FedNLP: A Research Platform for Federated Learning in Natural Language Processing

Bill Yuchen Lin[*], **Chaoyang He**[*], **Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren**[*], **Salman Avestimehr**[*]
University of Southern California
{yuchen.lin, chaoyang.he, saltanol, xiangren, avestime}@usc.edu

Abstract

Increasing concerns and regulations about data privacy, necessitate the study of *privacy-preserving* methods for natural language processing (NLP) applications. Federated learning (FL) provides promising methods for a large number of clients (i.e., personal devices or organizations) to collaboratively learn a shared global model to benefit all clients, while allowing users to keep their data locally. To facilitate FL research in NLP, we present the **FedNLP**[1], a research platform for federated learning in NLP. FedNLP supports various popular task formulations in NLP such as text classification, sequence tagging, question answering, seq2seq generation, and language modeling. We also implement an interface between Transformer language models (e.g., BERT) and FL methods (e.g., FedAvg, FedOpt, etc.) for distributed training. The evaluation protocol of this interface supports a comprehensive collection of non-IID partitioning strategies. Our preliminary experiments with FedNLP reveal that there exists a large performance gap between learning on decentralized and centralized datasets — opening intriguing and exciting future research directions aimed at developing FL methods suited to NLP tasks.

1 Introduction

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

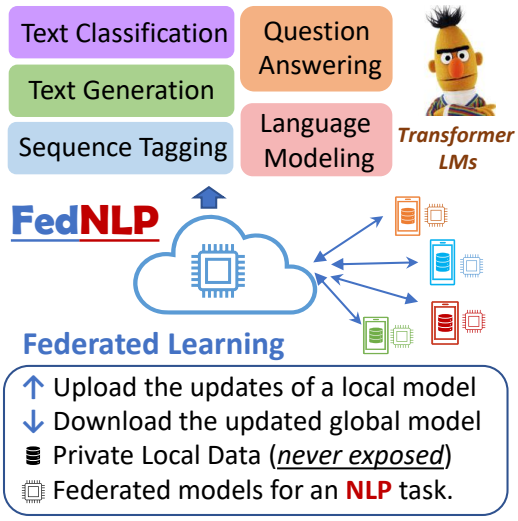

Figure 1: The proposed FedNLP platform.

To respect the privacy of the users and abide by these regulations, we have to 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 smart phones, virtual assistants (e.g., Amazon Alexa, Google Assistant, etc), or a personal computer. However, from a machine

[*] Bill Yuchen and Chaoyang contributed equally; Xiang and Salman are equal advisors.

[1] https://github.com/FedML-AI/FedNLP

learning persepective models trained with a centralized dataset that combines the data from all organizations or devices usually enjoy better performance in the NLP domain. Therefore, it 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* (Kairouz et al., 2019a; Li et al., 2020b) (FL) aims to enable many individual clients to jointly train their models, 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 to be 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 growing progress in the FL domain, research into and application of federated learning in NLP is has been rather limited. Indeed, the FL community now primarily focuses on developing new methods on synthetic datasets (e.g., federated Shakespeare (Caldas et al., 2018), or utilizing data from authentic users but with a simple task formulation and non-I.I.D. partition (e.g., next word prediction with *StackOverflow* (Reddi et al., 2020)). Complicating matters further, there is no dedicated platform covering comprehensive tasks in NLP and various non-I.I.D. data segmentation scenarios. Thus, benchmarking and analyzing existing FL methods for realistic NLP applications is still an open problem.

To effectively utilize FL in the NLP domain, herein, we present **FedNLP**, a research platform for studying federated learning in NLP. Our goal is to build a platform to support various types of research in federated learning for NLP, as shown in Figure 1. Specifically, we propose FedNLP, an open-source, research-oriented platform with:

- a federated data manager in various task formulations of NLP applications (e.g., classification, sequence tagging, question answering, text generation, language modeling);
- a unified, extensible interface between many *FL methods* and the *Transformer-based language models* (e.g., BERT);
- evaluation protocols with comprehensive partitioning strategies for simulating non-IID client distributions, in terms of their distribution shifts on labels, quantity and features.

These features when combined create a unique environment for research in both NLP and FL domains, paving the way for more private, personalized, robust NLP systems that can be deployed in realistic applications.

We conducted a series of experiments on the FedNLP platform to prove its functional integrity and availability. Specifically, using FedNLP, Transformer models can be trained efficiently and effectively on various NLP tasks in the distributed computing environment. Regarding data management and synthesis, we have conducted rigorous testing and data distribution analysis to ensure that our data processing is reasonable for future research exploration. Under the non-IID partition of different degrees of distribution shift, the experimental results found that the performance of the FL algorithm in these algorithms meets the theoretical expectations (i.e., the more significant the distribution shift, the lower the model accuracy is). Besides, our preliminary experiments have also pointed out directions that deserve to be explored in the future. For example, on a large distribution shift data set, all FL algorithms cannot obtain a reasonable accuracy and behave differently as results reported on the CV dataset, indicating the need to improve the algorithm or model based on the dataset and platform we provide.

The remainder of this paper is structured as follows. We introduce the background knowledge of federated learning and talk the motivation and challenges for the studying FL in NLP (§2). Then, we present the interested task formulations and selected datasets for realistic NLP applications in §3. We further show the details of our partitioning strategies (§4) for creating non-IID partitions for studying FL. In §5, we illustrate the details of each component of FedNLP. We present our experimental results, analysis, and findings in §6. Then, we discuss the related works (§7), future directions (§8) and finally conclude the work in §9.

Algorithm 1: FEDAVG algorithm for FL.

Notation: $N = \#$ clients, $R = \#$ rounds of communication, $E = \#$ epochs; $B =$ sample size; $\mathcal{D}_i$ and $\mathcal{M}_i$ as the local dataset and model of client C_i; M_i^t are models at t-th round; learning rate η.

In: $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$.

Out: M_G^R, the global model shared by all clients after R rounds of aggregation.

M_G^0.initialize() /* Initialize a global model. */

for $t = 0, 1, ..., R-1$ **do**

  /* Sample B clients from the total clients. */

  $\mathcal{S}_t \leftarrow \Gamma(N, B)$

  /* Get # examples in sampled clients. */

  $n \leftarrow \sum_{i \in \mathcal{S}_t} |\mathcal{D}_i|$

  foreach $c \in \mathcal{S}_t$ ***in parallel*** **do**

    $M_i^t \leftarrow M_G^t$ /* Sync the local model. */

    M_i^t.train($\mathcal{D}_\text{i}$, E, η) /* Locally train. */

    $\Delta_i \leftarrow M_i^t - M_G^t$

    /* Send the local updates Δ_i to server. */

  $M_G^{t+1} \leftarrow M_G^t + \eta \sum_{i \in \mathcal{S}_t} \frac{|\mathcal{D}^i|}{n} \Delta_i$

  /* Until all clients in $\mathcal{S}_t$ finished local training, update the current global model by averaging the updates of all local client models. */

2 Background and Motivation

In this section, we first introduce the concepts of federated learning (FL) with the de facto example – `FedAvg` (Section 2.1), and discuss the motivation (Section 2.2) and challenges (Section 2.3) for the FL+NLP research.

2.1 Federated Learning Concepts

Suppose we have N clients, where each client C_i has its own private dataset denoted as $\mathcal{D}_i = \{(x_i, y_i)\}$. We use $\mathcal{D}_G$ to denote the global dataset, which is the combination of all $\mathcal{D}_i$. In conventional supervised learning, we use $\mathcal{D}_G$ to learn a global model M_G shared by all clients in the end, which is only possible when client datasets are shareable and centralized. On the contrary, in federated learning, due to the fact that client datasets are private, we have to set up a learning schema such that clients communicate with each other via transmitting their model weights — each client maintains a local model M_i and committing local model updates to the server. We take the `FedAvg` (McMahan et al., 2017a), a de facto FL algorithm, as the example to illustrate the common pipeline.

As shown in Algorithm 1, in each round, the `FedAvg` method's pipeline is as follows: 1) sample a subset of all clients, 2) synchronize the local models with the current global model by communicating with the server, 3) locally train the local models on their local dataset, 4) send the weight updates of the local models back to the server, 5) the server updates the global model by averaging the weight updates of sampled local models. In such a straightforward yet effective way, we can learn a global model to benefit all clients while preserve their data privacy. Recent advanced FL methods, such as `FedOpt` (Reddi et al., 2020) and `FedProx` (Li et al., 2020c), follow a similar pipeline while improve the client-sampling method, model aggregation, etc.

Evaluation Protocol. In the standard FL setting, we usually use a global test set to evaluate the performance of the final global model M_G that is shared by all clients. We can also use the local datasets of clients to investigate the personalized performance on each client. As the non-IID partitions are the main challenge of FL, our evaluation protocol also support a wide range of possible partitioning strategies (Section 4). In addition, the proposed FedNLP platform can also show the learning curve to visualize the performance gain over the communication rounds easily for analyzing the efficiency of different FL methods.

2.2 Motivation Behind FL+NLP

Many realistic NLP services highly rely on users local data, e.g., text messages, documents and their tags, questions and selected answers, etc., which can locate at either personal devices or larger data-silos for organizations. These local data are usually considered as highly private and thus not directly accessible by anyone, according to many data privacy regulations, while 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 ques-

tions 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 concern 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 a lot of relevant research areas.

2.3 Challenges of Applying FL in NLP

Given the promising benefits of study 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, while lacking the ability to connect the study on 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 3.1). As the non-IID data partition over clients is the major feature of FL problems, it is also a challenge how to simulate the realistic non-IID partition for existing NLP datasets (Section 4). Finally, a platform also needs to 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 5).

3 Constructing Benchmark Datasets

In this section, we first introduce the basic task formulations in NLP (Section 3.1), and then present which *initial* selected datasets of each task formulation that we use for developing FedNLP (Section 3.2). Finally, we discuss our partitioning strategies to simulate realistic *non-IID* data segmentation over clients (Section 4) — one of the most important challenges in federated learning research.

3.1 Basic Formulations of NLP Tasks

There are various types of NLP applications while many of them share similar task formulation (i.e., input-and-put formats). We here 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 task. For example, we can use TC models for classifying the topic of a news article to be *political*, *sports*, *entertainment*, etc., or analyzing a 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 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 start 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.

Language Modeling (LM) The left-to-right language modeling task considers a sequence of words as the input $x = [w_1, w_2, \dots, w_n]$ and a token $y = w_{n+1}$ as the output. The output token

Task	Datasets	# Training Ex.	# Test Ex.	# Labels	# Clients	# Avg Tr.Ex.	# Clusters	Metrics
TC	AGnews	101,480	26,120	4	1,000	101.48	-	Acc/F1
TC	20News	15,037	3,805	20	100	150.37	-	Acc/F1
TC	SST2	6,969	1,772	2	30	232.30	-	Acc/F1
TC	Sentiment140	1,280k	320k	2	10,000	128.04	-	Acc/F1
	...							
ST	PLONER	14,000	3,501	7	50	280	7*	F1
ST	W-NUT	3,721	960	13	30	124.03	10	F1
ST	WikiNER	228,396	58,099	9	1,000	228.40	10	F1
	...							
QA	SQuAD	97,680	24,645	N/A	300	325.60	30	EM/F1
	...							
NLG	Movie-Dialogs	177,292	44,324	N/A	617	287.35	*	BLEU/ROUGE
NLG	CNN/DM	249,588	62,497	N/A	100	2495.88	50	ROUGE
	...							
LM	Shakespeare	3,803,542	422,616	N/A	1,129	3368.94	*	Acc
LM	Reddit	50,928,609	5,658,734	N/A	1,660,820	30.66	*	Acc
	...							

Table 1: An shortened list of datasets supported in the FedNLP platform. * stands that this dataset has a natural factor for creating client partitions. # clusters is the number of clusters we used for creating feature-based non-IID partitions. The full list of supported datasets can be found at our GitHub repository (linked in Abstract).

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 is also a great 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, while our extensible platform (detailed in Section 5) 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.

3.2 Selected Initial Datasets

Our standard for selecting the initial datasets for FedNLP is three-fold: 1) publicly available from the web, 2) having realistic applications for federated learning scenarios, 3) large enough for simulating different partitioning strategies.

For text classification, we choose three initial datasets: 1) 20News (Lang, 1995), 2) AGNews (Zhang et al., 2015), and 3) SST-2 (Socher et al., 2013). The first two datasets are for topic classification task, which is a very common NLP application on both personal devices and data-silos. 20News dataset is a collection contains 18,846 documents, partitioned evenly across 20 different categories. AGNews is extracted from web with 4 largest classes, where each class includes 30,000 training samples and 1900 testing samples. The SST-2 dataset is built for binary sentiment classification task in The *Stanford Sentiment Treebank*, which is obtained from movie reviews with fine-grained sentiment labels.

We mainly use PLONER (Fu et al., 2020) and CoNLL-2003 (Tjong Kim Sang, 2002) for the sequence tagging task. The PLONER dataset was constructed by collecting examples with person names, location, and organizations from multiple datasets (e.g., CoNLL2003, OntoNotes, and W-NUT). As a widely used question answering benchmark dataset, SQuAD (v1.1) (Rajpurkar et al., 2016) is a representative dataset for our reading comprehension formulation. The SQuAD dataset was created on Wikipedia articles with a collection of associated questions, where the answer to every question is a segment of text.

As for sequence-to-sequence generation tasks, we use Cornell Movie-Dialog Corpus (Danescu-Niculescu-Mizil and Lee, 2011a) here. This dataset contains a large collection of fictional conversations and its rich metadata including characters and movies can help us to apply it on federated learning reasonably. CNN/Daily Mail (Nal-

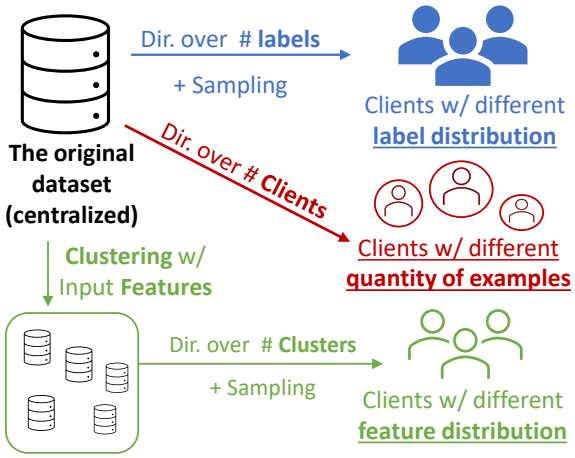

Figure 2: The three types of non-IID partitioning strategies, each of which split the original, centralized dataset to a collection of clients, with different focuses on modeling non-IIDness (e.g., label distribution, quantity of examples, input features).

lapati et al., 2016) is also an appropriate dataset that aims for doing text summarization. It collects summaries and corresponding passages from CNN and Daily Mail website.

In language modeling tasks, a large corpus would be preferable such as Shakespeare (Shakespeare; McMahan et al., 2017b), StackOverflow (Reddi et al., 2020) corpus and RealNews (Zellers et al., 2019). The *Shakespeare* corpus is a dataset constructed from *The Complete Works of William Shakespeare*, and each speaking role can be treated as different clients in federated learning. The *StackOverflow* corpus is a large dataset related to programming. The content in this dataset like posts and tags are suitable for federated learning settings. For the RealNews dataset, it crawls news over millions of web pages.

4 Non-IID Partitioning Strategies

To enable the study of federated learning in *non-IID* data partitions for the above NLP task formulations, we extend the common practice, which is widely used in prior works in generating synthetic FL benchmarks (Li et al., 2021). We first introduce how we control the *label distribution shift* for the classification task, 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., question answering), as shown in Figure 2.

4.1 Label Distribution Shift

Here we present how we synthesize the data partition such that clients share same (or similar) quantity of examples, but their *label distributions* are different from each other. In our *text classification* task, we assume on every client training examples are drawn independently with class 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 $\boldsymbol{q} \sim \mathrm{Dir}_L(\alpha \boldsymbol{p})$ 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 $\boldsymbol{q}_j$.

Note that this might cause a few clients may not 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.

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. We experiment with several values for α (i.e., 0.5, 1, 5, 10, 100) to generate populations that cover a spectrum of identicalness. As shown in Figure 3, we show a series heatmaps for visualizing the distribution differences between each client, where each entry at position (i, j) in the heatmap is the *Jensen–Shannon divergence* between i-th and j-th client (the darker the more different in terms of their label distribution). We can see that when α is higher, the overall label distribution shift is larger. Figure 4 shows an example of the concrete label distributions for all clients with different α.

4.2 Controlling the Quantity Shift

It is also common that different clients have very different data quantity while share similar label

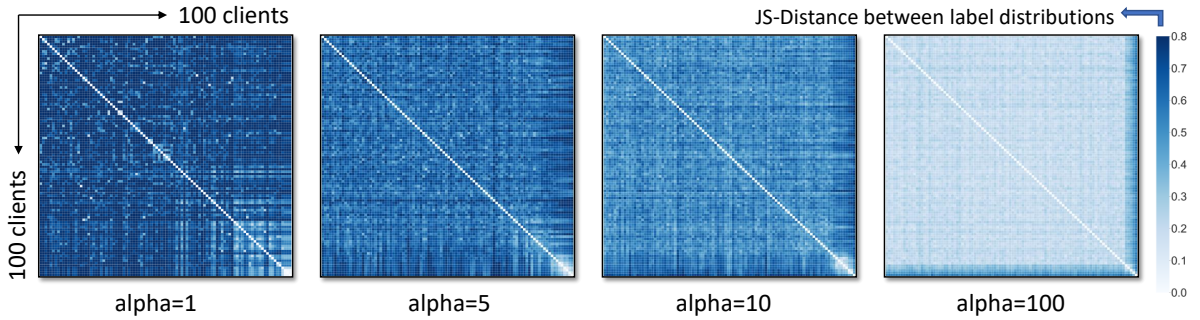

Figure 3: The *Jensen–Shannon divergence* matrix between 100 clients on the *20News* dataset when $\alpha \in \{1, 5, 10, 100\}$. Each sub-figure is a 100x100 symmetric matrix. The intensity of a cell (i, j)'s color here represents the distance between the label distribution of Client i and Client j. It is expected that when α is smaller, the data partition over clients is more non-IID in terms of their label distributions.

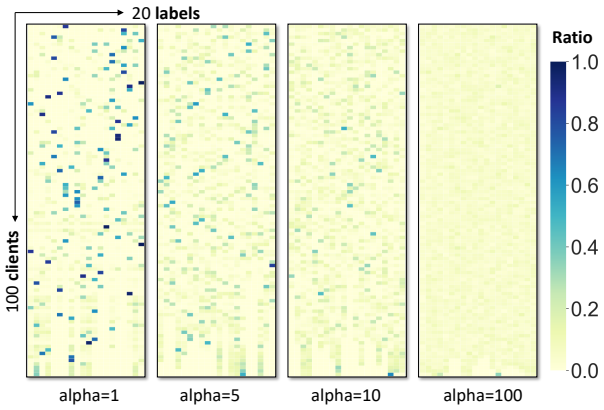

Figure 4: Visualizing the label distributions on *20News* with α being $\{1, 5, 10, 100\}$. Each sub-figure is a 100x20 matrix, where 100 is the number of clients, and 20 is the number of labels. The intensity of a cell here represents the ratio of a particular label in the local data of a client. When 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.

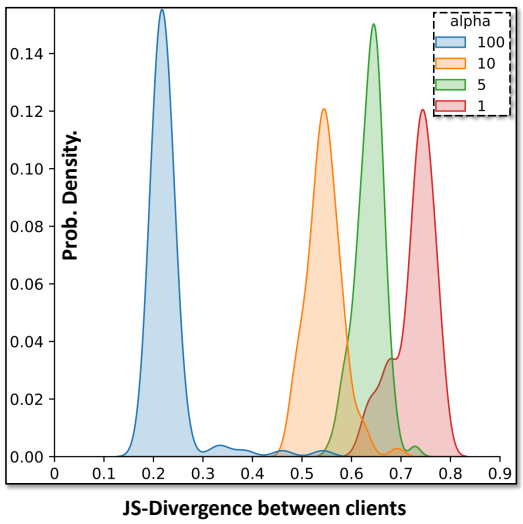

Figure 5: The probability density of the JS-divergence between 100 clients (reduced by summing on each client) on the *20News* dataset with different α — i.e., the matrices in Figure 6.

distribution. We thus also provide a quantity-level Dirichlet allocation $\boldsymbol{z} \sim \mathrm{Dir}_N(\beta)$ 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 share similar number of examples. A concrete example is shown in Figure 6.

4.3 Controlling the Input Feature Shift

Although straightforward and effective, the above label-based Dirichlet allocation method has a major limitation — they are only suitable for text classification tasks where each example's output can be modeled as a category-based random variable. Also, it can To create synthetic partitions for other non-classification NLP tasks and model distribution shift over input features, we thus propose a novel partition method based on clustering. Specifically, we use SentenceBERT (Reimers and Gurevych, 2019) to embed each example to a dense vector by their input text, then we apply K-Means clustering to get the cluster label of each example, and finally we use these cluster labels (as if they were classification tasks) for the steps in modeling *label distribution shift*.

There are two obvious benefits of this clustering-based Dirichlet partition method:

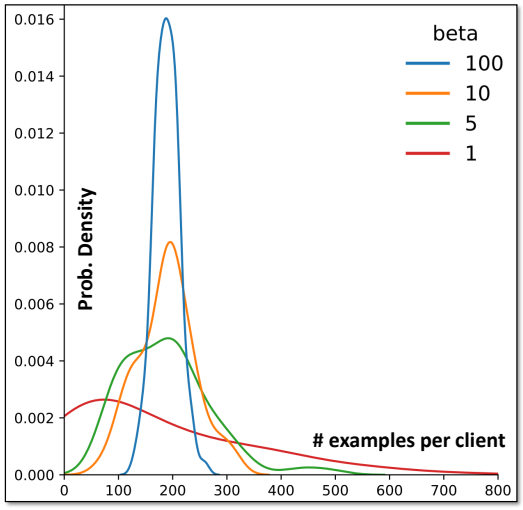

Figure 6: The probability density of quantity of training examples in each of the 100 clients on the *20News* dataset with different β. When β is larger, then all clients share more similar numbers of examples; when β is smaller, then the range of the quantity is much wider — i.e., the larger differences between clients in terms of their sizes of datasets.

1) It enables us to easily synthesise the FL datasets for non-classification tasks (i.e., ST, QA, NLG, LM) as they do not have a discrete label for every input; 2) The BERT-based clustering results naturally imply different sub-topics of a dataset, and thus input feature shift can be seen as shift of latent-labels — we can reuse the same method for label-based Dirichlet partition method.

4.4 Natural Factors

For a few datasets, we have natural factors that can be used for partition the data to multiple clients. For example, in the `Cornell Movie-Dialog` corpus (Danescu-Niculescu-Mizil and Lee, 2011b) we can use the movie_id to separate the dialogues into multiple clients — i.e., the dialogue history of one movie is only associated with one particular client so that the partition is closer to the realistic users' private data.

5 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.

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)
```

5.1 Overall Workflow

The module calling logic flow of the whole framework is shown on the left of Figure 7. 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).

5.2 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.

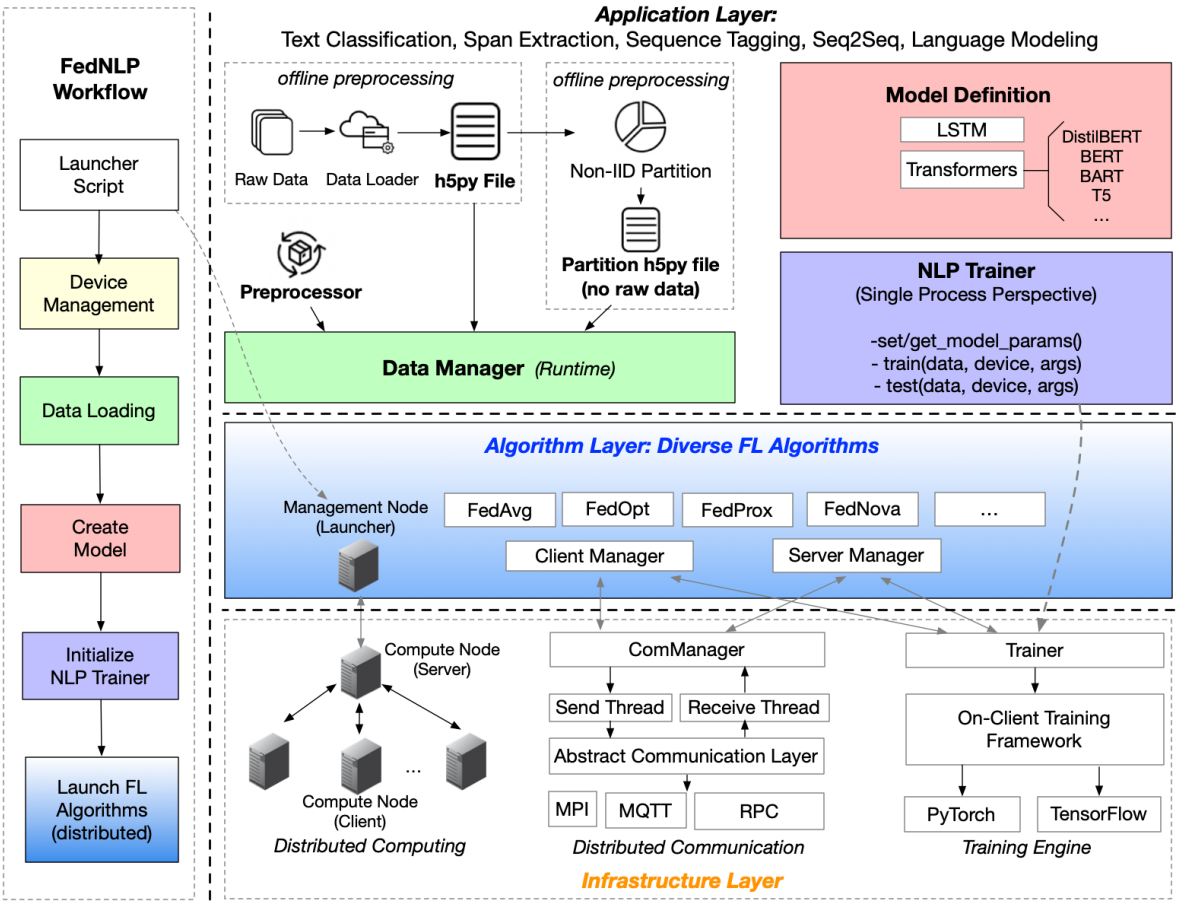

Figure 7: The overall workflow and system design of the proposed FedNLP platform.

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 of 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.

5.3 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:

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) has been illustrated in Section 2.1 and Algorithm 1, which 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 check 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.

5.4 The Infrastructure Layer

The infrastructure layer includes three modules:

1) Users can write distributed scripts to manage GPU resources 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 complex algorithmic 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.

6 Preliminary Experimental Results

We experimented with our FedNLP platform in a few common settings with an initial analysis. Note that the goal of these *preliminary* experiments here is to show the functionality of our FedNLP platform while leaving the development of state-of-the-art performance as to future work.

Set-up. As an initial step to developing a comprehensive platform for research in FL+NLP, we select a few NLP models and FL methods that are commonly used (e.g., DistilBERT/BERT for NLP models and `FedAvg`, `FedOPT`, `FedProx` for FL methods), while the FedNLP platform supports a much larger set of options. We choose to use DistilBERT (Sanh et al., 2019) for our preliminary experiments, as it is the lite version of BERT model and has a 7x speed improvement over BERT-base on mobile devices — a common scenario for FL. For `FedOPT` algorithm, we set the client optimizer as AdamW and the server optimizer as SGD. For `FedProx`, we tuned hyper-parameters learning rate in the range $\{0.1, 0.01, 0.001\}$, and μ in the range $\{1, 0.1, 0.01, 0.001\}$, respectively. We use uniform sampling to select 10 clients for each round when the client number in a dataset is very large. The local epoch number is set to 1.

To compare FL methods in different tasks, we select a representative dataset for each of the three popular task formulations: text classification (`20News`), sequence tagging (`PLONER`), and question answering (`SQuAD`). Please refer to our public repository for more results on other datasets and task formulations in the near future.

Note that the proposed FedNLP platform provides a much more diverse set of possibilities of the NLP models, FL methods, and options than we present here. Please find the full results in our repository on GitHub linked in Table 2.

Task Formulation	Data	Partition	FedAvg	FedProx	FedOpt	Centralized*
Text Classification	20news	Uniform($\alpha = \infty$)	61.0%	60.9%	69.1%	86.9%
Text Classification	20news	$\alpha = 1$ (label shift)	15.3%	15.2%	35.5%	86.9%
Text Classification	20news	$\alpha = 5$ (label shift)	51.4%	51.5%	53.5%	86.9%
Text Classification	20news	$\alpha = 10$ (label shift)	49.6%	49.6%	52.3%	86.9%
Text Classification	20news	$\beta = 5$ (quantity shift)	62.1%	61.9%	70.1%	86.9%
Sequence Tagging	PLONER	$\alpha = 0.1$ (feature shift)	84.5%	85.9%	91.4%	98.1%
Question Answering	SQuAD	$\alpha = 0.1$ (feature shift)	64.2%	64.4%	68.9%	82.1%
...	...	...	...	...	...	...

Table 2: The preliminary results on FedNLP. We cover three popular task formulations in NLP and three common federated learning methods, under different partition methods. Here we use *DistilBERT* as the base NLP model for all experiments. The full results on more tasks and options can be found at `https://github.com/FedML-AI/FedNLP/tree/master/experiments`.

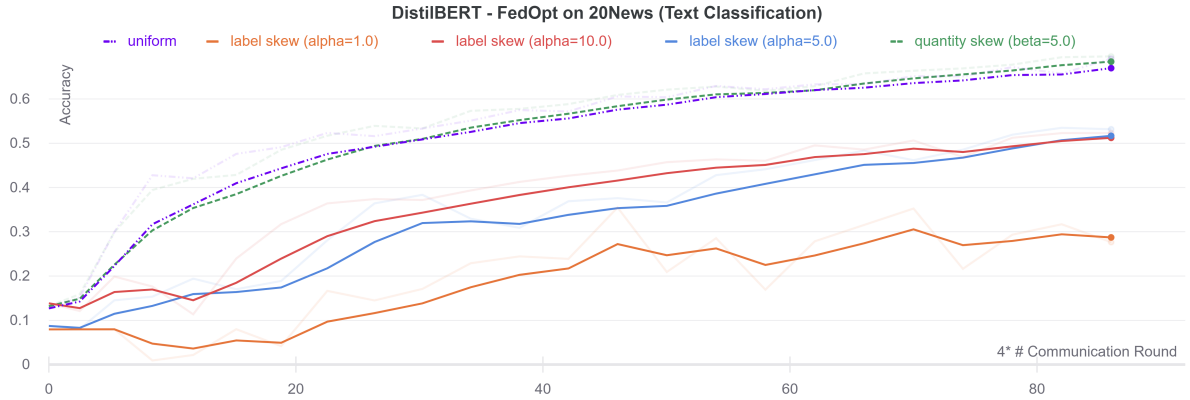

Figure 8: Testing `FedOPT` with `DistilBERT` for `20News` under different data partition strategies.

Preliminary experimental results are shown in Table 2. Our observations are as follows.

Accuracy in different degrees of data heterogeneity. The FedNLP platform supports users to investigate the performance of a federated learning algorithm with a wide range of data partition strategies, as discussed in Section 4. Here we look at the training curves of the `FedOPT` algorithm with different partition methods, as shown in Figure 8. The experiments reveal several interesting findings:

- When α is smaller (i.e., the partition is more non-IID in terms of their label distribution), the performance tends to be worse, looking at 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) in general.
- Quantity skew does not introduce a great challenge for federated learning when the label distribution is closer to the uniform one.

Comparing different FL methods. By observing the performance of the three algorithms on different data sets, we have the following findings:

- `FedOPT` performed best. This proves that using the client optimizer `AdamW` is an effective method, although this `FedOPT` variant has not yet rigorously proven in its convergence properties. In other words, naive SGD used in the client optimizer like `FedAvg` and `FedProx` does not work well in the training of Transformer models.
- When the α becomes smaller ($\alpha = 0.1$ in TC, meaning more non-IIDness), although `FedOPT` performs better, the accuracy of the three algorithms is very low. This indicates that we need to improve the FL algorithm further to adapt to a wide range of NLP tasks

with data heterogeneity.

- On the relatively simple datasets SQuAD and PLONER, FedProx performs slightly better than FedAvg, indicating that L2 regularization has a certain degree of benefit, but it is not obvious enough.

Different models and fine-tuning methods. In FL, understanding the trade-off between model performance and system efficiency is important. To study this, we consider two common settings of pre-trained model weights: *freezing* the backbone or *fine-tuning* the entire model. The advantage of the former is that it lowers the communication burden between clients and servers as it results in much smaller trainable model parameters, while the latter one usually enjoys the better performance. We will update the result of this exploration to our GitHub repository in the near future.

FedNLP is fully open and welcomes benchmarking experimental contributions from the machine learning and NLP community.

7 Related Work

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 care personalization and fairness in federated training.

For trustworthiness, security and privacy are the two main research directions, 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., 2019b) 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.

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 the 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 of these platforms only focus on designing a unified framework for federated learning methods, while do not provide a dedicated environment for studying NLP problems with FL methods. LEAF (Caldas et al., 2018) include a few text dataset, however, it is limited to classification and next word prediction datasets and do 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 starting 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., 2018), sentence-level text intent classification using TextCNN (Zhu et al., 2020), and pretraining and fine tuning of BERT using medical data from multiple silos without fetching all the 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 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 privacy of sensitive medical data and learn data in different platforms excluding the need of exchanging data between different platforms.

Our work in this paper 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 a general-purpose model. The above prior works would thus be a particular instance of the settings supported by the FedNLP platform.

8 Future Directions

We present several important open problems in the FL+NLP research, which people can use the proposed FedNLP for developing new solutions to.

Minimizing the performance gap. In centralized training, the current trend in the NLP community is to improve the model performance of downstream tasks via fine-tuning large pre-trained models, such as BERT and GPT-3. However, in the federated learning 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 on NLP tasks is the first priority.

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 heterogeneous model architecture for each client in the FL network.

Improving the system efficiency and scalability. Transformer models used for NLP normally are over-parameterized. Resource-constrained edge devices may not be able to run such large models. Designing efficient models or training framework is a practical problem worth solving. Also, the number of edge devices is million-level in a practical FL system. How to adopt a reasonable user selection mechanism to avoid stragglers and speed up the convergence of training algorithms is also an urgent problem to be solved.

Trustworthy, Robust and Privacy-preserving FedNLP. Existing works on trustworthy FL only evaluate their efficacy on small-scale datasets or non-NLP tasks. These methods may not be effective on various federated NLP tasks introduced in this paper. Thus, we design FedNLP with high flexibility. We advocate users of FedNLP to explore trustworthiness in various NLP tasks.

9 Conclusion

We present the FedNLP platform, an open-source framework aiming to facilitate FL research in NLP. The FedNLP platform supports various popular task formulations for realistic NLP applications, based on our flexible interface between Transformer-based models and a variety of FL methods. In addition, our non-IID partitioning strategies provide users with an extensive space for them to customize their own evaluation protocols. We believe the proposed FedNLP platform would benefit researchers in both NLP and FL areas, helping them advance many intriguing research questions, such as improving privacy-preserving methods for general NLP tasks, learning personalized language models, studying robustness and trustworthy issues, etc.

References

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 arXiv:2007.14390*.

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 arXiv:1812.01097*.

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.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011a. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87, Portland, Oregon, USA. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011b. Chameleons in imagined conversations: A

new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87, Portland, Oregon, USA. 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*.

A. Elkordy and A. Avestimehr. 2020. Secure aggregation with heterogeneous quantization in federated learning. *ArXiv*, abs/2009.14388.

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

Jinlan Fu, Pengfei Liu, and Qi Zhang. 2020. Rethinking generalization of neural models: A named entity recognition case study. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7732–7739. AAAI Press.

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*, abs/2003.09288.

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*, abs/1811.03604.

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 arXiv:2007.13518*.

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 arXiv:1910.04956*.

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)*, pages 1–8.

P. Kairouz, H. McMahan, B. Avent, Aurélien Bellet, Mehdi Bennis, A. Bhagoji, Keith Bonawitz, Z. Charles, Graham Cormode, R. Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, D. Evans, Josh Gardner, Zachary A. Garrett, A. Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Z. Harchaoui, Chaoyang He, Lie He, Z. Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, T. Javidi, Gauri Joshi, M. Khodak, Jakub Konecný, A. Korolova, F. Koushanfar, O. Koyejo, T. Lepoint, Yang Liu, Prateek Mittal, M. Mohri, R. Nock, Ayfer Özgür, R. Pagh, Mariana Raykova, Hang Qi, D. Ramage, R. Raskar, D. Song, Weikang Song, S. Stich, Ziteng Sun, A. T. Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, L. Xiong, Zheng Xu, Q. Yang, F. Yu, Han Yu, and Sen Zhao. 2019a. Advances and open problems in federated learning. *ArXiv*, abs/1912.04977.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019b. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

K. Lang. 1995. Newsweeder: Learning to filter netnews. In *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.

Q. Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2021. Federated learning on non-iid data silos: An experimental study. *ArXiv*, abs/2102.02079.

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*, 37:50–60.

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.

D. Liu and T. Miller. 2020. Federated pretraining and fine tuning of bert using clinical notes from multiple silos. *ArXiv*, abs/2002.08562.

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 arXiv:2012.06337*.

Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. 2019. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115.

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.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

NVIDIA. 2019. Nvidia clara.

Saurav Prakash and Amir Salman Avestimehr. 2020. Mitigating byzantine attacks in federated learning. *arXiv preprint arXiv:2010.07541*.

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*, 39(1):233–250.

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*, abs/1906.04329.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

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 arXiv:1811.04017*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

W. Shakespeare. Complete works of william shakespeare.

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*, 2(1):441–451.

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*, 2(1):479–489.

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.

Joel Stremmel and Arjun Singh. 2020. Pretraining federated text models for next word prediction. *ArXiv*, abs/2005.04828.

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.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

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 arXiv:2007.05084*.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020b. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

T. Yang, G. Andrew, Hubert Eichner, Haicheng Sun, W. Li, Nicholas Kong, D. Ramage, and F. Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *ArXiv*, abs/1812.02903.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems 32*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

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.