A Real-time Contribution Measurement Method for Participants in Federated Learning

1st Boyi Liu & Bingjie Yan
University of Macau &
Hainan University
by.liu@ieee.org &
beiyuouo@foxmail.com

2nd Yize Zhou
School of Science
Hainan University
Haikou, China
Yizezhou20001203@163.com

3rd Jun Wang
School of Information and
Communication Engineering
Hainan University
Haikou, China
20180581310080@hainanu.edu.cn

4th Li Liu
School of Computer Science and
Cyberspace Security
Hainan University
Haikou, China
hainan_lily2001@163.com

5th Yuhan Zhang
School of Information and
Communication Engineering
Hainan University
Haikou, China
email address

6th Xiaolan Nie
School of Computer Science and
Cyberspace Security
Hainan University
Haikou, China
email address

***Abstract*—In recent years, individuals, business organizations or the country have paid more and more attention to their data privacy. At the same time, with the rise of federated learning, federated learning is involved in more and more fields. However, there is no good evaluation standard for each agent participating in federated learning. This paper proposes an online evaluation method for federated learning and compares it with the results obtained by Shapley Value in game theory. The method proposed in this paper is more sensitive to data quality and quantity.**

***Index Terms*—Federated Learning, Contribution Evaluation, Multi-Agent**

I. Introduction

Today, there are many models that support smart behavior on mobile devices. For example, image classification can be used on mobile phones and tablets to predict. Its photos are most likely to be viewed or shared multiple times in the future. You can also use the language model to predict the next word or even the entire reply [1]. Thereby it improves the speech recognition and text input functions on the touch screen keyboard. In addition, service models such as insurance companies that use data models to integrate information in order to obtain better feedback also use related models [2]. There are some problems in these industries or smart terminals. That is, they all have more or less information leakage and privacy sensitivity in the process of achieving their goals. In image classification and language models, all photos taken by users and potential training data for everything they type on their mobile keyboards (including passwords, messages, etc.) can be sensitive to privacy. In the insurance industry, many companies have been concerned about protecting their data and are reluctant to share it with other entities. And by introducing federated learning(FL), data from multiple agents can be used to solve privacy issues in machine learning.

Federated learning(FL) is the equivalent of creating an information system. It can be used by multiple agents to build models while protecting data privacy for participants [3]. Federated learning is not a direct transfer of data to a centralized data warehouse used to build machine learning models. Instead, it allows agents to have data in their place, and still allows agents to build machine learning models together. This can be achieved in the following ways. By building a central model from the sub-models built by the agents, only the model parameters are transmitted. Or by using encryption technology to allow secure communication between different agents.

In order to make federated learning this system works effectively. It is necessary to encourage different agents to contribute their data and join collaborative alliances. In this process, credit allocation and contribution reward mechanisms are essential for federated learning to motivate current and potential participants. Fairly measuring the contribution of all agents in federated learning can achieve fair distribution. It plays a key role in the validity and practicability of the model for the data contributed by all agents. Therefore, we need a way to fairly measure overall data quality. In order to determine the contribution of all agents involved in model training.

We introduce a contribution incentive mechanism combined with federated learning to achieve this goal. It can be well optimized for federated learning. In the distributed case, the attention mechanism of model aggregation is used to calculate the attention weight of all agents in federated learning. Each agent trains locally and uploads the parameters to the server. The server then adjusts to the central model and distributes it to the agents. In the context of the attention mechanism, "attention" is assigned to each agent. The server then updates

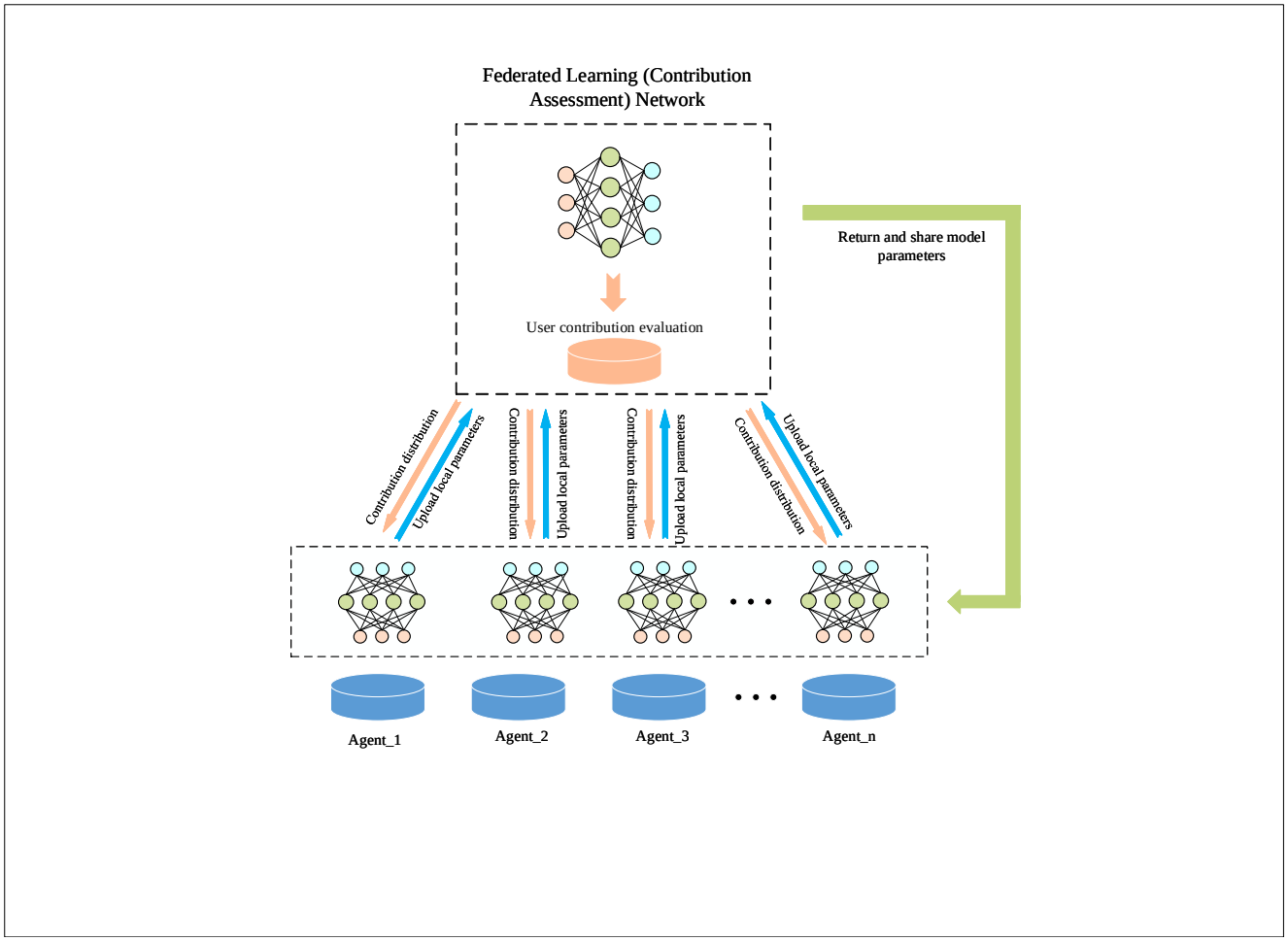


Fig. 1. Experimental results with randomize the word sequence.

the parameters based on the "attention" of each agent. In this way, we can calculate the impact of each agent on the server, and get the impact of each agent on each layer of the server model. Then, the initial and final values of the server parameters are used to obtain the changed value of each parameter of the server. Finally, a mathematical formula is established to solve using the attention-based RNN model [4]. The evaluator can compare the ideal results in the data transfer update to determine the contributions of all agents. And evaluate the contribution of each agent.

The method proposed in this paper is an important attempt to study model contributions and reward allocation in the framework of federated learning.

In this paper, we introduce the model aggregation attention mechanism in the framework of federated learning. So as to determine the contribution of all agents. The advantages of proposed method over the previous federated learning original paper simple sampling and averaging method are: 1) consider the relationship between the server model and the agents model and their weights, 2) optimize the server model and the agents model in the parameter space distance to learn a generic server model.

Our contributions in this paper are as follows:

- We propose a method to measure agents' contribution and compare this method with Shapley Value.
- The method we propose is sensitive to data volume and data quality, and can be used for mutual comparison between agents.
- In the training process, the contribution to each agent can be obtained in time, with low computational complexity.

II. Related Work

This paper refers to federated learning, contribution evaluation mechanism, attention mechanism. Companies applying federated learning can apply this contribution incentive mechanism in the case of commission distribution based on agents contribution.

A. Federated Learning

The training data will be distributed on each mobile device, not all of them will be sent to the central server, and only the updated data on each device will be aggregated to the central server. After federated optimization, the central server returns to the global state of each device, and continues to accept the

updated model parameters calculated by each agent in the new global state. This method is Federated Learning [5]. Federated Learning [6] can solve the problem of unprotected large-scale private data and complete updating learning of devices without exchanging large amounts of data.

This centralized training model approach provides privacy, security, regulation, and economic benefits [7]. Federated Learning presents new statistical and system challenges when training models on distributed device networks [8]. Federated Learning, which relies on scattered data, brings many aspects of research: Fei Chen et al. identified the combination of Federated Learning and Meta-learning as a major advance in Federated Learning [9]. Konstantin Sozinov et al. have made some progress in applying Federated Learning to human activity identification [10].

B. Contribution and Attention Mechanisms

The way to assign different excitation by measuring the contribution of different agents to global optimization is called contribution mechanism. Guan Wang et al. measured the contributions of all agents from the perspectives of horizonal federated learning and vertical federated learning [11]. They used the deletion method to calculate the influence of grouping instances on the horizontal federated learning and the Shapley value on the vertical federated learning to calculate the characteristic values of grouping, which equitably and accurately realized the measurement of contributions of all agents. We effectively optimize Federated Learning by distributing different incentives to agents based on their contributions.

Contribution incentive mechanism has also been widely used and improved. Robin C.goyer et al. proposed a federated learning optimization algorithm based on differential privacy protection of the agents that can hide the contribution of the agents [12]. In order to balance the contribution of the soft training model and ensure the collaborative convergence, Zirui Xu et al. proposed the corresponding parameter aggregation scheme [13].

Shaoxiong Ji et al. also investigated attention mechanisms and proposed a new model of aggregation. This method considers the contribution of the agents model to the global model and combines optimization techniques in the process of server aggregation. Their method minimizes the weighted distance between the server and the agents and improves the communication efficiency [14]. The attention mechanism is just a vector, used for directed perception, derived from the study of computer vision. Our experiment is based on the RNN model of attention.

III. Methodology

In order to solve the above issues, we propose a corresponding method which does not need to obtain agents data and data scale and can measure agents contribution. This method of measuring agents contributions is suitable for evaluating agents at a particular time point.

A. Federated Learning

In this section, we will introduce the basic architecture of federated learning and how parameters are updated. federated learning is a distributed learning method in which the server maintains an overall primary model and distributes it to individual user terminals. For privacy problems, the server can not obtain the data of the user terminal, so the computing power of the user terminal is used to learn at the user local terminal. A server will set up a fraction C, to extract the user terminal proportionally for the server's central model update. Then the extracted user's improved model parameters are uploaded to the server for parameter updating of the server model. It is then distributed to the user terminal to improve the model of the user terminal. In this way, we continue to improve the central model of the server and the local model of the user terminal. Under the premise of ensuring the correctness and privacy of the user terminal, using this method can make use of the computing power of the user terminal and a large amount of user data for learning, and at the same time maintain an excellent central model.

Algorithm 1 Federated Averaging Algorithm

Server model update

K is the total number of agents, w_t is the parameters of current central server model, w_t^k is the parameters of current agents model with index k.

Input: server parameters w_t at t, agent parameters w_{t+1}^k at $t+1$

Output: server parameter w_{t+1} at $t+1$

Initialize w_0

for each round $t = 1, 2, \cdots$ **do**

  $m \leftarrow max\{C \cdot K, 1\}$

  $s_t \leftarrow$ random set of m agents from all K agents

  for each $k \in S_t$ **do**

    $w_{t+1}^k \leftarrow$ ClientUpdate(k, w_t^k) //agents update model on local device

  end for

  $w_{t+1} \leftarrow$ ServerOptimization(w_t,w_{t+1}^k)

end for

Local model update

K is the total number of agents, B is the local mini-batch size, E is the number of local epochs, S is a set of all agents, w_t is the parameters of current central server model, w_t^k is the parameters of current agents model with index k, and η is the learning rate.

Input: index of client k, user private data x_k

Output: client parameters w_{t+1}^k

$B \leftarrow$ Split user data into local mini-batch size B

for each local epoch e from 1 to E **do**

  for batch $b \in B$ **do**

    $w \leftarrow w - \eta \bigtriangledown L(w; b)$

  end for

end for

return w to server

B. Attention Aggregation

In this section, we will introduce a FedAtt algorithm proposed by Shaoxiong Ji, Shirui Pan, Guodong Long et al. , which is outperformed to the common FedAvg and FedSGD algorithm. The proposed attention mechanism in Federated Learning is used after the agents transmitted the parameters to the server, the server uses (1) to calculate the value of attention α of each layer parameter that should be allocated to the agent, and multiplies each layer parameter by the corresponding attention to update the central model. The server integrates the update parameters passed by the agents according to theirs own parameter information, and finally completes an update of the central model through the communication between the server and the agents.

$$\alpha_k^l = \text{Softmax}(s_k^l) = \frac{e^{s_k^l}}{\sum_i e^{s_i^l}} \tag{1}$$

The s_l^k is the norm difference from the central model. We use w^l to represent layer l parameters of the server and w_k^l for lth layer parameters of the terminal model of user k. So the definition of s_l^k is as follows

$$s_k^l = ||w^l - w_k^l||_p \tag{2}$$

So for m user terminals that are selected to update the central model, the method of updating becomes

$$w_{t+1} \leftarrow w_t - \epsilon \sum_{k=1}^{m} \bigtriangledown(w_t^k) = w_t - \epsilon \sum_{k=1}^{m} \alpha_k (w_t - w_t^k) \tag{3}$$

To protect the users data privacy, you can add the randomized mechanism before the agent passes parameters to the server. Randomly generate a random vector that obeys the standard distribution $\mathcal{N}(0, \sigma^2)$, multiply the corresponding weight β, the results of the final update parameters as shown in Formula(4)

$$w_{t+1} \leftarrow w_t - \epsilon \sum_{k=1}^{m} \alpha_k (w_t - w_t^k + \beta \mathcal{N}(0, \sigma^2)) \tag{4}$$

The implementation process of the whole algorithm is shown as Algorithm 2

C. Agents Contributions

In this section, the method for measuring agents contribution proposed in this paper will be introduced in detail.

Before calculating each agent contribution, we make the following assumptions: we think each terminal does not tamper with the updated gradient itself during the previous process of passing and updating parameters between the server and the agents.

The agents contribution to the server should be calculated after each layer of agent parameter attention. When the server has calculated the attention of agent K, it can calculate the impact of this agent K on the server model parameters in the T update. We define aft_t^k as follows:

Algorithm 2 Attentive Federated Optimization

l is the ordinal of neural layers; ϵ is the stepsize of server optimization
Input: server parameters w_t at t, agents parameters $w_{t+1}^1, w_{t+1}^2, \cdots, w_{t+1}^m$ at $t+1$
Output: server parameters w_{t+1} at $t+1$
procedure ATTENTIVE OPTIMIZATION
Initialize attention $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$
for each layer l in model do **do**
  for each agents $k \in S_t$ from 1 to m do **do**
    $S_k^l = ||w^l - w_k^l||_p$
  end for
  $\alpha_k^l = softmax(s_k^l) = \frac{e^{s_k^l}}{\sum_i e^{s_i^l}}$
end for
$\alpha_k = \{\alpha_k^1, \alpha_k^2, \cdots, \alpha_k^l\}$
$w_{t+1} \leftarrow w_t - \epsilon$
$\sum_{k=1}^{m} \alpha_k (w_t - w_t^k + \beta \mathcal{N}(0, \sigma^2))$

$$aft_t^k = \epsilon \alpha_k (w_t - w_t^k + \beta \mathcal{N}(0, \sigma^2)) + \gamma \cdot aft_{t-1}^k \tag{5}$$

where $\gamma \in (0, 1)$ is forgetting coefficient, because large variations in the early stage of the training model, the model is not stable, and the impact of the previous endpoint should be reduced after multiple iterations. If the agent k is not in the selected M agents this time, we think that the impact of the agent t round is $aft_t^k = aft_{t-1}^k$, that is, only the number of rounds that the agent participates in the update is calculated. Note that the server's attention to the agents are the attention of each layer, so when calculating the impact it also calculates the impact of each layer by reweighting and averaging.

We will be each terminal attention MinMaxScaler normalized limited range, then Softmax, to obtain the contribution of each agent. We use con_t^k to represent the contribution of agent k at t time. The whole method of measuring agents contribution proposed in this paper is shown in Algorithm 3

Algorithm 3 Measure Agents Contributions

Input: l is the ordinal of neural layers; ϵ is the stepsize of server optimization, server parameters w_t at time $t+1$, agents parameters $w_{t+1}^1, w_{t+1}^2, \cdots, w_{t+1}^m$ at time $t+1$, γ is forgetting coefficient, S_t is selected agents set.
Output: Agents contributions con_t at t
procedure CONTRIBUTION CALCULATION
for each agents k from 1 to K **do**
  if $k \in S_t$ **then**
    $aft_t^k = \epsilon \alpha_k (w_t - w_t^k + \beta \mathcal{N}(0, \sigma^2)) + \gamma \cdot aft_{t-1}^k$
  else
    $aft_t^k = aft_{t-1}^k$
  end if
end for
$con_t = \text{Softmax}(\text{MinMaxScaler}(aft_t))$

IV. EXPERIMENT

In this section, we will introduce the verification experiments performed for our proposed agents' contribution evaluation method.

A. Experimental Environment and Configuration

The system used in our verification experiment is Ubuntu 18.04 LTS, the backend used is the GPU version of Pytorch, with the NVIDIA GTX1660Ti GPU acceleration for model calculation. The GRU-based agents model used the language processing model of RNN. For a detailed model description, see the subsection C under this Section.

B. Dataset

We perform experimental verification on public language datasets of Penn Treebank[1] to verify the effectiveness of our proposed algorithm and its sensitivity to special variables.

C. Model

In natural language processing, the LSTM model is often used for processing. We used a smaller GRU-based agent language model. First take texts as input, and convert words into word vectors according to a pre-built dictionary. The converted word vector is then used as an input to the LSTM model, and the prediction result is finally output.

D. Parameters

This section enumerates the meanings and values of the various parameters involved. At the same time, we believe that the data of each agent is independent and identically distributed. The parameters and their descriptions are listed in Tab. I

TABLE I
TABLE OF PARAMETERS MENTIONED

Name	Represent letter	Value
Number of agents	K	20
Server training rounds	round	30
The number of local epoch	epoch	5
The fraction of agents	C	0.1
Learning rate of agents	η	0.01
Learning rate of server	η'	0.001
Batch size	B	128
Step size	ϵ	1.2
Differential privacy	β	0.001
Embedding dimension	none	300
Forgetting coefficient	γ	0.9

E. Experimental Results

We use testing perplexity as an indicator of the evaluation model. In information theory, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. The perplexity of a discrete probability distribution is defined as:

$$ppl(x) = 2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)} \tag{6}$$

In the above formula, $H(p)$ is the expected probability distribution. If the prediction result of our model is $m(x)$, then the perplexity of the language model is defined as:

$$ppl(x) = 2^{H(p,m)} = 2^{-\sum_x p(x) log_2 m(x)} \tag{7}$$

Obviously, $H(p) \leqslant H(p, m)$. Therefore, the smaller the perplexity, the more representative the probability distribution can be to better predict the sample distribution.

And we compared with the results of Shapley Value evaluation. Shapley Value is originated from coalitional game theory and has proven theoretical properties. It provides a way to measure the impact and contribution of various agents. The definition of Shapley Value is:

$$\varphi_i(x) = \sum_{Q \subseteq S \backslash \{i\}} \frac{|Q|! - (|S| - |Q| - 1)!}{|S|!} (\Delta_{Q \cup \{i\}}(x) - \Delta_Q(x)) \tag{8}$$

S is the set of all agents, $Q \subset S = 1, 2, \cdots, n$ is a subset of the agent set S, i is the index of the agents, $||$ represents the size of the set, $\Delta_Q(x) = aft_Q$ denotes the affect of agent set Q.

Because the complexity of directly calculating Shapley value is too high, we use the following estimation method to get the $\varphi_i(x)$ of each Agents

$$\varphi_i(x) = \frac{1}{M} \sum_{m=1}^{M} (\Delta_{Q^m \cup \{i\}}(x) - \Delta_{Q^m}(x)) \tag{9}$$

where M is the number of iterations. $\Delta_{Q^m}(x)$ denotes the affect of random set Q^m. Finally, all the data obtained by Shapley Value is subjected to the same regularization processing as ours.

To ensure the fairness of the evaluation contribution in this experiment, we ensure that each agent is drawn the same number of times.

We randomly divided the data in the dataset, and distributed them evenly to all agents. Then, the corresponding special processing is performed on the data of the last few agents: reduce the amount of data by 30% and 70%, randomly generate the word sequence. Comparing the evaluation results after special processing with the results when unprocessed data, it is concluded that the sensitivity of variables such as data size and data quality is evaluated.

1) Experimental results of normal evaluation: We did not do any special treatment to any agents, and randomly divided the data set into each agent. The result is shown in Fig. 2.In the figure, the abscissa is the index of agents, and the ordinate is the agents' contribution ratios. It can be seen that the deviation of each agent is not large, only individual agents are too high or too low. And the evaluation result is similar to Shapley Value.

[1]Penn Treebank is available at https://github.com/wojzaremba/lstm/tree/master/data

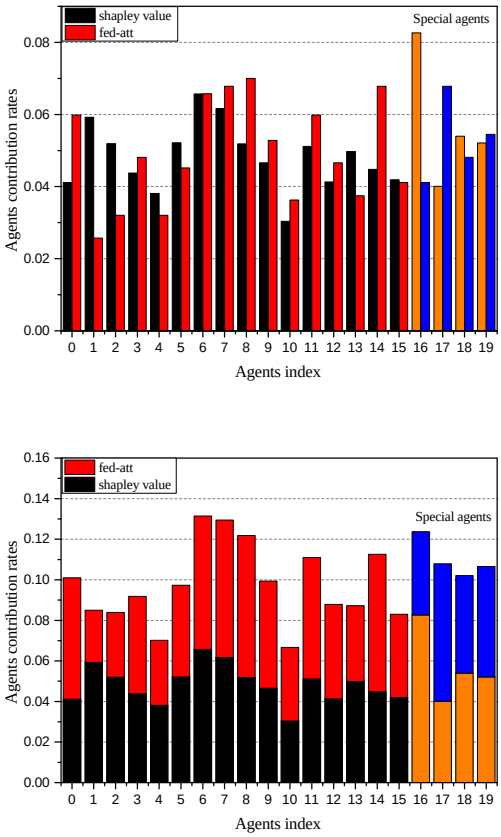


Fig. 2. Experimental results of normal evaluation.

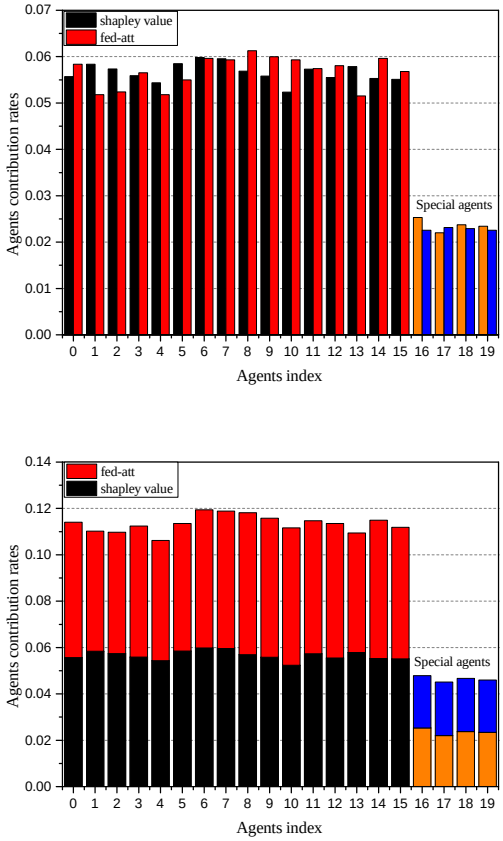


Fig. 3. Experimental results with randomize the word sequence.

2) Experimental results with randomize the word sequence: In this experiment, we modified the datasets of the next four agents into random word sequences. These should be regarded as dirty data by the model, so as to get a smaller return. The results are shown in Fig. 3. Both the method proposed in this paper and the Shpaley Value method can identify these bad agents and give them a small contribution.

3) Experimental results with reduce the amount of data: To demonstrate the sensitivity of our evaluation method to the amount of data, we performed the following experiments.

In this experiment, we reduced the data amount of the last 4 agents by 70%, and the data amount of other agents remained unchanged. As you can see in Fig. 4, the contribution of the specially treated agents are significantly reduced. But it is relatively not obvious in the evaluation results of Shapley Value.

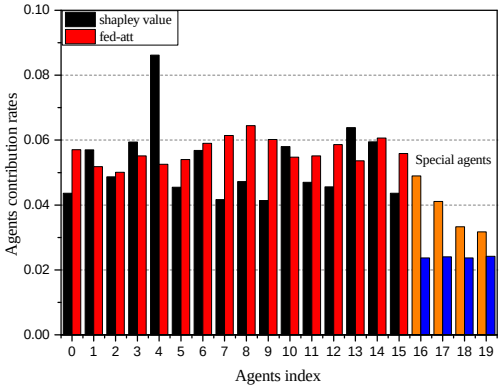
In order to reflect the relative relationship between the amount of data, we processed the data of the last 4 agents: agents with index 16 to 17 reduced the amount of data by 30%, agents with index 18 to 19 reduced the amount of data by 70%.

It can be seen in Fig. 5 that the method proposed in this paper can reflect the reduced amount of data, while Shapley Value cannot show it well.

V. CONCLUSION

A reasonable and fair completion of the assessment of the contribution of each participant in federated learning and the establishment of an incentive mechanism are essential for the development of federated learning. In this paper, we use a "FedAtt" method to train the models on Penn Treebank dataset.At the same time, we perform several special processing on the agents data, and compare the evaluation results of the special processing with the evaluation results of the unprocessed data. Experimental results show that we use this method to reasonably establish a measurement mechanism to evaluate the sensitivity of indicators such as data size and data quality. The data is evaluated and trained by this method, and the calculated results are real-time and fast, and the contribution rate reflects accurately.

For future work, we hope to find a better algorithm to reduce the test confusion of unbundled medium and unbundled large models; at the same time, we should consider the negative contribution of agent evaluation and the comprehensiveness of the evaluation. Preliminary classification (i.e., positive and negative) is carried out at the end to avoid attacks under the federated learning framework mechanism [15]; at the same time, the concept of game theory is introduced, such



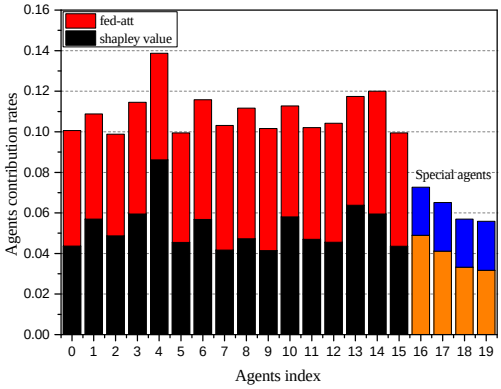
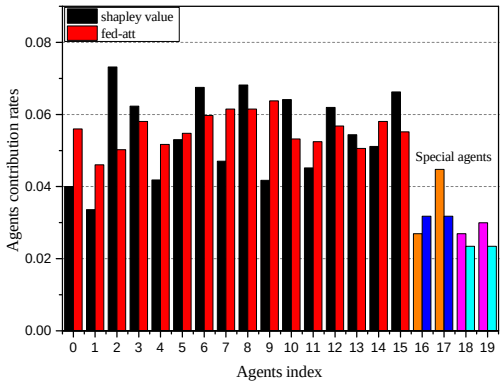


Fig. 4. Experimental results with reduce the amount of data.



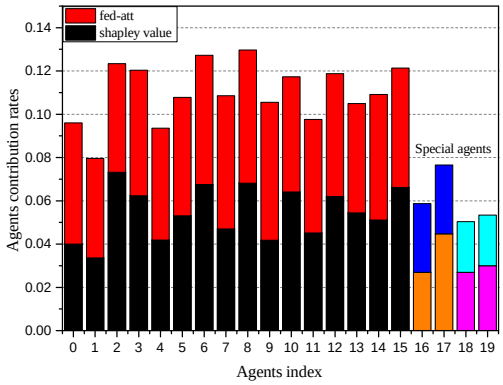


Fig. 5. Experimental results with reduce the amount of data.

as the establishment of a PVCG mechanism on the supply side [16]. Improvements in these directions will promote the implementation and application of the federated learning incentive mechanism.

REFERENCES

[1] J. General Chair-Rekimoto, T. General Chair-Igarashi, J. O. Program Chair-Wobbrock, and D. Program Chair-Avrahami, "Proceedings of the 29th annual symposium on user interface software and technology," in *Symposium on User Interface Software and Technology*, 2016.

[2] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning."

[3] J. Konen, H. B. Mcmahan, F. X. Yu, P. Richtrik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency."

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780.

[5] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[6] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

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

[8] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[9] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[10] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 2018, pp. 1103–1111.

[11] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," *arXiv preprint arXiv:1909.08525*, 2019.

[12] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[13] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen, "Elfish: Resource-aware federated learning on heterogeneous edge devices," *arXiv preprint arXiv:1912.01684*, 2019.

[14] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[15] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2018.

[16] M. Cong, X. Weng, H. Yu, J. Qu, and S. M. Yiu, "Optimal procurement auction for cooperative production of virtual products: Vickrey-clarke-groves meet cremer-mclean," 2020.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try

“R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.