technique which is resilient to a large extent to data heterogeneity. In sum, what is especially novel about our work is

addressing cold start issues (with respect to items and services) within recommender systems using federated learning, and integrating some specific intelligent reasoning techniques for selecting the recommenders that will participate in this process. In summary, the main contributions of our work are the following:

- We propose a federated learning-based recommendation system that aims to provide accurate recommendations on cold-start items. To the best of our knowledge, our work is the first to investigate the application of federated learning to address the cold-start problem in recommendation systems.
- We design a trust establishment mechanism that enables the recommender system to establish trust relationships toward potential recommenders, while relying on resource utilization data from the devices along with credibility scores of the recommenders to derive the aggregate trust scores.
- We propose a recommender selection strategy that relies on double deep Q learning that capitalizes on devices' trust score and energy levels to select the subset of IoT devices whose participation in the recommendation of the cold-start services contributes in improving the accuracy of the recommendations.
- We demonstrate the value of federated learning for a specific new area of application, i.e., cold start item recommendation. We provide insights into how best to design intelligent algorithms that support the federated learning process for the considered problem.

1.3. Paper's Organization

In Section 2, we review the relevant literature and highlight the unique features of our solution. In Section 3, we provide the details of our solution and explain the trust establishment, trust aggregation, recommender selection and federated cold-start recommendation. In Section 4, we conduct and discuss simulation comparisons with the state-of-the-art. Finally, in Section 5, we conclude and provide the main insights of our work.

2. Related Work

In what follows, we survey the two main approaches that are used to address the cold-start problem, i.e., content-based and collaborative filtering approaches. We also discuss the federated learning approaches that addressed problems related to recommendation systems. Note that federated learning has not yet been used to address the cold-start problem.

2.1. Content-based Approaches

In [20], the authors design a meta-learning approach that makes recommendations for each new user using a stochastic process that maps the user's observed interactions to a predictive distribution. A task-adaptive mechanism is integrated into the solution to achieve a balance between the model's capacity and the adaptation reliability. The proposed mechanism allows the model to learn the relevance of tasks and personalize the global knowledge to the task-related parameters when deriving the user's preferences.

In [15], the authors propose to extract implicit information from newcomer users' media social stream to generate a behavioral profile for each user. Machine learning models (i.e., decision tree and random forests) are then applied to make predictions on the user's preferences.

The authors of [17] propose an ontology-based content recommendation system to address the cold-start problem. First, a domain ontology is created to save information about the recommenders, recommenders' log data and cold-start item. Then, the similarities among the recommenders are computed using the ontology's domain knowledge. Finally, a list of top n items is generated using a content-based recommendation technique.

In a previous work [37], we propose an endorsement-based bootstrapping solution for newcomer cloud services. The solution comprises a (1) mechanism to enable potential recommenders to self-asses their aptitude to provide accurate recommendation; (2) machine learning approach that computes the similarities between existing and cold-start services; and (3) credibility update method to assess the credibility of the recommenders.

This type of recommendation entails high complexity owing to the need to manually create encodings for each item. This makes it quite inefficient to be applied in large-scale scenarios in which a large number of items need to be recommended.

2.2. Collaborative Filtering Approaches

In [46], the authors design a hybrid recommendation approach that capitalizes on neural networks to extract user-item ratings for collaborative filtering. These ratings are used to further understand the complex structure of user-interaction ratings. The proposed approach leverages item embedding as well to catch the content feature and employs it as an auxiliary to addressing the cold-start problem.

In [23], a generalized recommendation approach to address the cold-start problem is proposed. The approach takes as inputs user demographics and item genre, and employs them along with the rating matrix to derive recommendations on cold-start items. The proposed approach combines Generalized Matrix Factorization (GMF) for the rating matrix, Multilayer Perceptron (MLP) for the metadata and Neural Matrix Factorization to aggregate the outputs from GMF and MLP to make the rating prediction.

In [42], the authors put forward a recommendation mechanism to address the cold-start problem. The proposed mechanism combines deep learning and collaborative filtering in the sense that it relies on deep learning to extract content features from content descriptions to be then used as the key item factor vectors for the recommendation model.

In [19], a three-level approach is proposed to tackle the problem of cold-start users. First, a Naive Bayes approach is adopted to allocate new users to existing clusters. Second, the authors propose an algorithm to discover the neighbors of new users, based on which a method for computing the similarity between each newcomer user's characteristics and those of its neighbors is put forward. Finally, the authors discuss a prediction technique to assess the final rating of the newcomer user for each existing item through computing the weighted sum of the ratings given by the user's set of neighbors on the underlying items.

The main limitation of collaborative filtering approaches is that their performance

degrades in extreme scenarios wherein newly registered users seek recommendations on newly deployed services. In such a case, these approaches fail to learn the preferences of the newly registered users and are hence unable to capitalize on the similarities between the specifications of existing and newly deployed items/users to generate appropriate recommendations. Another main limitation of the collaborative filtering approach is related to privacy concerns. In fact, most collaborative filtering approaches are based on a centralized approach in which users are supposed to share their data with a central server, which in its turn performs the machine learning on these data. This discourage users from participating in the recommendation process to protect the privacy of their data.

2.3. Federated Learning-based Recommendation Systems

We survey and explain hereafter the main approaches that employ federated learning to address issues related to recommender systems. A federated meta-learning approach is proposed in [6] to facilitate the process of building personalized recommendations. The main idea is to perform the information sharing among the participating clients at the algorithm level instead of doing it at the model or data levels. The recommendation algorithm is trained using a two-level meta-training process, where at each iteration a collection of tasks is sampled. The model is trained by the clients on a support set and then assessed on an independent query set to estimate the testing loss gradient. Then, the recommendation algorithm is modified with the test feedback obtained from clients using their local data. The authors of [16] propose a federated learning-based personalized recommendation system that consists of a global training phase and a local training phase. In the global training phase, edge devices carry out some steps of the Stochastic Gradient Descent (SGD) method and then forward their parameter vector to the central cloud server. This latter aggregates the parameter vectors and sends the aggregated version back to the edge devices, which then capitalize on this aggregate vector to reinitialize the model parameters for the next steps of the SGD. Once a certain predetermined number of iterations has been attained, edge devices end their communication with the central server and start the local training phase. In this phase, edge devices keep on training their models locally over several iterations

in order to fine-tune the global parameter vector (trained by all devices) to fit each particular device's local data. In [14], the authors compare the performance of federated learning with that of gossip learning in a recommendation system scenario that is based on low-rank matrix decomposition. Gossip learning is a fully distributed learning approach wherein no central parameter server is needed. Instead, clients directly exchange and aggregate the model. The authors report that, based on their considered scenario and parameters, gossip learning achieves comparable performance to federated learning and outperforms it in scenarios wherein compression techniques are used to reduce the communication overhead.

Although federated learning has been already investigated in the context of recommendation systems, no existing work has yet studied federated learning to address the cold-start problem. Moreover, the existing federated learning-based approaches do not embed any technique to intelligently select the users that will participate in the on-device training of the federated learning model.

2.4. Client Selection in Federated Learning

Plenty of client selection and scheduling approaches have been proposed for federated learning [27, 26, 25, 29, 9, 38] The authors in [21] propose a reputation-based client selection for federated learning. To compute the reputation scores, three metrics are used: (1) difference between the accuracy of each device and the average test accuracy of local models in each epoch; (2) accuracy difference between the temporary global model and each client's local model; and (3) accuracy difference between the previous global model from previous epoch and each client's local model. In [24], the authors design an efficient technique called *FedFast* for boosting the speed of training recommendation systems in federated learning settings. The solution is composed of two parts, i.e., (1) *ActvSAMP* that aims to improve the selection of participating users through grouping them based on their similarities, and (2) *ActvAGG* which aims to aggregate the local models in a faster fashion. In [43], a client selection mechanism called *FedCS* is proposed in device-heterogeneous federated learning environments. The mechanism aims to accelerate the training convergence by considering the resource availabilities on the devices during the selection. In [40], the authors design

a resource allocation mechanism in wireless network-based federated learning environments which aims to ensure that the learning rounds are temporally interdependent and have different significance towards the final learning result. Thereafter, a stochastic optimization problem is designed to jointly achieve client selection and bandwidth allocation for energy-constrained clients. Departing from the observation that clients with different degrees of non-IID data result in heterogeneous weight divergence, the authors in [45] propose a client selection mechanism called *CSFedAvg* which favors the clients with weaker degree of non-IID data.

The originality of our solution compared to existing client selection mechanisms in federated learning is that we base the selection of recommenders on the trust and energy levels of the potential recommenders. Moreover, in our trust modeling approach, we take into consideration both the device perspective characteristics perspective and recommender credibility perspective to derive trust scores for the potential recommenders, followed by a technique to aggregate these two sources of trust. This is important to ensure fairness in the trust computation process and to avoid having biased decisions. Moreover, we would like to clarify that we tune the DDQN model to our problem through modeling the reward as a function of trust level of recommenders, energy levels of the devices and bandwidth available on the devices, all of which are important factors to guarantee accurate and timely recommendations on cold-start items.

3. Federated Learning-based Cold-Start Item Recommendation

In the following, we describe the details of our solution and explain its different phases. Also, we depict in Fig. 1 the different phases of the solution and the relationships among them and define in Table 1 the different notations that are used throughout the paper.

3.1. Trust Establishment

Trust is a fundamental concept towards secure communicative complex systems [7, 8]. We explain in the following the trust establishment mechanism for the potential recommenders which is divided in two phases, i.e., device perspective and recom-

Table 1: Notations

Symbol		Meaning
I	:	Set of IoT devices.
t	:	Current time moment.
M	:	Set representing the resource metrics, i.e., $M = \{CPU, energy, bandwidth\}$.
ν^m	:	Array recording the amounts consumed by a collection of IoT devices in terms of resource metric $m \in M$ to execute federated learning tasks having similar resource requirements in the time interval $[t-\epsilon, t]$.
ω_i^m	:	Array recording the utilizations of IoT device i in terms of metric $m \in M$ during the time interval $[t-\epsilon, t]$.
$Q1^m$	:	The point in the dataset containing utilization data on metric $m \in M$, under which 25% of the values fall.
$Q2^m$	:	The median of the dataset containing utilization data on metric $m \in M$.
$Q3^m$	:	The point in the dataset containing utilization data on metric $m \in M$, above which 25% of the values are.
$OverCons_i^m$	:	Sum of IoT device i's over-consumption in terms of resource metric $m \in M$ (initialized to 0).
$FrequencyOver_i^m$	:	Counter that records the number of times IoT device i has over-consumed the resource metric $m \in M$ (initialized to 0).
$UnderCons_i^m$	:	Sum of IoT device i's under-consumption in terms of resource metric $m \in M$ (initialized to 0).
$FrequencyUnder_i^m$	:	Counter that records the number of times IoT device i has under-consumed the resource metric $m \in M$ (initialized to 0).
$AvgFrequencyOver_i^m$	:	The average over-consumption of IoT device i in terms of resource metric $m \in M$.
$AvgFrequencyUnder_i^m$	:	The average under-consumption of IoT device i in terms of resource metric $m \in M$.
$OverConsProp_i^m$	:	The over-consumption of IoT device i in terms of resource metric $m \in M$, proportionally to the upper consumption limit of m.
$UnderConsProp_i^m$	:	The under-consumption of IoT device i in terms of resource metric $m \in M$, proportionally to the lower consumption limit of m.
$\lvert OverConsMetrics_i \rvert$	:	The number of metrics over-consumed by IoT device i.
$\lvert UnderConsMetrics_i \rvert$	:	The number of metrics under-consumed by IoT device i.
τ_i	:	Initial objective belief of the server in IoT device i's trustworthiness.
R	:	Set of recommenders.
C	:	Set of cold-start items.
$\chi_{r,c}$	:	Recommendation given by recommender $r \in R$ on item $c \in C$.
$\overline{\chi_r}$	:	The average of the recommendations given by recommender $r \in R$ on the different items.
ζ_r	:	The credibility score of recommender r.
$\chi_{Avg,c}$	:	Average recommendation of all recommenders on item c.
$\overline{\chi_{Avg}}$	:	Average recommendation of all recommenders in the community over all items.
$Trust(r)$	:	Final aggregate trust score for recommender r.
S	:	Set of global states in the Double Deep Q Network.
A	:	Set of joint actions of the IoT devices in the Double Deep Q Network.
T	:	Transition probability function in the Double Deep Q Network.
ς_i	:	Individual reward function of IoT device i.
κ	:	The global reward function of the Double Deep Q Network over all IoT devices.
γ	:	Discount factor that ensures assigning higher weight to the most recent rewards.
$\vartheta(i)$	:	Amount of energy available on IoT device i.
ω_i	:	A binary variable that accepts two values, i.e., 1 meaning that the server assigns a cold-start recommendation task to the IoT device i and 0 meaning that no cold-start recommendation task is assigned to device i.
ϖ_i	:	Amount of energy that IoT device i needs to download the global federated learning model from the server, train the model locally and upload the updated model back to the server.
θ_i	:	Bandwidth cost of transmitting the federated learning model from the server to IoT device i.
θ^{max}	:	Maximum bandwidth cost that could be utilized to transmit the federated learning model from the server to an IoT device.
Φ^*	:	The optimal policy in the Double Deep Q Network in terms of actions to be taken at each state to maximize the cumulative reward κ.
$Q(s,a)$	:	The expected future discounted reward that results from taking action $a \in A$ in a certain state $s \in S$.
φ_l	:	Network parameters of the online Double Q Learning model at the l^{th} iteration.
φ'_l	:	Network parameters of the target Double Q Learning model at the l^{th} iteration.
E	:	Number of local training epochs in the federated learning model.
ρ	:	Desired level ρ of federated learning model accuracy.
$\Im$	:	Total number of time moments over which the credibility scores of recommender r are computed.

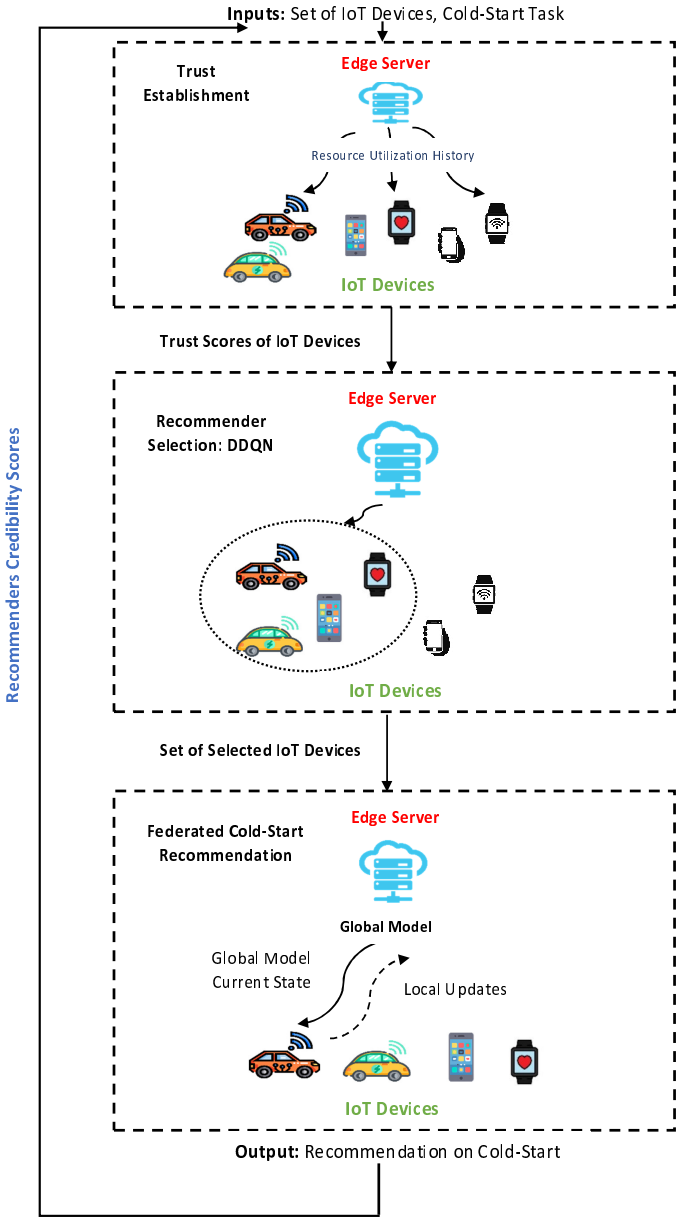


Figure 1: Diagram of our solution

mender perspective. We then present the Bayesian inference method to aggregate the trust scores from both phases.

3.1.1. Device Perspective

In this phase, we aim to build trust relationships between the server and the IoT devices that are eligible to participate in recommending a cold-start service. To do so, we propose a monitoring-based trust establishment strategy that enables the recommendation system (edge server) to monitor the CPU, energy and bandwidth consumption of the devices across time to identify those devices that over- or under-utilize their resources during the federated training process [36]. The objective is to detect those devices that (1) over-utilize their resources to include some malicious steps into the their local optimization problem to delude the classification results and/or (2) do not devote the necessary resources to perform the local training (i.e., under-utilize their resources).

The detailed steps of the monitoring-based trust establishment process are illustrated in Algorithm 1. The input to the Algorithm is an array containing the average CPU, energy and bandwidth consumption over a set of IoT devices that participated in the training of a set of cold-start recommendation tasks that share the same range of resource requirements. The server applies the Interquartile Range (IQR) statistical method on the array to identify the abnormal utilization bounds. The main idea of the IQR method is to split a given set of data into disjoint quartiles (i.e., $Q1$, $Q2$, $Q3$). The first quartile $Q1$ represents to the point in the data series under which 25% of the values fall. The second quartile $Q2$ represents the median of the data series. The third quartile $Q3$ stands for the point in the data series above which 25% fall. Then, the IQR measure is obtained by subtracting the first quartile from the third quartile. The server then adds the IQR to $Q3$ and multiplies the value by 1.5 to determine the upper consumption limit for each underlying metric (i.e., CPU, energy and bandwidth). Simply speaking, any value that is more than one and a half times above the upper quartile is considered to be an outlier based on Tukey analysis [33]. The upper consumption limit represents the pattern of maximal habitual utilization of the IoT devices for tasks of similar resource requirements. Similarly, the server subtracts the IQR from $Q1$ and multiplies the value

Algorithm 1: Trust Establishment on Recommenders

Input: t, M, ω_i^m

procedure MONITORINGBASEDTRUST

  for IoT device $i \in I$ **and** metric $m \in M$ **do**

    Repeat

      Sort the values in ν^m in ascending order

      Compute the median Q2^m of ν^m

      Compute the 25th percentile Q1^m as the median of ν^m's lower half

      Compute the 75th percentile Q3^m as the median of ν^m's upper half

      Compute the interquartile range: $IQR^m = Q3^m - Q1^m$

      Compute the upper consumption bound $U^m = Q3^m + IQR^m * 1.5$

      Compute the lower consumption bound $L^m = Q1^m - IQR^m * 1.5$

      for each data point $x_i^m \in \omega_i^m$ **do**

        if $x_i^m > U^m$ **then**

          $OverCons_i^m = OverCons_i^m + x^m$

          $FrequencyOver_i^m = FrequencyOver_i^m + 1$

        end if

        else if $x_i^m < L^m$ **then**

          $UnderCons_i^m = UnderCons_i^m + x_i^m$

          $FrequencyUnder_i^m = FrequencyUnder_i^m + 1$

        end if

      end for

      if $FrequencyOver_i^m > 0$ **then**

        $AvgFrequencyOver_i^m = OverCons_i^m / FrequencyOver_i^m$

        $OverConsProp_i^m = U^m / AvgFrequencyOver_i^m$

        $|OverConsMetrics_i| = |OverConsMetrics_i| + 1$

      end if

      if $FrequencyUnder_i^m > 0$ **then**

        $AvgFrequencyUnder_i^m = UnderCons_i^m / FrequencyUnder_i^m$

        $UnderConsProp_i^m = L^m / AvgFrequencyUnder_i^m$

        $|UnderConsMetrics_i| = |UnderConsMetrics_i| + 1$

      end if

    end for

    if $|OverConsMetrics_i| = 0$ **and** $|UnderConsMetrics_i| = 0$ **then**

      $\tau_i = 1$

    else

      $\tau_i = \frac{\sum_{m \in M} OverConsProp_i^m + UnderConsProp_i^m}{|OverConsMetrics_i| + |UnderConsMetrics_i|}$

    end

  until ϵ **elapses**

  end for

end procedure

by 1.5 to determine the lower consumption limit for each underlying metric (i.e., CPU, energy and bandwidth). This limit represents the pattern of minimal habitual utilization of the IoT devices for tasks of similar resource requirements.

Having computed the upper and lower consumption bounds, the server then checks, for any future consumption of a certain IoT device, if there is any consumption pattern that exceeds/falls under the upper/lower consumption bounds for any task having resource requirements in the same range as the tasks for which the upper/lower consumption limits were computed. The abnormal (over and under-utilization) events are added each to a table that records each device's over/under-utilization incidents. Using these tables, the average over and under utilization are separately computed. The monitoring-based trust τ_i of the underlying IoT device i is then computed through dividing the sum of average abnormal (over and under) utilizations over all the metrics by the number of over-consumed and under-consumed metrics. In case no metric has been under-consumed and overconsumed, the initial belief in the device's trust is set to 1, which represents a full (initial) trust in that device. This whole process is periodically repeated after a certain period of time ϵ to constantly capture the dynamism in the device's performance and behavior.

3.1.2. Recommender Perspective

To complement the monitoring-based trust method described in Section 3.1.1, we propose here a credibility-based trust method in which we assess the credibility of the recommenders after having participated in providing an opinion on a cold-start service. This helps avoid biased trust decisions, especially when we have some devices that might exhibit some abnormal behavior (i.e, over-utilization or under-utilization) due to some out of control circumstances rather than some malicious behavior. The idea of our credibility-based trust method, which is inspired by [3], is that if a user's recommendation on a certain item is similar to the average of recommendations given by all the users on that item, the user's recommendation would be deemed credible; otherwise it would be considered non-credible. Formally, let $R = \{r_1, r_2, \ldots, r_{|R|}\}$ be the set of recommenders, and $C = \{c_1, c_2, \ldots, c_{|C|}\}$ be the set of cold-start items. Let $\chi_{r,c}$ be the recommendation given by recommender $r \in R$ on item $c \in C$ which

is a real number in the range [1,5]. Let also $\overline{\chi_r}$ be the average of the recommendations given by user r on the different items.

The credibility ζ_r of recommender r is computed using the absolute value of the Pearson correlation coefficient [3] as per Eq. (1).

$$\zeta_r = \alpha \times \left| \frac{\sum_{c=1}^{|C|} (\chi_{r,c} - \overline{\chi_r})(\chi_{Avg,c} - \overline{\chi_{Avg}})}{\sqrt{\sum_{c=1}^{|C|} (\chi_{r,c} - \overline{\chi_r})^2} \sqrt{\sum_{c=1}^{|C|} (\chi_{Avg,c} - \overline{\chi_{Avg}})^2}} \right| \tag{1}$$

where $\chi_{Avg,c}$ is the average recommendation of all recommenders on item c, $\overline{\chi_{Avg}}$ is the average recommendation of all recommenders in the community over all items, and α is a significance factor that is equal to 1 in case the number $|\chi_r|$ of recommendations provided by recommender r is higher than a threshold θ and $\frac{1}{|\chi_r|}$ otherwise. The idea here is to give the recommenders that already participated in a higher number of recommendation processes a higher chance of having a higher credibility score. The generated credibility score ζ_u is a value from the interval $[0, 1]$.

3.2. Trust Aggregation

To aggregate the monitoring-based and credibility-based trust mechanisms, we employ the Bayesian inference method.

$$Trust(r) = \sum_{t=1}^{\Im} \frac{\zeta_r^t + (\Im \times \tau_r)}{2 \times \Im} \tag{2}$$

where τ_r is the trust score obtained in Algorithm 1 from the monitoring-based trust approach, ζ_r^t is the credibility score of recommender r at time moment t, and $\Im$ is the total number of time moments over which the credibility scores of recommender r are computed.

3.3. Recommender Selection

To select the IoT devices that will participate in the giving recommendations on the cold-start services, we employ in this work a trust and energy-aware recommender selection mechanism that relies on a dynamic Double Deep Q Network (DDQN) approach. The mechanism relies on a Markov Decision Process (MDP)-based multi-layered neural network that is determined by the tuple $\langle S, A, T, R, \gamma \rangle$, where:

- S: Set of global states.
- A: Set of IoT devices' joint actions.
- T: Transition probability function: $T(s, a, s') = Pr(s'|s, a)$, where $s, s' \in S$ and $a \in A$.
- $\kappa : S \times A \times S \mapsto \mathbb{R}$: the reward function of the model.
- γ: a discount factor that ensures assigning higher weight to the most recent rewards.

Let I be the set of available IoT devices that are eligible to participate in a certain cold-start recommendation problem, i.e, devices that are (1) connected to a free wireless network, (2) idle and (3) reachable to a base station [39]. Let S_i represent the set of local states of a certain IoT device i. The global system's state is acquired through applying a Cartesian product of the local states of the different IoT devices, i.e., $S = \prod_{i \in I} S_i$.

Each local state $s_i \in S_i$ is as follows:

$$s_i = (Trust(i), \vartheta(i)); \quad Trust(i) \in [0, 1], \vartheta(i) \in \{0, 1, \ldots, \vartheta^{max}\} \tag{3}$$

where $Trust(i)$ is the trust score of the IoT device i computed as per (2) and $\vartheta(i)$ is the amount of energy available on i. The local action $a_i \in A$ of an IoT device i is defined as per Equation (4):

$$a_j = (\omega_i, \varpi_i, \theta_i); \quad \omega_i \in \{0, 1\}, \theta_i \in \mathbb{R} \tag{4}$$

where ω_i is a binary variable with $\omega_i = 1$ reflecting the fact that the server assigns a cold-start recommendation task to IoT device i and $\omega_i = 0$ meaning that no cold-start recommendation task is assigned to i, ϖ_i quantifying the amount of energy that the device i needs to download the global model from the server, train the model locally and upload the updated model back to the server, and θ_i quantifying the bandwidth cost of communicating the model from the server to device i. The global action space at the server is the joint action space of each device: $A = \prod_{i \in I} A_i$ with A_i being the set of local actions of device i.

An action is said to be feasible from one global state s to another s', if and only if:

$$\varpi_i(s'_i) \leq \vartheta_i(s_i) \quad \forall i \in I \tag{5}$$

with $\varpi_i(s'_i)$ being ϖ_i in the action leading to s'_i and $\vartheta_i(s_i)$ being ϑ_i in state s_i.

We are now ready to design the reward function κ. The objective of the reward function is to maximize the likelihood of selecting trustworthy devices that enjoy sufficient energy levels to participate in the recommendation process, and to minimize the communication cost θ_i, proportionally to a maximum cost threshold θ^{max}. Thus, we define, in Equation (6), each device i's reward function ς_i as a function of state $s_i \in S_i$ and action $a_i \in A_i$:

$$\varsigma_i(s_i, a_i) = \begin{cases} Trust(i).\vartheta(i) - \frac{\theta_i}{\theta^{max}}, & \text{if } \varpi_i \leq \vartheta(i). \\ -\frac{\theta_i}{\theta^{max}}, & \text{otherwise.} \end{cases} \tag{6}$$

The global reward at the server is simply the summation of all IoT devices' individual rewards, as per Equation (7).

$$\kappa(s, a) = \sum_{i \in I} \varsigma_i(s_i, a_i) \tag{7}$$

Given Equation (7), the server aims at finding the optimal policy $\Phi^* : S \rightarrow A$ that involves the actions that should be taken at each state to maximize the cumulative reward. Q-learning [41] is a model-free reinforcement learning algorithm that is used to find Φ^*. It consists of updating the Q-value of a state-action pair, $Q(s, a)$, which represents the expected future discounted reward that results from taking action a in a certain state s. The optimal action-value function is given by Equation (8).

$$Q^*(s, a) = \max_{\Phi} Q_{\Phi}(s, a) \tag{8}$$

This optimal action-value function can represented with the Bellman optimality equation [31] as per Equation (9):

$$Q^*(s, a) = \kappa(s, a) + \gamma \sum_{s' \in S} Pr(s'|s, a). \max_{a' \in A} Q^*(s', a') \tag{9}$$

The Q-learning approach is useful in scenarios consisting of small state and action spaces solely, but becomes highly complex when the number of participants augments, which might be the case in our scenario. Deep Q-learning [12] is an approach that integrates deep neural networks into Q-learning to address the limitation of Q-learning in large-scale environments. In Deep Q-learning, the deep neural network receives as input one of the online network's states and outputs the Q-values $Q(s, a; \sigma)$ of all possible actions, where σ is the weight matrix of the deep neural network. To get the approximate values $Q^*(s, a)$, the deep neural network gets trained using a series experiences.

We define in Equation (10) the loss function of the Deep Q-learning using the Bellman optimality equation.

$$L(\theta_i) = E[(\kappa(s, a) + \gamma \arg \max_{a' \in A} Q(s', a'; \varphi'_l) - Q(s, a; \varphi_l))^2] \tag{10}$$

with φ_l and φ'_l being the online network and target network's parameters at the l^{th} iteration, respectively and $E[.]$ representing the expected value. Note that the ϵ-greedy policy is employed to determine the action a, achieving a balance between exploration (working on improving the current knowledge of the server about each action, leading to long-term benefit) and exploitation (pushing the agent to select the greedy action that leads to the highest reward) [11].

Given that the max-Bellman operator [22], which is used in standard Q-learning and Deep Q-learning, relies on the same Q-values to select and to evaluate an action, there is a high probability that the operator would select overestimated values (resulting in overoptimistic estimates). To counter this problem, we capitalize on the Double Deep Q-network (DDQN) [34] approach to separate the action selection and evaluation processes. This is accomplished through using two independent deep neural networks with two different valuation functions, one online network with weight set φ and one target network with weight set φ'. The weight set of the online deep neural network is updated at each iteration, whereas the weight set of the target network remains unchanged to derive the greedy policy. The target function of the DDQN error is defined in Equation (11) [28].

$$T(s, a, s') = \kappa(s, a) + \gamma Q(s',\ \arg\max_{a' \in A} Q(s', a'; \varphi); \varphi') \tag{11}$$

To derive the optimal value $Q(s', a'; \varphi)$, the weight φ of the online network capitalizes on the next state s' to select an action, whereas the weight φ' of the target network capitalizes on the next state s' to evaluate the selected action. Thereafter, the weight set φ of the online network is updated using the stochastic gradient descent method in the light of the incurred loss.

3.4. Federated Cold-Start Recommendation

The federated recommendation process is composed of three phases, i.e., global training, local training and testing. In the global training phase, the recommender system initializes a vector consisting of a set of predefined weights and sends the vector to the subset of selected IoT devices as per Section 3.3. At each global training iteration, every IoT device performs, using the shared vector, the training on its own data and sends the updated model weights to the recommender system. This latter aggregates, using the Federated Averaging approach [5][6], the model weights collected from the different devices and sends back the aggregate parameter vector to the devices for the next training epoch. This process repeats until a certain desired accuracy level is attained. At this stage, the communication between the recommender system and IoT devices stops and the devices start the local training phase. During the local training, each device seeks to fine-tune the global model trained by a large number of devices in such a way to fit its own data. This is done through allowing each device to iteratively train the global model on its own data. Finally, in the testing phase, after a certain number of local training epochs, each IoT device tests its own fine-tuned model on a separate set of data.

[6]Other federated learning aggregation approaches can be easily integrated into our solution instead of the Federated Averaging.

Algorithm 2: Federated Recommendation

Input: Number of local training epochs E
Input: Desired level ρ of model accuracy
procedure FEDERATEDRECOMMENDATION
    The recommender system initializes the parameter vector
    The recommender system employs the proposed scheduling solution to select a subset $R' \subseteq R$ of recommenders
    GlobalTraining:
        Repeat
            for each recommender $r \in R'$ **do**
                r performs training on its own data
                r sends the updated parameter vector to the recommender system
            end for
            The recommender system aggregates the parameter vectors from the different recommenders
            The recommender system sends the aggregated parameter vector back to the recommenders
        Until ρ is attained

    LocalTraining
        for each local iteration $e = 1$ to E **do**
            for each recommender $r \in R'$ **do**
                r executes local training to using the global model to fine-tune the global parameter vector to fit its own data
            end for
    Testing
        for each recommender $r \in R'$ **do**
            Evaluate the fine-tuned model on a separate set of data
        end for
    return local model θ
end procedure

Table 2: Comparison Datasets

Dataset	**Description**
MovieLens 1M	1 million ratings from 6000 users, each of which having at least 20 ratings on 4000 movies, with a rating scale between 0 and 5.
Epinions	Large-scale Web community dataset consisting of $\approx 140,000$s services and items rated by $50,000$ users, making a total of $\approx$ $1660,000$ reviews..

4. Simulations

4.1. Simulation Setup

To carry out our experiments, we use two datasets. The first dataset is that of *MovieLens 1M* [7], which consists of 1 million ratings from 6000 users. Each user has at least 20 ratings on 4000 movies, with a rating scale between 0 and 5. The dataset records as well demographic information on the users such as gender, age and occupation. The data is non-IID and unbalanced, making it quite suitable to be applied in a federated learning setting. The second dataset is that of *Epinions*[8], which is a large-scale Web community enabling users to voice their opinions on a large set of services and items (e.g., movies). It consists of $\approx$ $140,000$ services and items rated by $50,000$ users, making a total of $\approx 1660,000$ reviews. Table 2 summarizes these two datasets that are used in our comparisons.

We partitioned the data across a set of 6000 IoT devices. For each of the devices, a percentage of 20% of its own ratings was selected to be in the test set; the rest constitute the training set. To mimic the cold-start scenario in the simulations, we follow the strategy proposed in [19]. Specifically, among the set of items in the used datasets, we set a number (varying from 100 to 5000) of these items as being registered while the rest are considered cold-start. We train a Deep Neural Network (DNN) in a federated fashion as per Algorithm 2. The DNN is composed of a simple three-layer neural network that comprises one embedding layer and two fully-connected layers. Each

[7] `https://grouplens.org/datasets/movielens/1m`

[8] http://www.epinions.com

Table 3: Comparison Approaches

Approach	**Description**
Decision Tree [37]	Endorsement-based bootstrapping solution for cold-start cloud services that employs decision tree to compute the similarities between existing and cold-start services.
Naive Bayes [19]	Naive Bayes approach that predicts the final rating of the newcomer user for each existing item through computing the weighted sum of the ratings given by the user's set of neighbors on the underlying items.
Traditional Machine Learning, e.g., [42]	Requires the devices to send their data to a central server on which the machine learning logic is executed.
Fed-MVMF [10]	Federated multi-view matrix factorization method that extends the federated learning framework to matrix factorization with multiple data sources is proposed, mainly to address the challenge of cold-start items.
Fed-SGD [16]	Federated learning-based personalized recommendation system in which the client devices use the Stochastic Gradient Descent (SGD) method to perform the local training.

layer has 50 neurons, followed by dropout and batch normalization techniques, and a nonlinear function. We employ the TensorFlow Federated (TFF) platform, which provides an open source framework for decentralized machine learning. TFF facilitates a variety of collaborative learning scenarios on a number of heterogeneous devices with different resource characteristics. We compare the performance of our solution with five existing approaches which are summarized in Table 3.

4.2. Simulation Results

In Fig. 2, we compare the performance of our solution with the decision tree [37] and naive Bayes [19] approaches in terms of Mean Absolute Error (MAE) while varying the number of recommenders from 100 to 1000. The three approaches have been trained based on the federated learning approach. MAE quantifies the mean of the absolute values of each prediction error on all the observations of the test set, with prediction error being the difference between the predicted value and actual value for that observation. We observe from the figure that our solution yields the least MEA compared to the two other approaches. The reason is that we embed into our solution a trust and resource-aware recommender selection mechanism to intelligently select the

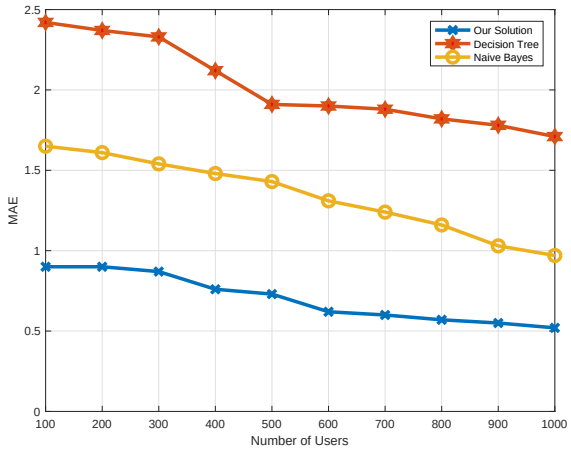


Figure 2: Our solution reduces the MAE compared to Naive Bayes [19] and decision tree [37] approaches

subset of recommenders whose participation improves the prediction efficiency. On the other hand, in the two other approaches, however well-performing the machine learning is, its performance can be easily burdened by some untrustworthy recommenders whose contributions might sway the prediction results. The second observation that can drawn from this figure is that increasing the number of recommenders has a positive impact on decreasing the MAE as having more recommenders leads to having more heterogeneity in the data, thus enabling the federated learning model to cover a wider set of cold-start scenarios.

In Fig. 3, we compare the performance of our solution with the decision tree and Naive Bayes approaches in terms of Root Mean Square Error (RMSE) while varying the number of recommenders from 100 to 1000. As in Fig. 2, the three approaches have been trained based on the federated learning approach. RMSE quantifies the standard deviation of the prediction errors and is used to show how far the predictions are from measured true values using the Euclidean distance. We notice from the figure that our solution yields the lowest RMSE compared to the two other approaches. As discussed in the context of MAE, the improvement brought by our solution in terms of RMSE is due to the trust and resource-aware recommender selection approach that our solution includes. Similarly, increasing the number of participants further contributes in decreasing the RMSE by enriching the federated learning model with more

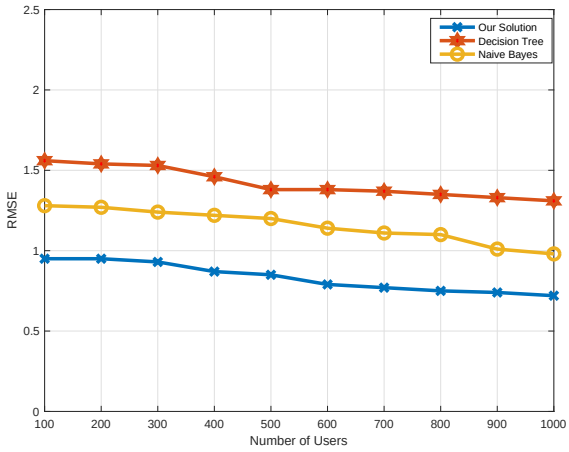
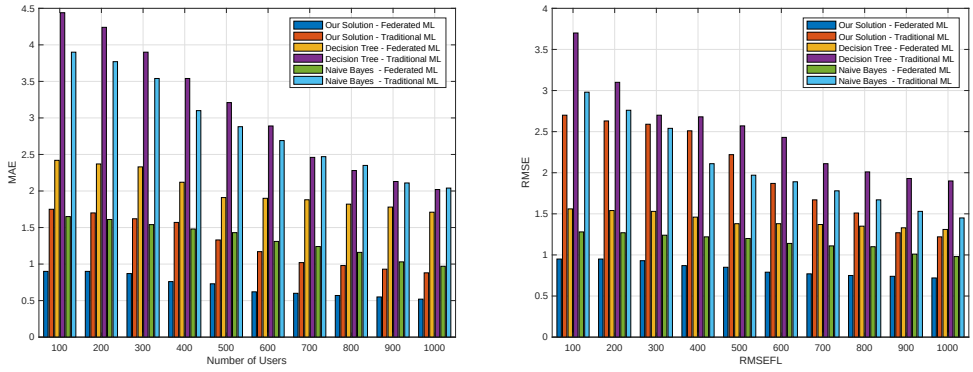


Figure 3: Our solution reduces the RMSE compared to Naive Bayes [19] and decision tree [37] approaches

heterogeneous data.

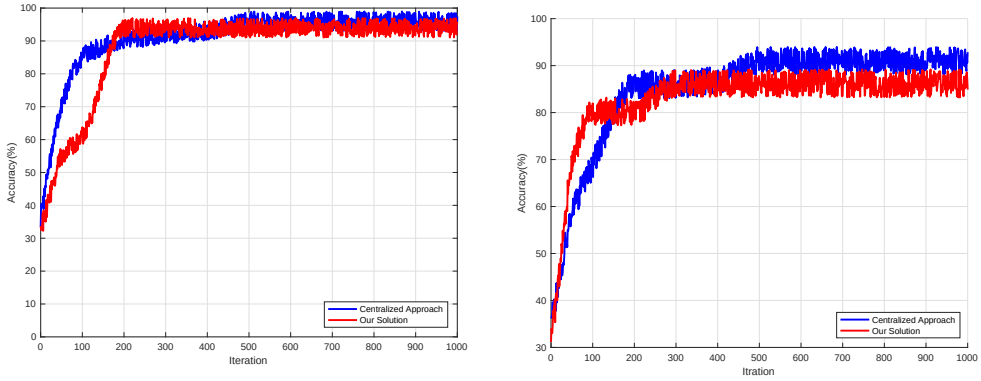
In Fig. 4, we compare the performance in terms of MAE (Fig. 4a) and RMSE (Fig. 4b) of our solution, the decision tree and Naive Bayes approaches under both the federated learning approach and traditional machine learning approach. The objective is to know whether or not adopting the federated learning approach has a positive impact on the accuracy of recommending cold-start services. In the traditional machine learning approach, data from the different recommenders are sent to a central server (i.e., recommender system) to be analyzed in a centralized fashion. On the other hand, in the federated learning approach, as explained earlier, the analysis is done on-device and only the learning parameters are shared between the devices and the server. We notice from Fig. 4a that the federated learning version of the different compared approaches outperforms the traditional machine learning version. The reason is that the federated learning approach enables access to a wider collection of data that are heterogeneous by nature. This is because users feel more confident to contribute in analyzing the data on their devices rather than sharing them with a third-party. Moreover, the federated learning approach that we adopt in our work (Algorithm 2) enables us to obtain personalized models by allowing each device to further fine-tune the global model based on their own data (local training phase).

In Fig. 5, we study the accuracy of both our solution and the centralized machine



(a) MAE (b) RMSE

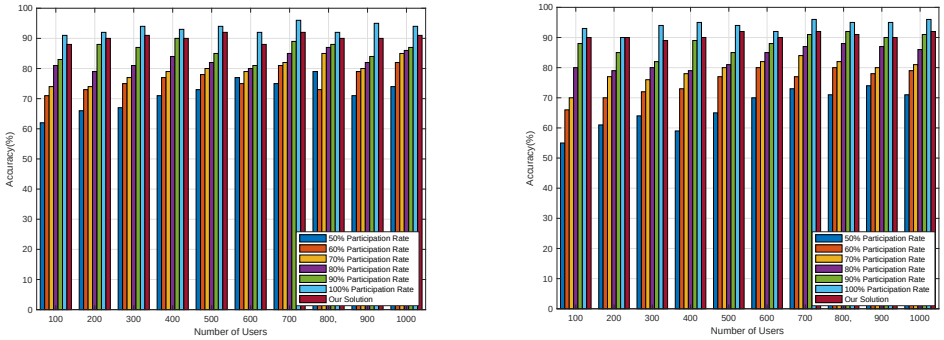
Figure 4: The federated learning approach outperforms the traditional machine learning approach through enriching the predictions with more heterogeneous data



(a) MovieLens (b) Epinions

Figure 5: Recommendation Accuracy: The centralized and federated learning approaches yield close accuracy levels on both the *MovieLens* and *Epinions* datasets with a slight advantage of the centralized approach.

learning approach. We compare both approaches on two different datasets, i.e., the *MovieLens* dataset (Fig. 5a) and the *Epinions* dataset (Fig. 5b). The objective is to generate results that are not biased toward any data type, size or features. Note that in this simulation, both compared approaches share the phases of trust establishment and recommender selection. They differ however in the way they run the machine learning logic to analyze recommenders' data. In our solution, the federated learning approach is adopted where the machine learning logic is executed in a cooperative fashion on the recommenders' devices. On the other hand, the centralized machine learning approach requires the devices to send their data to a central server on which the machine learning logic is executed. By observing Fig. 5, we notice that both compared approaches yield close accuracy levels on both datasets with a slight advantage of the centralized approach. This advantage is due to the fact that in this approach all the data is stored on a single powerful server whose computation power is usually more guaranteed compared to a set of distributed resource-constrained devices in the case of federated learning. However, this case is ideal and sometimes unrealistic for centralized machine learning, where in real-life this approach is challenged by many privacy and communication factors. In fact, users often have privacy concerns in terms of sharing their data with a third-party, especially in the light of the new regulations that are being issued by governments and authorities to support users' data privacy. Moreover, users' devices are usually small devices that are limited by their bandwidth capacity. This means that some users will abstain (either intentionally due to privacy concerns or unintentionally due to bandwidth limitations) from sharing their data with the server. To study this phenomenon and its impact on the centralized machine learning approach, we study in Fig. 6 the accuracy of our solution and that of the centralized machine learning approach while varying the user participation rate for the centralized approach on both the *MovieLens* (Fig. 6a) and *Epinions* datasets (Fig. 6b). By carefully examining the figure, we notice that the accuracy of the centralized approach starts to significantly drop even for a slight drop in the participation rate. For example, a drop of 10% in the participation rate leads to an accuracy drop of $\approx 10\%$ on the *MovieLens* dataset and $\approx 6\%$ on the *Epinions* dataset, making our solution to outperform the centralized approach.



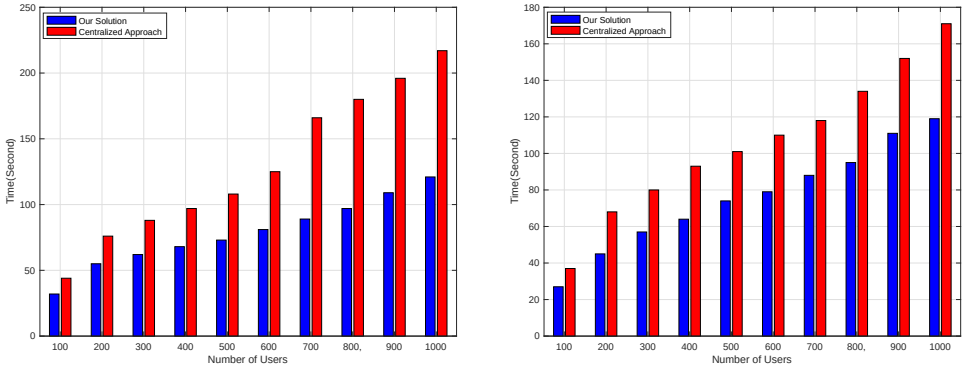
(a) *MovieLens* Dataset

(b) *Epinions* Dataset

Figure 6: Recommendation Accuracy: Our solution starts to outperform the centralized machine learning approach starting from 10% drop in the participation rate

In Fig. 7, we measure the running time of our solution against the centralized machine learning approach on the *MovieLens* (Fig. 7a) and *Epinions* (Fig. 7b) datasets, while varying the number of recommenders. We observe from the Figure that our solution reduces the running time by up to 70 seconds for $1,000$ participating users on the *MovieLens* dataset and by up to 45 seconds for the same number of participating users on the *Epinions* dataset, compared to the centralized approach on both datasets. The reason is that in the centralized approach, users' devices need to transmit large amounts of data to the sever at each communication round, which takes long time and slows down the recommendation process. On the other hand, since no raw data need to be shared in our solution but instead the model updates only, our solution boosts the efficiency of the cold-start recommendation process. This is very important especially when it comes to recommending mission-critical services (e.g., healthcare services).

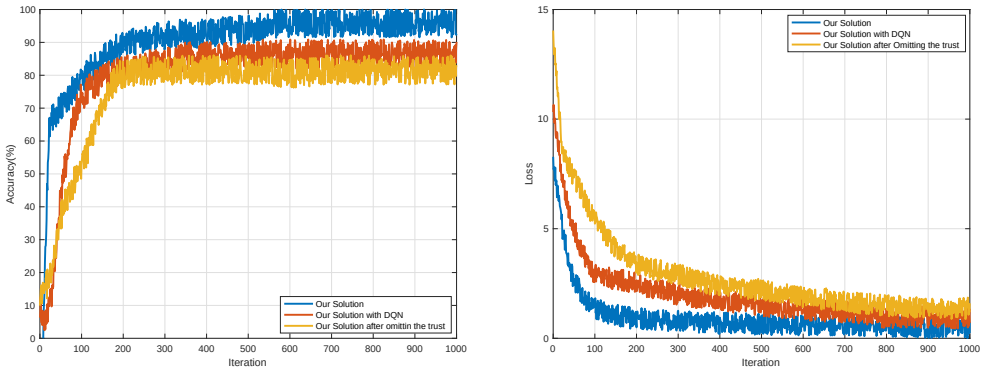
In Fig. 8, we aim to study the impacts of having the trust mechanism and DDQN scheduling approach on the performing of the overall solution. In the blue plot, we include all the components of our solution (i.e., the trust mechanism, the DDQN scheduling and the federated learning) and we measure the accuracy (Fig. 8a) and loss (Fig. 8b) of the solution over 1000 iterations. In the orange plot, we replace the DDQN with a Deep Q Network (DQN) approach. In the ywllow plot, we use DDQN but we omit the trust mechanism, which means that the selection of the recommenders is only



(a) MovieLens

(b) Epinions

Figure 7: Running Time: Our solution reduces the running time compared to the centralized machine learning approach as it required no raw data sharing with the server



(a) Accuracy entailed by our solution

(b) Loss entailed by our solution

Figure 8: We study in this figure how accuracy and loss react to the omission of some of our solution's phases

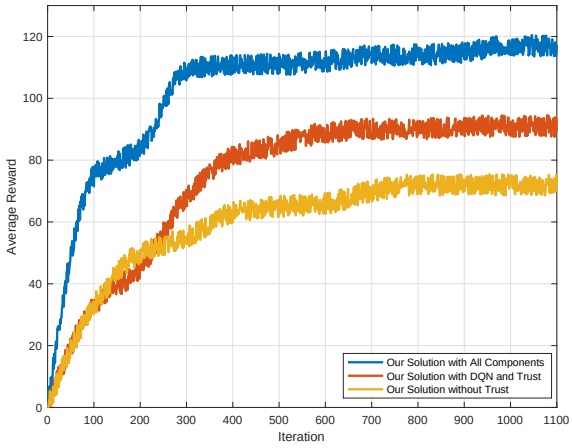
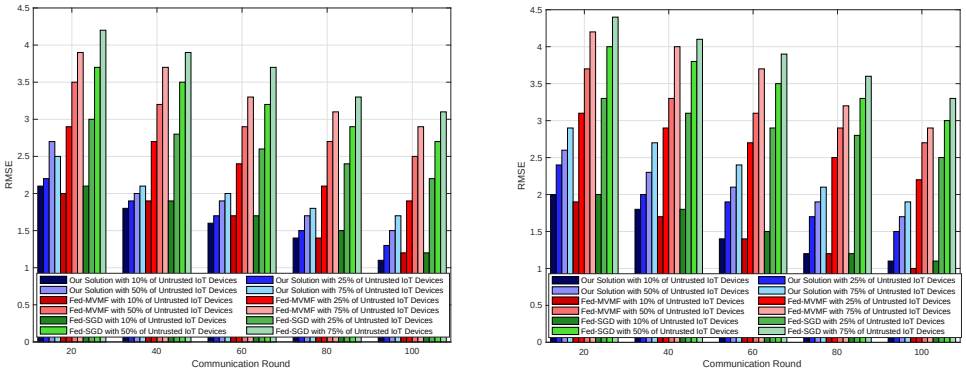


Figure 9: Average Reward: The DDQN and trust mechanism largely contribute in increasing the reward

based on the energy levels of the devices. By carefully observing the figure, we notice that our solution which combines both the DDQN scheduling and trust mechanism achieves the highest accuracy and smallest loss. Moreover, we notice that replacing the DDQN with the DQN has a bigger negative impact on the accuracy and loss compared to omitting the trust mechanism. More specifically, the DQN leads to some instability in the accuracy of the recommendations. DDQN can avoid this unstability thanks to its second Q-function approximator, which reduces the risk of obtaining overestimated values.

In Fig. 9, we measure the cumulative reward of our solution while (1) including all of its components; (2) replacing the DDQN scheduling with DQN; and (3) omitting the trust mechanism. We notice from the figure that our solution with both the DDQN and trust components yields the highest cumulative reward, followed by our solution without trust and then our solution with DQN. This shows the importance of the different components of our solution in enabling the recommender system to better learn the scheduling policy that best maximizes the reward.

In Figures 10, 11 and 12, we study the RMSE, accuracy and computation time, respectively, of our solution against two existing federated learning and cold-start-based recommendation solutions, i.e., Fed-SGD [16] and Fed-MVMF [10]. We conduct our study on both the *MovieLens* and *Epinions* datasets to better understand the generaliz-

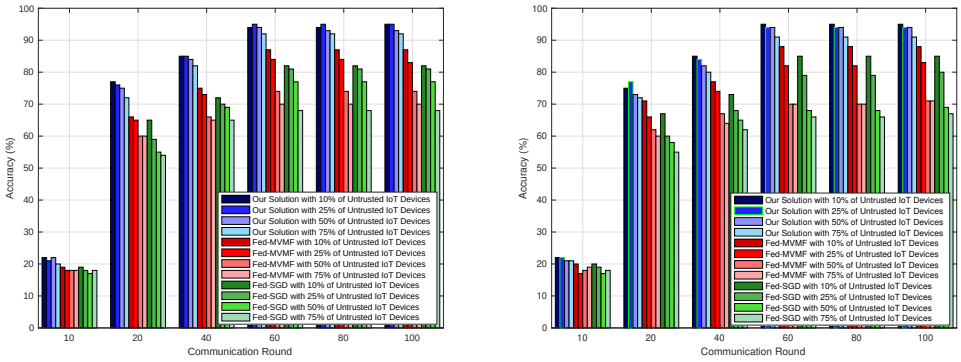


(a) Epinions

(b) MovieLens

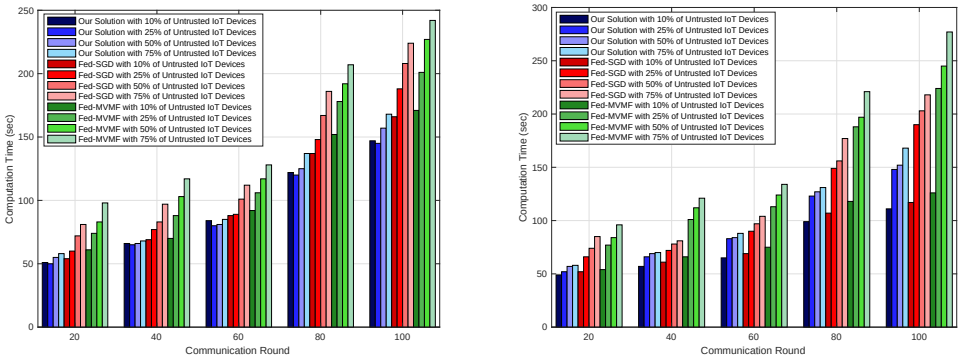
Figure 10: RMSE performance with respect to communication round on two datasets with varying the percent of untrusted IoT devices.

ability of the different solutions. In these Figures, we vary the number of communication rounds from 20 to 100 and percentage of untrusted IoT devices from 10% to 75%. Fed-SGD [16] capitalizes on federated learning to propose a personalized recommendation system in which the edge devices use the Stochastic Gradient Descent (SGD) method to perform the local training. In Fed-MVMF [10], a federated multi-view matrix factorization method that extends the federated learning framework to matrix factorization with multiple data sources is proposed, mainly to address the challenge of cold-start items. By carefully looking at the figures, we notice that our solution significantly reduces the RMSE and computation time and increases the accuracy on both the *Epinions* and *MovieLens* datasets compared to the two other approaches. Our solution also shows a considerable resilience to the increase in the percentage of untrusted IoT devices compared to the other approaches. This is because, unlike the two other approaches, our solution includes a mechanism to select only the trustworthy devices that can make a positive contribution to the federated recommendation process. Another observation that can be drawn from these figures is that the RMSE keeps decreasing and accuracy keeps increasing in all the studied approaches while the number of communication rounds increases. This is natural as having more communication rounds helps the federated learning model better improve the predictions and generalize on wider volumes of data.



(a) Epinions (b) MovieLens

Figure 11: FL accuracy with respect to communication round on two datasets with varying the percent of untrusted IoT devices.



(a) Epinions (b) MovieLens

Figure 12: Computation time with respect to communication round on two datasets with varying the percent of untrusted IoT devices.

5. Conclusion

In this work, we designed a federated learning-based approach to provide accurate recommendations on cold-start items. Our solution consists of a (1) trust mechanism that capitalizes on devices' resource utilization and recommendation credibility to derive trust scores on potential recommenders; (2) DDQN intelligent scheduling strategy that relies on recommenders' energy levels and trust scores to select the recommenders that will participate in the predictions; and (3) federated learning algorithm that allows recommenders to perform an on-device training of the machine learning model (to predict recommendations on cold-start items) without having to share their own data. Simulations conducted on the *MovieLens 1M* and *Epinions* datasets show that our solution outperforms, in terms of MAE, RMSE and accuracy five existing federated learning and cold-start recommendation approaches under different scenarios and settings.

Acknowledgments

This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, N. Kara, FoGMatch: an intelligent multi-criteria IoT-Fog scheduling approach using game theory, IEEE/ACM Transactions on Networking 28 (4) (2020) 1779–1789.

[2] M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendation, Communications of the ACM 40 (3) (1997) 66–72.

[3] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson Correlation Coefficient, in: Noise Reduction in Speech Processing, Springer, Berlin, Heidelberg, 1–4, 2009.

[4] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-based Systems 46 (2013) 109–132.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, J. Roselander, Towards Federated Learning at Scale: System Design, in: Proceedings of Machine Learning and Systems, vol. 1, 374–388, 2019.

[6] F. Chen, M. Luo, Z. Dong, Z. Li, X. He, Federated meta-learning with fast convergence and efficient communication, CoRR arXiv:1802.07876v2 (2019) [cs.LG].

[7] N. Drawel, J. Bentahar, A. Laarej, G. Rjoub, Formalizing Group and Propagated Trust in Multi-Agent Systems, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI), ijcai.org, 60–66, 2020.

[8] N. Drawel, H. Qu, J. Bentahar, E. M. Shakshuki, Specification and automatic verification of trust-based multi-agent systems, Future Gener. Comput. Syst. 107 (2020) 1047–1060.

[9] R. Estrada, R. Mizouni, H. Otrok, A. Mourad, Task coalition formation for Mobile CrowdSensing based on workers' routes preferences, Vehicular Communications 31 (2021) 100376.

[10] A. Flanagan, W. Oyomno, A. Grigorievskiy, K. E. Tan, S. A. Khan, M. Ammad-Ud-Din, Federated multi-view matrix factorization for personalized recommendations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 324–347, 2020.

[11] G. J. Gordon, Reinforcement Learning with Function Approximation Converges to a Region, in: Advances in Neural Information Processing Systems 13 (NIPS), Denver, CO, USA, MIT Press, 1040–1046, 2000.

[12] S. Gu, T. Lillicrap, I. Sutskever, S. Levine, Continuous deep q-learning with model-based acceleration, in: International Conference on Machine Learning, PMLR, 2829–2838, 2016.

[13] G. Guo, J. Zhang, N. Yorke-Smith, TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings, in: Proceed-

ings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI), Austin, Texas, USA, AAAI Press, 123–129, 2015.

[14] I. Hegedűs, G. Danner, M. Jelasity, Decentralized recommendation based on matrix factorization: a comparison of gossip and federated learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 317–332, 2019.

[15] J. Herce-Zelaya, C. Porcel, J. Bernabé-Moreno, A. Tejeda-Lorente, E. Herrera-Viedma, New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests, Information Sciences 536 (2020) 156–170.

[16] A. Jalalirad, M. Scavuzzo, C. Capota, M. Sprague, A simple and efficient federated recommender system, in: Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, 53–58, 2019.

[17] J. Joy, V. G. Renumol, An ontology-based hybrid e-learning content recommender system for alleviating the cold-start problem, Education and Information Technologies 26 (4) (2021) 4993–5022.

[18] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Processing Magazine 37 (3) (2020) 50–60.

[19] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Systems with Applications 41 (4) (2014) 2065–2073.

[20] X. Lin, J. Wu, C. Zhou, S. Pan, Y. Cao, B. Wang, Task-adaptive Neural Process for User Cold-Start Recommendation, in: WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, 2021, ACM / IW3C2, 1306–1316, 2021.

[21] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, A. Lawlor, Fedfast: Going beyond average for faster training of

federated recommender systems, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1234–1242, 2020.

[22] O. Nachum, M. Norouzi, K. Xu, D. Schuurmans, Bridging the Gap Between Value and Policy Based Reinforcement Learning, in: Advances in Neural Information Processing Systems 30 (NeurIPS), Long Beach, CA, USA, 2775–2785, 2017.

[23] R. Nahta, Y. K. Meena, D. Gopalani, G. S. Chauhan, Embedding metadata using deep collaborative filtering to address the cold start problem for the rating prediction task, Multim. Tools Appl. 80 (12) (2021) 18553–18581.

[24] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: The IEEE International Conference on Communications (ICC), IEEE, 1–7, 2019.

[25] G. Rjoub, J. Bentahar, O. Abdel Wahab, A. Saleh Bataineh, Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems, Concurrency and Computation: Practice and Experience 33 (23) (2021) e5919.

[26] G. Rjoub, J. Bentahar, O. A. Wahab, BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments, Future Generation Computer Systems 110 (2020) 1079–1097.

[27] G. Rjoub, J. Bentahar, O. A. Wahab, A. Bataineh, Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud, in: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 189–196, 2019.

[28] G. Rjoub, O. A. Wahab, J. Bentahar, A. Bataineh, A Trust and Energy-Aware Double Deep Reinforcement Learning Scheduling Strategy for Federated Learning on IoT Devices, in: International Conference on Service-Oriented Computing, Springer, 319–333, 2020.

[29] G. Rjoub, O. A. Wahab, J. Bentahar, A. S. Bataineh, Improving Autonomous Vehicles Safety in Snow Weather Using Federated YOLO CNN Learning, in: International Conference on Mobile Web and Intelligent Information Systems, Springer, 121–134, 2021.

[30] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, 285–295, 2001.

[31] M. Sniedovich, A new look at Bellman's principle of optimality, Journal of optimization theory and applications 49 (1) (1986) 161–176.

[32] P. B. Thorat, R. Goudar, S. Barve, Survey on collaborative filtering, content-based filtering and hybrid recommendation system, International Journal of Computer Applications 110 (4) (2015) 31–36.

[33] J. W. Tukey, Exploratory data analysis, Addison-Wesley series in behavioral sciences, Reading, Mass. : Addison-Wesley Pub. Co., 1977.

[34] H. van Hasselt, A. Guez, D. Silver, Deep Reinforcement Learning with Double Q-Learning, in: Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI), Phoenix, Arizona, USA, AAAI Press, 2094–2100, 2016.

[35] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, How to distribute the detection load among virtual machines to maximize the detection of distributed attacks in the cloud?, in: 2016 IEEE International Conference on Services Computing (SCC), IEEE, 316–323, 2016.

[36] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, Optimal Load Distribution for the Detection of VM-based DDoS Attacks in the Cloud, IEEE Transactions on Services Computing 13 (1) (2020) 114–129.

[37] O. A. Wahab, R. Cohen, J. Bentahar, H. Otrok, A. Mourad, G. Rjoub, An Endorsement-based Trust Bootstrapping Approach for Newcomer Cloud Services, Information Sciences 527 (2020) 159–175.

[38] O. A. Wahab, N. Kara, C. Edstrom, Y. Lemieux, MAPLE: A machine learning approach for efficient placement and adjustment of virtual network functions, Journal of Network and Computer Applications 142 (2019) 37–50.

[39] O. A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Communications Surveys & Tutorials 23 (2) (2021) 1342–1397.

[40] Y. Wang, B. Kantarci, A Novel Reputation-Aware Client Selection Scheme for Federated Learning within Mobile Environments, in: 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), IEEE, 1–6, 2020.

[41] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

[42] J. Wei, J. He, K. Chen, Y. Zhou, Z. Tang, Collaborative filtering and deep learning based recommendation system for cold start items, Expert Systems with Applications 69 (2017) 29–39.

[43] J. Xu, H. Wang, Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective, IEEE Transactions on Wireless Communications 20 (2) (2020) 1188–1200.

[44] R. Yera, L. Martinez, Fuzzy tools in recommender systems: A survey, International Journal of Computational Intelligence Systems 10 (1) (2017) 776–803.

[45] W. Zhang, X. Wang, P. Zhou, W. Wu, X. Zhang, Client Selection for Federated Learning With Non-IID Data in Mobile Edge Computing, IEEE Access 9 (2021) 24462–24474.

[46] Y. Zhang, Z. Liu, C. Sang, Unifying paragraph embeddings and neural collaborative filtering for hybrid recommendation, Applied Soft Computing (2021) 107345.