Personalized Federated Learning for Text Classification with Gradient-Free Prompt Tuning

Anonymous ACL submission

Abstract

In this paper, we study personalized federated learning for text classification with Pretrained Language Models (PLMs). We identify two challenges in efficiently leveraging PLMs for personalized federated learning: 1) *Communication.* PLMs are usually large in size, *e.g.*, with hundreds of millions of parameters, inducing huge communication cost in a federated setting. 2) *Local Training.* Training with PLMs generally requires back-propagation, during which memory consumption can be several times that of the forward-propagation. This may not be affordable when the PLMs are trained locally on the clients that are resource constrained, *e.g.*, mobile devices with limited access to memory resources. Additionally, the proprietary PLMs can be provided as concealed APIs, for which the back-propagation operations may not be available. In solving these, we propose a training framework that includes an approach of discrete local search for *gradient-free* local training, along with a compression mechanism inspired from the linear word analogy that allows communicating with *discretely* indexed tokens, thus significantly reducing the communication cost. Experiments show that our *gradient-free* framework achieves superior performance compared with baselines.

1 Introduction

Personalized federated learning (Fallah et al., 2020; Chen et al., 2018; Shamsian et al., 2021) involves collaborative training with non-shareable private data from multiple clients. For each client, we aim to train a personalized model that fits to its local data, leveraging knowledge from other clients. Personalized federated learning has been increasingly attended in the federated learning community due to its ability to account for data heterogeneity across clients (Li et al., 2021). On the other hand, the advent of Pretrained Language Models (PLMs) (Liu et al., 2019; Kenton and Toutanova, 2019) has yielded remarkable performance for natural language processing, *e.g.*, text classification. However, such PLMs are usually large in size, *e.g.*, with hundreds of millions of parameters. There has been limited works investigating how to efficiently train with such large PLMs in federated learning scenarios (Guo et al., 2022; Zhao et al., 2022). In this paper, we investigate on efficient training with PLMs in personalized federated learning for the task of text classification.

One challenge of training PLMs in a federated learning scenario is how to reduce communication cost. Federated learning generally requires communicating updated trainable model parameters between a central server and all the clients (McMahan et al., 2017; Li et al., 2020). When training with PLMs, their sheer size may introduce huge communication cost between the server and clients, thus reducing the training efficiency. To solve this problem, recent works propose to leverage prompt tuning (Guo et al., 2022; Zhao et al., 2023). Specifically, prompt tuning learns with a sequence of trainable prompt embeddings inserted into the input layer of the PLMs. By only training and communicating the prompt embeddings and freeze the pretrained parameters of the PLMs, the communication cost is largely reduced compared with training all the parameters of the PLMs. However, in these works prompt tuning is not realistic for federated learning. The main reason is that the local training, *i.e.*, when the PLMs are trained locally on each client, requires back-propagating through the PLMs in order to calculate the gradient of the prompt embeddings. The memory consumption of back-propagating is several times higher (depending on implementation) than that of forward-propagation[1] (Baydin et al., 2022; Belouze, 2022). Such memory consumption is proportional to the

[1] This is because back-propagation requires saving the intermediate results of a computational graph, while the forward-propagation does not.

size of the PLM, *e.g.*, with hundreds of millions of parameters. Therefore, back-propagating with the PLMs can be extremely memory consuming. Unfortunately, the clients in federated learning usually have limited access to the resources (Rabbani et al., 2021; Deng, 2019), *e.g.*, edge devices with limited memory. As a result, the memory footprint during the local training with back-propagation can exceed the memory capacity of the client devices, making the training infeasible. Further, the PLMs may be provided as concealed APIs, for which the back-propagation operation may not be available (Sun et al., 2022b).

To address these issues, we propose a gradient-free training framework that saves both the memory and communication cost in federated learning with PLMs. Specifically, during local training with client data, the PLM is trained via a gradient-free approach of discrete local search with natural language tokens, which saves the memory consumption of back-propagation via only requiring forward pass with PLMs during training. By keeping the prompts from local training to be natural language tokens, we further proposes a compression mechanism that compresses the aggregated prompt embeddings according to linear word analogies (Ethayarajh et al., 2018; Nissim et al., 2020; Drozd et al., 2016). Such a novel compression strategy allows the server and clients to communicate with *discrete* token indices, thus significantly reducing the communication cost. Our contributions are as follows:

- We propose a noval gradient-free personalized federated learning framework for text classification with PLMs. To the best of our knowledge, we are the first to consider gradient-free training in federated learning with PLMs.
- Our framework includes a gradient-free training approach with discrete local search, along with a communication mechanism that allows communicating with discrete token indices.
- Experiments on various datasets show that our gradient-free framework can achieve superior performance, while substantially reducing the communication cost during training.

2 Related Work

Federated Learning with PLMs: Previous works studying PLMs under the federated learning setting mostly consider the training efficiency in terms of the communication cost, but rarely account for memory footprint. For instance, Lit et al. (2022) propose to reduce the communication cost by only communicating the lower layers of the PLMs between server and clients. Inspired by the superior performance and efficiency of prompt tuning (Lester et al., 2021; Liu et al., 2022), (Guo et al., 2022; Zhao et al., 2023; Guo et al., 2023) propose to further reduce the communication cost via only training and communicating the continuous prompt embeddings. The drawback of these works is that they all require gradient computing with back-propagation, which ignores the huge memory consumption caused by back-propagation through the PLMs. As mentioned before, this can be problematic for clients with constrained computation resources, *e.g.*, edge devices with limited memory capacity. Additionally, the PLMs can be served as black-box APIs (Sun et al., 2022b), for which the back-propagation operation may not be available for model training.

Gradient-Free Training with PLMs: Sun et al. (2022b); Cao et al. (2023) assumes the PLMs are concealed in black-box APIs and propose to train the input prompt embeddings of the PLMs with CMA-ES (Hansen and Ostermeier, 2001), a gradient-free method that only requires forward-propagation. This setting is termed Language-Model-as-a-Service (LMaaS), *e.g.*, with GPT 3.5/4 (Koubaa, 2023; OpenAI, 2023), where the client data is transferred to an external server with the API of PLMs. This violates the privacy-preserving principle of federated learning. Sun et al. (2022a) further considers gradient-free training with prompts inserted into the intermediate layer of the PLMs, which contradicts our assumption about black-box APIs. Deng et al. (2022); Diao et al. (2022) model the prompts of the inputs layer of PLMs with a prompt generator that is trained with gradients from reinforcement learning. This may not be suitable for federated learning, since back-propagating with prompt generators (*e.g.*, implemented with another PLM) can introduce additional large memory consumption for clients during local training. Hou et al. (2022); Prasad et al. (2022) also study gradient-free training of PLMs, but it is unclear how to apply their approach for federated learning. To illustrate, Hou et al. (2022) adopts boosting with prompts, requiring ten times the computation for model inference compared to without boosting, thus is not compatible with clients equipped with constrained computation resources. Importantly, none of the above works are studying federated learning.

3 General Setup

Let M be the number of clients in federate learning, and $\{D_i\}_{i=1}^M$ be their local datasets. In personalized federated learning, these datasets are from different domains or tasks. We have $\mathcal{D}_i = \{\boldsymbol{X}_n, \boldsymbol{Y}_n\}_{n=1}^N$, for $i = 1, \ldots, M$ with totally N training samples, where $\boldsymbol{X}_n$ is the n^{th} text sequence and $\boldsymbol{Y}_n$ is its label for text classification. Let $\boldsymbol{f}_i(\cdot)$ be the model for client i, with $\boldsymbol{f}_i(\boldsymbol{X}_n)$ being the predicted probability distribution for $\boldsymbol{X}_n$ over all possible labels in client i. The model $\boldsymbol{f}_i$ is implemented as prompt tuning. Specifically, let $\boldsymbol{H}$ be the pretrained encoder of the PLM and $\boldsymbol{p}_i \in \mathbb{R}^{T\times D}$ represent a sequence of T prompt token embeddings. In experiments, we follow (Sun et al., 2022b) with $T = 50$. D is the dimension of the pretrained token embeddings. $\boldsymbol{f}_i(\boldsymbol{X}_n)$ can be written as,

$$Temp = [\boldsymbol{p}_i; \boldsymbol{e}(\boldsymbol{X}_n); \boldsymbol{e}(It\ is\ [MASK])] \quad (1)$$

$$\boldsymbol{f}_i(\boldsymbol{X}_n) = \mathrm{softmax}(\boldsymbol{H}(Temp) \cdot \boldsymbol{V}_l^T), \quad (2)$$

where $[;]$ denotes row concatenation, $\boldsymbol{p}_i$ is the learnable prompt, $\boldsymbol{e}(\cdot)$ is the embedding layer of the PLM that convert each token in $\boldsymbol{X}_n$ into a token embedding. $\boldsymbol{H}$, and $\boldsymbol{e}$ are frozen during prompt tuning. (1) defines the template for the text classification input, which contains a *[MASK]* token. The output from $\boldsymbol{H}$ on the position of *[MASK]* is compared via inner product with the *verbalizer* $\boldsymbol{V}_l$, which contains embeddings of words that are representative of each label. For instance, we can have $\boldsymbol{V}_l = \boldsymbol{e}([good, bad])$ for sentiment classification.

We see that the only trainable parameter in $\boldsymbol{f}_i(\cdot)$ is the prompt $\boldsymbol{p}_i$. The training loss for client i is,

$$\mathcal{L}(\boldsymbol{p}_i; \mathcal{D}_i) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{cross\_entropy}(\boldsymbol{f}_i(\boldsymbol{X}_n), \boldsymbol{Y}_n), \quad (3)$$

When training with personalized federated learning for text classification, the general objective is to find $\{\boldsymbol{p}_i\}_{i=1}^M$ that minimizes,

$$\frac{1}{M}\sum_{i=1}^{M} \mathcal{L}(\boldsymbol{p}_i; \mathcal{D}_i), \quad (4)$$

while keeping $\{\mathcal{D}_i\}_{i=1}^M$ locally for each client. This is achieved via corrdinating the training with a server that iteratively receives $\{\boldsymbol{p}_i\}_{i=1}^M$ from local training and distribute their aggregated version, denoted as $\boldsymbol{p}$ in Section 4.1. Unlike the PLMs with online APIs (e.g. GPT-3.5/4 (OpenAI, 2023)) that requires uploading user data to an external server, it is reasonable that the PLMs is deployed locally (*i.e.*, without data uploading), for better data privacy with federated learning. Since PLMs can be proprietary due to its costly pre-training (Qiu et al., 2020; Zhou et al., 2023), its parameters (as mentioned above) can be concealed in an API for which back-propagation is not available.

4 Our Framework

4.1 General Procedures

Our proposed framework of federated learning is composed of the following four steps (also shown in Alg 1), which are executed iteratively multiple rounds of federated learning:

- *Local Training:* Each client trains its own $\boldsymbol{p}_i$ with its local data. Section 4.2 introduces our proposed gradient-free approach of discrete local search for $\boldsymbol{p}_i$.
- *Upload*: The learnt $\{\boldsymbol{p}_i\}_{i=1}^M$ is uploaded to the server via converting each $\boldsymbol{p}_i$ to its corresponding index (Section 4.2).
- *Aggregate:* The server aggregates information from different clients by generating a global prompt $\boldsymbol{p}$ from $\{\boldsymbol{p}_i\}_{i=1}^M$ to generate, *i.e.*,

$$\boldsymbol{p} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{p}_i, \quad (5)$$

  where we adopt FedAvg (McMahan et al., 2017) and assume uniform weighting.
- *Download:* $\boldsymbol{p}$ is downloaded to each client as the initialization of local training (with $\boldsymbol{p}_i$) for the next round. Section 4.3 proposes a compression method that approximates $\boldsymbol{p}$ with reduced communication cost (denoted as $\boldsymbol{p}'$).

Note that we assume the API of the PLM has been downloaded to each client before the start of federated learning, so that we only need to communicate the prompts during federated learning. We claim that downloading the API to clients is a practical assumption. This is because it avoids the necessity of uploading client data to an external server (with API) for model inference, compared with the recent Language-Model-as-a-Service (Sun et al., 2022b) where the API is only store on an online server. This is especially important for federated learning where the privacy is of prime concern.

Algorithm 1 Overall Algorithm.

Input: Datasets $\{\mathcal{D}_i\}_{i=1}^{M}$, the PLM (API and its pretrained embedding matrix $\boldsymbol{e}(\mathcal{V})$).
Output: The resulting prompt $\boldsymbol{p}'$.

Initialize $\boldsymbol{p}$ with natural token embeddings.
$\boldsymbol{p} = \boldsymbol{p}' = \boldsymbol{p}'_{-1}$
Download the PLM API and $\boldsymbol{p}'_{-1}$ to each client.
% General procedures for federated learning.
for $r = 1, \cdots, n\_round$ **do**
  % Iterate with the M clients.
  for $i = 1, \cdots, M$ **do**
    % Local Training: Section 4.2, Alg. 3.
    $\boldsymbol{p}_i$ = Local_Training($\boldsymbol{p}'$, $\mathcal{D}_i$)
    % Upload: Section 4.2.
    Upload the index of $\boldsymbol{p}_i$ to the server.
  end for
  % Aggregation: Section 4.1
  Aggregate $\{p_i\}_{i=1}^{M}$ with (5), generating $\boldsymbol{p}$.
  % Download: Section 4.3, Alg. 2
  $\boldsymbol{p}'$ = Compress_Download($\boldsymbol{p}'$, $\boldsymbol{p}'_{-1}$, $\boldsymbol{e}(\mathcal{V})$)
  $\boldsymbol{p}'_{-1} = \boldsymbol{p}'$
end for

4.2 Gradient-Free Local Training

In updating each client i, its prompt $\boldsymbol{p}_i$ is firstly initialized with the global prompt $\boldsymbol{p}$ (or $\boldsymbol{p}'$ in Section 4.3) from the previous round of federated learning, then fine tuned on the local dataset $\mathcal{D}_i$. As mentioned before, gradient-based fine tuning of $\boldsymbol{p}$ with back-propagation can be extremely memory consuming with PLMs. Additionally, the back-propagation operation may not be available for PLMs concealed behind APIs. So motivated, we study gradient-free client update of the prompt $\boldsymbol{p}$, which does not need gradient computation with back-propagation and is compatible with the APIs. Specifically, we propose an update mechanism based on discrete local search with natural language tokens. Let $\mathcal{V}$ be the vocabulary of the PLM and superscript t denote the t^{th} row of a matrix. For each iteration update, given a randomly sampled position of the prompts t, $t \in [1, T]$, and a set of candidate tokens $\boldsymbol{C}(\boldsymbol{p}_i^t) \subset \mathcal{V}$, we update $\boldsymbol{p}_i^t$ via,

$$\boldsymbol{p}_i^t = \underset{\boldsymbol{w}\{\boldsymbol{e}(c)|c\in\boldsymbol{C}(\boldsymbol{p}_i^t)\}}{\operatorname{argmin}} \mathcal{L}(\operatorname{rep}(\boldsymbol{p}_i, \boldsymbol{w}, t), \mathcal{D}_i), \quad (6)$$

Note that $\boldsymbol{p}_i^t$ on the left side is the updated prompt of the next iteration, while the one on the right is that of the previous iteration. Further, $\operatorname{rep}(\boldsymbol{p}_i, \boldsymbol{w}, t)$ denotes replacing the t^{th} row of $\boldsymbol{p}_i$ with $\boldsymbol{w}$. We randomly choose one position t for each update iteration. The candidate set $\boldsymbol{C}(\boldsymbol{p}_i^t)$ is selected with,

$$\boldsymbol{C}(\boldsymbol{p}_i^t) = \underset{\boldsymbol{C}\subset\mathcal{V}, |\boldsymbol{C}|=K}{\operatorname{argmin}} \sum_{c\in\boldsymbol{C}} \cos(\boldsymbol{e}(c), \boldsymbol{p}_i^t), \quad (7)$$

where $\cos(\cdot)$ is the cosine distance. We only select K candidate tokens in $\boldsymbol{C}$ with the most similar semantics as $\boldsymbol{p}_i^t$ (low cosine distance), which avoids large change of $\boldsymbol{p}_i^t$ in a single iteration. K is the number of local search for each step that controls the training efficiency and is discussed in Section 5. The general procedures are shown in Algorithm 3.

Such a simple update mechanism has two benefits. Firstly, since $\boldsymbol{w}$ on the right side of (6) can take the value of $\boldsymbol{p}_i^t$, the value of $\mathcal{L}(\boldsymbol{p}_i, \mathcal{D}_i)$ should be non-increasing during client update. Secondly, by constraining the candidate embeddings to be from the natural language tokens, *i.e.*, $\boldsymbol{C}(\boldsymbol{p}_i^t) \subset \mathcal{V}$, the updated positions of $\boldsymbol{p}_i$ can be saved by only keeping its token index. This significantly reduces the communication cost when uploading prompts to the server, compared with previous works of continuous prompt tuning Guo et al. (2022); Zhao et al. (2022) that upload all the prompt parameters. For instance, the vocabulary size of the Roberta-Large (Liu et al., 2019) model is 50,264 with $D = 1024$, which implies that each token index can be encoded with 16 bits. For positions of $\boldsymbol{p}_i$ that are not modified during client update, we can indicate it with a special index using a 16-bit integer, *e.g.*, 50,265 (not natural token indices). Thus, we only need to upload 16 Bits for each position of p_i. Comparatively, uploading the whole prompt vector to the server requires communicating $16 * 1024 \approx 16$KB for each position, provided that the continuous parameters are encoded into float16 during communication. As the result, we reduce the communication cost by 1000 times (16 Bits *vs* 16 KB).

Note that previous works (Li and Liang, 2021; Liu et al., 2021) claim that discrete tokens are less expressive than continuous tokens, thus the model capacity may be limited when trained with discrete tokens. However, as described in Section 5.1, datasets of different clients in personalized federated learning may represent different domains/tasks. For such cases, training with continuous prompts via joint training may result in the updated $\boldsymbol{p}_i$ to overfit to the domain/task of client i, causing negative knowledge transfer to other clients when $\boldsymbol{p}_i$ is aggregated with (5). In experiments, we will show that our approach can produce better accuracy compared with joint training with continuous prompt embeddings, while also reducing the communication cost.

4.3 Embedding Compression

After the client update, the uploaded $\boldsymbol{p}_i$, for $i = 1 \ldots, M$, are aggregated with (5). We can observe that the results $\boldsymbol{p}$ after aggregation can no longer be represented with a single token index, thus cannot be compressed as in Section 4.2 when being downloaded to clients. Below we propose to compress $\boldsymbol{p}$ after aggregation with the pretrained token embeddings of the PLM, *i.e.*, estimating $\boldsymbol{p}$ with the matrix of pretrained token embeddings $\boldsymbol{e}(\mathcal{V}) \in \mathbb{R}^{|\mathcal{V}| \times D}$.

This draws from the intuition in previous works on linear word analogies (Ethayarajh et al., 2018; Nissim et al., 2020; Drozd et al., 2016), which show interesting examples with linear operations among the pretrained word/token embeddings, *e.g.*, $\boldsymbol{e}(king) - \boldsymbol{e}(man) + \boldsymbol{e}(woman) \approx \boldsymbol{e}(queen)$ or $\boldsymbol{e}(doctor) - \boldsymbol{e}(man) + \boldsymbol{e}(woman) \approx \boldsymbol{e}(nurse)$. These indicate that a pretrained token embedding can be estimated by a few embeddings of tokens with similar or relevant semantics. As for our $\boldsymbol{p}$, its prompt embeddings is assumed to be within the convex hull of the natural token embeddings. This can be observed from (5), *i.e.*, even $\boldsymbol{p}_i$ that is not updated in client i should also be aggregated from natural token embeddings that appeared as updates in previous rounds. Therefore, it should be viable to estimate $\boldsymbol{p}$ with a few or fixed number of natural token embeddings. For each round of federated learning with aggregated prompt $\boldsymbol{p}$, let $\boldsymbol{p}'$ be the prompts received by the clients from the server after compression in the current round. We denote $\boldsymbol{p}'_{-1}$ as the prompts received by the clients after compression in the previous round. Below, we elaborate on how to compress $\boldsymbol{p}$ into $\boldsymbol{p}'$ for the current update round, given $\boldsymbol{p}'_{-1}$ and $\boldsymbol{e}(\mathcal{V})$.

We should note that different from $\boldsymbol{p}$, the compressed $\boldsymbol{p}'_{-1}$ is accessible by both the server and clients, since it was generated by the server and received by the clients. Thus, instead of directly compressing $\boldsymbol{p}$, we only compress the increment (residual) of $\boldsymbol{p}$ between the previous and current rounds. Specifically, for each position t, we define the residual as $\boldsymbol{R}^t = \boldsymbol{p}^t - {\boldsymbol{p}^{t}}'_{-1}$. For each position t, we want to find a sparse projection from $\boldsymbol{e}(\mathcal{V})$ to $\boldsymbol{R}^t$ so it can be represented/estimated with a limited number of pretrained embeddings. Let $\boldsymbol{I}$ be a sequence of token indices, initialized as $\boldsymbol{I} = [1 \cdots, |\mathcal{V}|]$. We define $\boldsymbol{e}(\mathcal{V})_{\boldsymbol{I}}$ be the rows in $\boldsymbol{e}(\mathcal{V})$ indexed by $\boldsymbol{I}$. Formally, we optimize the following,

$$\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x}} \, ||\boldsymbol{e}(\mathcal{V})_{\boldsymbol{I}}^T \cdot \boldsymbol{x} - \boldsymbol{R}^t||_2^2 + \alpha ||\boldsymbol{x}||_1, \quad (8)$$

$$\boldsymbol{I}_x = \text{argmax}_{|\boldsymbol{I}_x|=L} \sum_{j \in \boldsymbol{I}_x} |\boldsymbol{x}^*[j]|, \; \boldsymbol{I} = \boldsymbol{I}[\boldsymbol{I}_x], \quad (9)$$

where $\boldsymbol{I}[\boldsymbol{I}_x]$ is the value of $\boldsymbol{I}$ indexed by $\boldsymbol{I}_x$. $x \in \mathbb{R}^{|\mathcal{I}| \times 1}$ is the learnt projection, $|| \cdot ||_1$ and $|| \cdot ||_2$ are the one and two norms, respectively, and $|\cdot|$ denotes the absolute value. We solve a sparse $\boldsymbol{x}^*$ with LASSO regularization as in (8), with α being the regularization weight. We empirically set $\alpha = 0.2$ for all datasets and clients. $\boldsymbol{x}^*[j]$ is the j^{th} element of $\boldsymbol{x}^*$. Note that (9) takes the top L token indices with the largest absolute projection values in the resulting $\boldsymbol{x}^*$. To minimize the error in estimating $\boldsymbol{R}^t$, the final projection $\boldsymbol{x}_f^* \in \mathbb{R}^{\boldsymbol{I} \times 1}$ is,

$$\boldsymbol{x}_f^* = \text{argmin}_{\boldsymbol{x}_f} \, ||\boldsymbol{e}(\mathcal{V})_{\boldsymbol{I}}^T \cdot \boldsymbol{x}_f - \boldsymbol{R}^t||_2^2. \quad (10)$$

We denote the cardinal of resulting $\boldsymbol{I}$ in (10) as Φ, the number of token embeddings used to approximate $\boldsymbol{R}^t$. Instead of downloading with the aggregated $\boldsymbol{p}$, we download $\{\boldsymbol{I}, \boldsymbol{x}_f^*\}$ to each client. As the result, we only need to download $16 \times 2\Phi$ Bits for each prompt token, consider that both the token index in $\boldsymbol{I}$ and continuous variable in $\boldsymbol{x}_f^*$ are encoded with 16 Bits, as in Section 4.2.

The client will reconstruct the residual $\boldsymbol{R}$ via $\hat{\boldsymbol{R}} = \boldsymbol{e}(\mathcal{V})_{\boldsymbol{I}}^T \cdot \boldsymbol{x}_f$ Finally, the compressed prompt received by the clients for the current round is,

$${\boldsymbol{p}^{t}}' = {\boldsymbol{p}^{t}}'_{-1} + \hat{\boldsymbol{R}}^t, \quad (11)$$

$\boldsymbol{p}' = [{\boldsymbol{p}^{1}}', \cdots, {\boldsymbol{p}^{T}}']$ will be further saved as $\boldsymbol{p}'_{-1}$ for the next round of federated learning. In the experiments, $\boldsymbol{I}$ is selected with two iterations of (8) and (9), as in Algorithm 2.

After the last round of federated learning, we follow (Fallah et al., 2020; Chen et al., 2018) that further fine-tunes $\boldsymbol{p}'$ with a post tuning process for the final $\boldsymbol{p}_i$ (no communication cost). The post tuning is to adapt the resulting $\boldsymbol{p}_i$ to the task/domain of test client i for more personalization. To avoid forgetting of the global knowledge encoded by $\boldsymbol{p}'$, we adopt the gradient-free method of BBT (Sun et al., 2022b) that allows $\boldsymbol{p}'$ being trained in a constrained continuous subspace with a small learning rate. Please refer to Appendix B for more details.

5 Experiments

5.1 Experiment Setting

Training: Following pLF-Bench (Chen et al., 2022), we adopt the datasets of Sentiment140

Method	Upload	Download	BP?
A. Prompt Tuning	0	0	Yes
B. Prompt Tuning (Fed)	819 KB	819 KB	Yes
C. Meta Prompt Tuning (Fed)	819 KB	819 KB	Yes
D. pFedMe	819 KB	819 KB	Yes
E. FedKD	1.3 GB	1.3 GB	Yes
F. Fine Tuning (Fed)	5.3 GB	5.3 GB	Yes
G. BBT	0	0	No
H. BBT (Fed)	8 KB	8 KB	No
I. Ours ($\Phi = 3$)	0.8 KB	4.8 KB	No
G. Ours ($\Phi = 5$)	0.8 KB	8 KB	No
K. Ours (FullDownload)	0.8 KB	819 KB	No

Table 1: Illustration of our approaches and baselines (cited/explained in Appendix C). *Upload* and *Download* shows the Bits that is uploaded and downloaded per round of federated learning. *BP?* indicates whether the method requires back-propagation. Our approaches can save the memory consumption of back-propagation, while significantly reduce the communication cost. We index the mapproaches with A-K for the convenience of Figure 2.

(Twitter) (Go et al., 2009), CoLA (Warstadt et al., 2018) and SST2 (Socher et al., 2013) for experiments of text classification with federated learning. We additionally adopt FDU-MTL (Liu et al., 2017) that contains 16 text domain. We train and evaluate on all the 16 domains of FDU-MTL (each client with a unique text domain). Please refer to Appendix A for more training details and data splits. Table 1 lists our approaches and considered baselines, which are also detailed in Appendix C. We follow (Sun et al., 2022b) that uses Roberta-Large in our experiments. We do not adopt larger models, *e.g.* LLaMA (Touvron et al., 2023) , due to our practical assumption that the federated learning clients are given limited access to computation resources (Section 1).

Evaluation: The performance of the PLM from federated learning is evaluated via the average classification accuracy over clients that it is tested on. We conduct two kinds of testing: *i) P*: Testing on the *P*articipant clients of federated learning. This evaluated how much a PLM can capture the knowledge from clients during training. *i) NP*: Testing on the *N*on-*P*articipant clients of federated learning. This evaluated the PLM can generalize to unseen clients. For Sentiment140, CoLA and SST2, our partition of participant and non-participant clients follows (Chen et al., 2022). For FDUMTL, we first set all its 16 domain/clients as participant for the evaluation of *P*. In evaluating *NP*, we conduct a 4-split cross-validation that split the 16 clients into 4 groups. We iteratively treat the clients of each group as non-participant while those from others groups as participant (Table 3). In this way, each client is treated as non-participant once and we average the results for *NP*.

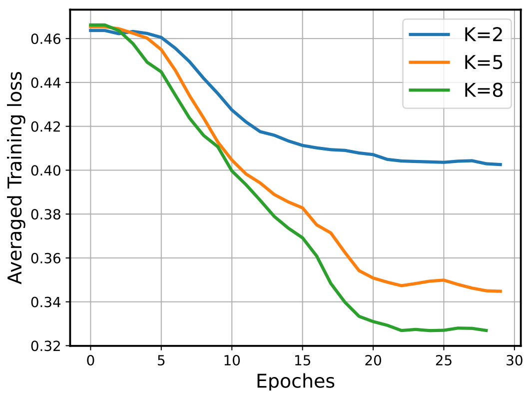

Figure 1: Averaged training loss during joint training of *Ours* ($\Phi = 5$) with different values of K.

In Table 1, we also report on: *1)* Whether the method requires back-propagation, *i.e.*, does the model consume a large memory footprint for local training? *2)* The communication cost, *i.e.*, the number of communicated Bits between server and clients for each round of federated learning. In calculating the Bits, we assume the token indices are encoded with 16-bit and continuous parameters are converted into float16 during communication, as in Sections 4.2 and 4.3. Importantly, we calculate the upload and download cost separately, due to the fact that the upload bandwidth is usually smaller than the download bandwidth (Hegedűs et al., 2021), *i.e.*, upload is more expensive than download with the same number of Bits. For instance, with prompt length $T = 50$ (Appendix A), the upload communication cost for Ours ($\Phi = 5$) is $50 \times 16 = 0.8K$ (Section 4.2) and its download cost is $50 \times 2 \times 5 \times 16 = 8K$ (Section 4.3)

5.2 Local Search with Different K Values.

As discussed in Section 4.2, discrete prompt tokens might be less expressive than continuous prompt embeddings trained with gradients (Li and Liang, 2021; Liu et al., 2021). Thus, one may be concerned about the capability of discrete local search in minimizing the loss functions of different tasks of different clients. From (6), we can observe that such capability is large and determined by the search number K for each step of local search. Ideally, in maximizing the optimization ability of our local search, we can set $K = |\mathcal{V}|$, *i.e.*, and try with the whole vocabulary instead of searching lo-

	Sentiment140		FDUMTL		CoLA		SST2		Avg	
	P	NP	P	NP	P	NP	P	NP	P	NP
Prompt Tuning	73.22	N/A	83.41	N/A	71.89	N/A	79.87	N/A	77.10	N/A
Prompt Tuning (Fed)	73.44	74.67	84.28	83.76	74.22	73.03	81.22	81.49	78.29	78.24
Meta Prompt Tuning (Fed)	73.95	74.89	84.20	83.89	73.17	73.46	81.96	82.44	78.32	78.67
pFedKD	72.75	73.11	84.03	83.86	72.56	71.34	78.65	79.57	77.00	76.97
pFedMe	75.66	74.95	84.60	84.79	**74.95**	72.27	81.78	81.65	79.25	77.66
Fine Tuning (Fed)	74.17	75.52	85.98	85.09	74.01	**74.35**	80.96	79.42	78.78	78.60
BBT	73.17	N/A	84.34	N/A	74.26	N/A	80.34	N/A	78.03	N/A
BBT (Fed)	73.87	73.58	86.12	86.44	75.88	73.07	81.46	80.67	79.33	78.69
Ours ($\Phi = 3$)	74.08	74.94	86.64	86.66	75.22	72.97	81.78	82.14	79.43	79.18
Ours ($\Phi = 5$)	**76.17**	75.34	87.14	87.00	74.86	73.31	82.36	**82.88**	80.13	79.63
Ours (FullDownload)	75.16	**76.00**	**87.71**	**87.31**	75.75	73.78	**82.95**	82.73	**80.39**	**80.00**

Table 2: Results with our considered datasets for federated learning. "P" and "NP" denotes the mean accuracy on *Participant* and *Non Participant* clients of federated learning, respectively. *Prompt Tuning* and *BBT* are not federated learning methods, thus all clients are treated as *Participants* Please note that, in addition to performance, our approaches are also superior in terms of memory consumption and computation cost. Please refer to Table 1 for more details.

cally. However, such a combinatorial optimization is computationally expensive, thus not compatible with resource constrained clients. There should be a trade-off between the optimization ability and training efficiency for discrete local search.

In this section, we investigate how the optimization ability of our proposed local search is affected by the search number K. In Figure 1, we plot the averaged training loss (4) over all the clients in FDU-MTL when training *Ours* ($\Phi = 5$) with different K values. We can observe that our local search can effectively minimize the loss function during training. Additionally, we find that the performance gain, *i.e.*, the difference in the optimized loss value, is diminishing when switching from $K = 2$ to $K = 5$ and from $K = 5$ to $K = 8$. However, the introduced computation cost from $K = 2$ to $K = 5$ is the same as that from $K = 5$ to $K = 8$. With such observation, we take $K = 5$ as a trade-off between the computation efficiency and optimization ability, since *1)* local search with $K = 5$ is not very expensive, *e.g.*, comparing the implementation of BBT (Sun et al., 2022b) that requires 20 searches each step. *2)* The performance gain from $K = 5$ to $K = 8$ is much smaller than that $K = 2$ to $K = 5$, thus increasing the value of K from 5 may not be cost-effective. Therefore, we keep $K = 5$ for all our experiments. Note that such a parameter selection of K only leverages the

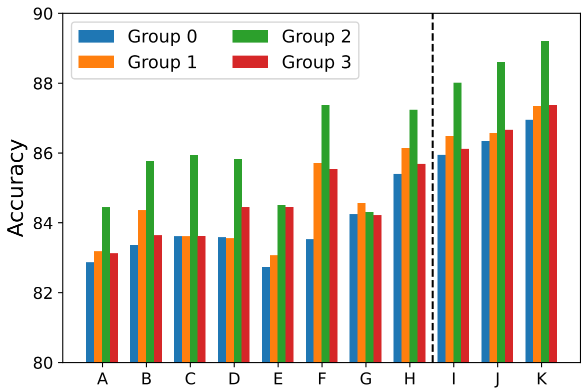

Figure 2: Results on each group of non-participant clients in FDUMTL. For convenience, we denote each method with the index defined in Table 1. Our approaches are the right of the vertical line.

training data of clients, with no development or testing data involved.

5.3 Result Analysis

Table 2 shows our results with considered datasets. Our approaches can achieve the highest accuracy, with comparable or much lower communication cost than the baselines (Table 1). This is especially obvious with the upload communication, *i.e.*, the upload cost of our approaches is 10 times smaller than the closest baselines (BBT (Fed)), which thanks to our proposed discrete local search mechanism (Section 4.2) that only requires uploading the pretrained token indices to the server. As mentioned in Appendix B, BBT (Sun et al., 2022b) works by randomly projecting the prompt parameters (with a fixed random matrix $\boldsymbol{A}$) into a small subspace, within which a low-dimensional vector $\boldsymbol{z}$ is trained. However, there is no guarantee that such a random projected subspace can cover directions that capture knowledge that is generalizable across clients. On the contrary, though our local search algorithm is constrained with discrete natural language tokens, such tokens should capture rich semantics of natural language that are expressive enough to describe a pattern that is generalizable across clients. This might explain why our approach of discrete local search with natural language tokens yeilds higher accuracy in training with data of different clients. Moreover, we can observe that compressing using $\Phi = 3$ and $\Phi = 5$ can maintain comparable performance for text classification as with *Ours (FullDownload)*, while substantially decreasing download communication cost.

Among the gradient-based approaches (*i.e.*, BP?=Yes), *Fine Tuning (Fed)* achieves competitive accuracy than other gradient-based approaches (*i.e.*, *BP?=Yes*), but at the expense of huge computation cost. *FedKD* (Wu et al., 2022) generally has lower classification accuracy, which might because its student model (DistilRoberta-base) is less capable than Roberta-Large as used in other approaches. We follow (Wu et al., 2022) that uses a small student model for FedKD to save the communication cost. We can observe that these gradient-based baselines may produce results that are inferior to gradient-free approaches. This may be counterintuitive since these gradient-based prompt tuning approaches allow training in the whole (more expressive) parameter space of prompt parameters, compared to gradient-free approaches with which the search space for the prompt parameters is usually constrained (Sun et al., 2022b). However, previous works of gradient-free training with PLMs (Sun et al., 2022b,a) also show results that are better than gradient-based approaches, especially with the scenario of few-shot training. Such a phenomenon may be explained by the over-expressiveness of prompts trained with gradients, *i.e.*, subject to overfitting with limited training data. Also, as discussed in Section 4.2, the prompts trained with gradients may overfit to the task/domain of the clients during local client update, inducing negative knowledge transfer from other clients.

X, ros, Target, himself, turn, Europe, WORK, Energy, scored, *, shortly, balls, TV, yearly, 2012, Race, International, ', Marketplace, conference, io, os, modifications, IG, troopers, inside, Forms, publishes, cellphone, CO, legal, executive, fight, ings, hope, Summer, Officers, football, Property, #, book, parents, expenses, ac, manager, create, age, email, market, mainline

Figure 3: The learnt prompt from the *apparel* domain of FDU-MTL, using our proposed discrete local search.

In Figure 2, we detailed results of *NP* for FDUMTL with each of its groups. We can find that our approach consistently outperform the baselines with in terms of group-wise *NP* accuracy. We also provide detailed participant accuracy for each client in Table 4 and 5.

Privacy with the learnt prompts. Figure 3 shows the prompts learnt with data from the *apparel* domain of FDU-MTL, using the proposed discrete local search in client update (Section 4.2). We can find it is hard to interpret, and we cannot infer that the client data is about "apparel" given the prompt tokens. Such a lack of interpretability reduces the chance of client privacy leakage, when uploading the learnt prompts to the server after client update. Inspired by recent approaches of evaluating with Large Language Models (LLMs) (Peng et al., 2023), we further conduct a privacy leakage analysis in Appendix H. Specifically, given a prompt trained from a certain client/domain of FDU-MTL, we investigate how GPT-4 (OpenAI, 2023) can link the prompt to its training domain. We find that none of the 16 clients/domains can be inferred from their prompts using GPT-4 predictions, indicating less chance of privacy leakage.

6 Conclusions

In this paper, we propose a gradient-free framework that trains with discrete local search on natural language token during personalized federated learning. Compared with gradient-based approaches, the discrete local search circurvents gradient computation and saves the huge memory consumption caused by back-propagation. We additionally propose a compression mechanism inspired by linear word analogy that allows the server-client communication with discretely indexed tokens. Experiments on multiple benchmarks show that our proposed gradient-free framework can achieve superior performance, while significantly reducing the upload communication cost.

7 Limitations

Our proposed approach considers communicating and compressing the pretrained embeddings of the natural language tokens, which is only applicable to the domain of natural langauge processing. It would be more comprehensive for our study to further explore applying our approach for visual tokens (Wu et al., 2020; Yin et al., 2022) during federated learning.

8 Ethics Statement

Ours study of personalized federated learning is intended to protect client privacy during training, avoiding malicious use of client private information. Additionally, the datasets in our experiments are all publicly available.

References

Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. 2022. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*.

Gabriel Belouze. 2022. Optimization without backpropagation. *arXiv preprint arXiv:2209.06302*.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*.

Tingfeng Cao, Liang Chen, Dixiang Zhang, Tianxiang Sun, Zhengfu He, Xipeng Qiu, Xing Xu, and Hai Zhang. 2023. Competition for gradient-free tuning of large language models: approaches, results, current challenges and future directions. *National Science Review*, 10(6):nwad124.

Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pfl-bench: A comprehensive benchmark for personalized federated learning. *Advances in Neural Information Processing Systems*, 35:9344–9360.

Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Yunbin Deng. 2019. Deep learning on mobile devices: a review. In *Mobile Multimedia/Image Processing, Security, and Applications 2019*, volume 10993, pages 52–66. SPIE.

Shizhe Diao, Xuechun Li, Yong Lin, Zhichao Huang, and Tong Zhang. 2022. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, pages 3519–3530.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2018. Towards understanding linear word analogies. *arXiv preprint arXiv:1810.04882*.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Karl Pearson F.R.S. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Tao Guo, Song Guo, and Junxiao Wang. 2023. pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, pages 1364–1374.

Tao Guo, Song Guo, Junxiao Wang, and Wenchao Xu. 2022. Promptfl: Let federated participants cooperatively learn prompts instead of models–federated learning in age of foundation model. *arXiv preprint arXiv:2208.11625*.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195.

István Hegedűs, Gábor Danner, and Márk Jelasity. 2021. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124.

Bairu Hou, Joe O'Connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2022. Promptboosting: Black-box text classification with ten forward passes. *arXiv preprint arXiv:2212.09257*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, pages 4171–4186.

Anis Koubaa. 2023. Gpt-4 vs. gpt-3.5: A concise showdown.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Zhengyang Lit, Shijing Sit, Jianzong Wang, and Jing Xiao. 2022. Federated split bert for heterogeneous text classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 1–15.

Malvina Nissim, Rik van Noord, and Rob van der Goot. 2020. Fair is better than sensational: Man is to doctor as woman is to doctor. *Computational Linguistics*, 46(2):487–497.

OpenAI. 2023. Gpt-4 technical report.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. 2019. A generic communication scheduler for distributed dnn training acceleration. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 16–29.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Tahseen Rabbani, Brandon Feng, Yifan Yang, Arjun Rajkumar, Amitabh Varshney, and Furong Huang. 2021. Comfetch: Federated learning of large networks on memory-constrained clients via sketching. *arXiv preprint arXiv:2109.08346*.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. 2022a. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. 2022. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*.

Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2022. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):2032.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10818.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. 2022. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. *arXiv preprint arXiv:2208.12268*.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. 2023. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*.

A Additional Explanation

Our model architecture for prompt tuning is the same as in (Sun et al., 2022b). Specifically, the backbone of the PLM is the Roberta-Large model (embedding dimension $D = 1024$), with $T = 50$ prompt tokens inserted into the input layer. The model is trained with 50 rounds of federated learning for FDU-MTL, SST2 and CoLA, with each client updated 40 steps for each round. For Sentiment140, we train for 100 rounds and we only sample 50 clients for training during each round (due to the large number of clients in Sentiment140). Following (Chen et al., 2022), clients for SST2 and CoLA are partitioned with Dirichlet distribution, denoted as $Dir(\gamma)$, where γ controls the client heterogenity. We follow (Chen et al., 2022) that set $\gamma = 0.4$ in the main results. We also experiment with different values of γ in table 7. The implementation of BBT in the both our approaches and the baselines follows (Sun et al., 2022b).

Following previous works of gradient-free learning (Sun et al., 2022b; Hou et al., 2022), we consider the few-shot scenario for each testing client. Specifically, we assume there are 16 samples for each class in each testing client during post-tuning. For FDU-MTL, these datasets are sampled from the development split in each domain. For sentiment140, these are sampled from the datasets of each testing client, with the rest data of each client used for testing after post tuning. We additionally sample a development dataset (not overlapped with data for training) from the development split for each client for FDUMLT with the same size as the training set, since development datasets are also used in previous works of gradient-free training (Sun et al., 2022b; Hou et al., 2022). We evaluate the classification accuracy of the resulting models on the test set of each client, averaged over four random seeds. We do not sample development datasets for Sentiment140 since no development datasets are provided. Note that our experiments are based only on English datasets and it would also be interesting for future works studying multilingual federated learning (Weller et al., 2022). We provide the algorithm for Local_Training and Compress_Download in Algorithm2 and 3, respectively.

B Black Box Tuning (BBT)

We briefly introduce a prior radient-free method of BBT (Sun et al., 2022b). For prompt $\boldsymbol{p}_i$, suppose

	Domains
Group 1	apparel, mr, baby, books
Group 2	camera, dvd, electronics, health
Group 3	imdb, kitchen, magazines, music
Group 4	software, sports, toys, video

Table 3: Group of domains in FDUMTL. In testing the performance on non-participant clients, we do 4-split cross-validation with FDUMTL. Specifically, we iteratively treat the domains from a group as non-paticipant clients that are held-out from federated learning, *i.e.*, we train with domains/clients of the other three groups during federated learning and test on domains of the held-out group.

we want to train its tth prompt token of $\boldsymbol{p}_i^t$, the BBT approach first reparameterizes $\boldsymbol{p}_i^t$ as,

$$\boldsymbol{p}_i^t = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{p}^t, \tag{12}$$

where $\boldsymbol{z} \in \mathbb{R}^d$, $d << D$, and $\boldsymbol{A} \in \mathcal{R}^{D \times d}$ is a randomly valued fixed matrix that project $\boldsymbol{z}$ into the space of $\boldsymbol{p}^t$. $\boldsymbol{z}$ is the only learnable parameter and is trained with CMA-ES (Hansen and Ostermeier, 2001), a gradient-free method without back-propagation. In post tuning, we set a small training step size (denoted as σ in (Sun et al., 2022b)) of CMA-ES, *i.e.*, $\sigma = 0.1$, while keeping $\sigma = 1$ in the other cases. Please refer to (Sun et al., 2022b) for more details.

C Baselines

All of our baselines are trained with the same model as used in (Sun et al., 2022b). We list the considered baselines are listed as follows:

- *Prompt Tuning* (Li and Liang, 2021): Train separated prompt parameters locally on each testing client with back-propagation. We have learning rate as 1e-2 and batch size 16.
- *Prompt Tuning (Fed)*. The prompts are initially trained with FedAvg (McMahan et al., 2017) on all the clients, then fine tuned on each testing client, as with our framework.
- *Meta Prompt Tuning (Fed)*: Same as Prompt Tuning (Fed), except that we follow (Fallah et al., 2020) that the prompts are trained using federated meta learning with MAML (Finn et al., 2017).
- *pFedMe* (T Dinh et al., 2020): We train ans communicate the prompt parameters with pFedMe, where there is an L2 regularization between the global prompt and personalized prompts for for each client.
- *FedKD* (Wu et al., 2022): Compressing the Roberta-Large into a smaller student model (DistilRoberta-base) via knowledge distillation, and only communicate the student model to save communication cost. For joint training with FedKD, we follow its original paper (Wu et al., 2022) that fine-tunes all the parameters of both the Roberta-Large and DistilRoberta-base. We did not implement the SVD compression in communicating the parameters, in order to show an upper bound of its classification performance. The learning rate for joint training is 1e-3. The resulting model is post tuned using gradient descent with a learning rate of 1e-5.
- *Fine-Tuning (Fed)*: We fine-tune and communicate all the parameters of Roberta-Large in joint training, while post tuning with all the model parameters. The learning rates are the same as in *FedKD*.
- *BBT* (Sun et al., 2022b): Train separated prompts locally on each testing client with the gradient-free method of CMA-ES (Hansen and Ostermeier, 2001), as in Section B. This is like the post tuning stage of our approach.
- *BBT (Fed)*: Federated training of $\boldsymbol{z}$ in (12) with BBT on training clients and FedAvg on the server. The resulting $\boldsymbol{z}$ is further fine tuned with BBT on the local dataset of each client, *i.e.*, the same as Section B.

In addition, we also implement different variations of our approach: *1) Ours (Φ=3 or 5)*. We experiment with different values of Φ, controlling the degree of the embedding compression in Section 4.3. *2) Ours (FullDownload)*. We directly download the aggregated $\boldsymbol{p}$ from (5), without embedding compression.

D Ablation study with α

In this section, we conduct an ablation study for the regularization parameter α (default to $\alpha = 0.2$) for the lasso loss in (8). In Table 6, we take Ours ($\Phi = 5$) as an example and report results with $\alpha =$

0.2 (same as in the main paper) and $\alpha = 0$. We can find that the results with $\alpha = 0$ is generally lower than that with $\alpha = 0.2$, indicating the importance of encouraging sparsity with the lassso loss in (8).

E Comparing with PCA compression and quantization

In Section 4.3, we present our proposed embedding compression method to reduce the download communication cost. To further validate the effectiveness of the proposed embedding compression, we compare it with the two additionaly baselines: PCA compression and quantization.

PCA Compression: Principled Component Analysis (PCA) (F.R.S., 1901) is a common way of dimensional reduction, *i.e.*, compress the embeddings via representing then with fewer dimensions. Previous works (Cai et al., 2021; Rabbani et al., 2021; Gao et al.) have shown that the learnt token embeddings (contextualized or not) of pretrained models are distributed in a narrow cone of the embedding space. In other words, the embeddings vectors are generally biased toward the top principled components of learnt embedding matrix. Specially, following the notation of Section 4.3, let $\boldsymbol{e}(\mathcal{V}) \in \mathbb{R}^{|\mathcal{V}|\times D}$ be the matrix of pretrained token embeddings. We can compute the principled components of $\boldsymbol{e}(\mathcal{V})$, denoted as,

$$\boldsymbol{E}_c = PCA(\boldsymbol{e}(\mathcal{V})) \tag{13}$$

where each column of $\boldsymbol{E}_c \in \mathbb{R}^{D\times D}$ is a principled component of $\boldsymbol{e}(\mathcal{V})$. We have $\boldsymbol{E}_c^T \cdot \boldsymbol{E}_c = \boldsymbol{I}$, with $\boldsymbol{I} \in \mathbb{R}^{D\times D}$ is the identity matrix. The informativeness of different principled component can be measured by the variance after projecting $\boldsymbol{e}(\mathcal{V})$ onto each of the components,

$$\boldsymbol{v} = \mathrm{Var}(\boldsymbol{e}(\mathcal{V}) \cdot \boldsymbol{E}_c) \tag{14}$$

where Var computes the variance for each row. Assume the index of each component, *i.e.*, the row index of $\boldsymbol{E}_c$, has been ranked by $\boldsymbol{v} = [\boldsymbol{v}_i]_{i=1}^D$ (from high to low). We plot the ratio of variance ($\boldsymbol{v}/\sum \boldsymbol{v}_i$) verse the index of each component for Roberta-Large in Figure 4a. We can find that the distribution of $\boldsymbol{e}(\mathcal{V})$ id highly an-isotropic, with much larger variation being captured by the top principled components. Thus, we can represent/compress the aggregated prompt $\boldsymbol{p} \in \mathbb{R}^{T\times D}$ from (5) with the top principled components[2] before downloading it to clients. Specifically, we compress $\boldsymbol{p}$ via,

[2]From Section 4.1, each token of p is a convex combination of $\boldsymbol{e}(\mathcal{V})$, thus should also be biased toward (more represented by) the top principled components.

Algorithm 2 Compress_Download.

Input: The prompt $\boldsymbol{p}$ without compression, the pretrained embedding matrix $\boldsymbol{e}(\mathcal{V})$.
Output: The reconstructed $\boldsymbol{p}'$.
$\boldsymbol{I} = [1, \cdots, |\mathcal{V}|]$
for $t = 1 \cdots, T$ **do**
  % Embedding compression.
  for $L = [100, 5]$ **do**
    Compute $\boldsymbol{I}$ with (8) and (9).
  end for
  Solve $\boldsymbol{x}_f^*$ with (10).
  % Download.
  Download $\{\boldsymbol{I}, \boldsymbol{x}_f^*\}$ to the clients.
  Compute $\boldsymbol{p}^{t\prime}$ on both server and clients
end for
return $\boldsymbol{p}' = [\boldsymbol{p}^{1\prime}, \cdots, \boldsymbol{p}^{T\prime}]$

$$\hat{\boldsymbol{p}} = \boldsymbol{p} \cdot \boldsymbol{E}_c[: n, :]^T \tag{15}$$

where $\hat{\boldsymbol{p}}\mathbb{R}^{T\times n}$ is the compressed prompt and $E_c[: n, :]$ denotes the top-n principled components. After downloading, each client reconstructs $\boldsymbol{p}$ via,

$$\boldsymbol{p} = \hat{\boldsymbol{p}} \cdot \boldsymbol{E}_c[: n, :] \tag{16}$$

In this way, we only need to download n integers (16 bits each) for each prompt token in $\boldsymbol{p}$. The total download bits per communication round is $T \times n \times 16 = 800n$ bits. In comparison with our approach, we experiment with $n = 10$ (denoted as PCA10), so that it has the same download communication cost for each round (8KB) as Ours $\Phi = 5$. We additionally experiment with $n = 300$ (denoted as PCA300), where the prompts are represented by more principled components but also with much larger download communication cost each round (0.24MB).

Quantization: We also compare our approach with quantizing each dimension of $\boldsymbol{p}$ from (5) before downloading. Following previous works (Courbariaux et al., 2015; Tao et al., 2022) of compressing pretrained language models, we quantize each element w of $\boldsymbol{p}$ via,

$$w_q = \beta \cdot Q(clip(w, -\beta, \beta)/\beta) \tag{17}$$

where Q is a quantization function that maps $clip(w, -\beta, \beta)$ to its closest value in $\{-1, -\frac{k-1}{k}, \cdots, 0, \cdots, \frac{k-1}{k}, 1\}$, $k = 2^{b-1} - 1$.

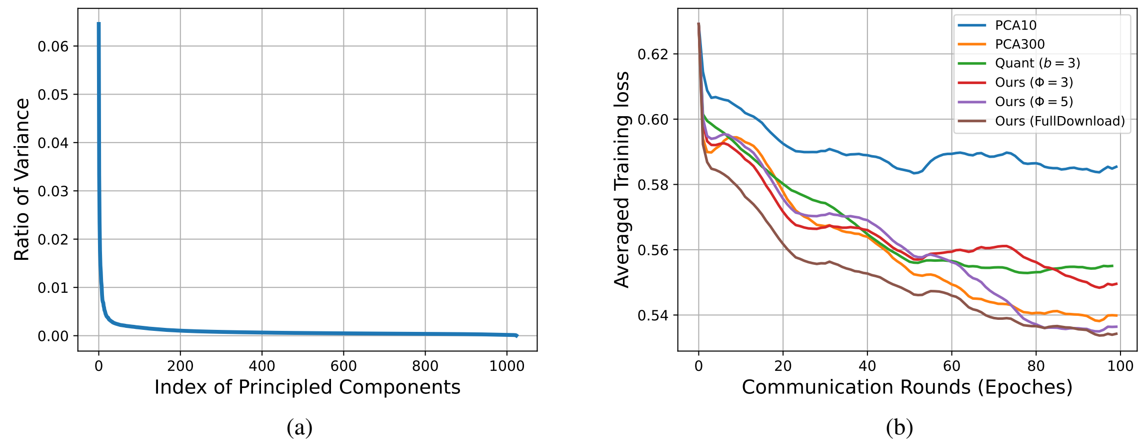

Figure 4: (a) The ratio of variance ($\boldsymbol{v}/\sum \boldsymbol{v}_i$) captured by each principled component of the pretrained Roberta-Large Token embeddings. (b) The training loss on Sentiment140 averaged over different clients in each communication round of federated learning for different compression methods. We have the same random seeds and order of training batches for all the methods.

Method	Upload	Download	BP?	FM(*apparel*)	FM(*mr*)	FM(*baby*)	FM(*books*)	FM(*camera*)	FM(*dvd*)	FM(*electronics*)
A. Prompt Tuning	0	0	Yes	83.42	81.75	79.95	86.38	80.05	86.52	84.18
B. Prompt Tuning (Fed)	819 KB	819 KB	Yes	83.56	81.06	81.05	87.83	81.80	87.96	84.93
C. Meta Prompt Tuning (Fed)	819 KB	819 KB	Yes	82.78	83.35	80.23	88.12	80.34	87.31	84.45
D. pFedMe	819 KB	819 KB	Yes	84.67	81.26	81.47	86.92	80.56	87.92	81.26
E. FedKD	1.3 GB	1.3 GB	Yes	83.67	79.89	80.46	86.92	81.07	87.08	79.89
F. Fine Tuning (Fed)	5.3 GB	5.3 GB	Yes	86.93	79.82	80.46	86.92	81.07	88.48	87.50
G. BBT	0	0	No	85.93	83.75	81.22	86.10	80.56	85.96	87.76
H. BBT (Fed)	8 KB	8 KB	No	87.44	81.02	82.99	90.19	81.84	87.92	87.74
I. Ours ($\Phi = 3$)	0.8 KB	4.8 KB	No	87.44	80.07	85.53	90.74	82.33	88.48	88.03
G. Ours ($\Phi = 5$)	0.8 KB	8 KB	No	88.54	80.05	86.55	90.21	82.61	88.08	87.78
K. Ours (FullDownload)	0.8 KB	819 KB	No	89.04	81.03	86.78	90.97	83.73	87.18	88.88

Table 4: Detailed results with FDUMLT on paticipant clients. Please refer to Figure 2 for non-paticipant clients. We report the accuracies for each of the 16 domains/clients (denoted as FM(*domain name*)) and their average (denoted as FM(*Avg*)).

Method	FM(*health*)	FM(*imdb*)	FM(*kitchen*)	FM(*magazines*)	FM(*music*)	FM(*software*)	FM(*sports*)	FM(*toys*)	FM(*video*)	FM(*Avg*)
Prompt Tuning	81.98	92.42	82.14	80.68	82.52	83.77	82.41	84.01	82.32	83.41
Prompt Tuning (Fed)	82.74	92.71	83.61	82.97	83.75	84.29	82.89	84.76	82.60	84.28
Meta Prompt Tuning (Fed)	82.34	92.41	84.53	83.25	83.56	83.48	83.58	85.26	82.21	84.20
pFedMe	84.51	93.00	84.44	82.25	83.60	84.29	83.42	85.53	84.53	84.60
FedKD	84.26	92.71	82.91	80.94	81.48	84.82	82.40	85.28	85.36	84.03
Fine Tuning (Fed)	85.79	93.00	86.99	85.12	84.39	84.82	85.46	86.80	85.08	85.98
BBT	84.01	92.13	81.38	81.46	82.28	85.08	82.40	85.53	83.86	84.34
BBT (Fed)	87.06	93.00	85.13	85.90	84.92	84.03	85.46	87.92	85.36	86.12
Ours ($\Phi = 3$)	87.06	92.42	86.73	86.95	85.98	84.55	86.73	87.31	85.91	86.64
Ours ($\Phi = 5$)	87.82	92.71	88.78	87.73	85.19	85.60	86.48	87.31	87.29	87.14
Ours (FullDownload)	89.57	94.27	88.75	87.44	86.34	85.44	87.86	89.31	86.86	87.71

Table 5: Results with FDUMLT on participant clients (continue).

In this way, $Q(clip(w, -\beta, \beta)/\beta)$ can be encoded with b bits. Following (Tao et al., 2022), the scaling factor for each element is shared within the same prompt token embedding. Let $\boldsymbol{p}[i,:]$ be the embedding of the ith prompt token, the scaling factor for each of its element is the maximum absolute value in $\boldsymbol{p}[i,:]$,

$$\beta = max(|\boldsymbol{p}[i,:]|) \tag{18}$$

For each prompt token with dimension D, we have to download the scaling factor β (16 bits) and b bits for each dimension, so that the clients can reconstruct w_q. We experiment with $b = 3$, denoted as Quat ($b = 3$). The total download communication cost for each round is $(D \times b + 16) \times T \approx 0.15$MB. Compared with Quat ($b = 3$) that quantizes each dimension of each prompt, our proposed approaches

Method	Upload	Download	Sentiment140	FDUMTL	CoLA	SST2	Avg
PCA10	0.8KB	8KB	72.37/73.26	83.25/83.89	72.45/72.11	79.65/78.34	76.93/76.90
PCA300	0.8KB	0.24MB	75.22/75.05	86.71/85.79	74.09/73.66	81.23/81.67	79.31/79.04
Quant ($b = 3$)	0.8KB	0.15MB	73.45/74.44	85.46/84.33	73.89/73.17	80.98/80.56	78.45/78.13
Ours ($\Phi = 5, \alpha = 0$)	0.8KB	8KB	74,77/74.35	85.80/86.41	74.94/73.11	81.45/82.12	79.24/79.00
Ours ($\Phi = 5, \alpha = 0.2$)	0.8KB	8KB	76.17/75.34	87.14/87.00	74.86/73.31	82.36/82.88	80.13/79.63
Ours (FullDownload)	0.8KB	819KB	75.16/76.00	87.71/87.31	75.75/73.78	82.95/82.73	80.39/80.00

Table 6: Results with different compression methods and α. We report the accuracy in the format of "P/NP", where P and NP follow Table 2.

of embedding compression can be regarded as quantizing on the token level, *i.e.*, representing each prompt with pretrained embeddings of discrete tokens.

Results: We report the results with different compress methods in Table 6. We can find that PCA10 has much lower accuracies than Ours ($\Phi = 5$), though sharing the same communication cost. This is because the top 10 principled components cannot capture enough information about the token embeddings, although the distribution of token embeddings are biased toward the top principled components (Figure 4a). We need to increase the value of n to hundreds in order to get comparable results with our approaches ((*i.e.*, PCA300)), which is at the expense of much higher communication cost. Additionally, we can notice that Quant ($b = 3$) also induces higher download communication cost than our approaches, but yeilding lower accuracies. These results validate the effectiveness of our proposed embedding compression. Additionally, Figure 4b shows the loss values averaged over training clients during federated learning. We can find that our approaches are effective in minimizing the loss function during training (also discussed in Section 5.2). We can also find that the final loss values are generally positively correlated with the accuracies in Table 6.

Algorithm 3 Local_Training.

Input: Dataset $\mathcal{D}_i$ for client i, $\boldsymbol{p}'$ from the previous round of communication.
Output: $\boldsymbol{p}_i$ after the client update.
$\boldsymbol{p}_i = \boldsymbol{p}'$
% Training with discrete local search.
for $s = 1 \cdots, S$ **do**
  Randomly sample position t.
  Update $\boldsymbol{p}_i^t$ using (6) and (7) with $\mathcal{D}_i$.
end for
return $\boldsymbol{p}_i$

Method	$\gamma = 0.1$	$\gamma = 0.4$	$\gamma = 5$
Ours (FullDownload)	77.79/78.41	82.95/82.73	83.68/83.27
Ours ($\Phi = 5$)	76.85/77.10	82.36/82.88	83.12/82.64
pFedMe	76.05/75.37	81.78/81.65	82.31/81.43
BBT (Fed)	76.33/76.79	81.46/80.67	81.88/80.47

Table 7: SST2 with varied client heterogeneity. We follow (Chen et al., 2022) that varies the dirichlet factor γ with values of $[0.1, 0.4, 5]$. We choose each of a competitive gradient-based baseline (pFedMe) that has moderate communication cost. We also choose a gradient-free baseline (BBT (Fed)). The results show taht our approaches are consistently better than baselines in various heterigenity. We report thr accuracy in the format of "P/NP".

F The number of floating-point operations during federated learning

From the previous work (Sun et al., 2022b) of gradient-free training for PLMs, the number of floating-point operations with gradient-free training can be evaluated via the number of model queries (i.e., how many times a model is forwarded). For all the methods in the paper, we have the same number of communication rounds and same number of update steps for each client per round. Thus, the number of floating-point operations is proportional to the number of model queries per step when training on each client. We keep all the discussed approaches with the proposed discrete local search method having 5 model queries per step (i.e., $K = 5$ as in Section 5.2), including the approaches denotes with "Ours" and those in Appendix E. Thus, all these approaches have the same number of model queries during federated learning. Comparably, our gradient-free federated learning baseline (i.e., BBT(Fed), there was no previous works on gradient-free federated learning with pretrained models) have 20 model queries per step, following the original implementation of (Sun et al., 2022b). This implies that our methods (5 queries per step) only use $1/4$ $(5/20)$ times of floating-point operations during federated learning, while having better performance than BBT(Fed). Since we target the scenario that clients has limited memory access, where back-propagation might not be viable (Section 1), we mostly compare the number of floating-point operations of our methods with gradient-free federated learning baselines. Provided the number of floating-point operations during federated learning, the training efficiency can be further enhanced by system designs, *e.g.*, the parallelism strategy (Narayanan et al., 2019) or communication scheduler (Peng et al., 2019), which are out of the scope of this paper.

G Overhead

Our way of converting the prompt token index of each position to 16 bits (Section 5.1) induces no computational overhead, if we save the 16 bits index for each position during training (50 prompt positions in total, *i.e.*, $T = 50$). The uploading of such bits is the same as uploading any model parameters in federated learning. There is not need of additionaly designed software implementation. Actually, by only uploading 16 bits for each position, we save the upload time compared with uploading the prompy embedding (the gradient-based methods in Table 4 and 5).

H Inferring the text domain with GPT-4

As mentioned in Section 5.3, we leverage GPT-4 (OpenAI, 2023) to infer the text domain (client) from the prompt trained on it, in order to investigate on the risk of privacy leakage by uploading prompt from clients to the server. This is inspired by recent approaches of evaluating with Large Language Models (LLMs) (Peng et al., 2023). Specifically, try to ask GPT4 with the following template,

> *Given the following prompt sequence learnt from Roberta-Large:*
> *{prompt}*
> *Can you infer that this is trained from a {domain} dataset?*

where *{prompt}* and *{domain}* refer to a prompt and the text domain (client) from which the prompt is trained on, respectively. For example, with *{prompt}* as in Figure 3 and the *{domain}* being *apparel* in FDU-MTL, the GPT-4 answers as,

> *The given list of words and phrases doesn't provide sufficient evidence to conclude that it is trained from an apparel dataset. ....*

We tried with 16 domains from FDUML and none of them result in a positive answer i.e., GPT-4 answers with positive semantics that it can infer the *{domain}* from *{prompt}*. In another word, the frequency that GPT-4 can infer the *{domain}* from the *{prompt}* is zero, indicating less chance of client privacy leakage.