World Wide Web (2020) 23:2653–2669
https://doi.org/10.1007/s11280-019-00775-w

Decentralized Knowledge Acquisition for Mobile Internet Applications

Jing Jiang[1] · **Shaoxiong Ji**[2] · **Guodong Long**[1]

Received: 17 March 2019 / Revised: 11 September 2019 / Accepted: 23 December 2019 /
Published online: 13 March 2020

Abstract
Mobile internet applications on smart phones dominate large portions of daily life for many people. Conventional machine learning-based knowledge acquisition methods collect users' data in a centralized server, then train an intelligent model, such as recommendation and prediction, using all the collected data. This knowledge acquisition method raises serious privacy concerns, and also violates the rules of the newly published General Data Protection Regulation. This paper proposes a new attention-augmented federated learning framework that can conduct decentralized knowledge acquisition for mobile Internet application scenarios, such as mobile keyboard suggestions. In particular, the attention mechanism aggregates the decentralized knowledge which has been acquired from each mobile using its own data locally. The centralized server aggregates knowledge without direct access to personal data. Experiments on three real-world datasets demonstrate that the proposed framework performs better than other baseline methods in terms of perplexity and communication cost.

Keywords Federated learning · Mobile internet applications · Decentralized knowledge acquisition

This article belongs to the Topical Collection: *Special Issue on Application-Driven Knowledge Acquisition*
Guest Editors: Xue Li, Sen Wang, and Bohan Li

✉ Guodong Long
guodong.long@uts.edu.au

Jing Jiang
jing.jiang@uts.edu.au

Shaoxiong Ji
shaoxiong.ji@aalto.fi

[1] Central for Artificial Intelligence, School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, Australia

[2] Department of Computer Science, Aalto University, Espoo, Finland

1 Introduction

According to a report in 2017 [29], the majority of respondents spent five hours or more a day on their smartphones daily. The increasing popularity of smartphones and mobile wearable devices, such as smart watches has generated a massive amount of distributed data, such as text messages, click behavior, travel trajectories and health status. Effectively using these data will provide additional advantages for these application service providers.

Intelligent functions, such as recommendation and personalized suggestions, can greatly improve the quality of services and provide a more satisfying experience with the application. A conventional intelligent approach usually collects big data into a centralized server, then applies a machine learning algorithm to train an intelligent model. The personalized intelligent model is also trained in the centralized sever to leverage data from many other users. This centralized knowledge acquisition method raises serious privacy concerns for end users who have no idea as to how the company will use their data. One of the best ways to address this issue is to store the data on local nodes rather than upload private data onto the central server. This takes into account the recently published General Data Protection Regulation (GDPR) that lays out strict rules for data usage alongside protection that will bring new challenges for existing centralized knowledge acquisition methods. Decentralized knowledge acquisition generates knowledge in each node locally. It is critically important in protecting sensitive data and preserving privacy. As most existing intelligent models are trained using machine learning algorithms, we discuss decentralized knowledge acquisition in the machine learning scenario. In general, most distributed machine learning algorithms can be categorized into two types: data-distributed and model-distributed. The first splits the data into many nodes and trains the models in distributed nodes locally. Then, the central sever acquires knowledge from these distributed nodes by aggregating their model parameters. The model-distributed method copies the entire data to many nodes and splits a big model into many pieces to be allocated to nodes accordingly. A piece of the model will be trained in each node and the central server acquires knowledge by integrating many of these pieces.

In recent years, federated learning [16] has attracted broad interest from the research community[4, 7]. The fundamental technology is derived from distributed machine learning. However, federated learning focuses on solving the non-IID problem which commonly exists in the mobile internet environment. In particular, the data distribution of each mobile user is different. Both distributed machine learning and federated learning still need a centralized server to aggregate acquired knowledge. Using a centralized server to aggregate knowledge is much more efficient and incurs lower communication costs across a large-scale network. Some applications have been developed using this federated learning framework: [11] recommending photos a mobile user would be the most likely to share on social websites, [2] suggesting the next word for mobile keyboards and [8] ranking notifications by importance and classifying spam messages.

Most existing federated learning methods use only a centralized aggregated model to store acquired knowledge which is basically common knowledge with limited information. This paper proposes a novel, federated learning-based, decentralized knowledge acquisition framework by using an attention-augmented mechanism to model aggregation, one that suits the capture of both common and special knowledge of all distributed nodes. A centralized aggregation model is used to store the common knowledge of all distributed nodes and an attention mechanism is designed as an augmentation tool to capture the special knowledge for each. In particular, the aggregation model has shared parameter settings for all nodes while the attention mechanism automatically attends to the weights or scores of the relation

between the central agent and various distributed nodes. The attention score is then taken to minimize the expected distance between the central agent and distributed nodes. The proposed method considers the relation between the server model and client models: their weights and can optimize the distance between the central node and distributed nodes in the parameter space to learn a well-generalized server model.

Our contributions in this paper are as follows:

- This is the first paper whereby attention mechanism is applied to augment the decentralized knowledge acquisition framework. The attention mechanism is designed specifically to capture the special knowledge of each distributed node, using an attention score to represent the relation between the central node and each distributed node.
- In the server optimization, we propose a novel attention-based Federated loss that can simultaneously learn the aggregated modes and attention scores between central node and distributed nodes.
- The proposed framework can solve privacy issues from two perspectives. In the first step, data are stored securely on local devices and model training is on-device. To protect the model parameters from inverse engineering, differential privacy via randomization is employed on the transferred model parameters.
- The experiment results demonstrate the effectiveness of the proposed method on a mimic mobile internet application using real-world datasets that include the data of private neural language modeling data and social media.

The paper's remaining parts are organized as follows. Section 2 discusses the related works by reviewing federated learning and attention mechanism. Section 3 introduces the proposed attentive federated aggregation. Section 4 details the experiments undertaken and our conclusion is drawn in Section 5.

2 Related work

The decentralized knowledge acquisition in this paper relates to the federated learning framework. Our proposed method using the attention mechanism augments this framework.

2.1 Federated learning

Federated learning was first proposed by McMahan et al. [16] to decouple training procedures from data collection by an iterative averaging model. It can train a model from decentralized data using distributed training and communication-efficient learning, and the decentralized learning framework has great potential to develop various applications with a focus on privacy preservation. For example, [7] applies a differential privacy-preserving mechanism on distributed nodes to protect the privacy of the data from gradient-based reverse engineering. The federated learning model can be fine-tuned to learn a language model from the decentralized data [21]. Many applications and new settings have been re-designed using the federated learning framework. For example, [12] proposed federated-based tensor factorization that could be applied to discover medical concepts from electronic health records; [4] combines federated learning with meta-learning for recommendation; [30] introduces federated knowledge transferring for online social care; and [24] tackles multi-task learning using federated learning.

With regard to mobile Internet applications, communication efficiency is one of the performance metrics used to evaluate a deployed decentralized learning algorithm, such as federated learning techniques. To improve communication efficiency, [13] proposed updates - both structured and sketched - to reduce up-link communication costs; [1] proposed quantized gradient compression and the encoding of stochastic gradient descent to improve the efficiency of communication and [25] used ternary gradients to reduce the communication cost.

2.2 Attention mechanism

The attention mechanism is designed to orient perception using numerical vectors and has attracted broad interest in the field of computer vision. Mnih et al. [18] apply attention onto recurrent neural network (RNN) to tackle image classification tasks. In [3], the authors apply the attention mechanism in a sequence-to-sequence neural network of natural language processing tasks that include neural machine translation. Luong et al. [14] extended attention-based RNNs and proposed two new attention mechanisms - global and local. The attention mechanism can also be used in convolution neural networks for sentence encoding. For example, ABCNN [27] illustrates this for modeling sentence pairs. [26] proposed hierarchical attention networks for document classification tasks. Shen et al. proposed directional self-attention for language understanding [33, 34].

3 Methodology

In this section, we introduce the preliminaries of federated learning, a decentralized knowledge acquisition framework. Building on this, we propose an attentive federated optimization algorithm to simultaneously learn attentive weights for each distributed node and aggregated model's parameters. As for the client learner, we apply the gated recurrent unit (GRU) [5] as the client model for language modeling. Furthermore, we add a randomized mechanism [7] for learning differential private client models.

3.1 Federated learning framework

Federated learning framework is composed of two components: central server and distributed nodes. The central server accumulates common knowledge using model aggregation among the model parameters collected from many distributed nodes. This is similar to the works on meta-learning by learning a good initialization for quick adaptation [6, 19], graph meta-knowledge [31, 32] and transfer learning by transferring knowledge between domains [20]. The algorithm of federated learning framework [16] comprises two parts - server optimization in Algorithm 1 and local training in Algorithm 2.

Updating server model in federated learning The server firstly chooses a client learning model and initializes the parameters of the client learner before setting the clients' fraction. It waits for online clients for local model training. Once the selected number of clients has finished the model update, it receives the updated parameters and performs 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.

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

Updating the client model in federated learning Some clients are selected to receive the parameters from the server model, after which each client performs secure local training on their own devices using their private data. Stochastic gradient descent is performed to update the supervised learning-based local model, e.g. GRU-based language modeling as introduced below in Section 3.5. Once the training of the local model has been completed - usually in several epochs - the clients send the parameters of their newly trained local models to the central server over a secure connection. During this local training, clients' confidential data can be stored on their own devices, thereby protecting their privacy.

Algorithm 2 Train a local model on clients.

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

3.2 Federated learning-based knowledge acquisition

The ultimate goal of federated learning-based knowledge acquisition is to find the shared knowledge that is the best fit for all distributed nodes. Specifically, the best fit is defined as that which offers the greatest help in training the local model across all the distributed nodes. To provide this help, the federated learning framework sends the model parameters from the centralized server model to the distributed nodes, then each node uses the received common knowledge to initialize the parameters of the local model. To acquire and accumulate the

common knowledge, the central server aggregates the model parameters collected from the distributed nodes.

During the model aggregation step in the central server, each node's knowledge (special knowledge) has a scale of contribution different from the common knowledge. How to define the different scale is a crucial in the setting of model aggregation. One of the simplest settings is use an equal contribution for each node. The original federated learning uses the data size of each node to measure the contribution. This paper proposes to use the attention score to measure the contribution, the details of which are given below.

3.3 Attentive model aggregation in federated learning

In this paper, a novel definition of the knowledge acquisition objective is proposed in the federated learning optimization framework. A new aggregation method is designed next to accumulate common knowledge effectively from decentralized client models. The proposed aggregation method is Attentive Federated Aggregation (`FedAttOpt`) in a jointly optimization manner. It begins by introducing the attention mechanism for federated aggregation by aggregating the contribution of the local models of the selected clients with the global model in the central server. An illustration of our proposed attentive federated aggregation is shown in Fig. 1 where the left box represents the distributed client models and the right box represents the attentive aggregation in the central server. Each box has a series of w_i to represent the i-th layer's parameters of a deep neural network-based model. This illustration shows only a single iteration t where the central server updates the common knowledge by aggregating model parameters weighted with attention score, namely attentive federated updating.

The thinking behind the proposed attentive federated updating is to find the optimal solution of common knowledge that can offer the most help to all client nodes. To achieve this, the shared model parameters should be easily generalized to all the clients models to minimize their loss. However, the loss of each model is difficult to infer due to the fine-tuning process in the local model. There is a well-known empirical assumption: if the model parameters' initialization is close to the global optimum, the model is more likely to converge to

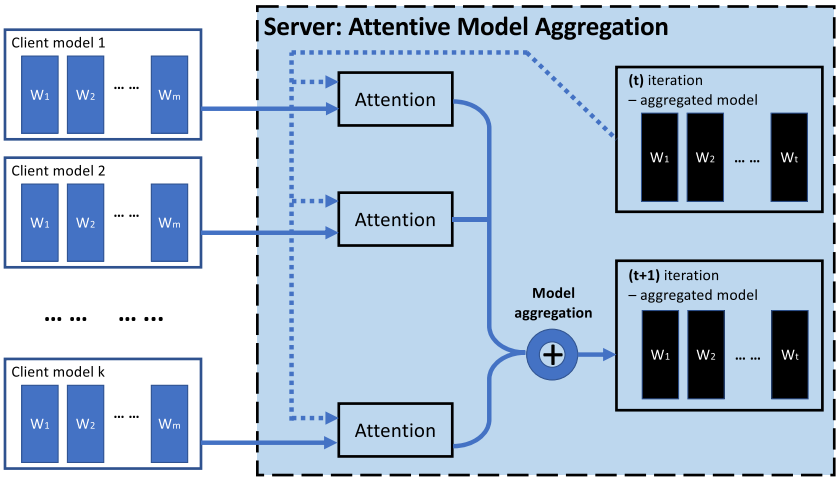

Figure 1 The illustration of our proposed layer-wise attentive federated aggregation in the t-th iteration

the global optimum by updating the parameters with training data and learning function. The objective of the overall optimization is to minimize loss function as below, namely Attentive Federated Loss.

$$L = \sum_{k=1}^{m} [\alpha(\theta, \theta_k) * d(\theta, \theta_k)] = \sum_{k=1}^{m} [\alpha_k * d_k], \tag{1}$$

where a_k is the attention score for center to attend node k, and d_k is the distance of model parameters θ between center and node k. In particular, the θ is a vector by concatenating the parameters from the multi-layer neural network. The attention score is defined as below.

$$a_k = \frac{e^{s_k}}{\sum_{j=1}^{n} e^{s_j}} \tag{2}$$

where s_k is the Scaled Dot-Product of two vectors: θ and θ_k as below.

$$s_k = \frac{\theta^T \theta_k}{\sqrt{n}} \tag{3}$$

The distance between two models' parameters are define as p-norm.

$$d_k = ||\theta_t - \theta_{t+1}^k||_p, \tag{4}$$

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 attention score to measure the importance of the client models. To facilitate the illustration, we set $p = 2$ in the following discussion.

3.4 Optimization for the proposed model

As discussed above, the proposed Attentive Federated Loss is to the weighted distance between the server model and the client models by taking a set of self-adaptive scores as the attentive weights. However, the current objective function includes two co-related components: attention score a_k and distance d_k between central model and local models. To facilitate the calculation of gradients, the objective function in (1) could be transformed to its log loss.

$$L' = \log L = \sum_{k=1}^{m} (\log a_k + \log d_k) \tag{5}$$

For the given loss function in (5), the proposed attentive Federated learning framework is to minimize the loss as below optimization function.

$$\theta = \arg\min_{\theta} L = \arg\min_{\theta} L' \tag{6}$$

We could use stochastic gradient descent in (7) to solve the optimization function with logarithmic loss L'.

$$\theta_{t+1} \leftarrow \theta_t - \epsilon \frac{\nabla L'}{\nabla \theta}, \tag{7}$$

where ϵ is the step size.

The gradient $\nabla L'/\nabla\theta$ can be calculated as below function:

$$\nabla = \frac{\nabla L'}{\nabla \theta} = \sum_{k=1}^{m} [\frac{1}{a_k} * \frac{\nabla a_k}{\nabla \theta} + \frac{1}{d_k} * \frac{\nabla d_k}{\nabla \theta}], \tag{8}$$

where m is the total number of distributed nodes. The two parts of the gradient can be calculated as below according to (2) and (4).

$$\frac{\nabla a_k}{\nabla \theta} = \frac{a_k}{\sqrt{n}}\theta_k \tag{9}$$

$$\frac{\nabla d_k}{\nabla \theta} = \sum_{k=1}^{m} 2(\theta_t - \theta_{t+1}^k) \tag{10}$$

For the selected group of m clients, we perform a gradient descent to update the parameters of the global model in (7). 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 model.

Algorithm 3 Server optimization.

k is the ordinal of clients; k 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 user k **do**
        $d_k = \|\theta - \theta_k\|^2$ ▷ E.q. 4
        $s_k = \frac{\sqrt{n}}{\theta^T \theta_k}$ ▷ E.q. 3
    $\alpha_k = softmax(s_k) = \frac{e^{s_k}}{\sum_{k=1}^{m} e^{s_k}}$ ▷ E.q. 2
    $\frac{\nabla a_k}{\nabla \theta} = \frac{a_k}{\sqrt{n}}\theta_k$ ▷ E.q. 9
    $\frac{\nabla d_k}{\nabla \theta} = \sum_{k=1}^{m} 2(\theta_t - \theta_{t+1}^k)$ ▷ E.q. 10
    $\frac{\nabla L'}{\nabla \theta} = \sum_{k=1}^{m} [\frac{1}{a_k} * \frac{\nabla a_k}{\nabla \theta} + \frac{1}{d_k} * \frac{\nabla d_k}{\nabla \theta}]$ ▷ E.q. 8
    $\theta_{t+1} \leftarrow \theta_t - \epsilon \frac{\nabla L'}{\nabla \theta}$ ▷ E.q. 7
    return θ_{t+1}

The advantages of our proposed attentive federated aggregation and its optimization are: 1) The aggregation of the client models is fine-grained considering the similarity between the client model and the server model in the parameter space. 2) The learned attention scores and central model's parameters form a summarized knowledge acquired from distributed knowledge and can be further applied to assist each node to train the local model.

3.5 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 language modeling as the target task to mimic the NLP-based applications on mobile phones. Therefore, we use the gated recurrent unit (GRU) [5] to learn the language modeling on the client side. The GRU is a well-known and simpler variant of the long short-term memory (LSTM) [9], reached by merging the forget gate and the input gates into a single gate, along with both the cell state and the hidden state. In the GRU-based neural language model, words or tokens are first 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 expressed as:

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

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.

3.6 Disturbing the transmitted parameters using differential privacy

To protect the client's data from an inverse engineering attack, we apply the randomized mechanism to federated learning. This ensures differential privacy on the client side without revealing the client's data [7]. Initially, this differentially private randomization was proposed as an application for federated averaging, where a white noise with the mean of 0 and standard deviation of σ is added to the client parameters in (11).

$$\hat{\theta}_{t+1}^{(k)} = \theta_{t+1}^{(k)} + \beta\mathcal{N}(0, \sigma^2)) \tag{11}$$

Here, we add magnitude coefficient $\beta \in (0, 1]$ on the randomization of normal noise to control the effect of the randomization mechanism on the performance of the federated aggregation. The $\hat{\theta}_{t+1}^{(k)}$ will be sent to the server for model aggregation.

4 Experiments

We conducted experiments 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. Federated learning-based knowledge acquisition is a totally new area where the design and background settings are different to conventional knowledge acquisition methods used with mobile internet applications. Our proposed work is a theoretical method based on the the federated learning framework, an area in which there are very few prior works. Therefore, we used some public datasets to mimic the situation of a knowledge acquisition method across the decentralized environment, e.g. a mobile internet application.

4.1 Datasets

We mimic a mobile internet application using neural language modeling tasks. Experiments are conducted on three English language datasets including Penn Treebank [15], WikiText-2 [17] and the Reddit Comments from Kaggle. Language modeling is one of the most suitable tasks with which to validate federated learning. It offers a large number of datasets to test performance and there is a real-world application in the form of the input keyboard application on smart phones.

Penn Treebank is an annotated English corpus. We use the data derived from Zaremba et al.[1] [28]. The WikiText-2 is available online.[2] The May 2015 Reddit Comments dataset is

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

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

Table 1 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

a portion of a large scale dataset of Reddit comments[3] from the popular online community – Reddit. It is available in the Kaggle Datasets.[4] We sampled 1‰ of the comments from this dataset to train our private language model as a representative of social networks data. Table 1 shows the statistical information of three datasets - the number of tokens in the training, validation, and testing set.

4.2 Baseline algorithms

We conducted to several groups of experiments for comparison, for example, performance with different model aggregation methods, the scale of client models, communication cost, and so forth. Two baselines are applied in these comparisons, i.e., `FedSGD` and `FedAvg`. The 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 weighted by data-size. It samples a fraction of users for each iteration and each client can perform several steps of gradient descent. The model aggregation is an average weighted by each client's data-size.
3. `FedAttOpt`: Federated Attentive learning. It treats attention mechanism as a component of the optimization procedure in Federated learning.

4.3 Experiment settings and data partitioning

Experiment settings We conducted experiments in the setting of federated learning using GRU-based private neural language modeling with Nvidia TITAN Xp GPU acceleration. The GRU-based client model first 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-embeddings and a hidden state of the 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 its communication cost. Tied embedding shares the weights and biases in both embedding and output layers to reduce greatly the number of trainable parameters.

Data partitions To mimic the mobile Internet application in a real-world scenario, we preprocess the data by partitioning these three public datasets into many small subsets and letting each represent the private dataset of one mobile client. First, we shuffle all the samples in the dataset. Second, we mimic a number of clients and their private datasets by

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

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

randomly collecting (without replacement) a few samples from the whole dataset. In this way the whole dataset is split into a number of shards to mimic the same number of users or clients. In particular, these three datasets were partitioned into the 100 subsets to mimic 100 mobile Internet clients who will participate in the training of federated learning.

4.4 Experiment analysis

Our proposed model uses the attention mechanism as an augmentation tool to capture more knowledge than the other federated learning algorithms in the decentralized knowledge acquisition scenario. We conduct experiments on these three datasets processed to mimic mobile Internet applications, and compare our method with other federated learning baseline methods. In general, the method with more knowledge is likely to achieve better performance on intelligent applications. Therefore, we use testing perplexity as the evaluation metric since perplexity is a standard measurement for probability distribution in natural language modelling-related tasks. It is one of the most commonly used metrics for word-level language modeling, one that we choose to mimic mobile Internet application. In particular, the perplexity of a distribution is defined as

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

where $H(p)$ represents the entropy of $p(x)$ - the distribution of probability among the prediction labels. In a language modeling task, a lower perplexity indicates a better prediction result is more likely.

We take 50 rounds of communication between the server and clients and compare the performance on the validation set to select the best model, then we test the performance on the testing set to obtain its perplexity. The results of testing perplexity of all three datasets are shown in Table 2. For FedAvg, 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 FedAttOpt outperforms others in terms of testing perplexity in all the three datasets other than WikiText-2 with fraction C as 0.5. When the client fraction C is 0.1 and 0.5 in the Penn Treebank and PTB respectively, the method we proposed here achieves 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 0.3, our proposed method obtains the best testing perplexity of 64.52 in the WikiText-2 dataset.

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 the additional exploratory experiments discussed in the following subsections, we explore the client fraction, the communication costs, the effect of different randomizations, and the scale of the models.

Table 2 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	244.49
0.1	FedAvg	95.27	138.13	243.76
	FedAttOpt	72.25	116.49	129.87
0.5	FedAvg	79.75	128.24	244.83
	FedAttOpt	75.58	117.82	212.37

4.5 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 Fig. 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 result 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 lower perplexity than FedAvg for all settings of client fraction.

4.6 Communication cost

Communication cost for parameter uploading and downloading between the clients and server is another important issue for decentralized learning. Communication, wired and wireless, depends heavily on Internet bandwidth and has an impact on the performance of federated optimization. To conserve 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 because 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: client fraction, epochs and batch size of client training. The small-scale language model is used as the client model with 10% of clients are selected for model aggregation. We set the testing perplexity for the termination of federated training at 90. When the testing perplexity

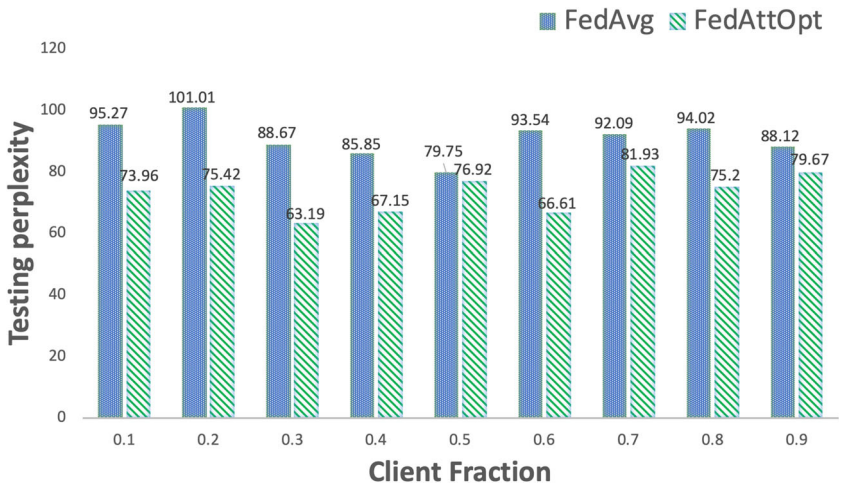

Figure 2 Testing perplexity of 50 communication rounds when a different number of clients are selected for federated aggregation

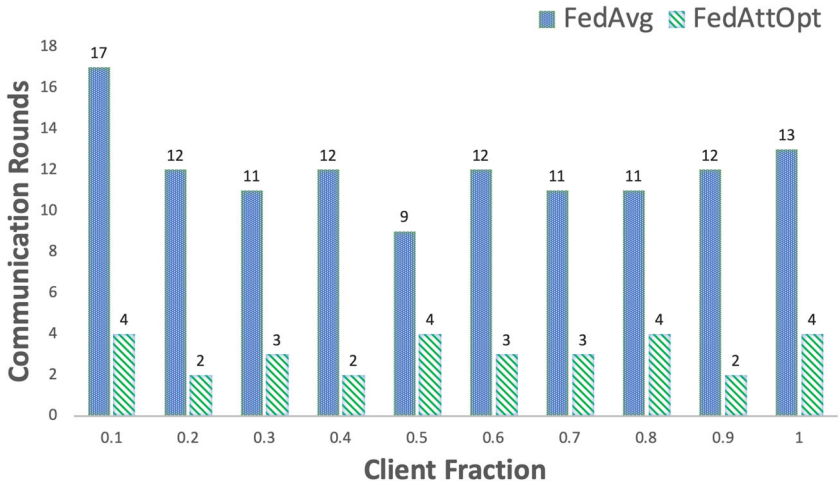

Figure 3 Rounds comparison by client fraction

is falls below that threshold, federated training comes to an end and we take the rounds of training as the communication rounds. As shown in Fig. 3, the communication round during training fluctuates when the number of clients increases. Furthermore, our proposed method is always better than FedAvg and incurs a lower communication cost. In most of cases, our proposed method saves half of the 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. FedAttOpt achieved a comparable communication cost with different epoch values and the batch size of local training, as shown in Figs. 4 and 5 respectively.

4.7 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 enhance the privacy preservation of decentralized training, we evaluate the magnitude of normal noise in the randomization

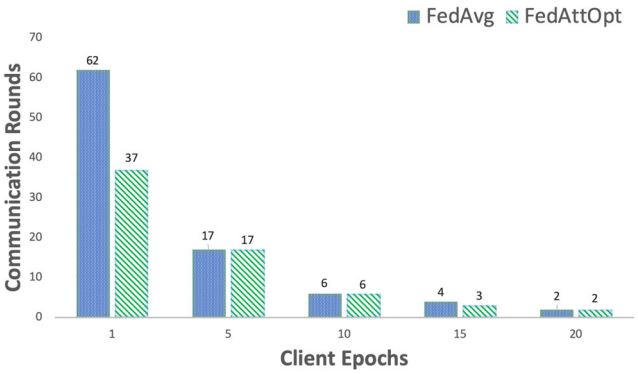

Figure 4 Rounds comparison by epochs

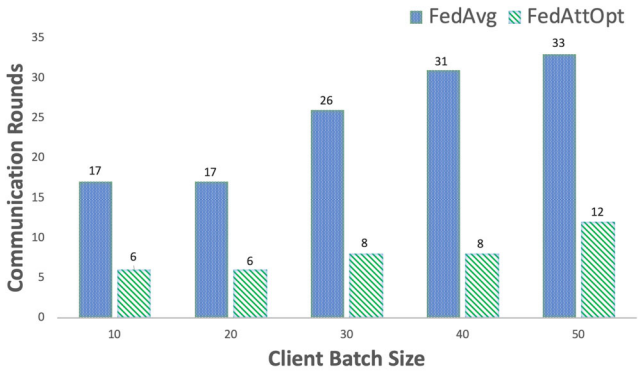

Figure 5 Rounds comparison by batch size

mechanism on the model parameters. Comparative experiments are conducted to analyze the effect of the magnitude on testing perplexity. The results are shown in Table 3 with both randomized and non-randomized settings. For the randomized version, four values of magnitude are chosen: 0.001, 0.005, 0.01, and 0.05.

As shown in the table, a very small noise does not affect the performance of either method. Actually, the testing perplexity in the randomized setting is slightly better than the result with the non-randomized setting. With increased noise, the performance deteriorates. For our proposed method, the testing perplexity is always lower than its counterpart FedAvg, showing that our method can resist greater noise and can better preserve privacy to ensure the perplexity of next-word prediction.

4.8 Scale of model

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

The vocabulary in most language modeling corpora is very extensive. To save training costs, the embedding weights and output weights are tied to reduce the number of trainable parameters [10, 22]. 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.

Table 3 Magnitude of randomization vs. testing perplexity using a small-scaled model with tied embedding

Randomization		FedAvg	FedAttOpt
Nonrandomized	$\beta = 0$	88.21	72.31
Randomized	$\beta = 0.001$	88.17	71.60
	$\beta = 0.005$	88.36	72.27
	$\beta = 0.01$	89.74	73.71
	$\beta = 0.05$	103.17	108.58

Table 4 Testing perplexity of 50 communication rounds vs. the scale of the model using a tied embedding or untied embedding model

Model		FedAvg	FedAttOpt
Small	tied	88.21	71.60
	untied	91.25	71.88
Medium	tied	103.07	76.21
	untied	96.67	73.80
Large	tied	77.51	74.49
	untied	82.97	72.85

The results of the model's scales on testing perplexity are shown in Table 4. The tied large-scale model achieves the best results with all three methods and the tied model is usually better than its untied counterpart of the same scale. Our proposed method achieves lower testing perplexity in five of the six settings, i.e., tied and untied in small model and large model, untied medium model. For the tied medium settings, the testing perplexity of our method is higher than all baselines. Overall, for real-world keyboard applications in practice, tied embedding can be used to conserve the number of trainable parameters, reducing communication cost while achieving a better performance.

5 Conclusion

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

To optimize server aggregation using federated averaging, we investigated the model aggregation and optimization on the central server in this paper. We proposed a novel attentive federated optimization to measure the importance of the knowledge acquired from each client. We partitioned three popular datasets - 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.

References

1. Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: Communication-efficient SGD via gradient quantization and encoding. In: Advances in Neural Information Processing Systems, pp. 1709–1720 (2017)
2. Arnold, K.C., Gajos, K.Z., Kalai, A.T.: On suggesting phrases vs. predicting words for mobile text composition. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, pp. 603–608. ACM (2016)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014)
4. Chen, F., Dong, Z., Li, Z., He, X.: Federated meta-learning for recommendation. arXiv:1802.07876 (2018)

5. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: 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), pp. 1724–1734 (2014)
6. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. arXiv:1703.03400 (2017)
7. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv:1712.07557 (2017)
8. He, H., Watson, T., Maple, C., Mehnen, J.: As Tiwari: A new semantic attribute deep learning with a linguistic attribute hierarchy for spam detection. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 3862–3869. IEEE (2017)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neur. Comput. **9**(8), 1735–1780 (1997)
10. Inan, H., Khosravi, K., Socher, R.: Tying word vectors and word classifiers: A loss framework for language modeling. arXiv:1611.01462 (2016)
11. Kim, E., Lee, J.-A., Sung, Y., Choi, S.M.: Predicting selfie-posting behavior on social networking sites: An extension of theory of planned behavior. Comput. Hum. Behav. **62**, 116–123 (2016)
12. Kim, Y., Sun, J., Yu, H., Jiang, X.: Federated tensor factorization for computational phenotyping. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 887–895. ACM (2017)
13. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492 (2016)
14. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421 (2015)
15. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The Penn Treebank. Comput. Linguis. **19**(2), 313–330 (1993)
16. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
17. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. arXiv:1609.07843 (2016)
18. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems, pp. 2204–2212 (2014)
19. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv:1803.02999 (2018)
20. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
21. Popov, V., Kudinov, M., Piontkovskaya, I., Vytovtov, P., Nevidomsky, A.: Distributed fine-tuning of language models on private data. In: International Conference on Learning Representation (ICLR) (2018)
22. Press, O., Wolf, L.: 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, vol. 2, pp. 157–163 (2017)
23. Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. arXiv:1709.04696 (2017)
24. Smith, V., Chiang, C.-K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems, pp. 4427–4437 (2017)
25. Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., Li, H.: Terngrad: Ternary gradients to reduce communication in distributed deep learning. In: Advances in Neural Information Processing Systems, pp. 1509–1519 (2017)
26. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: 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, pp. 1480–1489 (2016)
27. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Trans. Assoc. Comput. Linguis. **4**(1), 259–272 (2016)
28. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv:1409.2329 (2014)
29. Counterpoint Research @ Statista Inc.: Time spent on smartphone everyday worldwide in 2017 (in hours). https://www.statista.com/statistics/781692/worldwide-daily-time-spent-on-smartphone/ (2017)
30. Ji, S., Long, G., Pan, S., Zhu, T., Jiang, J., Wang, S., Li, X.: Knowledge transferring via model aggregation for online social care. arXiv preprint arXiv:1905.07665 (2019)
31. Liu, L., Zhou, T., Long, G., Jiang, J., Zhang, C.: Learning to propagate for graph meta-learning. In: Advances in Neural Information Processing Systems, pp. 1037–1048 (2019)

32. Liu, L., Zhou, T., Long, G., Jiang, J., Zhang, C.: Attribute Propagation Network for Graph Zero-shot Learning. In: Thirty-Fourth AAAI Conference on Artificial Intelligence (2020)
33. Shen, T., Zhou, T., Long, G., Jiang, J., Zhang, C.: Bi-directional block self-attention for fast and memory-efficient sequence modeling. In: International Conference on Learning Representations (2018)
34. Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)