Practical Takes on Federated Learning with Pretrained Language Models

Ankur Agarwal Mehdi Rezagholizadeh Prasanna Parthasarathi
Huawei Noah's Ark Lab, Montréal
{ankur.agarwal1,mehdi.rezagholizadeh,prasanna.parthasarathi}@huawei.com

Abstract

Real-world applications of language models entail data privacy constraints when learning from diverse data domains. Federated learning with pretrained language models for language tasks has been gaining attention lately but there are definite confounders that warrants a careful study. Specifically, understanding the limits of federated NLP applications through varying the effects of different aspects (such as data heterogeneity, the trade-off between training time and performance, the effect of different data, and client distributions and sensitivity of the shared model to learning local distributions) is necessary to evaluate whether language models indeed learn to generalize by adapting to the different domains. Towards that, we elaborate different hypotheses over the components in federated NLP architectures and study them in detail with relevant experiments over three tasks: Stanford Sentiment Treebank-2, OntoNotes-5.0 and GigaWord. The experiments with different Transformer inductive biases on the variety of tasks provide a glimpse at the understanding of federated learning at NLP tasks. Specifically, the analysis suggests that regularization due to the ensembling effect may be masquerading as domain adaptation of federated learning in NLP with pre-trained language models.

1 Introduction

The success of large pretrained language models (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; Lewis et al., 2019) have demonstrated their applicability in consumer-based natural language processing (NLP) applications (Otter et al., 2020). While there are such massive datasets(Kiela et al., 2021; Wang et al., 2021), making models trained on these datasets to reflect the data diversity is an important challenge towards building equitable NLP systems. Hence, treating the distribution of data over the users as non-IID (McMahan and Ramage, 2017; Xu et al., 2018; Liu and Mazumder, 2021) to better emphasize the preferences of users as personalization gets naturally extended to the consumer NLP applications.

But, recent studies highlight that pretrained language models (PLMs) (Devlin et al., 2019) tend to get their predictions skewed by the frequency effects of tokens in the data distribution (Wei et al., 2021), this is concerning from a privacy and personalization standpoint. Han and Eisenstein (2019); Ramponi and Plank (2020); Carlini et al. (2022) show that neural language models (even the large pretrained architectures) have challenges in adapting to different data distributions on generative and classification tasks alike.

Federated learning (FL) (McMahan and Ramage, 2017; Konečnỳ et al., 2016) has been gaining popularity in machine learning as a practical way to mitigate domain adaptation with the promise of data privacy. FL as a learning paradigm focuses on learning a shared model through training data distributed over several clients. Such approaches have only recently begun to focus on NLP applications (Mammen, 2021; Lin et al., 2021). Lin et al. (2021) suggest that the success of federated algorithms can be improved through adapting over the client distribution that improves the generalization performance across the client distributions.

However, the opaqueness of the pretraining routine — primarily, quantifying what, and how much of that a language model has learnt from the pretraining corpora (Zhu et al., 2015; Devlin et al., 2019; Gao et al., 2020) cast a shadow on evaluating the effectiveness of these architectures in learning from diverse domains. Understanding the roles of different inductive biases not limited to the architecture, loss functions and data distribution becomes imperative to carefully look at claims of "domain adaptation" (Kouw and Loog, 2018). The representation of texts guided by syntax and semantic elements makes generalization to non-IID distributions in NLP more challenging if not the same when

Findings of the Association for Computational Linguistics: EACL 2023, pages 454–471
May 2-6, 2023

compared to domains such as computer vision (Liu et al., 2020; Luo et al., 2021; Zhuang et al., 2021; Yang et al., 2021). Then, it becomes imperative to understand the role of different constituents in the federated learning in NLP setup to take steps in the right direction. In that regard, we investigate four major hypotheses detailing the confounding variables in federated NLP systems: (1) role of pretrained weights in FL's domain adaptation, (2) distribution of clients, (3) Data homogeneity, and (4) robustness of personalizing to local distribution. Although the primary focus of the paper is to generate discussions along these questions, we support the discussions with relevant experiments on 3 different NLP tasks on 2 large Transformer architectures.

2 Background

Federated Learning Federated learning assumes the set up of K clients $\{C_k\}_{k=1}^{K}$ with different data distributions and a single server model S. Each client model, θ_{C_k}, is initialized by the server model θ_S a dedicated copy of the server model and then updated locally on the k^{th} client data distribution using an optimizer Opt_{C_k}. This distributed learning is repeated iteratively over R rounds. At each round r, the client models, $\theta_{C_k^r}$, are initialized with the aggregated weights of all the client models from the previous round $(r-1)$, referred to as the central server model, θ_S^r. The rounds end with accumulation and aggregation of gradients from all the client models to update the server model with optimizer Opt_S and continued until convergence on an unseen set (D_{test}). FedOpt, a popular federated learning algorithm is shown in Algorithm 1.

Algorithm 1 FedOpt Algorithm (Asad et al., 2020)

Input: $\theta_S^0, \text{Opt}_{C_k}, \text{Opt}_S$
for $r = 1$ **to** R **do**
  for $k = 1$ **to** K in parallel **do**
    $\theta_{C_k}^r \leftarrow \theta_S^{r-1}$
    for $e = 1$ **to** E **do**
      $g_{C_k^r} \leftarrow \nabla_\theta(\theta_{C_k^r} \mid D_k)$
      $\theta_{C_k^r} \leftarrow \text{Opt}_{C_k}(\theta_{C_k^r}, g_{C_k^r})$
    end for
    $\Delta_k^r \leftarrow \theta_{C_k^r} - \theta_S^{r-1}$
  end for
  $\Delta^r \leftarrow \frac{1}{K}\Sigma_{k=0}^{K}\Delta_k^r$
  $\theta_S^r \leftarrow \text{Opt}_S(\theta_S^{r-1}, \Delta^r)$
end for

The Performance Gap Like in (Lin et al., 2021), we compare the performance of federated server, θ_S, with $\theta_{central}$ over a common unseen set, D_{test} as in Equation 1,

$$\Delta\text{Perf} = \text{Perf}_S - \text{Perf}_{central} \tag{1}$$

where $\theta_{central}$ is trained over $\{D_k\}_{k=1}^{K}$ until convergence. Accuracy, F1, or Rouge score (Lin, 2004) can be used for measuring Perf.

Generalization Personalization trade-off Let the best server generalization performance is measured over D_{test} be $\mathcal{P}^*_{server}$. The generalization performance of the client model trained in r^* is measured on D_{test} be $\mathcal{P}^*_{client_k}$. The performance of the client model on D_k in round r^* measured be $\tilde{\mathcal{P}}_{client_k}$, which acts as a proxy to the personalization on D_k. Then, we measure the difference in the test loss for every client, k, between $\theta_{C_k^*}$ and θ_S^{*-1} as ΔP.

$$\Delta\mathcal{P}_k\left(\boldsymbol{x}_i\right) = \mathcal{P}^*_{client_k}\left(\boldsymbol{x}_i\right) - \mathcal{P}^{*-1}_{server}\left(\boldsymbol{x}_i\right) \tag{2}$$

For every $\boldsymbol{x}_i \in D_{test}$, the correlation between $\Delta\mathcal{P}_k(\boldsymbol{x}_i)$ to that of $\tilde{\mathcal{P}}_{client_k}$ measures the mutual cost of personalization to D_k on the generalization performance on D_{test}. Also, we measure the average the empirical risk of $\theta_{C_k^*}$ over D_k as $\tilde{\mathcal{P}}_{client_k}$:

$$\tilde{\mathcal{P}}_{client_k} = \frac{1}{\mid D_k \mid}\sum_{j=0}^{|D_k|}\mathcal{P}_{client_k}\left(\boldsymbol{x_j}\right) \tag{3}$$

We now define the trade-off metric as the slope between $\Delta\mathcal{P}_k(\boldsymbol{x}_i)$ and $\tilde{\mathcal{P}}_{client_k}$ over all $\boldsymbol{x} \sim D_{test}$ and K. $m_{\Delta P}$ measures the unit increase in generalization performance for adapting to D_k. Or $m_{\Delta P}$ estimates the cost of personalizing over the client distribution. Consequently, we make the interpretations for the metric $m_{\Delta P}$ — (a) positive slope ($\nearrow$) indicates that learning on local distribution aids in better generalization, (b) negative slope ($\searrow$) indicates that generalization inhibits the learning from local distributions, or (c) neutral ($\longrightarrow$) shows that the model is unaffected by learning.

3 Related Work

Multi-Domain Learning Realtime applications of most tasks have shown diverse distribution of datapoints requiring domain adaptation strategies (Daumé III, 2009; Dredze and Crammer, 2008). The effect of such domain shift in NLP has been

a topic of study for a while (Blitzer et al., 2006; Quiñonero-Candela et al., 2008; Blitzer, 2008; Ben-David et al., 2010; Cui and Bollegala, 2019). The general topic of domain adaptation in NLP shares similarity with the topics of continual learning (Sun et al., 2019), transfer learning (Devlin et al., 2019; Radford et al., 2019), multi-task learning (Collobert and Weston, 2008), and federated learning (Lin et al., 2021). Federated learning however, is different from the other paradigms since it emphasizes on the notion of preserving privacy of different local data distributions. To that, sophisticated approaches to aggregate the gradients to transfer learning from clients to the shared model (e.g. FedProx (Li et al., 2020), FedAvg (McMahan et al., 2017a), and FedOpt (Asad et al., 2020)) have showcased improvements in the generalization of the shared parameters. On the other hand, due to the many interactions of the clients with the server, communication overhead is an important aspect, and FedOpt (Asad et al., 2020) has been shown to better address it over existing federated algorithms.

Overview of Federated Learning Federated learning (McMahan and Ramage, 2017; Mammen, 2021) addresses the challenge of learning from private data spanning over multiple clients. Although the evaluation of such architectures prioritizes the generalization of the shared model, Mendieta et al. (2022) highlight that learning from the local distributions is critical towards that. The key to such efficient learning in federated architectures has been shaped by homogeneous and heterogeneous data or model (Li and Wang, 2019) distribution in clients (model architectures across clients have similar or different parameters). Further, the emphasis on privacy of client data has also been mitigated through the recent progress in knowledge distillation. However, systematic studies (Kairouz et al., 2021; Li et al., 2021) over federated architectures have identified potential biases due to unbalanced data of clients or diversity in the label distribution among others. Chen et al. (2018) propose a meta learning approach for federated learning that improves personlaizing to the non-IID client distributions. Also, constraints on data privacy makes it difficult to import approaches (Kirkpatrick et al., 2017; Rolnick et al., 2019) that avoid catastrophic forgetting of distributions in continual learning tasks.

Federated Learning for language tasks Distributed training on language tasks with federated learning has been gaining some attention. McMahan et al. (2017b) trained a differentially private language model over non-IID data distributions while Ge et al. (2020) trained a recurrent + convolutional architecture for medical named entity recognition task. Recently, Lin et al. (2021) proposed a framework that enables using modern pretrained language models on different language understanding tasks. Lin et al. (2021) discuss and hypothesizes a gap between the performance of Transformer architectures between the federated and centralized setting with data heterogeneity. Dupuy et al. (2022) analyze the effect of having clients with different amounts of data gathered from Alexa devices and suggest that non-uniform selection of devices improves the performance of the shared model.

In this work, we attempt to investigate different possible confounders for domain adaptation claims of federated systems in NLP and elaborately analyze them in the language tasks of classification, sequence tagging and sequence generation tasks that are popularly used with Transformer architectures.

4 Experiments

Models and Tasks We experiment with a focus on the *pretrain-finetune* setup that is popular with Transformer architectures on many language tasks. Of the many, we pick three tasks— Stanford Sentiment Treebank 2 (SST-2) (Socher et al., 2013), OntoNotes (v5.0) (Weischedel et al., 2013) and Gigaword (Graff et al., 2003) that fall into the broad categories of text classification, sequence tagging, text-generation respectively. The data splits are as used in (Lin et al., 2021), please refer §B for details. As for the models[1], we use BART-Base (Lewis et al., 2019) for text generation and DistilBERT (Sanh et al., 2019) for the other two tasks. In the experiments we use the models *with* (——) and *without* (- - - -) pretrained weights[2][3].

Centralized Training We use batch-wise gradient descent with AdamW (Loshchilov and Hutter, 2017) as the optimizer along with a linear learning rate scheduler. We use cross-entropy for model selection across the tasks. Also, in our analysis, we

[1]We use the transformer weights shared in huggingface: 'distilbert-base-uncased' and 'facebook/bart-base'.

[2]Across the experiments the line style and the colour is used to denote the corresponding model performances.

[3]The without pretrained weights setting trains the models from scratch.

Dataset	Model	Metric	Pretrained	Cent.	Fed.	Δ(*Perf*)	Δ (Rel.%)
SST2	DistilBERT	Accuracy	✓	$89.0_{\pm 0.8}$	$87.8_{\pm 0.4}$	1.3	–
			✗	$69.2_{\pm 3.2}$	$67.8_{\pm 1.0}$	1.4	7.7 ▲
OntoNotes	DistilBERT	F1	✓	$85.9_{\pm 0.1}$	$84.4_{\pm 0.1}$	1.5	–
			✗	$65.1_{\pm 0.3}$	$55.3_{\pm 0.3}$	9.8	550 ▲
Gigaword	BART	Rouge1	✓	$34.6_{\pm 0.7}$	$32.5_{\pm 0.2}$	2.1	–
			✗	$6.1_{\pm 0.5}$	$2.9_{\pm 0.6}$	3.2	50 ▲

Table 1: Comparison between the Δ(Perf) of federated and corresponding centralized set up when using (✓) and *not* using (✗) pretrained transformer weights. Across the 3 tasks it can be seen that the gap increases when not using pretrained weights (▲) suggesting that the pretrained weights of transformer are possibly doing the heavy lifting in domain adaptation of federated learning in language tasks.

use cross-entropy of samples in the test set to evaluate the relative performance of models compared. The complete results of the experiments are in §F.

Federated Training For the federated experiments, we partition the training dataset for the clients and train them using the FedOpt algorithm (Asad et al., 2020) to estimate the server parameter updates. Further, we use AdamW with a linear learning rate scheduler to estimate the gradients in our experiments. The round with the best server test loss is selected as the best round (For the complete results please refer to §F; for their run-time refer to §E).

Evaluation Metrics The metric of evaluation ($Perf$) is accuracy for sentence classification, F1-span for sequence tagging and ROUGE1 score for text summarization.

4.1 Motivation

Federated NLP considers two powerful learning paradigms— Federated algorithms aggregate the gradient updates over clients trained with non-identical data distributions while maintaining privacy, and PLMs trained over large corpora with a generic objective that gives a better downstream performance. Stickland and Murray (2019) and Peng et al. (2020) show that PLMs are successful in tasks that require domain adaptation. The motivation primarily relies on verifying if federated algorithms and PLMs share a synergy in the extreme domain adaptation scenario.

To that, we first study the role of pretrained weights as a confounding variable in the federated setup. The null hypothesis being the federated learning not affected much by the pretrained weights should be supported with $\Delta(\text{Perf})$ remaining similar in both cases. But, in Table 1 across different tasks we see that $\Delta(\text{Perf})$ increases when *not* using pretrained weights. This suggests that the pretrained weights may be supporting the performance of federated learning in NLP applications. The corollary to this observation could be that the learning in a federated setting may not be happening from adapting to the client distributions. This raises concerns on personalization that if at all the federated NLP setup with PLM learns anything from the client distribution.

4.2 Estimating the Confounding Variables

Pretraining and Federated Learning Towards understanding the essence of the pretrained weight's semantic prior as a confounding role in the success of FL for NLP, we continue to control for it in the remainder of the experiments. Disentangling such observations is necessary to objectively analyze the federated algorithm for language tasks.

Contribution by Client size One major challenge in the realistic setting of federated learning for data-driven tasks is the data imbalances that naturally occur among the clients (Lin et al., 2021; Dupuy et al., 2022). The distribution of number of data samples that each client has creates two distinct classes of clients— *major* and *minor* players— whose updates may affect the parameters of the shared server model differently. Generalization aside, ensuring personalization to the local distribution of data in the clients also becomes necessary in different scenarios arisen from the diverse data distributions. While extreme distributions may provide a regularization effect due to the ensemble learning (Balaji et al., 2018; Kumar et al., 2020; Stanton et al., 2021), the objective being able to better generalize through adapting to different local distributions require careful consideration of the distribution of clients. Towards understanding the limits of learning under the influence of clients

of different sizes, we evaluate the role of minor clients by ablating clients smaller than a *threshold (τ)* number of samples.

Client Personalization and Server Generalization Personalization emphasizes the shared model's capacity to be representative of all clients' distributions alike. But, training on the local distribution may affect the generalized representation of the shared model similar to the catastrophic forgetting in continual learning (Kirkpatrick et al., 2017) or mode collapse in generative modeling (Salimans et al., 2016). Using PLMs due to their robust semantic representations could alleviate some of these challenges. More specifically, we formulate our questions as ablation experiments: (a) How are personalization and generalization related across different tasks and client distributions (b) Does removing updates from minor players affect this relation? We study these questions in experiments with the help of $m_{\Delta\mathcal{P}}$ metric.

Client data partitioning distribution Generalization to unseen distribution, and how well the local client distributions are personalized over the federated learning rounds is also affected by the distribution of samples over the clients. Towards understanding the effect of the sample distribution on the personalization-generalization relation we compare the learning of the server model between non-uniform distribution that is closer to real world scenario, and a more controlled uniform distribution of samples among the clients to (a) evaluate the effect of generalization by the shared model and the personalization to local distribution by varying the number of samples per client uniformly over all the clients, and (b) we perform ablations on the updates by thresholding the clients on heterogeneous data distribution set up to understand the ideal scenarios in different language tasks.

4.3 Additional Setup

Dataset Partitioning Strategies For the study, we use data selection for the clients using two different strategies. Please refer to §C for the choice of hyper-parameters for the two methods.

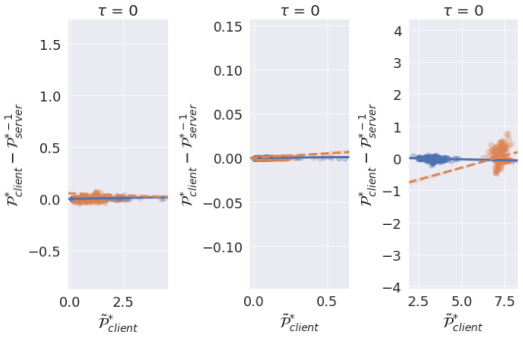
Random partitioning samples data over $\{\mathcal{C}\}_{k=1}^{K}$ by sampling from a Dirichlet distribution over the K clients with $\alpha = 0.1$ (Lin et al., 2021). For ablation on random distribution, we use hyperparameter τ that denotes the minimum number of samples in $\mathcal{C}_i$ for its parameters to be aggregated.

Uniform partitioning distributes the data uniformly over $\{\mathcal{C}\}_{i=1}^{K}$, which is controlled by a hyperparameter γ. Specifically, uniform distribution is constructed by sampling *at least* γ samples from each class for each client in the classification tasks. For the text summarization dataset, we cluster the SentenceBERT embedding (Reimers and Gurevych, 2019) and use KMeans++ [4](Arthur and Vassilvitskii, 2006) with 8 as the number of clusters, and sample uniformly over them.

Distributional Similarity We use MAUVE score (Pillutla et al., 2021) to measure the similarity between different text distributions in the experiments. The score uses GPT-2 (Radford et al., 2019) estimating the distributional similarity. Also, we use `mauve_scaling_parameter` set to 20. The score ranges between 0 and 1, where 1 indicates significant overlap.

5 Results

Following our motivation in §4.1 and questions raised for the confounders to the federated system for NLP tasks in §4.2, we structure the results to our investigation in this section.



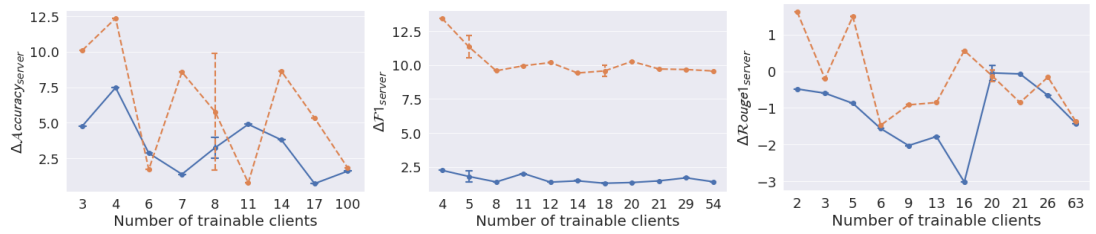
(a) SST2 (b) OntoNotes (c) Gigaword

Figure 1: We observe the effect of domain shift by measuring the δ change in the loss over every sample in the test set ($\mathcal{P}_{client} - \mathcal{P}^{*-1}_{server}$) and drawing a correlation with the average loss over the train distribution in the clients ($\tilde{\mathcal{P}}_{client}$).

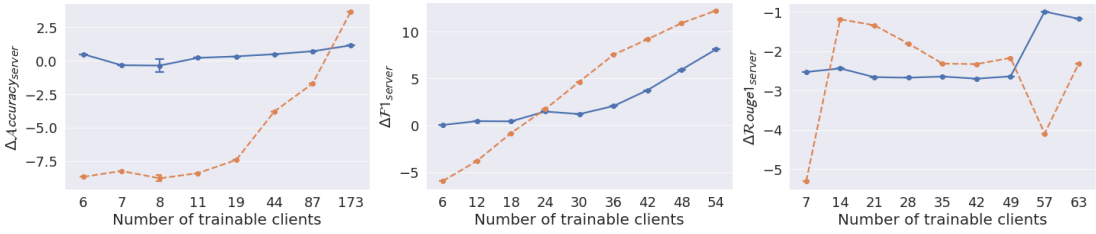
5.1 Effectiveness of Pretrained weights in adapting to client data distributions

An ideal model is expected to not discount the learning on the client distributions for better gener-

[4]We use the implementation in www.scikit-learn.org.



(a) SST2-Random (b) OntoNotes-Random (c) Gigaword-Random



(d) SST2-Uniform (e) OntoNotes-Uniform (f) Gigaword-Uniform

Figure 2: By varying the number of trainable clients in the random setting across the tasks, we measure the sensitivity of the shared model's performance (Cent. − Fed.: ▽ indicates federated model performing better) to data removal as high as $\sim 55\%$ of training data (SST-2: $\sim 55\%$, OntoNotes: $\sim 33\%$ and Gigaword: $\sim 20\%$ with the smallest number of clients).

alization performance. To verify whether such discounting happens, we analyze the personalization-generalization trade-off with the $m_{\Delta P}$ metric. In Figure 1, across the datasets we observe that the correlation stayed more neutral than positive, suggesting that the pretrained model may not be learning much from the local distributions. We observe a relatively positive $m_{\Delta \mathcal{P}}$ value when trained without the pretrained weights. This could be an anticipated behaviour, as the model relies on the information in the client distribution for generalization, unlike the pretrained weights that come with a semantic prior.

5.2 Client contributions to the metrics

We set up additional experiments to understand the contribution of clients in more detail. Particularly, we begin by studying our experiment of ablating updates from the clients that are below threshold (τ) on the *random* experiments. We hypothesize that as the threshold value is increased, the server model is restricted to learning from fewer client local distributions and the generalization performance should also decline as a result. On the contrary, we note in Figure 2 (top row) that the performance of the with pretrained model remained relatively unchanged across the datasets. However, the MAUVE scores estimated over the **A**blated, **U**nablated and **T**est (Unseen) distributions of data in Table 2 suggest that the test distribution are not close to the ablated or unablated clients' distributions[5]. The performance of the pretrained model in the federated setup, still being little affected only suggests that pretrained models may not be learning from the local distributions that hurt the claims of personalization to these distributions.

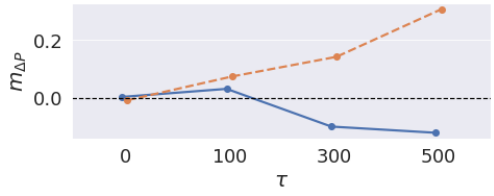
Dataset	*U —A*	*U —T*	*A —T*
SST2	$0.52_{\pm 0.06}$	$0.48_{\pm 0.07}$	$0.47_{\pm 0.07}$
OntoNotes	$0.64_{\pm 0.06}$	$0.53_{\pm 0.07}$	$0.51_{\pm 0.06}$
Gigaword	$0.58_{\pm 0.07}$	$0.45_{\pm 0.07}$	$0.45_{\pm 0.07}$

Table 2: The MAUVE score between the distribution of data in ablated (A), unablated (U) and unseen (T) splits with maximum value of τ in the random distribution setting. The values indicate the maximal distance between the distributions across the tasks.

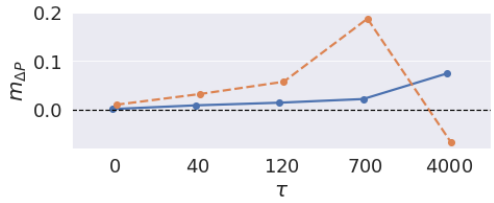
If not adapting to the local distributions, we further investigate whether the pretrained models use the updates from client distributions as regularization. To that, we repeat the same experiments on the same datasets with the more controlled uniform distribution setting.

Do client sizes affect the gap To have a clearer picture of the client size affecting the learning contribution, we use the *uniform* distribution with γ controlling the data partitioning size uniformly over all the clients. In Figure 2 (bottom row), we observe a trend showing that the models' performance

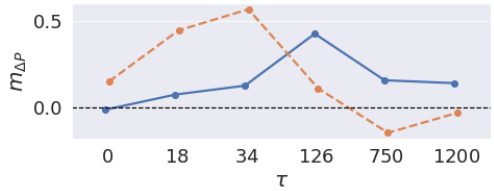
[5]Scores closer to 1 indicate significant overlap



(a) SST2



(b) OntoNotes



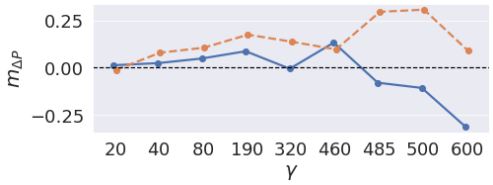
(c) Gigaword

Figure 3: Impact of personalization of clients on generalization of server —m_{Δ_P} values depict the impact in Random distribution strategy when ablating minor clients over different τ values.

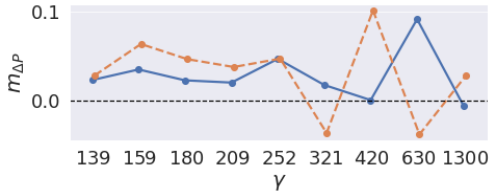
(pretrained or not) decreases drastically as the number of clients increases and the client partitioning sizes progressively become smaller. Similar observation across the different tasks suggests that the model requires a quorum of samples to minimize the gap across (Equation 1) the different tasks in a federated setup.

Do client sizes affect the trade-off Again here, we adjust for τ in *random* setting, where the clients with less than τ number of samples are restricted from updating the server parameters. We measure $m_{\Delta P}$ values for the varying τ values in Figure 3. We observe that with updates primarily from more minor clients (lower τ value), the generalization is less affected by personalization. But, the gap being lower as shown in Figure 2 suggests that the noisy updates with fewer clients be acting as a regularizer for the server updates. Further, as the value of τ increases (Figure 3), the trade-off remains healthier until a certain value of τ and then it drops. This trend could be attributed to the fact that with a higher τ value the number of clients updating the server becomes lesser with more data points, which provides better generalization but personalization due to variance in the client distribution gets challenging.

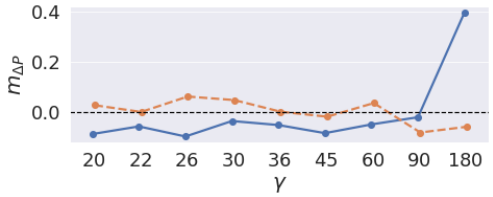
To understand the impact of varying client distributions on the trade-off, we perform the same analysis with *uniform* distribution shown in Figure 4. The pretrained model's generalization remains unaffected with more minor clients across SST-2 and GigaWord tasks supporting the alternate that updates from minor clients provide a regularization effect as we also see the gap to remain the same in Figure 2. On OntoNotes, while the m_{Δ_P} value stays the same the gap widens as observed in Figure 2. The varying results do not provide conclusive evidence on whether the pretrained models can learn to adapt to different domains in such extreme settings. Answering this is non-trivial which requires a careful consideration of the pretraining datasets and characterization of domains based on—task, topic, syntax, style etc.



(a) SST2



(b) OntoNotes



(c) Gigaword

Figure 4: Impact of personalization of clients on generalization of server—m_{Δ_P} values depicting the impact in Uniform distribution strategy with varying sizes of clients.

Time-Performance trade-off with varying client sizes The dropping of updates from the minor clients could also provide acceleration in the number of rounds, R, as federated learning has a communication overhead. We measure the performance of server with pretrained weights, Perf_S, over the different tasks and the number of rounds (R) taken

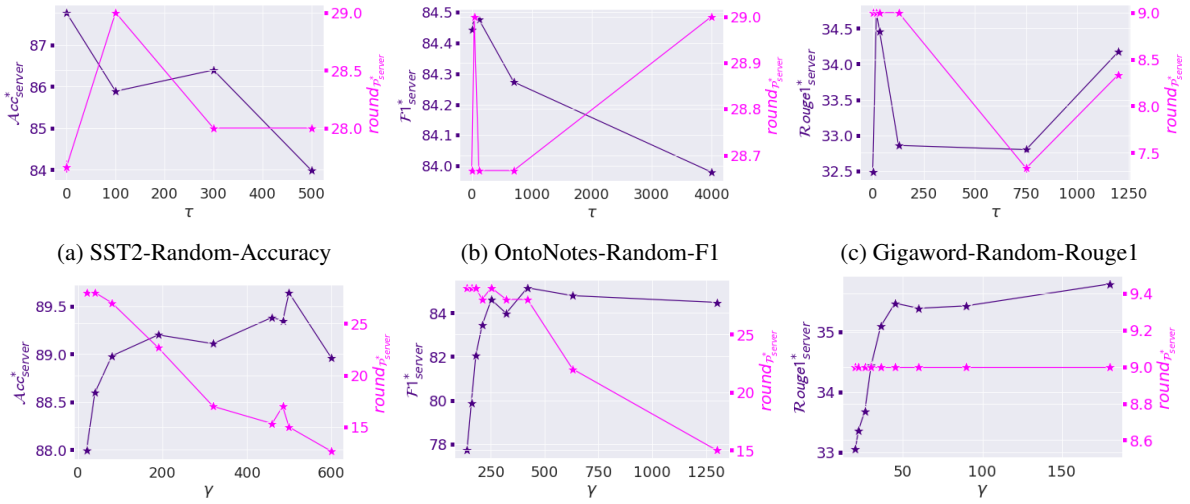


Figure 5: We compare R taken to converge when varying the number of clients in the uniform and in the random distribution settings with pretrained weights (✓). We also measure the corresponding Perf_S of the server model in the task.

by the set up to converge.

In Figure 5, we compare across the tasks with *random* distribution and the *uniform* distribution of samples with pretrained weights by varying the τ and γ respectively. In the random distribution experiments, as we are discarding parameter updates from clients we observe a *not-so-steep* drop in the Perf_S of the server model in SST-2 and OntoNotes. On the other hand, in Gigaword dataset experiments the impact of dropping the clients did not affect Perf_S. With respect to R, τ value being inversely proportional to the number of clients, we did not see a drastic acceleration to the number of rounds as lesser clients also increased the difficulty of the tasks.

In *uniform* setting, by varying the number of clients without the data loss, we make two observations: (1) The Perf_S is always better than when compared with *random* setting, (2) the performance saturates after a certain γ across the tasks, and (3) the number of rounds taken by the models to converge shows drastic decrease as the number of clients decreases. We hypothesize that with only major clients the gradient updates are stable to enable faster convergence. This contradicts with the observation in (Lin et al., 2021) that shows a wider gap in the performance when training pretrained transformer models in a federated set up, which we observe only when *not* using pretrained weights. Collectively, the results hint that the federated set up with PLMs suffer from personalizing to the client distributions, and the generalization on tasks may be a regularization of the distributed set-up.

6 Conclusion

This work explores pertinent questions that require a closer look at evaluating PLMs in the federated setting. Through empirical observations, we find that in federated learning, where the emphasis is more on personalization while ensuring privacy there could be a risk of pretrained models overlooking the client distributions. We also evaluated the effects of varying the client distributions which suggested that the gap between centralized and federated performance to be reduced when the samples are uniformly distributed over the clients. While that is ideal, the random distribution too does not suffer significant performance loss with pretrained weights. However, the critical aspect of the questions stems from the need to investigate the pretraining routine in identifying the *right* domain adaptation challenges for pretrained models. The gap being minimized while the personalization taking a toll calls for a deeper inspection to explore the limits of domain adaptation in PLMs with an appropriate evaluation framework (datasets, and metrics) that controls for the *leak* in the pretraining corpus.

Acknowledgements

We thank Guojun Zhang, and Xi Chen from the federated learning team at Huawei Noah's Ark Lab, Montréal for the many interesting discussions. We thank the anonymous reviewers for their insightful comments to our work. We also want to reproduce our results on Mindspore [6] in future, which is a new deep learning computing framework.

Limitations

The study, though, considers sample tasks from the different language tasks the downstream tasks generally are smaller in the size, and not much diversity with respect to the task complexity is considered. Though there is motivation for using FedOpt for training, the claims could have been further supported by exploring other possible federated algorithms. The scale of the experiments however do not play in favour of such an exhaustive study. Although the distributional similarity is measured with MAUVE, other aspects of texts n-gram, topic modelling could be explored to understand the domain shifts. Further, the study does not consider language models with different other inductive biases. The different transformer models and the effect of their respective pretraining datasets and task remain unexplored for future work. In addition to the above, the behaviour of different federated algorithms in the hypotheses we frame would also become interesting cases to scale our work.

Broader Impact

The trend of fine tuning transformer models for downstream tasks as both time and cost-effective solution for improving performance in downstream tasks has been gaining enough popularity. With federated algorithms giving access to learning from more public data while tackling the privacy concerns, it becomes worthwhile to use pretrained language models for language applications. Thus, understanding the adjustments to this federated language task learning with pretrained transformers on the claims of personalization-generalization trade-off becomes necessary. Knowledge and role of variables like client sizes and their distribution on the federated performance help identifying better decisions on setting up an appropriate domain for learning in downstream NLP tasks.

[6] https://www.mindspore.cn/

References

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2020. Fedopt: Towards communication efficiency and privacy preservation in federated learning. *Applied Sciences*, 10(8):2864.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1):151–175.

John Blitzer. 2008. *Domain adaptation of natural language processing systems*. Ph.D. thesis, University of Pennsylvania.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Xia Cui and Danushka Bollegala. 2019. Self-adaptation for unsupervised domain adaptation. *Proceedings-Natural Language Processing in a Deep Learning World*.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

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.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 689–697.

Christophe Dupuy, Tanya G Roosta, Leo Long, Clement Chung, Rahul Gupta, and Salman Avestimehr. 2022. Learnings from federated learning in the real world. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8767–8771. IEEE.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. Fedner: Privacy-preserving medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. *arXiv preprint arXiv:1904.02817*.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. 2021. Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Wouter M Kouw and Marco Loog. 2018. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*.

Ananya Kumar, Tengyu Ma, and Percy Liang. 2020. Understanding self-training for gradual domain adaptation. In *International Conference on Machine Learning*, pages 5468–5479. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2021. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*.

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

Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. 2021. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: learning during conversation. *Proceedings of AAAI-2021*.

Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. 2021. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984.

Priyanka Mary Mammen. 2021. Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*.

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

Brendan McMahan and Daniel Ramage. 2017. Federated learning: Collaborative machine learning without centralized training data. *Google Blog*.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017b. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*.

Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. 2022. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8397–8406.

Daniel W Otter, Julian R Medina, and Jugal K Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning*. Mit Press.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in nlp—a survey. *arXiv preprint arXiv:2006.00632*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems*, 29.

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

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.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. 2021. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. *arXiv preprint arXiv:2103.04044*.

Jason Wei, Dan Garrette, Tal Linzen, and Ellie Pavlick. 2021. Frequency effects on syntactic rule learning in transformers. *arXiv preprint arXiv:2109.07020*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. *OntoNotes Release 5.0*. Abacus Data Network.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Lifelong domain word embedding via meta-learning. *arXiv preprint arXiv:1805.09991*.

Dong Yang, Ziyue Xu, Wenqi Li, Andriy Myronenko, Holger R Roth, Stephanie Harmon, Sheng Xu, Baris Turkbey, Evrim Turkbey, Xiaosong Wang, et al. 2021. Federated semi-supervised learning for covid region segmentation in chest ct using multi-national data from china, italy, japan. *Medical image analysis*, 70:101992.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. 2021. Collaborative unsupervised visual representation learning from decentralized data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4912–4921.

A Reproducibility Checklist

A.1 For all reported experimental results:

1. *A clear description of the mathematical setting, algorithm, and/or model:* We define the details of our experimental setup in §4.

2. *Description of computing infrastructure used:* We use multiple servers equipped with 8 NVIDIA V100 (32 GB) GPUs and 72 cores CPU (754 GB) for running our experiments.

3. *The average runtime for each model or algorithm (e.g., training, inference, etc.), or estimated energy cost and number of parameters in each model:* The details of the runtime costs per experiment and the model have been reported in Table 8. The experiment runs have been tabulated in Table 9, Table 10 & Table 11.

4. *Corresponding validation performance for each reported test result:* Not applicable.

5. *Explanation of evaluation metrics used, with links to code:* This is specified with references in paragraph titled 'Evaluation' in §2.

A.2 For all experiments with hyperparameter search:

1. *The exact number of training and evaluation runs:* We run all centralized and random distribution on 3 seeds while the uniform distribution experiments are run on a single seed. The random distribution leads to different client size distributions while the uniform distribution has all clients of similar size.

2. *Bounds for each hyperparameter:* The tunable hyperparameters were batch size and learning rate in both centralized and federated training. For each dataset in both cases of fine-tuning and training from scratch, we find the best learning rate in the range $[0.01, 0.000001]$ for centralized and federated training by tuning on the exponent scale. For federated training we find the best hyperparameters in random distribution setting which we continue to use in other variants of our experiments. For the batch size, we explore in the set $8, 16, 32, 64$.

3. *Hyperparameter configurations for best-performing models:* The Table 4, Table 5, Table 6 & Table 7 records the best hyperparameters in use.

4. *Number of hyperparameter search trials:* The best hyperparameters are chosen over 3 seeds.

5. *The method of choosing hyperparameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy)* The best test loss resulting combination of hyperparameters is selected. The grid search method is used.

6. *Summary statistics of the results (e.g., mean, variance, error bars, etc.)* The tabulated results show mean and standard deviation results while the line plots are created using median as an estimator. The plots involving test or train loss account the evaluations done on a sample level. The plots using *Perf* evaluations use set of experiment level values.

A.3 For all datasets used:

1. *Relevant details such as languages, and number of examples and label distributions:* The datasets used are SST2, OntoNotes and Gigaword which are all in the English language.

2. *Details of train/validation/test splits:* This can be found tabulated in Table 3.

3. *Explanation of any data that were excluded, and all pre-processing steps:* For the task of text classification we use the complete sentences as samples instead the parsed phrases.

4. *A zip file containing data or link to a downloadable version of the data:* The references to the datasets are provided in §4.

5. *For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.* Not applicable.

B Dataset Splits

Dataset	Train set	Test set	Labels
SST2	$6,920$	$1,821$	2
OntoNotes	$59,924$	$8,262$	37
Gigaword	$10,000$	2000	$N.A.$

Table 3: Statistics for the 3 different dataset used.

C Client Data Distribution

During the gradient accumulation, we normally use uniform weightage. For sanity check if uniform weighting is the best choice, we made comparison of the random distribution with SST2 dataset using weighted aggregation where the client gradients are weighted to their size proportions. We did not see any advantage and hence continued using the uniform weightage. The comparison in performance can be seen in Table 9.

C.1 Random Distribution

Dataset	Clients	τ
SST2	3-100	{0, 100, 300, 500}
OntoNotes	4-54	{0, 40, 120, 700, 4000}
Gigaword	7-63	{0, 18, 34, 126, 750, 1200}

Table 4: Statistics for random distribution strategy experiments. The number of clients are effected by the ablation threshold for minor clients (τ).

C.2 Uniform Distribution

Dataset	Clients	γ
SST2	5-173	{600, 500, 485, 460, 320, 190, 80, 40, 20}
OntoNotes	6-54	{1300, 630, 420, 321, 252, 209, 180, 159, 139}
Gigaword	7-63	{180, 90, 60, 45, 36, 30, 26, 22, 20}

Table 5: Statistics for uniform distribution strategy experiments. The client count increases as the number of samples per label in a client (γ) decreases.

D Hyperparameters

The experiments on random distribution for all the datasets were carried out with 3 different seeds. However, for the uniform distribution we use only a single seed for OntoNotes and Gigaword datasets. Unlike the random distribution where the sampled client sizes keeps varying dramatically, the uniform distribution has all clients with almost the same number of data samples. Thus, we relax the need for repeating experiments with multiple seeds in the uniform distribution.

Dataset	Pretraining	Epochs	Batch size	L.R.
SST2	✓	10	8	1.00E-05
	✗	10	8	1.00E-05
OntoNotes	✓	5	8	2.00E-05
	✗	5	32	2.00E-04
Gigaword	✓	5	8	3.00E-05
	✗	5	8	3.00E-05

Table 6: The hyperparameters used for the centralized training experiments.

Dataset	Pretraining	Rounds	Batch size	L.R.
SST2	✓	30	64	1.00E-05
	✗	50	64	1.00E-05
SST2 (Weighted Aggregation)	✓	30	64	1.00E-04
	✗	50	64	1.00E-04
OntoNotes	✓	30	64	2.00E-05
	✗	50	64	2.00E-05
Gigaword	✓	10	8	3.00E-05
	✗	15	8	3.00E-05

Table 7: The hyperparameters used for the federated training experiments.

E Runtime of the experiments

Dataset	**Model (parameters)**	**Experiment**	**Pretrained**	**GPUs**	**Runtime (Hrs)**
SST2	distilbert-base-uncased (66.9M)	Centralized Training	✓	1	~0.5
			✗		~0.5
		Random Distribution	✓	8	5-6
			✗		8-10
		Uniform Distribution	✓	8	9-12
			✗		15-20
OntoNotes	distilbert-base-uncased (66.4M)	Centralized Training	✓	1	~1
			✗		~1
		Random Distribution	✓	8	4-5
			✗		6.5-7.5
		Uniform Distribution	✓	8	6-27.5
			✗		10-46
Gigaword	facebook/bart-base (139.4M)	Centralized Training	✓	1	~1
			✗		~1
		Random Distribution	✓	8	2-15
			✗		3-22.5
		Uniform Distribution	✓	8	3.5-17.5
			✗		5-26.5

Table 8: Time and resource costs per experiment run for the different datasets. The GPU refers to the NVIDIA V100 (32 GB) in a server having 8 of them.

F Master Results Tables

F.1 SST2

Centralized learning		
Pretraining	**Test Accuracy (%)**	**Epochs**
✓	89.02 ± 0.83	4.33 ± 2.87
✗	69.19 ± 3.2	5.33 ± 2.62

Random distribution (τ=0, 100 clients) (Weighted Aggregation)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	86.84 ± 0.1	30.0 ± 0.0
✗	52.99 ± 2.39	49.67 ± 0.47

Random distribution (τ=0, 100 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	87.77 ± 0.4	28.67 ± 1.89
✗	67.78 ± 1.04	50.0 ± 0.0

Random distribution (τ=100, 100 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	85.89 ± 1.08	30.0 ± 0.0
✗	64.34 ± 1.05	49.67 ± 0.47

Random distribution (τ=300, 100 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	86.44 ± 0.39	30.0 ± 0.0
✗	62.84 ± 0.5	50.0 ± 0.0

Random distribution (τ=500, 100 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	84.79 ± 1.42	30.0 ± 0.0
✗	61.14 ± 1.46	50.0 ± 0.0

Uniform distribution (γ=20, 173 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	88.05 ± 0.32	29.67 ± 0.47
✗	66.68 ± 1.17	50.0 ± 0.0

Uniform distribution (γ=40, 87 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	88.61 ± 0.23	30.0 ± 0.0
✗	72.29 ± 0.4	49.33 ± 0.47

Uniform distribution (γ=80, 44 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.07 ± 0.13	29.33 ± 0.94
✗	74.32 ± 0.64	48.67 ± 0.94

Uniform distribution (γ=190, 19 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.2 ± 0.2	23.67 ± 1.7
✗	78.0 ± 0.4	47.67 ± 1.7

Uniform distribution (γ=320, 11 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.11 ± 0.32	18.0 ± 1.41
✗	79.06 ± 0.42	41.0 ± 2.94

Uniform distribution (γ=460, 8 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.38 ± 0.09	16.33 ± 0.47
✗	79.7 ± 0.14	34.33 ± 1.7

Uniform distribution (γ=485, 8 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.35 ± 0.68	18.0 ± 2.94
✗	79.64 ± 1.05	32.33 ± 2.87

Uniform distribution (γ=500, 7 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	89.64 ± 0.13	16.0 ± 0.82
✗	79.75 ± 0.32	31.33 ± 1.25

Uniform distribution (γ=600, 6 clients)		
Pretraining	**Test Accuracy (%)**	**Rounds**
✓	88.96 ± 0.12	13.67 ± 0.94
✗	79.33 ± 0.39	28.33 ± 2.05

Table 9: Results of all experiments on SST2 dataset after model selection on the best server test loss.

F.2 OntoNotes

Centralized learning		
Pretraining	**Test F1 (%)**	**Epochs**
✓	85.93 ± 0.13	3.33 ± 1.7
✗	65.1 ± 0.29	1.0 ± 0.0

Random distribution (τ=0, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.44 ± 0.05	29.67 ± 0.47
✗	55.31 ± 0.27	49.67 ± 0.47

Random distribution (τ=40, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.49 ± 0.05	30.0 ± 0.0
✗	55.41 ± 0.26	50.0 ± 0.0

Random distribution (τ=120, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.48 ± 0.1	29.67 ± 0.47
✗	55.4 ± 0.16	50.0 ± 0.0

Random distribution (τ=700, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.27 ± 0.25	29.67 ± 0.47
✗	55.08 ± 0.18	49.67 ± 0.47

Random distribution (τ=4000, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	83.98 ± 0.32	30.0 ± 0.0
✗	53.07 ± 1.23	50.0 ± 0.0

Uniform distribution (γ=139, 54 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	77.7	30.0
✗	52.67	50.0

Uniform distribution (γ=159, 48 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	79.86	30.0
✗	54.0	50.0

Uniform distribution (γ=180, 42 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	82.05	30.0
✗	55.72	50.0

Uniform distribution (γ=209, 36 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	83.43	29.0
✗	57.38	50.0

Uniform distribution (γ=252, 30 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.61	30.0
✗	60.26	50.0

Uniform distribution (γ=321, 24 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	83.99	29.0
✗	63.2	50.0

Uniform distribution (γ=420, 18 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	85.16	29.0
✗	65.22	48.0

Uniform distribution (γ=630, 12 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.79	23.0
✗	67.61	45.0

Uniform distribution (γ=1300, 6 clients)		
Pretraining	**Test F1 (%)**	**Rounds**
✓	84.49	16.0
✗	66.14	28.0

Table 10: Results of all experiments on OntoNotes dataset after model selection on the best server test loss.

F.3 Gigaword

Centralized learning				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Epochs**
✓	34.57 ± 0.66	15.92 ± 0.36	32.35 ± 0.59	3.0 ± 1.41
✗	6.07 ± 0.51	0.36 ± 0.08	5.86 ± 0.5	3.0 ± 1.63

Random distribution (τ=0, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	32.48 ± 0.22	14.28 ± 0.02	30.61 ± 0.19	10.0 ± 0.0
✗	2.89 ± 0.56	0.05 ± 0.02	2.89 ± 0.54	15.0 ± 0.0

Random distribution (τ=18, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	34.72 ± 0.25	15.57 ± 0.08	32.25 ± 0.16	10.0 ± 0.0
✗	1.87 ± 0.3	0.01 ± 0.01	1.83 ± 0.33	14.33 ± 0.94

Random distribution (τ=34, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	34.46 ± 0.54	15.48 ± 0.22	32.07 ± 0.36	10.0 ± 0.0
✗	2.0 ± 0.3	0.03 ± 0.01	1.96 ± 0.32	14.33 ± 0.94

Random distribution (τ=126, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	32.86 ± 0.97	14.45 ± 0.61	30.88 ± 0.82	10.0 ± 0.0
✗	1.54 ± 0.24	0.02 ± 0.01	1.51 ± 0.26	14.33 ± 0.47

Random distribution (τ=750, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	32.8 ± 1.73	14.54 ± 0.91	30.9 ± 1.39	8.33 ± 1.7
✗	3.29 ± 1.74	0.01 ± 0.01	3.22 ± 1.68	13.67 ± 1.89

Random distribution (τ=1200, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	34.17 ± 0.38	15.41 ± 0.23	32.09 ± 0.26	9.33 ± 0.47
✗	4.71 ± 0.98	0.0 ± 0.0	4.67 ± 0.97	11.33 ± 1.25

Uniform distribution (γ=20, 63 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	33.05	14.56	31.04	10.0
✗	2.41	0.0	2.4	15.0

Uniform distribution (γ=22, 57 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	33.36	14.89	31.35	10.0
✗	1.58	0.04	1.56	15.0

Uniform distribution (γ=26, 49 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	33.68	15.03	31.66	10.0
✗	2.07	0.0	2.09	15.0

Uniform distribution (γ=30, 42 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	34.44	15.43	32.32	10.0
✗	2.06	0.0	2.06	15.0

Uniform distribution (γ=36, 35 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	35.09	15.88	32.68	10.0
✗	2.18	0.0	2.17	15.0

Uniform distribution (γ=45, 28 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	35.48	16.19	33.13	10.0
✗	5.59	0.11	5.38	15.0

Uniform distribution (γ=60, 21 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	35.39	16.38	33.11	10.0
✗	6.44	0.17	6.23	15.0

Uniform distribution (γ=90, 14 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	35.43	16.32	33.15	10.0
✗	2.73	0.1	2.69	15.0

Uniform distribution (γ=180, 7 clients)				
Pretraining	**Test Rouge1 (%)**	**Test Rouge2 (%)**	**Test RougeL (%)**	**Rounds**
✓	35.8	15.99	33.2	10.0
✗	10.68	1.65	10.2	15.0

Table 11: Results of all experiments on Gigaword dataset after model selection on the best server test loss.