FedAssistant: Dialog Agents with Two-side Modeling

Haoran Li[*], **Ying Su**[*], **Qi Hu**, **Jiaxin Bai**, **Yilun Jin** and **Yangqiu Song**[†]
Department of Computer Science and Engineering, HKUST
{hlibt, ysuay, qhuaf, jbai, yilun.jin}@connect.ust.hk, yqsong@cse.ust.hk

Abstract

The success of large pre-trained neural generative conversation models benefits from publicly available datasets of various sources. However, some high-quality datasets held by separated companies and organizations are not exploited yet due to security and privacy concerns. In this work, we demonstrate a framework named FedAssistant to address the above issue by training neural dialog systems in a federated learning setting. Our framework can be trained on multiple data owners with no raw data leakage during the process of training and evaluating the models. Moreover, our proposed FedAssistant with two-side modeling can be easily deployed to any user of all data providers. In order to reduce the communication cost between data holders and the parameter server, FedAssistant further implements Top-k gradient sparsification. We conduct quantitative experiments to evaluate the performance of FedAssistant on various domains, and our experiment shows that FedAssistant can obtain lower perplexity on all testing data with acceptable communication cost.

1 Introduction

Dialogue systems play an important role in daily life and are widely used for recommendation, question answering, online customer services, and social chatbots. There are mainly two types of machine learning based dialogue systems: 1) utterance retrieval models that select responses from a given database and 2) neural generative language models that improvise their responses according to contexts. It is commonly believed that high-capacity generative language models trained on large datasets are the future trends for dialogue generation related tasks. Large pre-trained language models, especially GPT-2 [Radford *et al.*, 2019], obtain excellent performance on both task-oriented dialogue tasks [Ham *et al.*, 2020a; Budzianowski and Vulić, 2019a] and open-domain chit-chat chatbots [Zhang *et al.*, 2020].

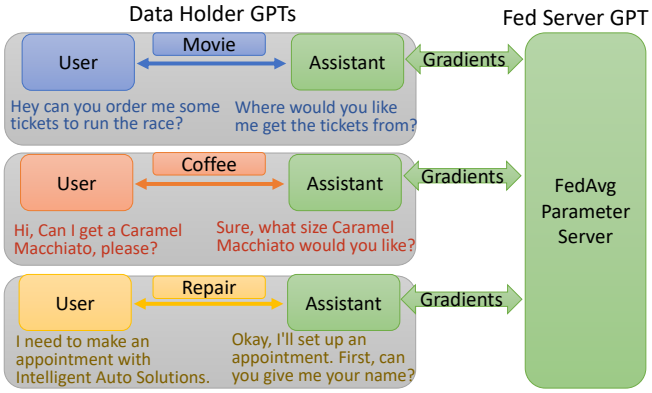

Figure 1: Overview of FedAssistant framework. Here Movie, Coffee and Repair are 3 examples of data holders.

Recently, Google proposed LaMDA [Thoppilan *et al.*, 2022] that incorporated knowledge grounding and prompts to language models to achieve superb performance on both task-oriented and chit-chat tasks. To train those language models, various sources of publicly available datasets have been used. However, for high-quality private datasets that are held by individual companies, institutes, and organizations, current language models still cannot have access to them due to the data silos. This is caused by strict laws like the EU's GDPR[1] that forbid companies to share personal data without consent. Moreover, even for sharing de-identified data, it is still possible to re-identify participants through data re-identification with auxiliary data [Narayanan and Shmatikov, 2008]. Therefore, owners of sensitive data like medical records are not willing to share their raw data directly. On the other hand, most data holders are also service providers and they indeed have the incentive to share their data to improve their services for users. For example, dialog models are able to answer more comprehensive and complicated questions after training on data from multiple knowledge domains. Hence, a better method for training language models on multiple data owners without revealing the raw data should be considered.

Federated machine learning [Yang *et al.*, 2019] has been proposed when training machine learning models with data silos and the privacy of different sources should be preserved.

[*] Equal contribution.
[†] Corresponding author.

[1] https://gdpr-info.eu

Most federated machine learning frameworks utilize a central server to perform variants of FedAvg Algorithm [Konečný *et al.*, 2016] that uses the weighted or unweighted average of clients' model parameters or gradients to update a global model and then the global model sends the update back to its corresponding clients. It is promising to consider training large pre-trained language models in a federated manner, so that the raw data from multiple owners can be utilized in dialogue agent while maintaining their data privacy at the same time. However, there are three main problems to consider. First, giant models typically consist of more than one hundred million model parameters thus updating client and global models iteratively is impractical due to huge communication costs. Second, the generative property of dialogue neural models can also be a constraint for service users: to produce a response with given context, the generative models have to improvise word by word, which is time-consuming for mobile users with limited computational resources. Third, simply deploying dialogue neural models to service providers may avoid the second problem, but it requires users to transfer their plaintexts to the server. These texts may become a potential threat to the users' privacy since private information can be mined from some sensitive content.

To overcome the above problems, we propose a two-side dialogue modeling framework based on the GPT-2 model, FedAssistant, where only hidden states are required to transfer between the two GPT-2 models with the FedAvg algorithm. FedAssistant exploits the transformer architecture to model next utterances where partial hidden states of the other side can be regarded as contexts. Figure 1 depicts an example of FedAssistant with three data holders: Movie, Coffee, Repair and each of them owns the conversations between users and assistants. Every data holder uses two GPT-2 models for modeling users' and assistants' utterances separately and FedAvg is only performed for assistants' GPT-2. As a result, data holders can better obtain their own auto-response assistants to further improve their services and users are able to obtain replies without sending raw utterances through one simple round of user side GPT-2 inference. In summary, we highlight the following contributions of FedAssistant.

- FedAssistant can avoid plaintext transmission during training and inference states. Only past keys and values of all transformer blocks are required as contexts for response generation.
- FedAssistant views data holders as user service providers and allows simple model deployment to their users that do not need to generate explicit responses word by word locally.
- Lastly, FedAssistant can perform well with the Top-k gradient sparsification technique to reduce the communication cost.

2 Related Works

In this section, we first introduce the details of FedAvg algorithm and approaches for gradient compression. Then several works about task-oriented dialog agents are presented.

2.1 FedAvg and Gradient Compression

FedAvg [McMahan *et al.*, 2017] combines local SGD [Ketkar, 2017] on each client and model averaging on a server, which makes the distributed modeling training under federated learning setting applicable. Each client needs to download an entire model and upload an updated model again. The communication cost of this process could be very high due to slow and unreliable network connections. To address this problem, researches on gradient compression in distributed SGD have been proposed can be classified into two streams, including unbiased estimate for gradients and biased gradient compression.

Approaches of unbiased estimate for gradients include Quantized SGD [Alistarh *et al.*, 2017] and sparsification [Lin *et al.*, 2017]. The Quantized SGD aims to trade-off between the communication cost and convergence guarantees. The sparsification method maintains the unbiasedness of sparsified stochastic gradient by dropping some coordinates of the gradient and amplify the remaining ones. However, the requirement for these methods is too stringent for empirical application.

Another direction is biased gradient compression that includes signSGD [Bernstein *et al.*, 2018] and DGC sparsification [Lin *et al.*, 2017]. SignSGD utilizes the sign of stochastic gradient to perform a 1-bit compressed communication between server and clients. Previous gradient sparsifications focus on reducing the size of transmitted gradients. [Strom, 2015; Aji and Heafield, 2017] proposed to send gradients or absolute values of gradients larger than a constant threshold and [Dryden *et al.*, 2016] proposed to send a fixed portion of gradients. Different from SignSGD, DGC sparsification sends the magnitude of gradients greater than a threshold while keep accumulating the local gradients in the meantime.

In this paper, we focus on the training of task-oriented dialog agents under the biased gradient estimate stream. To simplify the communication process, we adopt a top-k sparsification strategy similar to the idea in [Dryden *et al.*, 2016]. Each time the clients and server communicate with a fixed proportion of gradients selected by magnitude.

2.2 Dialog Agent

Task-oriented dialog assistants are very common in real-world now, such as Google Home, Apple Siri and Microsoft Cortana. Typically a traditional task-oriented dialog agent is built based on a pipeline architecture, including four parts, a natural language understanding (NLU) module [Sarikaya *et al.*, 2014; Ravuri and Stolcke, 2016], a dialog state tracking (DST) module [Henderson *et al.*, 2013; Mrkšić *et al.*, 2015], a dialog policy module [Lipton *et al.*, 2018; Chen *et al.*, 2018] and a natural language generation (NLG) module [Wen *et al.*, 2015]. The modules are built seperately which does not require an overall optimized performance.

Different from previous methods, current studies attempt to build end-to-end dialogue systems to complete the tasks by employing transfer learning framework based on large pre-trained language models [Madotto *et al.*, 2018; Lei *et al.*, 2018]. [Wolf *et al.*, 2019] incorporates a large-scale pre-trained language model to build a chit-chat dialog system. [Budzianowski and Vulić, 2019b] bypasses explicit pol-

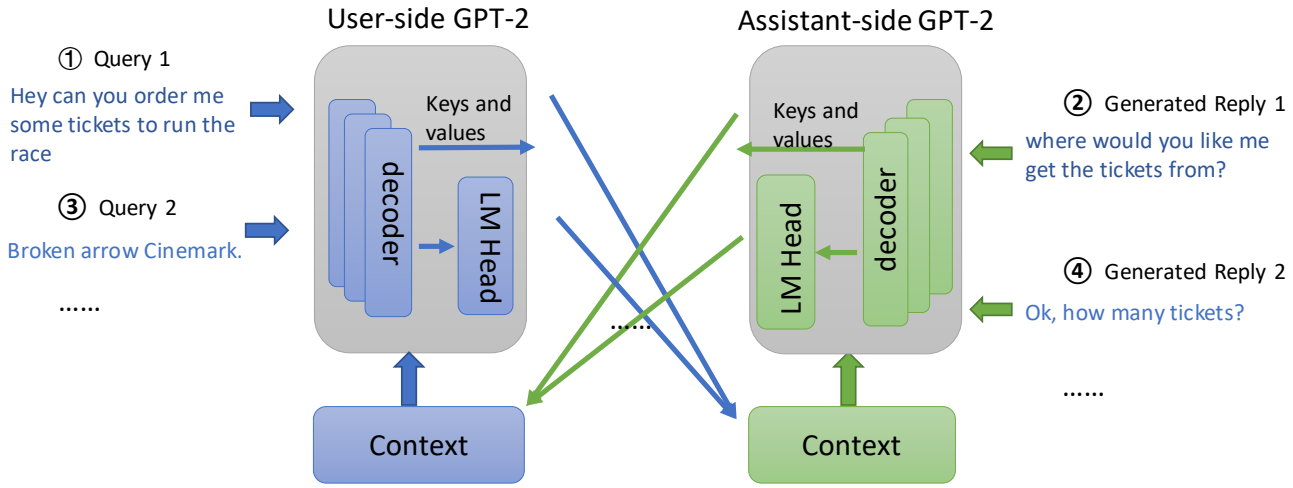

Figure 2: Two-side modeling of FedAssistant. For any data holder, the user-side GPT-2 and the assistant-side GPT-2 model utterances of users and assistants, respectively. Keys and values of all transformer decoder blocks are transmitted as context for modeling the next utterance. Therefore, no raw text is passed between the user side and the assistant side.

icy and language generation modules by employing a dialogue model that operates solely on text input. [Ham *et al.*, 2020b] leverages GPT-2 and trains the model following the traditional dialog management pipeline. Recent researches continue making significant progress on the task oriented dialog benchmarks [Raffel *et al.*, 2019; Yang *et al.*, 2020; Peng *et al.*, 2021; Lee, 2021; Lu *et al.*, 2021].

Our work follows the the idea in [Ham *et al.*, 2020b] that we also build an end-to-end dialog system by using GPT-2 model. The difference is that we aim to leverage multiple data sources while keeping the data privacy under the federated setting. In order to realize this, we propose a two-side modeling module of dialogue agent.

3 FedAssistant Framework

In this section, we present detailed descriptions of FedAssistant. We first give the problem setup in Section 3.1. Then we give a detailed model architecture explanation for two-side modeling in Section 3.2. Finally, we will introduce federated learning for assistant side models in Section 3.3.

3.1 Problem Formulation

We define the set of data holders as $\mathcal{D} = \{D_1, D_2, ..., D_N\}$ where N is total number of data holders. For each data holder D_i, it owns a dataset of conversations $\mathcal{C}_i = \{c_i^1, c_i^2, ..., c_i^{n_i}\}$ where n_i refers to size of dataset of D_i. Each conversation consists of a sequence of utterances between its users and assistants. Our goal is to train an auto-response assistant that can chit-chat comprehensively for every data holder D_i.

3.2 Two-side Modeling

To achieve the above goal, we propose FedAssistant, a two-side modeling framework based on GPT-2 models without revealing any utterances. For any data holder D_i, it initializes two GPT-2 models, classified as a user-side GPT-2 and an assistant-side GPT-2, respectively. In practice, during inference, the auto-response assistant is responsible for answering users' queries. Hence we only need to care about the generation of the assistant-side GPT-2. For any conversation c_i^j of D_i, the user-side GPT-2 only models the users ' utterances while the assistant-side GPT-2 is responsible for modeling utterances from the assistant during training or generating replies based on the previous context during inference.

Conventionally, to train GPT-2 with multi-round utterances, all previous utterances will be concatenated together separated by <eos> tokens as context for training current utterances with language modeling (LM) [Bengio *et al.*, 2003]. Given current utterance $U = \{w_0, w_1, ..., w_{|U|-1}\}$ and previous context c, the objective of LM is to maximize the likelihood over every word token of U:

$$\mathcal{L}(U) = \sum_{i=1}^{|U|} \log(P(w_i|c, w_0, w_1, ..., w_{i-1})). \quad (1)$$

Maximizing the likelihood $\mathcal{L}(U)$ is done by minimizing the cross-entropy loss between generated probabilistic distribution and ground truth utterance U with teacher forcing [Williams and Zipser, 1989]. In particular, GPT-2 consists of a stack of transformer decoder blocks [Vaswani *et al.*, 2017]. For any position with its query (q), and all positions' keys (K) and values (V), the attention is computed by masking the future positions, which can be formulated as:

$$\text{Attention}(q, K, V) = \text{Softmax}\left(\frac{\text{Mask}(qK^\top)}{\sqrt{d_k}}\right)V, \quad (2)$$

where d_k refers to the dimension of keys and queries.

FedAssistant applies conventional approaches for training both user-side and assistant-side GPT-2. Moreover, FedAssistant can further avoid the raw utterances transmission between the user-side and the assistant-side GPT-2, by exploiting past keys and values of transformer blocks of GPT-2, as illustrated in Figure 2. One key observation of FedAssistant is that all previous keys and values are sufficient as context

for modeling the current utterance, since future keys and values are masked for transformer decoder blocks. For example, to model the third utterance, keys and values of all word tokens of first and second utterances are required. This allows FedAssistant to train its GPT-2 models by past keys and values transmission without revealing raw data. Moreover, it can speed up GPT-2's training and inference since the model does not need to recompute previous unchanged contexts. To train both GPT-2 models, in the beginning, the user-side GPT-2 models its first utterances U_0 and sends its computed keys and values to the assistant side as context c. Then the assistant-side GPT-2 models the response U_1 based on context c and transmits keys and values of both U_0 and U_1 to the user-side as the updated context. This process continues repeatedly until all utterances are fed to the models, and each model updates its own parameters through language modeling. If the assistant-side starts the conversation, then the user-side models U_0 and the training procedure is similar. Unlike previous works that update GPT-2 by all utterances, FedAssistant requires the user-side to train its model based only on users' utterances while the assistant-side only updates its parameters for assistants' utterances. After training, the user-side GPT-2 can be distributed to all users and its data holder holds the assistant-side GPT-2.

The advantage of the proposed two-side modeling is obvious. First, it requires no raw data transmission between the user side and the assistant side, so sensitive conversations are kept private. Second, unlike other GPT-2 implementations, users with limited resources don't have to generate response word by word locally and only one forward pass is required. The whole generation process is left to the assistant-side GPT-2. Third, [Wu *et al.*, 2021] showed that by using alternating memory recurrence, lower perplexity and better generation quality could be achieved. This suggests that two-side modeling brings no harm to language models.

3.3 Federated Learning for the Assistant Side

To further enhance the generation performance of FedAssistant, we perform FedAvg algorithm for all assistant-side GPT-2 models with a parameter server. All data holders initialize their user-side and assistant-side models with the same parameters (that can be pre-trained from a large corpus). After training a batch of conversations, each data holder sends its assistant-side update to the parameter server, and the server averages the updates to update server parameters. Then the updated server parameters will be sent back to all clients to start a new round of training for each data holder. The training stops until certain epochs are reached.

However, FedAvg for GPT-2 can cause huge communication costs since GPT-2 usually consists of hundreds of millions of trainable parameters. For each round of FedAvg, the parameter server needs to download previous weights and upload averaged new weights to all data holders. Thus, the parameter server bandwidth is likely to be the bottleneck for federated learning. To reduce the communication cost for the parameter transmissions, FedAssistant adapts top-k gradient sparsification [Dryden *et al.*, 2016] that is widely used for distributed optimization algorithms. For every data holder and parameter server, top-k gradient sparsification chooses k largest magnitude gradients and only k of them are used for FedAvg. Algorithm 1 describes federated learning with top-k sparsification for FedAssistant. The gradient on the server side is initialized as 0. In each iteration, top-k gradients are selected on the client side for uploading to server and top-k gradients are selected on the server side after averaging local gradients for downloading to clients.

Algorithm 1: FedAvg with top-k gradient sparsification

Input: clients C_1, ..., C_N
Input: round b for local training
Input: the number of clients N
Input: optimization function SGD
Input: training steps m
Input: init parameters $w = \{w[0], w[1], ..., w[N-1]\}$

$G^0 \leftarrow 0$;
for $t = 1$ **to** m **do**
  $G_k^t \leftarrow G_k^{t-1}$;
  for $i = 1$ **to** b **do**
    Sample dialog data x from client k;
    $G_k^t \leftarrow G_k^t + \frac{1}{Nb} \nabla f(x; w_t)$;
  end
  for $j = 1$ **to** N **do**
    $\hat{G}_k^t[j] \leftarrow topk(G_k^t[j])$;
  end
  All-reduce $G_k^t : G^t \leftarrow \frac{1}{N} \sum_{k=1}^{N} encoder(\hat{G}_k^t)$;
  $G^t \leftarrow topk(G^t)$;
  $w_{t+1} \leftarrow SGD(w_t, G^t)$;
end

4 Experiment

In this section, we present extensive comparative experiments for our proposed FedAssistant framework. In Section 4.1, we will introduce the setup for dataset and training. Then we will compare FedAssistant with other 4 GPT-2 based model setup in Section 4.2. Moreover, we will analyze how top-k gradient sparsification can affect the performance of FedAssistant in Section 4.3. Finally, we give the communication cost for FedAvg in Section 4.4.

4.1 Experimental Setup

Dataset. We evaluate our result on the Google Taskmaster-1 dataset [Byrne *et al.*, 2019] which consists of woz-dialogues and self-dialogues. Woz-dialogues are collected by recording two paid workers' conversations while self-dialogues are written by a single worker imaging the conversations between two people. Taskmaster-1 includes conversations between users and assistants of six domains: ordering pizza (Pizza), creating auto repair appointments (Repair), setting up ride service (Ride), ordering movie tickets (Movie), ordering coffee drinks (Coffee) and making restaurant reservations (Restaurant). We merge all conversations of woz-dialogues and self-dialogues and separate them according to dialogues' corresponding domains.

	Repair	Coffee	Movie	Restaurant	Pizza	Ride	All
GPT_local	10.10	10.23	9.72	12.62	9.59	9.87	22.16
GPT_all	9.54*	9.17*	9.01*	11.65*	8.65*	8.99*	9.51*
FedA_both	11.31	10.44	10.13	13.57	9.85	10.48	**10.96**
FedA	**10.62**	**10.36**	**9.94**	**13.40**	**9.74**	**10.07**	16.14
FedA_freeze	104.77	16.24	14.81	20.29	14.92	14.99	28.70

Table 1: Evaluation results on the testing data. Perplexity (PPL) is used as the metric for evaluating the performance of six data holders. "All" means using all the testing data to evaluate the overall performance and average the perplexity for all six data holders.

Training details. We treat each domain as one data holder and randomly split the data for training, validation and testing. Each domain has roughly 1,800 training dialogues, 200 validation and 200 testing dialogues. For model initialization, we use pre-trained DialoGPT-small (117M) [Zhang *et al.*, 2020] trained on Reddit data. The model performance is evaluated by perplexity (PPL) for testing data. For optimization, we use AdamW optimizer [Loshchilov and Hutter, 2019] and the learning rate is 3e-5 with linear warm-up and decay. Each data holder trains its data for 10 epochs. We use a single machine and warp each data holder with its own data and FedAssistant into a class, so that every data holder cannot access the data of others. For each data holder, 1 Nvidia V100 is used to train its data.

4.2 Evaluation on Model Architecture

For our experiment, We consider the following model architectures:

GPT_local: for each data holder, train one GPT-2 model with its own training data locally.

GPT_all: only train one GPT-2 model for all training data of 6 data holders.

FedA_both: for each data holder, train two-side modeling GPT-2 models with its own training data and perform FedAvg for both the user side and the assistant side.

FedA: for each data holder, train two-side modeling GPT-2 models with its own training data and perform FedAvg for the assistant side only.

FedA_freeze: for each data holder, train two-side modeling GPT-2 models with its own training data and perform FedAvg for only the assistant. The parameters of the user-side GPT-2 are frozen.

Both GPT_local and GPT_all can be considered as baselines by comparing two-side modeling with conventional modeling. FedA is the FedAssistant framework we propose in Section 3. FedA_both evaluates the performance for performing FedAvg on the user side as well. FedA_freeze freezes the training parameters for the user side, hence users are not required to update the user side model anymore. Intuitively, this approach leads better transferability with poorer evaluation results.

The evaluation result based on perplexity is shown in Table 1. To test the generalized ability, we also consider testing models based on all the testing data. GPT_all has the best performance over each data holder and all testing data, since the model is trained on the conventional approach with all the training data. GPT_local achieves the second-best evaluation result on each data holder, however, there is an obvious gap between the generalized ability on all testing data and adaptability on individual domains. Also, by comparing the gap between individual domains and all testing data, we can conclude that the data distribution in each domain is not identical. FedA_both obtains the lowest perplexity among all variants of FedAssistant for all testing data, which indicates that performing FedAvg on the user-side indeed improves FedAssistant's generalization. However, performing FedAvg for the user-side is not practical when the user-side model is deployed to the users. Compared with GPT_local, FedAssistant has similar performance for individual testing data and lower perplexity for all testing data. This justifies that FedAssistant can perform well on local data while being able to generalize via FedAvg. The perplexity of FedA_feeze on Repair is surprisingly high. We investigate the Repair data and find that users' phone numbers are frequently given for some users' utterances. Freezing parameters on the user side can lead to high perplexity when modeling users' utterances with phone numbers. And by training the user-side, the exploding perplexity can be reduced.

Overall, the above experiments demonstrate that FedAssistant can have competitive performance on individual data holder with better generalized ability.

4.3 Evaluation on Top-k

Here we analyze the experiments on perplexity after applying top-k gradient sparsification for both both FedA_freeze and FedA. Top-k gradient sparsification can significantly reduce the communication cost while sacrificing most minor updates. It is a trade-off to balance between utility and ability. We use No_TopK to indicate FedAssistant without using Top-k gradient sparsification and No_TopK_freeze to denote FedA_freeze without using Top-k gradient sparsification. We consider optimizing top 9.75M ($\frac{1}{12}$), 4.68M ($\frac{1}{25}$), 2.34M ($\frac{1}{50}$), 1.17M ($\frac{1}{100}$) of transmitted gradients for both the user side and the assistant side, namely, Top$\frac{1}{12}$, Top$\frac{1}{25}$, Top$\frac{1}{50}$ and Top$\frac{1}{100}$ for FedAssistant. Roughly speaking, we are cutting the transmitted gradients by half every time. We use the same data split for all data holders.

Table 2 depicts the evaluation results. Not surprisingly, FedAssistant without Top-k obtain the best performance for individual domains and all testing data. For FedAssistant, it is more robust towards Top-k gradient sparsification. As updated gradients are reduced by half, testing perplexity increases no more than 1 for most local data holders. Compared with reduced cost, the minor decrease in perplexity seems acceptable. However, for FedA_freeze, the performance drop is obvious: for Restaurant, perplexity increases

	Repair	Coffee	Movie	Restaurant	Pizza	Ride	All
No_TopK_freeze	104.77	16.24	14.81	20.29	14.92	14.99	28.70
No_TopK	10.62	10.36	9.94	13.40	9.74	10.07	16.14
No_TopK_A*	9.06	7.91	8.42	11.61	7.68	7.81	10.66
Top$\frac{1}{12}$_freeze	108.64	18.85	17.12	23.99	16.87	17.71	31.54
Top$\frac{1}{12}$	11.79	11.27	10.93	14.87	10.54	11.11	19.05
Top$\frac{1}{12}$_A*	10.61	9.11	9.84	13.79	8.73	9.12	13.93
Top$\frac{1}{25}$_freeze	111.13	20.45	18.69	26.55	18.28	19.37	33.40
Top$\frac{1}{25}$	12.37	11.80	11.58	15.70	10.978	11.75	20.92
Top$\frac{1}{25}$_A*	11.45	9.81	10.72	15.07	9.31	9.92	16.27
Top$\frac{1}{50}$_freeze	114.36	22.43	20.60	29.86	19.96	21.45	35.75
Top$\frac{1}{50}$	12.92	12.28	12.18	16.66	11.41	12.48	22.82
Top$\frac{1}{50}$ _A*	12.25	10.50	11.61	16.50	9.93	11.94	18.78
Top$\frac{1}{100}$_freeze	118.66	24.98	23.07	34.13	22.16	24.13	38.79
Top$\frac{1}{100}$	13.40	12.74	12.78	17.59	11.75	13.29	24.58
Top$\frac{1}{100}$_A*	13.00	11.14	12.48	17.94	10.40	11.76	21.21

Table 2: Evaluation results on overall perplexity of top-k gradient sparsification for FedAssistant_freeze, FedAssistant and assistant side models' perplexity. Suffix '_freeze' is used for FedA_freeze and '_A' is used for assistant side models' perplexity of FedAssistant. Top $\frac{1}{12}$, $\frac{1}{25}$, $\frac{1}{50}$, $\frac{1}{100}$ gradient sparsifications are used for all three settings as well as no top-k. "All" indicates all the testing data is used to evaluate the overall performance and the result is averaged.

from 20.3 to 34.1. For all testing data, when the fraction of updated gradients is larger than 50, both models fail to outperform GPT_local. This implies that top-k gradient sparsification with a ratio larger than $\frac{1}{50}$ might be a proper choice for FedAssistant. By comparing FedAssistant models with different Top-k ratios, we show that FedAssistant tends to be more robust towards top-k gradient sparsification.

Evaluation on the assistant side. According to the problem setting of FedAssistant, data holders are more interested in the assistant side for providing better services. Intuitively, FedAvg on the assistant side should improve the generalized ability of the assistant-side GPT-2. To better evaluate how FedAssistant performs, it is crucial to analyze the perplexity only on the assistant-side models.

Table 2 also includes the assistant side's perplexity for FedAssistant, named with suffix "_A". Compared with the perplexity of all utterances on FedAssistant and FedA_freeze, assistant side models achieve the best performance for all top-k and no top-k settings except the minor increase ($\sim$0.4) on the Restaurant with Top$\frac{1}{100}$. By comparing overall perplexity with assistant side utterances' perplexity, it shows that FedAssistant is able to better improve the assistant-side models and hence the FedAvg algorithm contributes to the improvement of the assistant side. Moreover, for all testing data, FedAvg can bring significant improvement for the assistant side, even the top $\frac{1}{50}$ gradient sparsification can achieve lower perplexity than GPT_local for all testing data.

The above comparison experiment shows the effectiveness of FedAvg on the assistant side with lower perplexity for both top-k and no top-k scenarios.

4.4 Communication Cost

The communication cost is acceptable for every data holder. When we apply top-k gradient sparsification for 2% parameters (Top$\frac{1}{50}$), the communication cost for any data holder of each round of FedAvg is around 18 Mb. This cost makes federated learning feasible for training large pre-trained language models.

5 Conclusion and Future Work

In this work, we present FedAssistant, a well-performed and federated two-side modeling framework for neural dialogue models. First, we exploit the past keys and values of transformer decoder blocks to avoid raw data leakage for both training and evaluation. Then we implement a two-side modeling architecture to model utterances according to the roles of speakers separately and distributedly. What's more, we apply FedAvg algorithm to further enhance the generalized ability of the assistant-side models for better customer services. To address the high communication cost for uploading and download parameters, we implement top-k gradient sparsification for our FedAvg algorithm. We conduct extensive comparison experiments to demonstrate the effectiveness of both two-side modeling and FedAvg with top-k gradient sparsification.

However, certain problems still remain. The first problem is related to federated learning with the non-IID problem of various domains. As the experiment shows, after FedAvg, though FedAssistant can obtain better performance on all testing data, it cannot obtain improvement over its local data. Another problem comes from FedA_freeze, which has more transferability than FedAssistant for all users and is more promising in practice. In our experiment, for the Repair data, FedA_freeze with exploded perplexity exemplifies the problem. In the future, we plan to apply a more dedicated and frozen pre-training method for every user-side model while improving the FedAvg algorithm to address the non-IID problem. We will also try to distill the model with fewer trainable parameters and apply quantization with less computational precision to further reduce the communication cost.

References

[Aji and Heafield, 2017] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[Alistarh *et al.*, 2017] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30:1709–1720, 2017.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Bernstein *et al.*, 2018] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[Budzianowski and Vulić, 2019a] Paweł Budzianowski and Ivan Vulić. Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, Hong Kong, November 2019. Association for Computational Linguistics.

[Budzianowski and Vulić, 2019b] Paweł Budzianowski and Ivan Vulić. Hello, it's gpt-2-how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, 2019.

[Byrne *et al.*, 2019] Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. Taskmaster-1:toward a realistic and diverse dialog dataset. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, Hong Kong, 2019.

[Chen *et al.*, 2018] Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1257–1268, 2018.

[Dryden *et al.*, 2016] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pages 1–8. IEEE, 2016.

[Ham *et al.*, 2020a] Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the ACL*, pages 583–592. ACL, 2020.

[Ham *et al.*, 2020b] Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, 2020.

[Henderson *et al.*, 2013] Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, 2013.

[Ketkar, 2017] Nikhil Ketkar. Stochastic gradient descent. In *Deep learning with Python*, pages 113–132. Springer, 2017.

[Konečný *et al.*, 2016] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[Lee, 2021] Yohan Lee. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303, 2021.

[Lei *et al.*, 2018] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, 2018.

[Lin *et al.*, 2017] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[Lipton *et al.*, 2018] Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Loshchilov and Hutter, 2019] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[Lu *et al.*, 2021] Yujie Lu, Chao Huang, Huanli Zhan, and Yong Zhuang. Federated natural language generation for personalized dialogue system. *arXiv preprint arXiv:2110.06419*, 10 2021.

[Madotto *et al.*, 2018] Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*, 2018.

[McMahan *et al.*, 2017] 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. PMLR, 2017.

[Mrkšić *et al.*, 2015] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David

Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*, 2015.

[Narayanan and Shmatikov, 2008] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[Peng *et al.*, 2021] Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. Soloist: Buildingtask bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824, 2021.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[Raffel *et al.*, 2019] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[Ravuri and Stolcke, 2016] Suman Ravuri and Andreas Stolcke. A comparative study of recurrent neural network models for lexical domain classification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6075–6079. IEEE, 2016.

[Sarikaya *et al.*, 2014] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, 2014.

[Strom, 2015] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[Thoppilan *et al.*, 2022] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, volume 30, 2017.

[Wen *et al.*, 2015] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

[Williams and Zipser, 1989] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[Wolf *et al.*, 2019] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*, 2019.

[Wu *et al.*, 2021] Qingyang Wu, Yichi Zhang, Yu Li, and Zhou Yu. Alternating recurrent dialog model with large-scale pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1292–1301, Online, April 2021. Association for Computational Linguistics.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM TIST*, 10(2):12:1–12:19, 2019.

[Yang *et al.*, 2020] Yunyi Yang, Yunhao Li, and Xiaojun Quan. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *arXiv preprint arXiv:2012.03539*, 2020.

[Zhang *et al.*, 2020] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of ACL: System Demonstrations*, pages 270–278. ACL, 2020.