FedNLP: Benchmarking Federated Learning Methods for Natural Language Processing Tasks

Anonymous ACL submission

Abstract

Increasing concerns and regulations about data privacy and sparsity necessitate the study of privacy-preserving, decentralized learning methods for natural language processing (NLP) tasks. Federated learning (FL) provides promising approaches for a large number of clients (e.g., personal devices or organizations) to collaboratively learn a shared global model to benefit all clients while allowing users to keep their data locally. Despite interest in studying FL methods for NLP tasks, a systematic comparison and analysis is lacking in the literature. Herein, we present the FedNLP, a benchmarking framework for evaluating federated learning methods on four common formulations of NLP tasks: text classification, sequence tagging, question answering, and seq2seq generation. We propose a universal interface between Transformer-based language models (e.g., BERT, BART) and FL methods (e.g., FedAvg, FedOPT, etc.) under various non-IID partitioning strategies. Our extensive experiments with FedNLP provide empirical comparisons between FL methods and help us better understand the inherent challenges of this direction. The comprehensive analysis points to intriguing and exciting future research aimed at developing FL methods for NLP tasks. [1]

1 Introduction

Fine-tuning large pre-trained language models (LMs) such as BERT (Devlin et al., 2019) often leads to state-of-the-art performance in many realistic NLP applications (e.g., text classification, named entity recognition, question answering, summarization, etc.), when large-scale, *centralized* training datasets are available. However, due to the increasing concerns and regulations about data privacy (*e.g.,* GPDR (Regulation, 2016)) emerging data from realistic users have been much more *fragmented* and *distributed*, forming *decentralized private datasets* of multiple "data silos" (a *data silo* can be viewed as an individual dataset) — across different clients (e.g., organizations or personal devices).

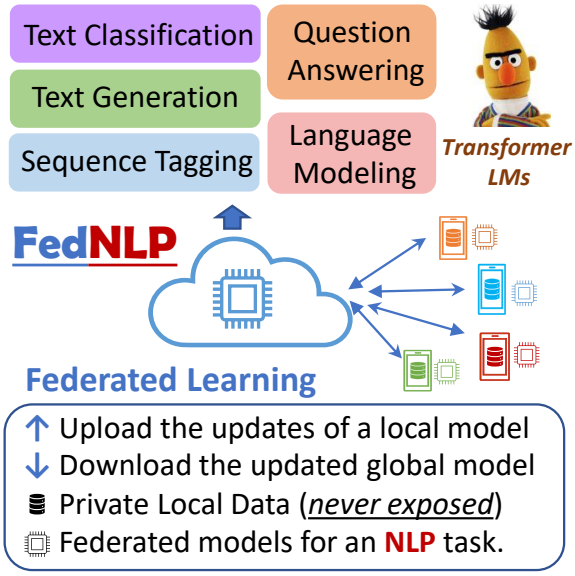

Figure 1: The FedNLP benchmarking framework.

To respect the privacy of the users and abide by these regulations, we must assume that users' data in a *silo* are not allowed to transfer to a centralized server or other clients. For example, a client cannot share its private user data (e.g., documents, conversations, questions asked on the website/app) with other clients. This is a common concern for *organizations* such as hospitals, financial institutions, or legal firms, as well as *personal computing devices* such as smartphones, virtual assistants (e.g., Amazon Alexa, Google Assistant, etc.), or a personal computer. However, from a machine learning perspective, models trained on a centralized dataset that combine the data from all organizations or devices usually result in better performance in the NLP domain. Therefore, it

[1] We have uploaded our code and will make it public.

is of vital importance to study NLP problems in such a realistic yet more challenging scenario — i.e., training data are distributed across different clients and cannot be shared for privacy concerns.

The nascent field of *federated learning* (et al, 2019; Li et al., 2020b) (FL) aims to enable many individual clients to train their models jointly while keeping their local data *decentralized* and completely *private* from other users or a centralized server. A common training schema of FL methods is that each client sends its model parameters to the server, which updates and sends back the global model to all clients in each round. Since the raw data of one client has never been exposed to others, FL is promising as an effective way to address the above challenges, particularly in the NLP domain, where many user-generated text data contain sensitive and/or personal information.

Despite the growing progress in the FL domain, research into and application for NLP has been rather limited. There are indeed several recent works on using FL methods for processing medical information extraction tasks (Sui et al., 2020). However, such prior work usually has its experimental setup and specific task, making it difficult to fairly compare these FL methods and analyze their performance in other NLP tasks. We argue that future research in this promising direction (FL for NLP) would highly benefit from a universal benchmarking platform for systematically comparing different FL methods for NLP. To the best of our knowledge, such a benchmarking platform is still absent from the literature.

Therefore, our goal in this paper is to provide comprehensive comparisons between popular FL methods (e.g., FedAvg (McMahan et al., 2017a), FedOPT (Reddi et al., 2020), FedProx (Li et al., 2020c)) for four mainstream formulations of NLP tasks: text classification, sequence tagging, question answering, and seq2seq generation. Although there are few available realistic FL datasets for NLP due to privacy concerns, we manage to use existing NLP datasets to create various non-IID data partitions over clients. These non-IID partitions simulate various kinds of distribution shifts (e.g., label, features, quantities, etc.) over the clients, which often happen in real-world NLP applications. As for the base NLP models, we use the Transformer architecture (Vaswani et al., 2017) as the backbone and support a wide range of pre-trained LMs such as DistilBERT (Sanh et al., 2019), BERT (Devlin et al., 2019), BART (Lewis et al., 2020), etc. To conduct extensive experiments, we need to support the experiments with multiple options on dimensions such as (1) *task formulations*, (2) *NLP models*, (3) *FL algorithms*, and (4) *non-IID partitions*. Therefore, we propose FedNLP, a modular framework with universal interfaces among the above four components, which is thus more extensible for supporting future research in FL for NLP.

We aim to unblock the research of FL for NLP with the following two-fold contributions:

- **Evaluation and analysis.** We systematically compare popular federated learning algorithms for mainstream NLP task formulations under multiple non-IID data partitions, which thus provides the first comprehensive understanding. Our analysis reveals that there is a considerably large gap between centralized and decentralized training under various settings. We also analyze the efficiency of different FL methods and model sizes. With our analysis, we highlight several directions to advance FL for NLP.
- **Resource.** The implementation of our experiments forms a general open-source framework, FedNLP, which is capable of evaluating, analyzing, and developing FL methods for NLP. We also provide decentralized NLP datasets of various task formulations created by various non-IID partitioning strategies for future research.

The remainder of this paper is structured as follows. We introduce the background knowledge of federated learning and several typical FL algorithms in §2. Then, we present the proposed non-IID partitioning strategies to create synthetic datasets for different task formulations in §3. Our results, analysis, and findings are in §4. Finally, we discuss related work (§5) and conclusions (§6).

2 Federated Learning for NLP

In this section, we first introduce the background knowledge of federated learning (FL) in the context of NLP tasks. Then, we illustrate a unified FL framework that we used to study typical FL algorithms. Based on this, we build our own research framework, a general pipeline for benchmarking and developing FL methods for NLP.

2.1 Federated Learning Concepts

Federated learning (FL) is a machine learning paradigm where multiple entities (clients) collaborate in solving a machine learning problem under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objectives (Kairouz et al., 2019). Therefore, federated learning has been seen as a promising direction to decrease the risk of attack and leakage, reduce the difficulty and cost of data movement, and meet the privacy-related data storage regulations.

In the basic conception of federated learning, we would like to minimize the objective function,

$$\begin{aligned} F(\boldsymbol{x}) &= \mathbb{E}_{i\sim\mathcal{P}}[F_i(\boldsymbol{x})], \\ \text{where} \quad F_i(\boldsymbol{x}) &= \mathbb{E}_{\xi\sim\mathcal{D}_i}[f_i(\boldsymbol{x},\xi)]. \end{aligned} \tag{1}$$

$\boldsymbol{x} \in \mathbb{R}^d$ represents the parameter for the global model, $F_i : \mathbb{R}^d \to \mathbb{R}$ denotes the local objective function at client i, and $\mathcal{P}$ denotes a distribution on the collection of clients $\mathcal{I}$. The local loss functions $f_i(\boldsymbol{x}, \xi)$ are often the same across all clients, but the local data distribution $\mathcal{D}_i$ will often vary, capturing data heterogeneity.

Federated averaging (FedAvg) (McMahan et al., 2017a) is a common algorithm to solve (1) by dividing the training process into rounds. At the beginning of the t-th round ($t \geq 0$), the server broadcasts the current global model $\boldsymbol{x}^{(t)}$ to a *cohort* of participants: a random subset of clients from $\mathcal{S}^{(t)}$ which includes M clients in total. Then, each sampled client in the round's cohort performs τ_i local SGD updates on its own local dataset and sends the local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}^{(t)}$ to the server. Finally, the server uses the aggregated $\Delta_i^{(t)}$ to update the global model: $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \frac{\sum_{i\in\mathcal{S}^{(t)}} p_i \Delta_i^{(t)}}{\sum_{i\in\mathcal{S}^{(t)}} p_i}$. where p_i is the relative weight of client i. The above procedure will repeat until the algorithm converges. In the *cross-silo* setting where all clients participate in training on every round (each cohort is the entire population), we have $\mathcal{S}^{(t)} = \{1, 2, \ldots, M\}$. Consequently, we can learn a global model to benefit all clients while preserving their data privacy.

2.2 Our Unified Framework for FL

In this work, we propose to use FedOPT (Reddi et al., 2020), a generalized version of FedAvg, to build the `FedNLP` platform. As the pseudo-code presented in Algorithm 1, the algorithm is parameterized by two gradient-based optimizers: CLIENTOPT and SERVEROPT with client learning rate η and server learning rate η_s, respectively. While CLIENTOPT is used to update the local models, SERVEROPT treats the negative of aggregated local changes $-\Delta^{(t)}$ as a pseudo-gradient and applies it to the global model. This optimization framework generalizes to many aggregation-based FL algorithms and simplifies the system design.

Algorithm 1: FEDOPT (Reddi et al., 2020)): A Generic FedAvg Algorithm

Input: Initial model $\boldsymbol{x}^{(0)}$, CLIENTOPT, SERVEROPT

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Sample a subset $\mathcal{S}^{(t)}$ of clients
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Initialize local model $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Compute local stochastic gradient $g_i(\boldsymbol{x}_i^{(t,k)})$
      Perform local update $\boldsymbol{x}_i^{(t,k+1)} =$ CLIENTOPT$(\boldsymbol{x}_i^{(t,k)}, g_i(\boldsymbol{x}_i^{(t,k)}), \eta, t)$
    Compute local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}_i^{(t,0)}$
  Aggregate local changes $\Delta^{(t)} = \sum_{i\in\mathcal{S}^{(t)}} p_i \Delta_i^{(t)} / \sum_{i\in\mathcal{S}^{(t)}} p_i$
  Update global model $\boldsymbol{x}^{(t+1)} =$ SERVEROPT$(\boldsymbol{x}^{(t)}, -\Delta^{(t)}, \eta_s, t)$

To make our research general, we explore different combinations of SEVEROPT and CLIENTOPT. The original FedAvg algorithm implicitly sets SEVEROPT and CLIENTOPT to be SGD, with a fixed server learning rate η_s of 1.0. FedProx (Li et al., 2020c), tackling statistical heterogeneity by restricting the local model updates to be closer to the initial (global) model, can be easily incorporated into this framework by adding L2 regularization for better stability in training. Moreover, given that AdamW (Loshchilov and Hutter, 2019) is widely used in NLP, we set it for `ClientOpt` and let the `ServerOpt` to be SGD with momentum to reduce the burden of tuning.

2.3 The Proposed FedNLP Framework

To support our research in this paper and other future work in the area of federated learning for NLP, we build a general research framework named FedNLP, based on the above universal optimization framework. We here briefly highlight its unique features and leave the details in the follow-

ing content and detailed design is shown in App. F. First, FedNLP is the very first framework that connects multiple FL algorithms with Transformer-based models, to our best knowledge. Also, we implement a flexible suite of interfaces to support different types of NLP tasks and models, as well as different non-IID partitioning strategies (Sec. 3.2. To study security and privacy guarantees, we also incorporate the state-of-the-art secure aggregation algorithms such as LightSecAgg (see F.5).

3 Benchmarking Setup with FedNLP

In this section, we introduce the creation of our benchmark datasets from a set of chosen NLP tasks with different non-IID partition methods. We evaluate various FL methods on these datasets.

3.1 Task Formulations, Datasets, and Models

There are numerous NLP applications, but most of them can be categorized based on four mainstream formulations: text classification (TC), sequence tagging (ST), question answering (QA), and seq2seq generation (SS). The formal definition of each formulation is detailed in Appendix §B. To cover all formulations while keeping our experiments in a reasonable scope, we select one representative task for each formulation:

- **Text Classification**: `20Newsgroup` (Lang, 1995) is a news classification dataset with annotations for 20 labels. We showcase our FedNLP with this dataset as it has a larger output space (20 labels) than sentiment-analysis datasets, which is an important factor for the label-distribution shift scenarios. .
- **Sequence Tagging**: `OntoNotes` (Pradhan et al., 2013) (5.0) is a corpus where sentences have annotations for the entity spans and types. We use it for the named entity recognition task, which is fundamental to information extraction and other applications.
- **QA**: `MRQA` (Fisch et al., 2019) is a benchmark consisting of 6 popular datasets[2]: `SQuAD` (Rajpurkar et al., 2016) (8529/431), `NewsQA` (Trischler et al., 2017) (11877/613), `TriviaQA` (Joshi et al., 2017) (4120/176) , `SearchQA` (Dunn et al., 2017) (9972/499) , `HotpotQA` (Yang et al., 2018b) , and `NQ` (Kwiatkowski et al., 2019) (9617/795).

[2]We only use part of the data to demonstrate and verify our hypothesis; we show the train/test split in brackets.

Task	Txt.Cls.	Seq.Tag.	QA	Seq2Seq
Dataset	20News	Onto.	MRQA	Giga.
# Training	11.3k	50k	53.9k	10k
# Test	7.5k	5k	3k	2k
# Labels	20	37*	N/A	N/A
Metrics	Acc.	F-1	F-1	ROUGE

Table 1: Statistics of the selected datasets for our experiments. *37 is the size of the tag vacabulary.

- **Seq2Seq Generation**: `Gigaword` (DBL, 2012) is a news corpus with headlines that is often used for testing seq2seq models as a summarization task. Other tasks such as dialogue response generation and machine translation can also be adapted to this format.

We show the basic statistics of the above datasets in Table 1. Note that our FedNLP as a research platform supports a much wider range of specific tasks of each formulation, while we only introduce the ones used in our experiments here with typical settings. Moreover, our contribution is more of a general FL+NLP benchmarking platform instead of particular datasets and partitions.

Base NLP Models. Fine-tuning pre-trained LMs has been the *de facto* method for NLP research, so we focus on testing Transformer-based architectures in FedNLP. Specifically, we choose to use BART (Lewis et al., 2020), a text-to-text Transformer model similar to the T5 model (Raffel et al., 2020), for seq2seq tasks.

3.2 Non-IID Partitioning Strategies

The existing datasets have been used for centralized training in NLP. As our focus here is to test decentralized learning methods, we need to distribute the existing datasets to a set of clients. It is the non-IIDness of the client distribution that makes federated learning a challenging problem. Thus, we extend the common practice widely used in prior works to the NLP domain for generating synthetic FL benchmarks (Li et al., 2021). We first introduce how we control the *label distribution shift* for TC and ST, then the *quantity distribution shift*, and finally how we model the distribution shift in terms of input features for non-classification NLP tasks (e.g., summarization).

Non-IID Label Distributions. Here we present how we synthesize the data partitions such that clients the share same (or very similar) number

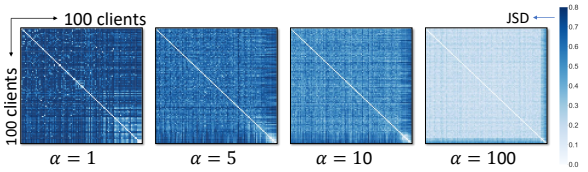

Figure 2: The *J-S divergence* matrix between 100 clients on the *20News* dataset when $\alpha \in \{1, 5, 10, 100\}$. Each sub-figure is a 100x100 symmetric matrix. The intensity of a cell (i, j)'s color here represents the distance between the label distribution of Client i and j. It is expected that when α is smaller, the partition over clients is more non-IID in terms of their label distributions.

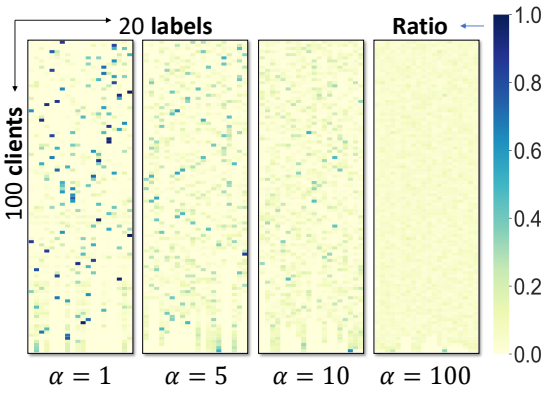

Figure 3: Visualizing the **non-IID label distributions** on *20News* with α being $\{1, 5, 10, 100\}$. Each sub-figure is a 100x20 matrix, where 100 is the number of clients, and 20 is the number of labels. The intensity of a cell here represents the ratio of a particular label in the local data of a client. When α is smaller (1, 5, 10), each client has a relatively unique label distribution, thus the differences between clients are larger; when $\alpha = 100$, every client has a nearly uniform label distribution.

of examples, but have different *label distributions* from each other. We assume that on every client training, examples are drawn independently with labels following a categorical distribution over L classes parameterized by a vector $\boldsymbol{q}$ ($q_i \geq 0, i \in [1, L]$ and $\|\boldsymbol{q}\|_1 = 1$). To synthesize a population of non-identical clients, we draw $\boldsymbol{q} \sim \text{Dir}_L(\alpha \boldsymbol{p})$ from a *Dirichlet* distribution, where $\boldsymbol{p}$ characterizes a prior class distribution over L classes, and $\alpha > 0$ is a concentration parameter controlling the identicalness among clients. For each client C_j, we draw a $\boldsymbol{q}_j$ as its label distribution and then sample examples without replacement from the global dataset according to $\boldsymbol{q}_j$. With $\alpha \to \infty$, all clients have identical distributions to the prior (i.e., uniform distribution); with $\alpha \to 0$, on the other extreme, each client holds examples from only one class chosen at random. In Fig. 2, we show heatmaps for visualizing the distribution differences between each client. Figure 3 shows an example of the concrete label distributions for all clients with different α. We can see that when α is smaller, the overall label distribution shift becomes larger.

Controlling non-IID Quantity. It is also common that different clients have very different data quantities while sharing similar label distribution. We thus also provide a quantity-level Dirichlet allocation $\boldsymbol{z} \sim \text{Dir}_N(\beta)$ where N is the number of clients. Then, we can allocate examples in a global dataset to all clients according to the distribution $\boldsymbol{z}$ — i.e., $|\mathcal{D}_i| = z_i |\mathcal{D}_G|$. If we would like to model both quantity and label distribution shift, it is also easy to combine both factors. Note that one could assume it is a uniform distribution $\boldsymbol{z} \sim U(N)$, (or $\beta \to \infty$) if we expect all clients to share a similar number of examples. A concrete example is shown in Figure 8 (Appendix).

Controlling non-IID Features. Although straightforward and effective, the above label-based Dirichlet allocation method has a major limitation — it is only suitable for text classification tasks where the outputs can be modeled as category-based random variables. To create synthetic partitions for other non-classification NLP tasks and model distribution shifts, we thus propose a partition method based on feature clustering. Specifically, we use Sentence-BERT (Reimers and Gurevych, 2019) to encode each example to a dense vector by their text then we apply K-Means clustering to get the cluster label of each example; finally, we use these cluster labels (as if they were classification tasks) to follow the steps in modeling *label distribution shift*. There are two obvious benefits of this clustering-based Dirichlet partition method: 1) It enables us to easily synthesize the FL datasets for non-classification tasks (i.e., ST, QA, SS) as they do not have discrete labels as output space; 2) The BERT-based clustering results naturally imply different sub-topics of a dataset, and thus feature shift can be seen as a shift of latent labels — we can reuse the same method for the label-based Dirichlet partition method.

Natural Factors For datasets like MRQA, we consider a cross-silo setting where each client is associated with a particular sub-dataset (out of the six datasets of the same format), forming a natural distribution shift based on the inherent factors such as data source and annotating style.

Task	Dataset	Partition	Clients	FedAvg	FedProx	FedOPT	# Rounds
Text Classification	20news	α =1 (label shift)	100	0.5142	0.5143	0.5349	22
Sequence Tagging	OntoNotes	α =0.1 (label shift)	30	0.7382	0.6731	0.7918	17
Question Answering	MRQA	natural factor	6	0.2707	0.2706	0.3280	13
Seq2Seq Generation	Gigaword	α =0.1 (feature shift)	100	0.3192	0.3169	0.3037	13

Table 2: The comparisons between different FL methods under the same setting on different NLP tasks. The number of workers per round are 10, expect for the MRQA task, which uses 6.

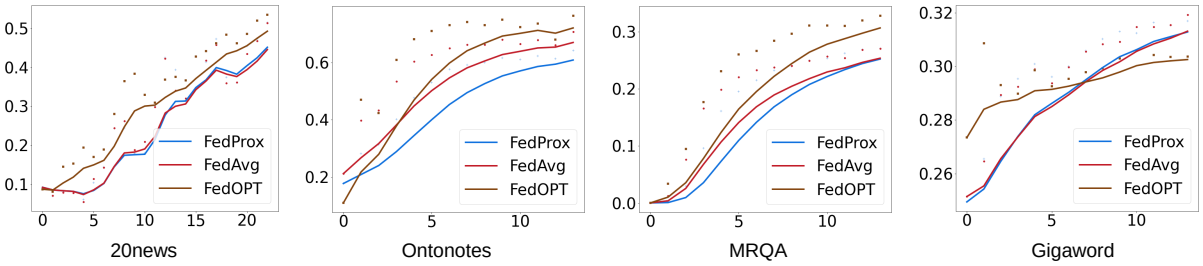

Figure 4: The learning curves of the three FL Methods on four different task formulations. The metrics used for these tasks are accuracy, span-F1, token-F1, and ROUGE respectively; The x-axis is the number of rounds in federated learning.

4 Experimental Results and Analysis

In this section, we aim to analyze typical federated learning methods (introduced in on our benchmark datasets with multiple dimensions with the base NLP models listed previously. We put more implementation details and additional results in Appendix. We organize our extensive experimental results and findings from the analysis as a collection of research questions with answers.

Experimental Setup and Hyper-parameters. We use DistilBERT and BART-base for most of our experiments, as the former is a distilled version of the BERT model and has a 7x speed improvement over BERT-base on mobile devices — a common scenario for FL applications; the BART-base model is the most suitable option considering the trade-off between performance and computation cost. We leave our implementation details and the selected hyper-parameters in the submitted supplementary materials.

Our experiments cover both cross-device and cross-silo settings. As shown in Table 2, in the cross-device setting, we use uniform sampling to select 10 clients for each round when the client number in a dataset is very large (e.g., 100). For the cross-silo setting, each round will select the same number of clients (we use 6 for the QA task). The local epoch number is set to 1 for all experiments. To make our results reproducible, we use *wandb.ai* to store all experiment logs and hyper-parameters as well as running scripts.

Q1: How do popular FL methods perform differently under the same setting?

We compare the three typical FL methods under the same setting (i.e., data partition, communication rounds, etc.) for each task formulation. As shown in Table 2, we report the results of FedAvg, FedProx, and FedOPT. We can see that overall FedOPT performs better than the other two methods, with the only exception being in the seq2seq generation task. FedAvg and FedProx perform similarly with marginal differences, but FedAvg outperforms FedProx in sequence tagging. These two exceptions are surprising findings, as many prior works in the FL community show that FedOPT is generally better than FedProx than FedAvg on vision tasks and datasets.

We conjecture that such inconsistent performance across tasks suggests the difference in terms of the loss functions have a great impact on FL performance. Seq2seq and sequence tagging tasks usually have more complex loss landscapes than text classification, as they are both typical structured prediction tasks, while the text classification has a much smaller output space. From Fig. 4, we see that the FedOPT outperforms the other two methods at the beginning while gradually becoming worse over time.

This tells us that the use of AdamW as the client optimizer may not always be a good choice, especially for a complex task such as the Seq2Seq ones, as its adaptive method for scheduling learn-

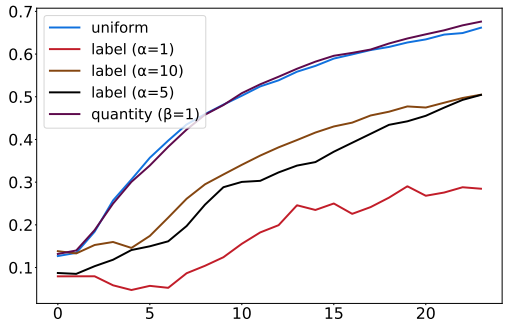

Figure 5: Testing FedOPT with `DistilBERT` for `20News` under different data partition strategies.

ing rates might cause implicit conflicts. These observations suggest that federated optimization algorithms need to be tailored for various NLP tasks, and exploring FL-friendly model architecture or loss function can also be promising directions to address these challenges.

Q2: How do different non-IID partitions of the same data influence FL performance?

The FedNLP platform supports users to investigate the performance of an FL algorithm with a wide range of data partitioning strategies, as discussed in §3.2. Here we look at the training curves of the FedOPT on different partitions, as shown in Figure 5. We reveal several findings:

- When α is smaller (i.e., the partition is more non-IID in terms of their label distribution), the performance tends to degrade, based on the three curves ($\alpha = \{1, 5, 10\}$).
- The variance is also larger when the label distribution shift is larger. Both uniform and quantity-skew partitions have a smoother curve, while the variance is smaller for a larger α (e.g., 10).
- Quantity skew does not introduce a great challenge for federated learning when the label distribution is closer to the uniform one.

These findings suggest that it is important to design algorithms to mitigate data heterogeneity. One promising direction is *personalized* FL, which enables each client to learn its own personalized model via adapting its local data distribution and system resources (Dinh et al., 2020; Fallah et al., 2020; Li et al., 2020a).

Q3: How does freezing of Transformers influence the FL performance?

Communication cost is a major concern in the federated learning process. It is thus natural to consider freezing some Transformer layers of the client models in order to reduce the size of the trainable parameters that will be transmitted between servers and clients. To study the influence of freezing layers on the FL performance, we conduct a series of experiments that freeze the layers from the embedding layer (E) to the top layer (L_5) of DistilBERT with both centralized training and FedOPT on the text classification task.

Frozen Layers	# Tunable Paras.	Cent.	FedOpt.
None	67.0M	86.86	55.11
E	43.1M	86.19	54.86
$E + L_0$	36.0M	86.54	52.91
$E + L_{0\to1}$	29.0M	86.52	53.92
$E + L_{0\to2}$	21.9M	85.71	52.01
$E + L_{0\to3}$	14.8M	85.47	*30.68*
$E + L_{0\to4}$	7.7M	82.76	16.63
$E + L_{0\to5}$	0.6M	*63.83*	12.97

Table 3: Performance (Acc.%) on 20news (TC) when different parts of `DistilBERT` are frozen for centralized training and FedOpt (at 28-th round). E stands for the embedding layer and L_i means the i-th layer. The significant lower accuracy are underlined.

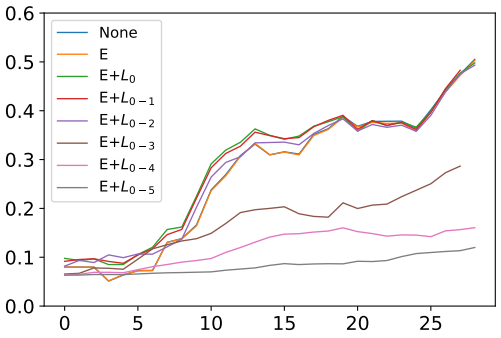

Figure 6: Testing `FedOPT` with `DistilBERT` for `20News` under different frozen layers.

We report our results in Table 3 and Figure 6. We find that in centralized training, the largest performance gain happens when we unfreeze the last layer, while in FedOPT we have to unfreeze the last three layers to enjoy a comparable performance with the full model. This suggests that reducing communication costs via freezing some layers of Transformer LMs is feasible, though one should be aware that the experience in centralized training may not generalize to the FL experiments.

Q4: Are compact model DistilBERT adequate for FL+NLP?

We know that BERT has a better performance than DistilBERT for its larger model size. However, is it cost-effective to use BERT rather than DistilBERT? To study this, we compare the perfor-

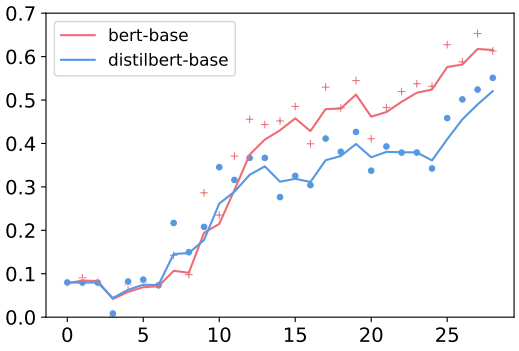

Figure 7: `FedOPT` for `20News` with different LMs.

mance of both models with FedOPT on text classification, sharing the same setting as the above experiments. As shown in Figure 7, although BERT-base achieves better performance, the performance of DistilBERT is not significantly worse. Considering the communication cost (BERT-base is almost 2x larger), we argue that using DistilBERT is a more cost-effective choice for both experimental analysis and realistic applications.

5 Related Work

FL benchmarks and platforms. In the last few years a proliferation of frameworks and benchmark datasets have been developed to enable researchers to better explore and study algorithms and modeling for federated learning, both from academia: LEAF(Caldas et al., 2018), FedML (He et al., 2020c), Flower (Beutel et al., 2020), and from the industry: PySyft (Ryffel et al., 2018), TensorFlow-Federated (TFF) (Ingerman and Ostrowski, 2019), FATE (Yang et al., 2019), Clara (NVIDIA, 2019), PaddleFL (Ma et al., 2019), Open FL (Intel®, 2021). However, most platforms only focus on designing a unified framework for federated learning methods and do not provide a dedicated environment for studying NLP problems with FL methods. LEAF (Caldas et al., 2018) contains a few text datasets, however, it is limited to classification and next-word prediction datasets and does not consider the pre-trained language models. We want to provide a dedicated platform for studying FL methods in realistic NLP applications with state-of-the-art language models.

Federated learning in NLP applications. There are a few prior works that have begun to apply FL methods in privacy-oriented NLP applications. For example, federated learning has been applied to many keyboard-related applications (Hard et al., 2018; Stremmel and Singh, 2020; Leroy et al., 2019; Ramaswamy et al., 2019; Yang et al., 2018a), sentence-level text intent classification using Text-CNN (Zhu et al., 2020), and pretraining and fine-tuning of BERT using medical data from multiple silos without fetching all data to the same place (Liu and Miller, 2020). FL methods also have been proposed to train high-quality language models that can outperform the models trained without federated learning (Ji et al., 2019; Chen et al., 2019). Besides these applications, some work has been done in medical relation extractions (Ge et al., 2020) and medical name entity recognition (Sui et al., 2020). These methods use federated learning to preserve the privacy of sensitive medical data and learn data in different platforms, excluding the need for exchanging data between different platforms.

Our work aims to provide a unified platform for studying various NLP applications in a shared environment so that researchers can better design new FL methods either for a specific NLP task or as a general-purpose model. The aforementioned prior works would thus be a particular instance of the settings supported by the FedNLP platform.

6 Conclusion and Future Directions

Our key contribution is providing a thorough and insightful empirical analysis of existing federated learning algorithms in the context of NLP models. Notably, We compare typical FL methods for four NLP task formulations under multiple non-IID data partitions. Our findings reveal both promise and the challenges of FL for NLP. In addition, we also provide a suite of resources to support future research in FL for NLP (e.g., a unifying framework for connecting Transformer models with popular FL methods and different non-IID partition strategies). Thus, we believe our well-maintained open-source codebase to support future work in this area.

Promising future directions in FL for NLP includes: 1) minimizing the performance gap, 2) improving the system efficiency and scalability, 3) trustworthy and privacy-preserving NLP, 4) personalized FL methods for NLP, etc. (Please see Appendix E for more details.)

Ethical Considerations and Limitations(*)

Ethical considerations. The key motivation of FedNLP (and FL) is to protect the data privacy of general users by keeping their data on their own devices while benefit from a shared model from a broader community. Among the risks that need to be considered in any deployment of NLP are that responses may be wrong, or biased, in ways that would lead to improperly justified decisions. Although in our view the current technology is still relatively immature, and unlikely to be fielded in applications that would cause harm of this sort, it is desirable that FedNLP methods provide audit trails, and recourse so that their predictions can be explained to and critiqued by affected parties.

Limitations. One limitation of our work is that we have not analyzed the privacy leakage of FL methods. We argue that novel privacy-centric measures are orthogonal to the development FL methods, which is beyond the scope of our work. How to fairly analyze the privacy leakage is now still an open problem for both FL and NLP, and it is only possible to study this when we have an existing platform like FedNLP.

References

2012. *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, AKBC-WEKEX@NAACL-HLT 2012, Montrèal, Canada, June 7-8, 2012*.

James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. 2020. Flower: A friendly federated learning research framework. *ArXiv preprint*.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *ArXiv preprint*.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. 2019. Federated learning of n-gram language models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 121–130, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. 2020. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. U. Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *ArXiv*.

A. Elkordy and A. Avestimehr. 2020. Secure aggregation with heterogeneous quantization in federated learning. *ArXiv*.

P. Kairouz et al. 2019. Advances and open problems in federated learning. *ArXiv*.

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

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and X. Xie. 2020. Fedner: Privacy-preserving medical named entity recognition with federated learning. *ArXiv*.

Andrew Hard, K. Rao, Rajiv Mathews, F. Beaufays, S. Augenstein, Hubert Eichner, Chloé Kiddon, and D. Ramage. 2018. Federated learning for mobile keyboard prediction. *ArXiv*.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020a. Fednas: Federated deep learning via neural architecture search.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020b. Group knowledge transfer: Federated learning of large cnns at the edge. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Yu Rong, Peilin Zhao, Junzhou Huang, M. Annavaram, and S. Avestimehr. 2021. Fedgraphnn: A federated learning system and benchmark for graph neural networks.

Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020c. Fedml: A research library and benchmark for federated machine learning. *ArXiv preprint*.

Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. 2019. Central server free federated learning over single-sided trust social networks. *ArXiv preprint*.

Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. 2020d. Milenas: Efficient neural architecture search via mixed-level reformulation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11990–11999. IEEE.

Alex Ingerman and Krzys Ostrowski. 2019. *TensorFlow Federated*.

Intel®. 2021. Intel® open federated learning.

Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. 2019. Learning private neural language modeling with attentive aggregation. *2019 International Joint Conference on Neural Networks (IJCNN)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *ArXiv preprint*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proc. of ICML*.

David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. 2019. Federated learning for keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6341–6345. IEEE.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Q. Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2021. Federated learning on non-iid data silos: An experimental study. *ArXiv*.

T. Li, Shengyuan Hu, A. Beirami, and Virginia Smith. 2020a. Ditto: Fair and robust federated learning through personalization.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and V. Smith. 2020b. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020c. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

D. Liu and T. Miller. 2020. Federated pretraining and fine tuning of bert using clinical notes from multiple silos. *ArXiv*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. 2020. Privacy and robustness in federated learning: Attacks and defenses. *ArXiv preprint*.

Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. 2019. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Dompuding*, (1).

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017a. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017b. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

NVIDIA. 2019. Nvidia clara.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Saurav Prakash and Amir Salman Avestimehr. 2020. Mitigating byzantine attacks in federated learning. *ArXiv preprint*.

Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Yair Yona, Shilpa Talwar, Salman Avestimehr, and Nageen Himayat. 2020. Coded computing for low-latency federated learning over wireless edge networks. *IEEE Journal on Selected Areas in Communications*, (1).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, (140).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Swaroop Indra Ramaswamy, Rajiv Mathews, K. Rao, and Franccoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. *ArXiv*.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. *ArXiv preprint*.

General Data Protection Regulation. 2016. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union. Available at: http://ec. europa. eu/justice/data-protection/reform/files/regulation_oj_en. pdf (accessed 20 September 2017)*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. *ArXiv preprint*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4424–4434.

Jinhyun So, Başak Güler, and A Salman Avestimehr. 2020. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*.

Jinhyun So, Başak Güler, and A Salman Avestimehr. 2021a. Codedprivateml: A fast and privacy-preserving framework for distributed machine learning. *IEEE Journal on Selected Areas in Information Theory*, (1).

Jinhyun So, Başak Güler, and A Salman Avestimehr. 2021b. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, (1).

Joel Stremmel and Arjun Singh. 2020. Pretraining federated text models for next word prediction. *ArXiv*.

Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. 2020. FedED: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2118–2128, Online. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020a. Attack of the tails: Yes, you really can backdoor federated learning. *ArXiv preprint*.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. 2020b. Federated learning with matched averaging. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, (2).

T. Yang, G. Andrew, Hubert Eichner, Haicheng Sun, W. Li, Nicholas Kong, D. Ramage, and F. Beaufays. 2018a. Applied federated learning: Improving google keyboard query suggestions. *ArXiv*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018b. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14747–14756.

Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical studies of institutional federated learning for natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 625–634, Online. Association for Computational Linguistics.

Appendix

A FL+NLP

Many realistic NLP services heavily rely on users' local data (e.g., text messages, documents and their tags, questions and selected answers, etc.), which can be located at either personal devices or larger data-silos for organizations. These local data are usually regarded as highly private and thus not directly accessible by anyone, according to many data privacy regulations; this makes it difficult to train a high-performance model to benefit users. Federated learning aims to solve machine learning under such a privacy-preserving use case, thus offering a novel and promising direction to the community: FL+NLP.

Apart from the goal of learning a shared global model for all clients, FL also provides a new perspective for many other interesting research questions in NLP. One related direction is to develop personalized models for NLP applications, which requires both protection of data privacy and transferred ability on users' own input feature distribution caused by language styles, interested topics and so on. The recent concerns on adversarial attacks and safety issues of NLP models are also highly related to FL+NLP. We thus believe FL+NLP is of vital importance for applying NLP technologies in realistic use cases and could benefit many relevant research areas.

A.1 Challenges of Applying FL in NLP

Given the promising benefits of studying FL+NLP, however, this research direction is currently blocked by the lack of a standardized platform providing fundamental building blocks: benchmark datasets, NLP models, FL methods, evaluation protocols, etc. Most of the current FL platforms either focus on unifying various FL methods and use computer vision models and datasets for their experiments, but lack the ability to connect the study of pre-trained language models, the most popular NLP , and realistic NLP applications of various task formulations.

The first challenge in developing a comprehensive and universal platform for FL+NLP is to deal with various task formulations for realistic NLP applications, which have different input and output formats (Section B). As the non-IID data partition over clients is the major feature of FL problems, it is also a challenge to simulate the realistic non-IID partition for existing NLP datasets (Section 3.2). Finally, a platform also must integrate various FL methods with the Transformer-based NLP models for a variety of task types, and thus a flexible and extensible learning framework is needed. In particular, the conventional trainer component of Transformers now needs to be modified for efficient and safe communications towards federated learning (Section F).

B Basic Formulations of NLP Tasks

There are various types of NLP applications, but many of them share a similar task formulation (i.e., input-and-put formats). We show four common task formulations that can cover most of the mainstream NLP applications: text classification, sequence tagging, question answering, sequence-to-sequence generation.

Text Classification (TC) The input is a sequence of words, $x = [w_1, w_2, \dots]$, and the output is a label y in a fixed set of labels $\mathcal{L}$. Many NLP applications can be formulated as text classification tasks. For example, we can use TC models for classifying the topic of a news article to be *political*, *sports*, *entertainment*, etc., or analyzing movie reviews to be *positive*, *negative* or *neutral*.

Sequence Tagging (ST) The input is a sequence of words, $x = [w_1, w_2, \dots, w_N]$, and the output is a same-length sequence of tags $y = [t_1, t_2, \dots, t_N]$, where t_i is in a fixed set of labels $\mathcal{L}$. The main difference between TC and ST is that ST learns to classify the label of each token in a sentence, which is particularly useful in analyzing syntactic structures (e.g., part-of-speech analysis, phrase chunking, and word segmentation) and extracting spans (e.g., named entity recognition).

Question Answering (QA) Given a passage $P = [w_1, w_2, \dots, w_N]$ and a question q as input, the task is to locate a span in the passage as the answer to the question. Thus, the output is a pair of token index (s, e) where $s, e \in \{1, 2, \dots, N\}$ for denoting the begin and end of the span in the passage. This particular formulation is also known as *reading comprehension*.

Natural Language Generation (NLG) Both input and output are sequence of words, $x = [w_1^i, w_2^i, \dots, w_N^i]$, $y = [w_1^o, w_2^o, \dots, w_M^o]$. It is shared by many realistic applications such as summarization, response generation in dialogue systems, machine translation, etc.

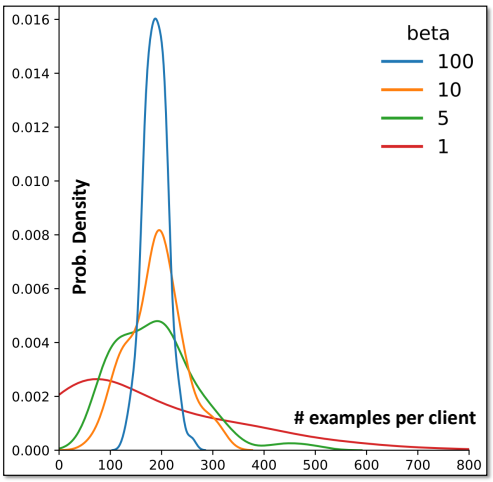

Figure 8: The probability density of quantity of training examples in each of the 100 clients on the *20News* dataset with different β. When β is larger, then all clients share more similar numbers of examples; when β is smaller, then the range of the quantity is much wider — i.e., the larger differences between clients in terms of their sizes of datasets.

Language Modeling (LM) The left-to-right language modeling task considers a sequence of words as the input $x = [w_1, w_2, \ldots, w_n]$ and a token $y = w_{n+1}$ as the output. The output token is expected to be the most plausible next word of the incomplete sentence denoted as x. Although the direct application of LM is limited, a high-performance pre-trained language model can benefit a wide range of NLP applications (as above) via fine-tuning. It also serves as an excellent test bed as it requires no human annotations at all.

Others. There are some other applications that not are covered by the above four basic formulations, and our extensible platform (detailed in Section F) enables users to easily implement their specific tasks. For each task formulation, we show which datasets are used in FedNLP and how we partition them in Section 3.

C Implementation Details

Non-IID. Label Distribution Note that this might cause a few clients not to have enough examples to sample for particular labels if they are already used up. Prior works choose to stop assigning early and remove such clients, but it consequently loses the other unused examples and also causes the inconsistency of client numbers. Thus, to avoid these issues, we propose a *dynamic reassigning* method which complement the vacancy of a label by filling in the examples of other labels based on their current ratio of remaining unassigned examples.

C.1 The FedNLP Training Pipeline: Security and Efficiency

Under the definition of federated learning in Algorithm 1, we design a training system to support the research of NLP in the FL paradigm. We highlight its core capabilities and design as follows.

Supporting diverse FL algorithms. `FedNLP` aims to enable flexible customization for future algorithmic innovations. We have supported a number of classical federated learning algorithms, including FedAvg (McMahan et al., 2017a), FedOPT (Reddi et al., 2020), and FedProx (Li et al., 2020c). These algorithms follow the same framework introduced in Algorithm 1. The algorithmic APIs are modularized: all data loaders follow the same format of input and output arguments, which are compatible with different models and algorithms and are easy to support new datasets; the method of defining the model and related trainer is kept the same as in centralized training to reduce the difficulty of developing the distributed training framework. For new FL algorithm development, worker-oriented programming reduces the difficulty of message passing and definition. More details are introduced in Appendix F.3.

Enabling secure benchmarking with lightweight secure aggregation. In particular, `FedNLP` enhances the security aspect of federated training, which is not supported by existing non-NLP-oriented benchmarking libraries (e.g., TFF, LEAF). This is motivated by the fact that model weights from clients may still have the risk of privacy leakage (Zhu et al., 2019). To break this barrier, we integrate secure aggregation (SA) algorithms to the `FedNLP` system. NLP researchers do not need to master security-related knowledge and also benefit from a secure distributed training environment. To be more specific, `FedNLP` supports state-of-the-art SA algorithms `LightSecAgg`, `SecAgg` (Bonawitz et al., 2017), and `SecAgg+` (Bell et al., 2020). At a high-level understanding, SA protects the client model by generating a single random mask and allows their cancellation when aggregated at the server. Consequently, the server can only see the aggregated model and not the raw model from each client. In this work, our main effort is to

design and optimize these SA algorithms in the context of the FedNLP system. We provide an algorithmic performance comparison in Appendix F.5.

Realistic evaluation with efficient distributed system design. FedNLP aims to support distributed training in multiple edge servers (e.g, AWS EC2) or edge devices (e.g., IoTs and smartphones). To achieve this, the system is designed with three layers: the application layer, the algorithm layer, and the infrastructure layer. At the application layer, FedNLP provides three modules: data management, model definition, and a single-process trainer for all task formats; at the algorithm layer, FedNLP supports various FL algorithms; at the infrastructure layer, FedNLP aims at integrating single-process trainers with a distributed learning system for FL. Specifically, we make each layer and module perform its own duties and have a high degree of modularization. We refer readers to Appendix F for a detailed description of the system architecture and design philosophy.

D More Related Works

Federated Learning Methods. Federated Learning (FL) is a widely disciplinary research area that mainly focuses on three aspects: statistical challenge, trustworthiness, and system optimization. Numerous methods have been proposed to solve statistical challenges, including FedAvg (McMahan et al., 2017b), FedProx (Li et al., 2020c), FedOPT (Reddi et al., 2020), FedNAS (He et al., 2020a,d), and FedMA (Wang et al., 2020b) that alleviate the non-IID issue with distributed optimization, and new formulations, MOCHA (Smith et al., 2017), pFedMe (Dinh et al., 2020), perFedAvg (Fallah et al., 2020), and Ditto (Li et al., 2020a), that consider personalization and fairness in federated training.

For trustworthiness, security and privacy are the two main research directions that are mainly concerned with resisting data or model attacks, reconstruction, and leakage during training (So et al., 2021b,a, 2020; Prakash et al., 2020; Prakash and Avestimehr, 2020; Elkordy and Avestimehr, 2020; Prakash et al., 2020; Wang et al., 2020a; Lyu et al., 2020). Given that modern deep neural networks are over-parameterized and dominate nearly all learning tasks, researchers also proposed algorithms or systems to improve the efficiency and scalability of edge training (He et al., 2020b,c, 2019, 2021). We refer readers to the canonical survey (Kairouz et al., 2019) for details.

Although tremendous progress has been made in the past few years, these algorithms or systems have not been fully evaluated on realistic NLP tasks introduced in this paper.

E Future Directions

Minimizing the performance gap. In the FL setting, we demonstrate that federated fine-tuning still has a large accuracy gap in the non-IID dataset compared to centralized fine-tuning. Developing algorithms for Transformer models with NLP tasks is of the highest priority.

Improving the system efficiency and scalability. Transformer models are usually large, while resource-constrained edge devices may not be able to run large models. Designing efficient FL methods for NLP tasks is thus a practical problem worth solving. How to adopt a reasonable user selection mechanism to avoid stragglers and speed up the convergence of training algorithms is also a pressing problem to be solved.

Trustworthy and privacy-preserving NLP. We argue that it is an important future research direction to analyze and assure the privacy-preserving ability of these methods, although our focus in this paper is the implementation and performance analysis of the FL methods for NLP tasks. It is now an open problem for both FL and NLP areas, while it is an orthogonal goal for improving the trustworthy of decentralized learning, and it is only possible to study privacy preservation when we have an existing FL+NLP platform. This is also part of our motivation in proposing FedNLP, and we believe our framework provides a set of flexible interfaces for future development to analyze and improve the privacy-preserving ability of FL methods for NLP tasks and beyond.

Personalized FedNLP. From the perspective of the data itself, user-generated text is inherently personalized. Designing personalized algorithms to improve model accuracy or fairness is a very promising direction. In addition, it is also an interesting problem to adapt the heterogeneous model architecture for each client in the FL network. We show that it is feasible to only fine-tune a small

Algorithm 2: The FedNLP Workflow

```
# using text classification (TC) as an example

# initialize distributed computing environment
process_id, ... = FedNLP_init()

# GPU device management
device = map_process_to_gpu(process_id, ...)

# data management
data_manager = TCDataManager (process_id, ...)
# load the data dictionary by process_id
data_dict = dm.load_federated_data(process_id)

# create model by specifying the task
client_model, ... = create_model(model_args,
    formulation="classification")

# define a customized NLP Trainer
client_trainer = TCTrainer(device,
                    client_model, ...)

# launch the federated training (e.g., FedAvg)
FedAvg_distributed(..., device,
                   client_model,
                   data_dict, ...,
                   client_trainer)
```

amount of the parameters of LMs, so it is promising to adapt recent prefix-tuning methods (Li and Liang, 2021) for personalizing the parameters of NLP models within the FedNLP framework.

F The System Design of FedNLP

The FedNLP platform consists of three layers: the application layer, the algorithm layer, and the infrastructure layer. At the application layer, FedNLP provides three modules: data management, model definition, and single-process trainer for all task formats; At the algorithm layer, FedNLP supports various FL algorithms; At the infrastructure layer, FedNLP aims at integrating single-process trainers with a distributed learning system for FL. Specifically, we make each layer and module perform its own duties and have a high degree of modularization.

F.1 Overall Workflow

The module calling logic flow of the whole framework is shown on the left of Figure 9. When we start the federated training, we first complete the launcher script, device allocation, data loading, model creation, and finally call the API of the federated learning algorithm. This process is expressed in Python-style code (see Alg. 2).

F.2 The Application Layer

Data Management. In data management, What `DataManager` does is to control the whole workflow from loading data to returning trainable features. To be specific, `DataManager` is set up for reading h5py data files and driving a `preprocessor` to convert raw data to features. There are four types of `DataManager` according to the task definition. Users can customize their own `DataManager` by inheriting one of the `DataManager` class, specifying data operation functions, and embedding a particular preprocessor. Note that the raw data's `H5Py` file and the non-IID partition file are preprocessed offline, while `DataManager` only loads them in runtime.

Model Definition. We support two types of models: Transformer and LSTM. For Transformer models, in order to dock with the existing NLP ecology, our framework is compatible with the *HuggingFace Transformers* library (Wolf et al., 2020), so that various types of Transformers can be directly reused without the need for re-implementation. Specifically, our code is compatible with the three main classes of `Tokenizer`, `Model`, and `Config` in *HuggingFace*. Users can also customize them based on `HuggingFace`'s code. Although LSTM has gradually deviated from the mainstream, we still support LSTM to reflect the framework's integrity, which may meet some particular use cases in federated setting.

NLP Trainer (single process perspective). As for the task-specific NLP `Trainer`, the most prominent feature is that it does not require users to have any background in distributed computing. Users of FedNLP only need to complete single-process code writing. A user should inherit the `Trainer` class in the application layer to implement the four methods as shown in the figure: 1. the `get_model_params()` interface allows the algorithm layer to obtain model parameters and transmit them to the server; 2. the `set_model_params()` interface obtains the updated model from the server's aggregation and then updates the model parameters of the local model; 3. the programming of the `train()` and `test()` function only needs to consider the data of a single user, meaning that the trainer is completely consistent with the centralized training.

F.3 The Algorithm Layer

In the design of the algorithm layer, we follow the principle of one-line API. The parameters of the API include model, data, and single-process trainer (as shown in Algorithm 2). The algorithms we support include:

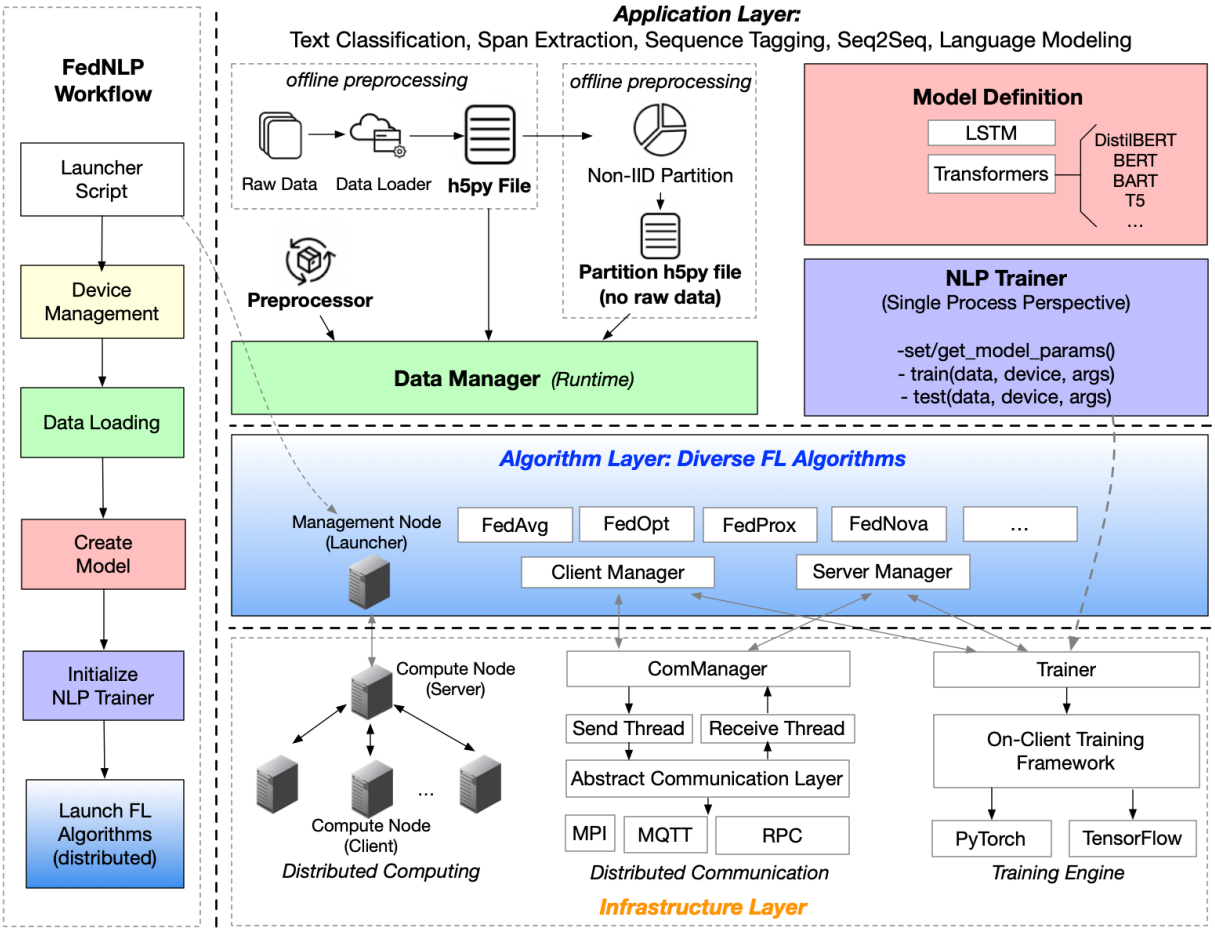

Figure 9: The overall workflow and system design of the proposed FedNLP platform.

Centralized Training. We concatenate all client datasets and use the global data $\mathcal{D}_G$ to train a global model — i.e., the conventional protocol for learning a NLP model on a dataset.

FedAvg (McMahan et al., 2017a) is the *de facto* method for federated learning, assuming both client and server use the *SGD* optimizer for updating model weights.

FedProx (Li et al., 2020c) can tackle statistical heterogeneity by restricting the local model updates to be closer to the initial (global) model with L2 regularization for better stability in training.

FedOPT (Reddi et al., 2020) is a generalized version of `FedAvg`. There are two gradient-based optimizers in the algorithm: `ClientOpt` and `ServerOpt` (please refer to the pseudo code in the original paper (Reddi et al., 2020)). While `ClientOpt` is used to update the local models, `SerevrOpt` treats the negative of aggregated local changes $-\Delta^{(t)}$ as a pseudo-gradient and applies it on the global model. In our FedNLP framework, by default, we set the `ClientOpt` to be AdamW (Loshchilov and Hutter, 2019) and the `SerevrOpt` to be SGD with momentum (0.9) and fix server learning rate as 1.0.

Each algorithm includes two core objects, `ServerManager` and `ClientManager`, which integrate the communication module `ComManager` from the infrastructure layer and the `Trainer` of the training engine to complete the distributed algorithm protocol and edge training. Note that users can customize the Trainer by passing a customized `Trainer` through the algorithm API.

F.4 The Infrastructure Layer

The infrastructure layer includes three modules:

1) Users can write distributed scripts to manage GPU resource allocation. In particular, FedNLP provides the GPU assignment API (`map_process_to_gpu()` in Algorithm 2) to assign specific GPUs to different FL Clients.

2) The algorithm layer can use a unified and abstract `ComManager` to complete a complex al-

gorithmic communication protocol. Currently, we support MPI (Message Passing Interface), RPC (Remote procedure call), and MQTT (Message Queuing Telemetry Transport) communication backend. MPI meets the distributed training needs in a single cluster; RPC meets the communication needs of cross-data centers (e.g., cross-silo federated learning); MQTT can meet the communication needs of smartphones or IoT devices.

3) The third part is the training engine, which reuses the existing deep learning training engines by presenting as the `Trainer` class. Our current version of this module is built on `PyTorch`, but it can easily support frameworks such as `TensorFlow`. In the future, we may consider supporting the lightweight edge training engine optimized by the compiler technology at this level.

F.5 Enhancing Security with Secure Aggregation (SA)

`FedNLP` supports state-of-the-art SA algorithms `LightSecAgg`, `SecAgg` (Bonawitz et al., 2017), and `SecAgg+` (Bell et al., 2020). Here, we provide a short performance comparison of these three algorithms. In general, `LightSecAgg` provides the same model privacy guarantees as SecAgg (Bonawitz et al., 2017) and SecAgg+ (Bell et al., 2020)) while substantially reducing the aggregation (hence run-time) complexity (Figure **??**). The main idea of `LightSecAgg` is that each user protects its local model using a locally generated random mask. This mask is then encoded and shared to other users, in such a way that the aggregate mask of any sufficiently large set of surviving users can be directly reconstructed at the server. Our main effort in `FedNLP` is integrating these algorithms, optimizing its system performance, and designing user-friendly APIs to make it compatible with NLP models and FL algorithms.