IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

Learning Private Neural Language Modeling with Attentive Aggregation

Shaoxiong Ji*, Shirui Pan†, Guodong Long‡, Xue Li*, Jing Jiang‡, Zi Huang*
* School of ITEE, Faculty of EAIT, The University of Queensland, Australia
† Faculty of Information Technology, Monash University, Australia
‡ Centre for Artificial Intelligence, Faculty of Engineering and IT, University of Technology Sydney, Australia
Email: shaoxiong.ji@uq.edu.au, shirui.pan@monash.edu, guodong.long@uts.edu.au,
xueli@itee.uq.edu.au, jing.jiang@uts.edu.au, huang@itee.uq.edu.au

***Abstract*—Mobile keyboard suggestion is typically regarded as a word-level language modeling problem. Centralized machine learning techniques require the collection of massive user data for training purposes, which may raise privacy concerns in relation to users' sensitive data. Federated learning (FL) provides a promising approach to learning private language modeling for intelligent personalized keyboard suggestions by training models on distributed clients rather than training them on a central server. To obtain a global model for prediction, existing FL algorithms simply average the client models and ignore the importance of each client during model aggregation. Furthermore, there is no optimization for learning a well-generalized global model on the central server. To solve these problems, we propose a novel model aggregation with an attention mechanism considering the contribution of client models to the global model, together with an optimization technique during server aggregation. Our proposed attentive aggregation method minimizes the weighted distance between the server model and client models by iteratively updating parameters while attending to the distance between the server model and client models. Experiments on two popular language modeling datasets and a social media dataset show that our proposed method outperforms its counterparts in terms of perplexity and communication cost in most settings of comparison.**

***Index Terms*—federated learning, language modeling, attentive aggregation**

I. Introduction

With the advances in mobile technology, smart phones and wearable devices like smart watches are becoming increasingly popular in modern life. A study conducted in 2017 shows that the majority of participants spent five hours or more every day on their smartphones[1]. These mobile devices generate a massive amount of distributed data such as text messages, travel trajectories and health status.

In the traditional machine learning approaches, cloud-based servers send a data collection request to the clients, collect data from clients, train a centralized model, and make predictions for clients. However, this centralized way of learning relies heavily on the central server. Moreover, there are privacy issues, especially in relation to sensitive data, if the central server is hacked into or misused by other third parties.

[1]Reported by The Statistics Portal available at https://www.statista.com/statistics/781692/worldwide-daily-time-spent-on-smartphone/, retrieved in Dec, 2018

Recently, a distributed learning technique called federated learning has attracted great interest from the research community [1]–[3] under the umbrella of distributed machine learning. It protects the privacy of data by learning a shared model by distributed training on local client devices without collecting the data on a central server. Distributed intelligent agents take a shared global model from the central server's parameters as initialization to train their own private models using personal data, and make predictions on their own physical devices. There are many applications of federated learning in the real world, for example, predicting the most likely photos a mobile user would like to share on the social websites [4], predicting the next word for mobile keyboards [5], retrieving the most important notifications, and detecting the spam messages [6].

Of these applications, mobile keyboard suggestion as a language modeling problem is one of the most common tasks because it involves with user interaction which can give instant labeled data for supervised learning. In practice, the mobile keyboard applications predict the next word with several options when a user is typing a sentence. Generally speaking, an individual's language usage expresses that person's language preference and patterns. With the recent advances in deep neural networks, a language model combined with neural networks, called neural language modeling, has been developed. Of these neural network models, recurrent neural networks (RNNs) which capture the temporal relations in sentences has significantly improved the field of language modeling. Specific RNNs include long short-term memory (LSTM) [7] and its variants such as gated recurrent unit (GRU) [8].

In the real-world scenario, users' language input and preferences are sensitive and may contain some private content including private personal profiles, financial records, passwords, and social relations. Thus, to protect the user's privacy, a federated learning technique with data protection is a promising solution. In this paper, we take this application as learning word-level private neural language modeling for each user.

Federated learning learns a shared global model by the aggregation of local models on client devices. But the original paper on federated learning [1] only uses a simple average on client models, taking the number of samples in each

978-1-7281-2009-6/$31.00 paper N-19564.pdf

client device as the weight of the average. In the mobile keyboard applications, language preferences may vary from individual to individual. The contributions of client language models to the central server are quite different. To learn a generalized private language model that can be quickly adapted to different people's language preferences, knowledge transferring between server and client, especially the well-trained clients models, should be considered.

In this paper, we introduce an attention mechanism for model aggregation. It is proposed to automatically attend to the weights of the relation between the server model and different client models. The attentive weights are then taken to minimize the expected distance between the server model and client models. The advantages of our proposed method are: 1) it considers the relation between the server model and client models and their weights, and 2) it optimizes the distance between the server model and client models in parameter space to learn a well-generalized server model.

Our contributions in this paper are as follows:

- Our work first introduces the attention mechanism to aggregate multiple distributed models. The proposed attentive aggregation can be further applied to improve broad methods and applications using distributed machine learning.
- In the server optimization, we propose a novel layer-wise soft attention to capturing the "attention" among many local models' parameters.
- As demonstrated by the experiments on private neural language modeling task for mobile keyboard suggestions, the proposed method achieves a comparable performance in terms of perplexity and communication rounds.

The structure of this paper is organized as follows. In Section II, related works including federated learning, attention mechanism and neural language modeling are reviewed. Our proposed attentive federated aggregation is introduced in Section III. Experimental settings and results are given in Section IV together with the comparison and analysis. In Section V, the conclusion is drawn.

II. Related Work

This paper relates to federated learning, language modeling, and attention mechanism.

A. Federated Learning

Federated learning is proposed by McMahan et al. [1] to decouple training procedures from data collection by an iterative averaging model. It can perform distributed training and communication-efficient learning from decentralized data to achieve the goal of privacy preservation. Geyer et al. proposed a differential privacy-preserving technique on the client side to balance the performance and privacy [2]. Popov et al. proposed the fine-tuning of distributed private data to learn language models [9]. The federated learning technique is useful in many fields. Chen et al. combined federated learning with meta learning for recommendation [3]. Kim et al. proposed federated tensor factorization to discover clinical concepts from electronic health records [10]. The federated setting can also be integrated into other machine learning settings. Smith et al. [11] proposed a framework that fits well with multi-task learning and federated setting to tackle the statistical challenge of distributed machine learning.

Communication efficiency is one of the performance metrics for federated learning techniques. To improve communication efficiency, Konečnỳ et al. proposed structured updates and sketched updates to reduce the uplink communication costs [12]. It is also studied under the umbrella of distributed machine learning. Alistarch et al. proposed quantized compression and the encoding of stochastic gradient descent [13] to achieve efficient communication. Wen et al. used ternary gradients to reduce the communication cost [14].

B. Neural Language Modeling

Language modeling, as one of the most crucial natural language processing tasks, has achieved better performance than classical methods using popular neural networks. Mikolov et al. used a simple recurrent neural network-based language model for speech recognition [15]. Recently, LSTM-based neural language models were developed to learn context over long sequences in large datasets [16]. To facilitate learning in a language model and reduce the number of trainable parameters, Inan et al. proposed tying word vectors and word classifiers [17]. Press and Wolf evaluated the effect of weight tying and introduced a new regularization on the output embedding [18].

C. Attention Mechanism

The attention mechanism is simply a vector serving to orient perception. It first became popular in the field of computer vision. Mnih et al. used it in recurrent neural network models for image classification [19]. It was then widely applied in sequence-to-sequence natural language processing tasks like neural machine translation [20]. Luong et al. extended attention-based RNNs and proposed two new mechanisms, i.e., the global attention and local attention [21]. Also, the attention mechanism can be used in convolutional models for sentence encoding like ABCNN for modeling sentence pairs [22]. Yang et al. proposed hierarchical attention networks for document classification [23]. Shen et al. proposed directional self-attention for language understanding [24].

III. Proposed Method

In this section, we firstly introduce the preliminaries of the federated learning framework, and then propose our attentive federated optimization algorithm to improve the generalizability for distributed clients by learning the attentive weights of selected clients during model aggregation. As for the client learner, we apply the gated recurrent unit (GRU) [8] as the client model for language modeling. Furthermore, we add a randomized mechanism [2] for learning differentially private client model.

A. Preliminaries of Federated Learning

Federated learning decouples the model training and data collection [1]. To learn a well generalized model, it uses model aggregation on the server side, which is similar to the works on meta-learning by learning a good initialization for quick adaptation [25], [26] and transfer learning by transferring knowledge between domains [27]. The basic federated learning framework comprises two main parts, i.e., server optimization in Algorithm 1 and local training in Algorithm 2 [1].

Central Model Update. The server firstly chooses a client learning model and initializes the parameters of the client learner. It sets the fraction of the clients. Then, it waits for online clients for local model training. Once the selected number of clients finishes the model update, it receives the updated parameters and performs the server optimization. The parameter sending and receiving consists of one round of communication. Our proposed optimization is conducted in Line 9 of Algorithm 1.

Algorithm 1 Optimization for Federated Learning on Central Server

1: K is the total number of clients; C is the client fraction; U is a set of all clients.
2: **Input**: server parameters θ_t at t, client parameters $\theta_{t+1}^1, ..., \theta_{t+1}^m$ at $t+1$.
3: **Output**: aggregated server parameters θ_{t+1}.
4: **procedure** SERVER EXECUTION ▷ Run on the server
5: initialize θ_0
6: **for** each round t=1, 2, ... **do**
7: $m \leftarrow max(C \cdot K, 1)$
8: $S_t \leftarrow \{u_i \mid u_i \in U\}_1^m$ ▷ Random set of clients
9: **for** each client k $\in$ S_t on local device **do**
10: $\theta_{t+1}^k \leftarrow$ ClientUpdate(k, θ_t)
11: $\theta_{t+1} \leftarrow$ ServerOptimization$(\theta_t, \theta_{t+1}^k)$

Private Model Update. Each online selected client receives the server model and performs secure local training on their own devices. For the neural language modeling, stochastic gradient descent is performed to update their GRU-based client models which is introduced in Section III-C. After several epochs of training, the clients send the parameters of their models to the central server over a secure connection. During this local training, user data can be stored on their own devices.

B. Attentive Federated Aggregation

The most important part of federated learning is the federated optimization on the server side which aggregates the client models. In this paper, a novel federated optimization strategy is proposed to learn federated learning from decentralized client models. We call this Attentive Federated Aggregation, or `FedAtt` for short. It firstly introduces the attention mechanism for federated aggregation by aggregating the layer-wise contribution of neural language models of selected clients to the global model in the central server. An illustration of our proposed layer-wise attentive federated aggregation is shown in Figure 1 where the lower box represents the distributed client models and the upper box represents the attentive aggregation in the central server. The distributed client models in the lower box contain several neural layers. The notations of "$\oplus$" and "$\ominus$" stand for the layer-wise operation on the parameters of neural models. This illustration shows only a single time step. The federated updating uses our proposed attentive aggregation block to update the global model by iteration.

Algorithm 2 Secure Local Training on Client

1: B is the local mini-batch size; E is the number of local epochs; β is the momentum term; η is the learning rate.
2: **Input**: ordinal of user k, user data X.
3: **Output**: updated user parameters θ_{t+1} at $t+1$.
4: **procedure** CLIENT UPDATE(k, θ) ▷ Run on the k-th client
5: B $\leftarrow$ ($split\ user\ data\ X\ into\ batches$)
6: **for** each local epoch i from 1 to E **do**
7: **for** batch b $\in$ B **do**
8: $z_{t+1} \leftarrow \beta z_t + \nabla L(\theta_t)$
9: $\theta_{t+1} \leftarrow \theta_t - \eta z_{t+1}$
10: send θ_{t+1} to server

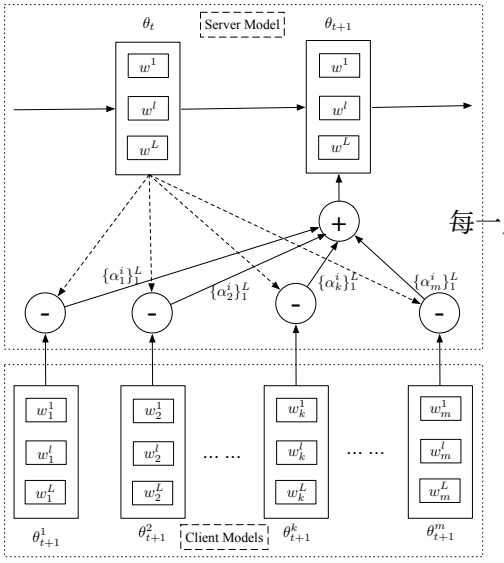


Fig. 1: The illustration of our proposed layer-wise attentive federated aggregation

The intuition behind the federated optimization is to find an optimal global model that can generalize the client models well. In our proposed optimization algorithm, we take it as finding an optimal global model that is close to the client models in parameter space while considering the importance of selected client models during aggregation. The optimization objective is defined as

全局模型 客户机模型

$$\arg\min_{\theta_{t+1}} \sum_{k=1}^{m} [\frac{1}{2} \alpha_k L(\theta_t, \theta_{t+1}^k)^2], \quad (1)$$

where θ_t is the parameters of the global server model at time stamp t, θ_{t+1}^k is the parameters of the k-th client model at time stamp $t+1$, $L(\cdot,\cdot)$ is defined as the distance between two sets of neural parameters, and α_k is the attentive weight to measure the importance of weights for the client models. The objective is to minimize the weighted distance between server model and client models by taking a set of self-adaptive scores as the weights.

To attend the importance of client models, we propose a novel layer-wise attention mechanism on the parameters of the neural network models. The attention mechanism in this paper is quite similar to the soft attention mechanism. Unlike the popular attention mechanism applied to the data flow, our proposed attentive aggregation is applied on the learned parameters of each layer of the neural language models. We take the server parameters as a query and the client parameters as keys, and calculate the attention score in each layer of the neural networks.

Given the parameters in the l-th layer of the server model denoted as w^l and parameters in the l-th layer of the k-th client model denoted as w_k^l, the similarity between the query and the key in the l-th layer is calculated as the norm of the difference between two matrices, which is denoted as:

$$s_k^l = \|w^l - w_k^l\|_p.$$

Then, we apply softmax on the similarity to calculate the layer-wise attention score for the k-th client in Equation 2.

$$\alpha_k^l = softmax(s_k^l) = \frac{e^{s_k^l}}{\sum_{k=1}^{m} e^{s_k^l}} \quad (2)$$

Our proposed attention mechanism on the parameters is layer-wise. There are attention scores for each layer in the neural networks. For each model, the attention score is $\alpha_k = \{\alpha_k^0, \alpha_k^1, \ldots, \alpha_k^l, \ldots\}$ in a non-parameter way.

Using the Euclidean distance for $L(\cdot,\cdot)$ and taking the derivative of the objective function in Equation 1, we get the gradient in the form of Equation 3.

$$\nabla = \sum_{k=1}^{m} \alpha_k (\theta_t - \theta_{t+1}^k) \quad (3)$$

For the selected group of m clients, we perform gradient descent to update the parameters of the global model in Equation 4 as

$$\theta_{t+1} \leftarrow \theta_t - \epsilon \sum_{k=1}^{m} \alpha_k (\theta_t - \theta_{t+1}^k), \quad (4)$$

where ϵ is the step size. The full procedure of our proposed optimization algorithm is described in Algorithm 3. It takes the server parameters θ_t at time stamp t and client parameters $\theta_{t+1}^1, ..., \theta_{t+1}^m$ at time stamp $t+1$, and returns the updated parameters of the global server.

Algorithm 3 Attentive Federated Optimization

k is the ordinal of clients; l is the ordinal of neural layers; ϵ is the stepsize of server optimization
Input: server parameters θ_t at t, client parameters $\theta_{t+1}^1, ..., \theta_{t+1}^m$ at $t+1$.
Output: aggregated server parameters θ_{t+1}.
procedure ATTENTIVE OPTIMIZATION(θ_t, θ_{t+1}^k)
    Initialize $\alpha = \{\alpha_1, \ldots, \alpha_k, \ldots, \alpha_m\}$ ▷ attention for each clients
    for each layer $l = 1, 2, \ldots$ **do**
        for each user k **do**
            $s_k^l = \|w^l - w_k^l\|_p$
            $\alpha_k^l = softmax(s_k^l) = \frac{e^{s_k^l}}{\sum_{k=1}^{m} e^{s_k^l}}$
    $\alpha_k = \{\alpha_k^0, \alpha_k^1, \ldots, \alpha_k^l, \ldots\}$
    $\theta_{t+1} \leftarrow \theta_t - \epsilon \sum_{k=1}^{m} \alpha_k (\theta_t - \theta_{t+1}^k)$
    return θ_{t+1}

The advantage of our proposed layer-wise attentive federated aggregation and its optimization is as follows: 1) The aggregation of client models is fine-grained on each layer of the neural models considering the similarity between the client model and the server model in the parameter space. The learned features of each client model can be effectively selected to to produce a fine-tuned global server model. 2) By minimizing the expected distance between the client model and the server model, the learned server model is close to the client models in the parameter space and can well represent the federated clients.

C. GRU-based Client Model

The learning process on the client side is model-agnostic. For different tasks, we can choose appropriate models in specific situations. In this paper, we use the gated recurrent unit (GRU) [8] for learning the language modeling on the client side. The GRU is a well-known and simpler variant of the Long Short-Term Memory (LSTM) [7], by merging the forget gate and the input gate into a single gate as well as the cell state and the hidden state. In the GRU-based neural language model, words or tokens are firstly embedded into word vectors denoted as $X = \{x_0, x_1, \ldots, x_t, \ldots\}$ and then put into the recurrent loops. The calculation inside the recurrent module is as follows:

$$\begin{aligned} z_t &= \sigma(w_z \cdot [h_{t-1},\ x_t]), \\ r_t &= \sigma(w_r \cdot [h_{t-1},\ x_t]), \\ \tilde{h_t} &= tanh(w \cdot [r_t * h_{t-1},\ x_t]), \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h_t}, \end{aligned}$$

where z_t is the update gate, r_t is the reset gate, h_t is the hidden state, and $\tilde{h_t}$ is a new hidden state.

D. Differential Privacy

To protect the client's data from an inverse engineering attack, we apply the randomized mechanism into federated

learning [2]. This ensures differential privacy on the client side without revealing the client's data [2]. This differentially private randomization was firstly proposed to apply on the federated averaging, where a white noise with the mean of 0 and the standard deviation of σ is added to the client parameters in Equation 5.

$$\theta_{t+1} = \theta_t - \frac{1}{m}(\sum_{k=1}^{K} \Delta\theta_{t+1}^k + \mathcal{N}(0, \sigma^2)) \tag{5}$$

Our proposed attentive federated aggregation can also add this mechanism smoothly using Equation 6. The randomization is added in before the clients send the updated parameters to the server, but it is written in the form of server optimization for simplicity.

$$\theta_{t+1} \leftarrow \theta_t - \epsilon \sum_{k=1}^{m} \alpha_k(\theta_t - \theta_{t+1}^k + \beta\mathcal{N}(0, \sigma^2)) \tag{6}$$

In practice, we add a magnitude coefficient $\beta \in (0, 1]$ on the randomization of normal noise to control the effect of the randomization mechanism on the performance of federated aggregation.

IV. Experiments

This section describes the experiments conducted to evaluate our proposed method. Two baseline methods are compared and additional exploratory experiments are conducted to further the exploration of the performance of our attentive method. Our code is available online at https://github.com/shaoxiongji/fed-att.

A. Datasets

We conduct experiments of neural language modeling experiments on three English language datasets for evaluation to mimic the real-world scenario of mobile keyboards in the decentralized applications. They are Penn Treebank [28], WikiText-2 [29], and the Reddit Comments from Kaggle. Language modeling is one of the most suitable tasks for the validation of federated learning. It has a large number of datasets to test the performance and there is a real-world application, i.e., the input keyboard application in smart phones.

Penn Treebank is an annotated English corpus. We use the data derived from Zaremba et al.[2] [30]. The WikiText-2 is available online[3]. May 2015 Reddit Comments dataset is a portion of a large scale dataset of Reddit comments[4] from the popular online community – Reddit. It is available in the Kaggle Datasets[5]. We sampled 1‰ of the comments from this dataset to train our private language model as a representative of social networks data. The statistical information, i.e., the number of tokens in the training, validation, and testing set of these three datasets is shown in Table I.

TABLE I: Number of tokens in training, validation and testing sets of three datasets

Dataset	# Train	# Valid.	# Test
Penn Treebank	887,521	70,390	78,669
WikiText-2	2,088,628	217,646	245,569
Reddit Comments	1,784,023	102,862	97,940

Data Partitioning. To mimic the scenario of real-world private keyboard applications, we perform data partitioning on these three popular language modeling datasets. At first, we shuffle the whole dataset. Then, we perform random sampling without replacement under the independently identical distribution. The whole dataset is partitioned into a certain number of shards denoted as the number of users or clients. We split these three datasets into 100 subsets as the user groups of 100 clients to participate in the federated aggregation after local training.

B. Baselines and Settings

We conducted to several groups experiments for comparison, for example, performance with different model aggregation methods, the scale of client models, communication cost, and so forth. There are two baselines totally in these comparisons, i.e., `FedSGD` and `FedAvg`. The basic definitions and settings of baselines and our proposed method are as follows.

1) `FedSGD`: Federated stochastic gradient descent takes all the clients for federated aggregation and each client performs one epoch of gradient descent.
2) `FedAvg`: Federated averaging samples a fraction of users for each iteration and each client can take several steps of gradient descent.
3) `FedAtt`: Our proposed FedAtt takes a similar setting as FedAvg, but uses an improved attentive aggregation algorithm.

We conduct experiments under the setting of federated learning using the GRU-based private neural language modeling with Nvidia GTX 1080 Ti GPU acceleration. The GRU-based client model firstly takes texts as input, then embeds them into word vectors and feeds them to the GRU network. The last fully connected layer takes the output of GRU as input to predict the next word. The small model uses 300 dimensional word embedding and hidden state of RNN unit. We deploy models of three scales: small, medium and large with word embedding dimensions of 300, 650 and 1500, respectively. Tied embedding is applied to reduce the size of the model and decrease the communication cost. Tied embedding shares the weights and biases in the embedding layer and output layer and greatly reduces the number of trainable parameters greatly.

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

[3]WikiText-2 is available at https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-2-v1.zip

[4]Avaliable at https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/, retrieved in Dec, 2018

[5]Reddit Comments dataset is available at https://www.kaggle.com/reddit/reddit-comments-may-2015

C. Results

We conduct experiments on these three datasets and three federated learning methods. Testing perplexity is taken as the evaluation metric. Perplexity is a standard measurement for probability distribution. It is one of the most commonly used metrics for word-level language modeling. The perplexity of a distribution is defined as

$$PPL(x) = 2^{H(p)} = 2^{-\sum_x p(x) \log \frac{1}{p(x)}}$$

where $H(p)$ is the entropy of the distribution $p(x)$. A lower perplexity stands for a better prediction performance of the language model.

We take 50 rounds of communication between server and clients and compare the performance on the validation set to select the best model, then test the performance on the testing set to get the testing perplexity. The results of testing perplexity of all three datasets are shown in Table II. For FedAvg and FedAtt, we set the client fraction C to be 0.1 and 0.5 within these results. According to the definition of FedSGD, the client fraction is always 1. As shown in this table, our proposed FedAtt outperforms FedSGD and FedAvg in terms of testing perplexity in all the three datasets. When the client fraction C is 0.1 and 0.5 in the Penn Treebank and WikiText-2 respectively, our proposed method gains a significant improvement over its counterparts. We also conduct experiments on the fine-grained setting of the client fraction C (from 0.1 to 0.9). When the client fraction is 0.7, our proposed method obtains the best testing perplexity of 67.59 in the WikiText-2 dataset.

TABLE II: Testing perplexity of 50 communication rounds for federated training using small-scaled GRU network as the client model

Frac.	Methods	WikiText-2	PTB	Reddit
1	FedSGD	112.45	155.27	128.61
0.1	FedAvg	95.27	138.13	126.49
	FedAtt	91.82	115.43	120.25
0.5	FedAvg	79.75	128.24	101.64
	FedAtt	69.38	123.00	99.04

We then further our exploration of the four factors in the WikiText-2 dataset to evaluate the performance of our proposed method with a comparison of its counterpart FedAvg. In additional exploratory experiments in the following subsections, we explored the client fraction, the communication costs, the effect of different randomizations, and the scale of the models.

D. Client Fraction

In real-world applications of federated learning, some clients may be offline due to a change in user behavior or network issues. Thus, it is necessary to choose only a small number of clients for federated optimization. To evaluate the effect of the client fraction C on the performance of our proposed attentive federated optimization, we explore the testing perplexity with various number of clients. The result is illustrated in Figure 2 where the client fraction varies from 0.1 to 0.9. The small-scaled neural language model is used in this evaluation. The testing perplexity fluctuates when the client fraction increases. There is no guarantee that more clients results in a better score. Actually, 70% of clients for model aggregation achieved the lowest perplexity in this experiment. This result indicates that the number of clients participating in model aggregation has an impact on the performance. But our proposed FedAtt can achieve much quite lower perplexity than FedAvg for all the settings of the client fraction.

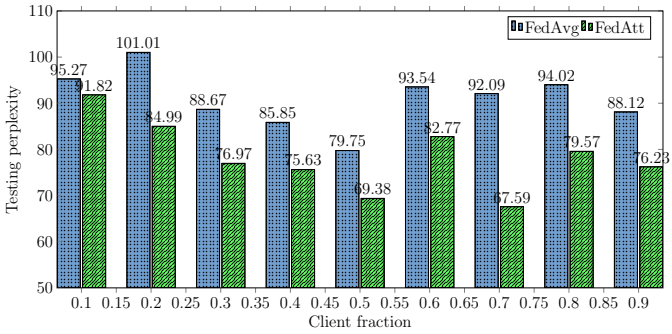


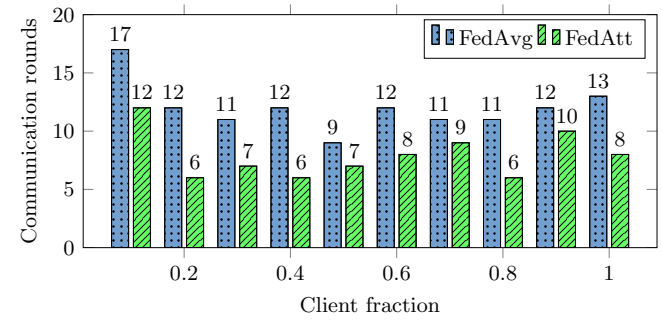
Fig. 2: Testing perplexity of 50 communication rounds when a different number of clients is selected for federated aggregation

E. Communication Cost

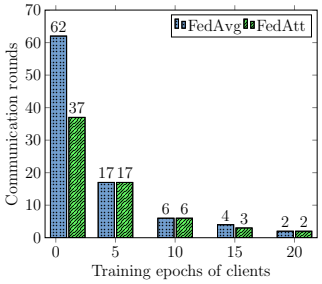
Communication cost for parameter uploading and downloading between the clients and server is another important issue for decentralized learning. Communication, both wired and wireless, depends on Internet bandwidth highly and has an impact on the performance of federated optimization. To save the capacity of network communication, decentralized training should be more communication-efficient. Several approaches apply compression methods to achieve efficient communication. Our method accelerates the training through the optimization of the global server as it can converge more quickly than its counterparts.

To compare the efficiency of communication, we take the communication rounds during training as the evaluation metric in this subsection. Three factors are considered, i.e., the client fraction, epochs and batch size of client training. The results are shown in Figure 3 where the small-scaled language model is used as the client model and 10% of clients are selected for model aggregation. We set the testing perplexity for the termination of federated training to be 90. When the testing perplexity is lower than that threshold, federated training comes to an end and we take the rounds of training as the communication rounds. As shown in Figure 3(a), the communication round during training fluctuates when the number of client increases. Furthermore, our proposed method is always better than FedAvg with less communication cost. When the client fraction C chosen is 0.2 and 0.4, our

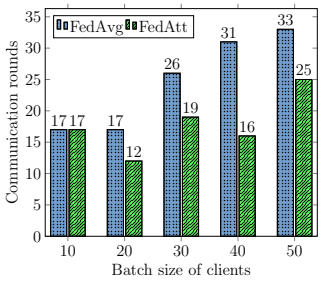
proposed method saves a half of communication rounds. Then, we evaluate the effect of the local computation of clients on the communication rounds. We take the local training epochs to be 1, 5, 10, 15, and 20 and the local batch size to be from 10 to 50. We proposed FedAtt to achieve a comparable communication cost in the comparison of different values of epoch and the batch size of local training, as shown in Figure 3(b) and Figure 3(c) respectively.



(a) Rounds vs. client fraction



(b) Rounds vs. epochs



(c) Rounds vs. batch size

Fig. 3: Effect of the client fraction, epochs, and batch size of clients on communication rounds when the threshold of testing perplexity is set to be 90 and small-scaled GRU-based language model is used

F. Magnitude of Randomization

The federated learning framework focuses on the privacy of the input data using distributed training on each client side to protect the user's privacy. To further the privacy preservation of the decentralized training, we evaluate the magnitude of normal noise in the randomization mechanism on model parameters. Comparative experiments are conducted to analyze the effect of the magnitude on the testing perplexity. The results are shown in Table III with both randomized and nonrandomized settings. For the randomized version, four values of magnitude are chosen, i.e., 0.001, 0.005, 0.01, and 0.05.

As shown in the table, a very small noise on both the methods does not affect the performance. Actually, the testing perplexity for the randomized setting is slightly better than the result of nonrandomized setting. With a larger noise, the performance becomes worse. For our proposed method, the testing perplexity is always lower than its counterpart FedAvg, showing that our method can resist a larger noise and can better preserve privacy to ensure the perplexity of next-word prediction.

TABLE III: Magnitude of randomization vs. testing perplexity using a small-scaled model with tied embedding

Randomization		FedAvg	FedAtt
Nonrandomized	$\beta = 0$	88.21	77.66
Randomized	$\beta = 0.001$	88.17	77.76
	$\beta = 0.005$	88.36	78.59
	$\beta = 0.01$	89.74	79.51
	$\beta = 0.05$	103.17	101.82

G. Scale of Model

Distributed training depends on communication between the server and clients, and the central server needs to optimize on the model parameters for the aggregation of the clients models. Thus, the central server will have a higher communication cost and computational cost when there are a larger number of clients and the local models have millions of parameters.

The size of the vocabulary in most language modeling corpus is very large. To save training costs, the embedding weights and output weights are tied which can reduce the number of trainable parameters [17], [18]. We compared three scales of client models with the word embedding dimensions of 300, 650 and 1500. Two versions of the tied and untied models are used. In the tied setting, the dimension of the RNN hidden state must be the same as the embedding dimension.

The results of the model's scales on the testing perplexity are shown in Table IV. The tied large-scale model achieves the best results for both FedAvg and FedAtt and the tied model is better than the untied model of the same scale. Our proposed method achieves lower testing perplexity in four out of the six settings, i.e., tied and untied small model, tied medium model, and tied large model. For the other two settings, the testing perplexity of our method is slightly higher than FedAvg. Overall, for real-world keyboard applications in practice, the tied embedding can be used to save the number of trainable parameters and the communication cost while achieving a better performance.

V. Conclusion

Federated learning provides a promising and practical approach to learning from decentralized data while protecting the private data with differential privacy. Efficient decentralized learning is significant for distributed real-world applications such as personalized keyboard word suggestion on mobile phones, providing a better service and protect user's private personal data.

To optimize the server aggregation by federated averaging, we investigated the model aggregation and optimization on the central server in this paper. We proposed a novel layer-wise

TABLE IV: Testing perplexity of 50 communication rounds vs. the scale of the model using a tied embedding or untied embedding model

Model		FedAvg	FedAtt
Small	tied	88.21	77.66
	untied	91.25	81.31
Medium	tied	103.07	77.41
	untied	96.67	96.71
Large	tied	77.51	76.37
	untied	82.97	83.40

attentive federated optimization for private neural language modeling which can measure the importance of selected clients and accelerate the learning process. We partitioned three popular datasets, i.e., Penn Treebank and WikiText-2 for the prototypical language modeling task, and Reddit comments from a real-world social networking website, to mimic the scenario of word-level keyboard suggestions and performed a series of exploratory experiments. Experiments on these datasets show our proposed method outperforms its counterparts in most settings.

ACKNOWLEDGEMENT

This research is funded by the Australian Government through the Australian Research Council (ARC) under grants LP150100671 partnership with Australia Research Alliance for Children and Youth (ARACY) and Global Business College Australia (GBCA).

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[2] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[3] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

[4] Eunice Kim, Jung-Ah Lee, Yongjun Sung, and Sejung Marina Choi. Predicting selfie-posting behavior on social networking sites: An extension of theory of planned behavior. *Computers in Human Behavior*, 62:116–123, 2016.

[5] Kenneth C Arnold, Krzysztof Z Gajos, and Adam T Kalai. On suggesting phrases vs. predicting words for mobile text composition. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 603–608. ACM, 2016.

[6] Hongmei He, Tim Watson, Carsten Maple, Jörn Mehnen, and Ashutosh Tiwari. A new semantic attribute deep learning with a linguistic attribute hierarchy for spam detection. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3862–3869. IEEE, 2017.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[9] Vadim Popov, Mikhail Kudinov, Irina Piontkovskaya, Petr Vytovtov, and Alex Nevidomsky. Distributed fine-tuning of language models on private data. In *International Conference on Learning Representation (ICLR)*, 2018.

[10] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Federated tensor factorization for computational phenotyping. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 887–895. ACM, 2017.

[11] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4427–4437, 2017.

[12] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[13] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[14] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519, 2017.

[15] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[16] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[17] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

[18] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163, 2017.

[19] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[21] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[22] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics*, 4(1):259–272, 2016.

[23] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[24] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*, 2017.

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[26] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[27] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[28] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[29] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[30] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.