Guidelines for Federated Aggregation Strategies and Architectures for Next Word Prediction

Yana Sakhnovych
SBA Research, Vienna, Austria

Rudolf Mayer
SBA Research, Vienna, Austria
rmayer@sba-research.org

***Abstract*—Federated learning is an important technique for training language models, which are frequently used for next word prediction, since federated learning allows utilising large quantities of real-life data without compromising the privacy of the data owners. Training a model that generalises well in this setting is a challenging task due to the inherent statistical heterogeneity of the training data, and due to the hardware limitations of private mobile devices. There are different approaches that address this issues, e.g. through model selection, different aggregation and learning strategies, update compression. In this paper, two popular model architectures, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are evaluated in centralised and federated settings. For federated learning, the vanilla Federated Averaging algorithm and two alternatives that try to address statistical heterogeneity, namely FedProx, which uses a proximal term to restrict the divergence from the global model during local model training, and Federated Attention, which has similar aims of reducing the distance between models as well to ensure faster convergence and improve generalisation, but is performing this during the aggregation station, are evaluated for their achieved perplexity and accuracy in various settings on two datasets. Based on these results, we provide guidelines on which methods to use, depending on the scenario.**

***Index Terms*—Federated Learning Federated Averaging Language Models Next Word Prediction**

I. Introduction

Next word prediction is a common feature to assist users in several mobile applications that require text input, like messaging apps, search engines, and many more. To train accurate language models for next word predictions, large volumes of training data are required. While several large benchmark datasets are available, language usage is highly dependent on the context and the domain. For this reason, it is important to train language models on data that is representative of the future application of a model, and user data is a prime fit. Such data may be difficult to gather due to privacy concerns. One solution is Federated Learning, where data that is collected distributed on end-user devices can stay on-device. In Federated Learning, a machine learning model is trained directly on-device on the local dataset, and only model updates need to be exchanged with a central coordinator. Removing the need centralise the data for training, this approach reduces the privacy risks related to the transmission, storage and analysis of confidential data.

This makes federated learning an attractive strategy for language modelling tasks like mobile keyboard prediction and text correction for mobile devices. However, there are still a few limitations in federated learning on mobile devices [1] [2] [3]:

- **Model size**: hardware limitations on the mobile devices mandate a trade-off between the model performance and size. Selecting smaller, but still well-performing models and applying techniques to reduce the number of parameters, without compromising performance too much, is necessary.
- **Communication Cost**: the transmission of updates may constitutes a bottleneck in the training process due to bandwidth constraints on mobile devices. Thus, reducing the number of communication rounds until convergence and the size of the model updates is important.
- **Distribution of Data**: since a client's dataset is heavily influenced by the behaviour of the user (e.g. by a specific writing style), the local datasets might be very different from the whole population. Additionally, large differences in sizes of the local datasets can occur since some users might use a service more frequently than others. Thus, algorithms need to be robust to heterogeneity of the datasets.

To address these challenges, several different architectures for learning language models in a federated manner have been investigated, e.g. recurrent neural network architectures such as the Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU). Further, different methods to perform the aggregation of the locally trained models to a global model have been proposed. While the Federated Averaging (FedAvg) [4] algorithm, which performs position-wise averaging of all local model parameters, remains a popular approach, other methods have been proposed to specifically address e.g. data heterogeneity, such as FedProx [5] or FedAtt [3].

In this paper, we perform an empirical analysis to determine (i) which aggregation methods and models are most suitable for federated next word prediction, (ii) what impact statistical heterogeneity has on federated next word prediction, and (iii) to what degree the negative impact can be mitigated with current state-of-the-art methods. To validate the performance of our federated learning models, we compare them also to a hypothetical centralized approach, where all data is aggregated before learning one single model.

The remainder of this paper is structured as follows. In Section II, we present related work on language modelling and federated learning. Section IV presents our evaluation.

Finally, we conclude and provide an outlook on future work in Section V.

II. Related Work

Language modeling is a major part of Natural Language Processing (NLP), and is used for a wide array of different downstream tasks such as speech recognition, machine translation and question answering. Language modeling deals with predicting a linguistic unit (such as characters, words, or sentences) given a sequence of previous units [6]. This can be achieved trough statistical methods by calculating the respective probability distribution of a sequence of linguistic units, or through Machine Learning based approaches. Language modeling can thus be used to predict the next word given an input sequence in order to make suggestions while typing text or to enable other text generation tasks [7].

Since there is a fixed number of valid labels, language modeling can be considered a classification task: each word that can be predicted is a distinct class. While earlier approaches would use e.g. probabilities on n-gram models, or shallow machine learning models, the past decade has seen rapid improvements mainly through the success of different neural-network-based approaches, which are now State-of-the-Art in the field. As language is a sequence of tokens, Recurrent Neural Networks (RNNs), where the current state depends on the outputs at earlier time steps, and due to the ability to process sequences of variable length, are specifically suited. Mikolov et al. [8] presented one of the first uses of RNNs for language modeling.

Major advancements to deal with issues in training were made when the Long Short Term Memory (LSTM) architecture [9] was used for language modeling [10]. An LSTM reduces the vanishing gradient problem by introducing input and forget-gates that control which parts of a sequence are taken into account and which are "forgotten". These gates control how the hidden state, which is passed to the cell in the next time step, is updated.

Another architecture that has become popular for language modeling is the Gated Recurrent Unit (GRU), which achieves similar performance to a LSTM, but requires fewer parameters to be learned [11]. GRU cells consist of only two gates, a reset gate and an update gate. LSTM and GRU based models are commonly used for federated language modeling due to their relatively small size [2], [3], [7].

Current state-of-the-art methods mostly use transformer-based [12] architectures, but since these models have millions of parameters, they require large amounts of training data and extensive training. Transformer-based models are less suited to federated learning on private mobile devices in particular, as the amount of data on a single device is limited, the computational power of a single device is low and the number of rounds required to train such a model would be very high. GPT-3 [13], for example, has 75 billion parameters, which would make it very difficult to allocate enough memory or computing power on mobile devices that participate in federated learning [1]. For these reasons, RNN-based language models are still frequently used in settings where these prerequisites are not fulfilled, such as mobile or edge devices.

Federated Learning encompasses methods to collaboratively learn a common model (global model) from distributed data sources, without the need to centralise this data in one location. The term was coined by [4]. One fundamental aspect in federated learning is how the central server (or aggregator) will aggregate these locally trained models. Most work focuses on neural networks and similar models (e.g. logistic regression, or also support vector machine), where positional parameters are aggregated. In many settings, there are multiple federated (communication) rounds (also called iterations), i.e. several iterations of locally training models, aggregating them, and continuing local training.

The authors of [4] adapted the synchronous distributed version of the stochastic gradient descent (SGD) optimization algorithm to use as a baseline for federated learning algorithms. In FedSGD, each client computes one step of gradient descent and sends the gradient to the server. The server then calculates a weighted average of the received gradients and adjusts the model parameters. [4] also proposed federated averaging (FedAvg) as an alternative. In FedAvg, the clients perform multiple steps of gradient descent locally, before transferring the model weights to the coordinator, who subsequently computes the average. The main advantage of this method is the greatly reduces the number of communication rounds compared to FedSGD, as multiple local epochs of training can be performed before aggregation.

While FedAvg has become the most common aggregation method in the federated setting, several modifications and alternatives have been developed to improve performance, communication cost, privacy, or robustness to non-independent and identically distributed data. Two major challenges in federated learning are system heterogeneity – meaning hardware and network differences between clients – and statistical heterogeneity, i.e. non-idd distribution of data. FedProx [5] addresses both of these issues through two modifications of FedAvg. To tackle system heterogeneity, FedProx allows the usage of partial computations, i.e. a device with better hardware may train its local model for multiple epochs, while slower devices can only train for one epoch. Still, both results are used for the model update. This leads to a reduction of devices that have to be completely dropped from computations, because they take too long for a full training round. The aggregation itself is the same in FedProx as in FedAvg.

Differences in the clients' datasets may lead to issues in model convergence, as local model updates may not generalize well and thus decrease the performance of the global model. In FedProx, a proximal term is used to restrict the divergence from the global model. The proximal term is added to the training loss on the client device, and the new loss is used for computing gradients.

The federated attention (FedAtt) algorithm [3] focuses on reducing the distance between models, and to ensure faster

convergence and improve generalization. The difference to FedProx is that in FedAtt, this happens at the aggregation step and not during local training. Instead of computing the average of the weights from the local models, [3] proposes a layer-wise attention-based aggregation to train a global model that better generalizes the client models. Attentive weights for the aggregation are computed for each client and layer, based on the distance of the weights of the layer at the client to layer at the global model. Thus, client weights that are closer to the global model are favored during aggregation. FedAtt is reported to outperform FedAvg in terms of perplexity and communication cost. The experiments conducted by the authors included both a comparison of perplexity after 50 communication rounds for different numbers of local epochs and a comparison of how many rounds it took to reach a perplexity threshold [3].

In this paper, we will compare FedAvg, FedProx and FedAtt will be compared in terms of efficiency and effectiveness, to derive guidelines which methods to use in which setting.

III. Experiment Setup

In the following, we describe the setup of our experimental evaluation, including the datasets we used, the vocabulary creating approach, and our evaluation metrics.

A. Datasets

We selected two datasets, name the Stackoverflow and a Reddit dataset, since they are sufficiently large, are frequently used in literature for language modelling, and we can thus compare our effectiveness to literature, and as these datasets can be divided among the federated learning clients in a meaningful manner (so that there is high heterogeneity between clients but lower heterogeneity when it comes to text within a client dataset).

1) Stackoverflow: Stackoverflow is a website where users can ask programming-related questions and receive answers by other programmers; this data have frequently been used for federated learning research [14], [15]. We utilise the pre-processed version available through TensorFlow Federated[1]. Further, ¡BOS¿ and ¡EOS¿ tokens were added to indicate the beginning and end of a comment. The comment is then split into 21-token-long sequences. Each sequence is transformed into input and target – where the first 20 tokens are the input and the target consists of 20 tokens starting with the second token, so that the resulting splits are overlapping.

The train set consists of 342,477 unique users with 102,387,303 sequences in total. However, of this set, only a maximum of 5,120 tokens per user are utilized; limiting the size of the local datasets was also done in [2], [15] to reduce the influence of very large local datasets on the training time. The validation set includes 38,758 unique users with 12,449,166 sequences and the test set consists of data from 204,088 unique users with 124,099,09 sequences. Due to the large size of the test set, only a fraction of the whole set was used for measuring the performance during training and the full dataset was used to get the final accuracy and perplexity at the end. A histogram of the numbers of tokens per local dataset is shown in Figure 1.

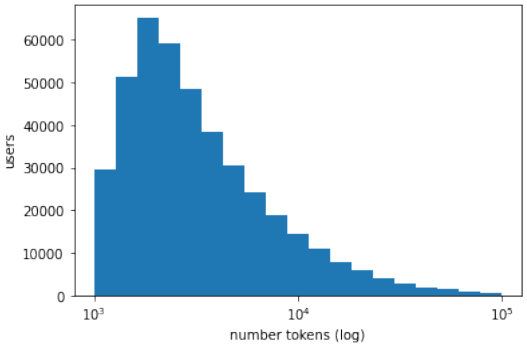


Fig. 1: Stackoverflow dataset: Token-per-user-histogram

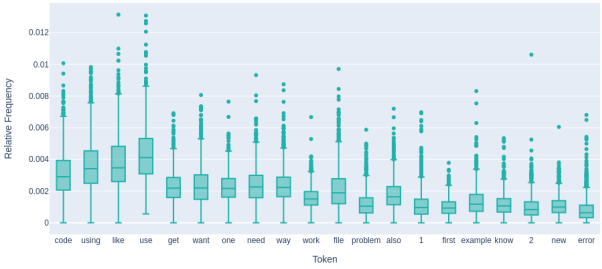


Fig. 2: Stackoverflow Dataset: Word Usage

Due to differences in language patterns of the users and topics that the posts related to, there is statistical heterogeneity in the dataset. The word usage of 1,000 randomly sampled users with their respective posts and comments shows this heterogeneity – each word in the vocabulary was used at least once by 143.62 out of the 1,000 users on average. Figure 2 depicts boxplots of the relative word frequency of 20 non-punctuation and non-stop-word tokens (words with little informational value like ”the”, ”is”, and ”that”) that were included by the highest numbers of users in this random sample.

2) Reddit Comments: Reddit is an internet-platform for sharing and discussing user-generated content. The website is organized in “subreddits” which are forums that are created and moderated by users and are dedicated to different topics. This makes Reddit a good source for a large amount of heterogeneous textual data, and it has thus been used for many language modeling tasks in literature in federated settings [2], [3], [16]–[18]; however, all these use different Reddit datasets, rendering the results not directly comparable. We use the dataset provided by the federated learning benchmark LEAF [19][2]. It provides a preprocessed version, as well as scripts

[1]https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow

[2]https://github.com/TalwalkarLab/leaf/tree/master/data/reddit

TABLE I: Dataset Overview

Dataset	Stackoverflow	Reddit
Users Train / Valid / Test	342,477 / 38,758 / 20,408	500,000 / 10,000 / 10,000
Avg. Local Train Set Size	3,040 Token	1,750 Token
Vocab. Size	1,004 Token	1,000 Token
Vocab. Selection	used by most users	most frequently used
Avg. # of Users of Vocab. Tokens	44.46 out of 1,000	143.62 out of 1,000

that were used for preprocessing and for vocabulary creation. A vocabulary that consists of the 10,000 most frequently used words. For our experiments, we slightly adjusted the preprocessing from LEAF to make the results more comparable to literature: the sequence length was increased to 20 tokens, since this setup is more common in other works [2], [3] and fully employs the advantage of RNNs in regard to processing longer sequences. 500,000 users were randomly selected for the train set. The test and validation set were sampled from the remaining users; both sets contain comments from 10,000 users each. Again, 1,000 users were sampled to analyze their word usage. On average, each word in the vocabulary was included in 44.46 out of 1,000 user datasets, meaning that the overlap in the word usage is much smaller on average than with the Stackoverflow dataset. As individual subreddits are dedicated to different topics, there is a high variety in topics that are covered by posts and comments. In addition, Reddit is not intended to be used in a "professional" setting, which may lead to more usage of slang, and language variations. These differences between the websites could be the reasons for the higher heterogeneity in the Reddit dataset.

Table I shows the characteristics of the two datasets. There are three main differences between the two datasets. Firstly, Stackoverflow is focused on a single subject, meaning that the language used is more homogeneous than Reddit. Secondly, Stackoverflow has more text available per user. This suggests that language modeling tasks may be more challenging for the Reddit dataset. Thus, a higher perplexity for a language model trained and evaluated on the Reddit dataset is expected (under the same experiment setup).

B. Vocabulary Creation

For each training corpus, there are some words which rarely occur or not at all present in the training data. It is not possible for a model to correctly predict that word in these cases. Usually, the vocabulary which a model learns is restricted by selecting a subset of all valid tokens. Most frequently this is done by counting the occurrences of each distinct token in the training corpus and selecting some fixed number of most frequent tokens. All other tokens get an encoding that indicates that they are unknown – per convention that is usually UNK for "unknown" or OOV for "out of vocabulary". Any prediction that is made where the target is UNK is usually either counted as false or is ignored.

In a federated setting where it is possible to freely access all client datasets, one might select the words for the vocabulary based on the number of clients that had that word in their local dataset instead of the overall frequency. We consider two cases. Our first vocabulary creation strategy is based on the collection frequency, describing the number of occurrences of a word in a collection of documents, while the second one is based on document frequency, which corresponds to the number of documents in which a term appears [20]. Here, a user dataset counts as a document.

C. Evaluation Metrics

There are multiple ways to measure the quality of next-word predictions. In this paper, we use two metrics that are frequently used in related work, namely perplexity and accuracy.

1) Perplexity: Perplexity is the most frequently used metric for language modeling tasks. It can be thought of as the weighted average branching factor of a language. The branching factor is the number of possibilities to continue a sequence of tokens [20]. Perplexity is the exponential of the cross-entropy (Equation (1)) [3].

$$PP(W) = 2^{H(W)} = 2^{(-1/N)\,log_2(P(w_1,...,w_n))} \tag{1}$$

This is equivalent to the normalized inverse of the probability of the test set (Equation (2)) [20].

$$PP(W) = \sqrt{P(w_1, ..., w_n)} \tag{2}$$

Intuitively, this means the lower the perplexity, the less "surprised" the model is by the sequences in the test set. A model that predicts an incorrect class with high confidence has a higher perplexity than a model that makes the same prediction, but with a lower confidence. The perplexity metric has a range of drawbacks. Firstly, it is highly dependent on the vocabulary of a language model. This means that it cannot be used to compare models with different vocabulary sizes or different tokens in the vocabulary. Secondly, while there is a correlation between the correctness of predictions and perplexity, a reduction of perplexity does not necessarily mean that the model makes fewer prediction mistakes [21].

2) Accuracy: Accuracy is the ratio of correct predictions to predictions overall. It is used as a metric for all kinds of classification tasks. As opposed to perplexity, it can be used to compare models that were trained using different vocabularies (where one includes different tokens than the other) on the same dataset (as long as all predictions of tokens outside the vocabulary are counted as false predictions). For some tasks, top-k-accuracy is used, where a prediction counts as correct if the true class is among the k most likely classes according to the model. However, as opposed to perplexity, accuracy does not measure with what confidence the model made a prediction, so it does not capture how close the model was to the correct label on a wrong prediction.

IV. Evaluation

First, we report the performance of the baseline models in centralised learning, followed by an analysis of the federated learning results. Effectiveness is evaluated on the validation and test data every 10 rounds; hyper-parameter tuning for

federated learning was performed for 500 to 2,000 training rounds, depending on the parameter, and the final federated experiments were performed for 2,500 rounds each. In each round of the base experiments, one epoch of training was performed per client.

A. Centralised and Local Baselines

The final test accuracy and perplexity of these baseline models are given in Table II. An accuracy of 20.97% on the Reddit dataset could be reached with LSTM; lacking a centralised comparison, this is 2% higher than [2] reached on Reddit comments in a federated setting. The Stackoverflow dataset achieves an accuracy of 27.52% with LSTM, which is 3.5% higher than the federated results in [14].

TABLE II: Centralized Baseline Performance

Dataset	Model	Accuracy	Perplexity
Reddit	GRU	20.64%	26.48
	LSTM	20.97%	25.87
Stackoverflow	GRU	27.34%	3.53
	LSTM	27.52%	3.51

Taking a closer look at the convergence of the GRU-models, we note that most of the improvements occur within the first 20 epochs. At the 20th epoch, the accuracy of the GRU model on Reddit dataset reaches 20.38% and improves only by 0.25% until the 80th epoch. On the Stackoverflow dataset, the model reached an accuracy of 26.74% by epoch 20 and improved until it reached 27.18% at the 55th epoch, where the validation loss stopped improving.

The performance of the local baseline models is shown in Table III. All four models have a poor performance – the LSTM-models reach an average accuracy of only 3.81% on the Reddit dataset and 5.02% on the Stackoverflow dataset, with a standard deviation of 1.38 and 1.53 respectively. The local model that reach the highest accuracy on the Reddit dataset was an LSTM-model and achieved only 5.8%. On the Stackoverflow dataset, a LSTM model reached 7.6% accuracy. This means that models that were only trained locally cannot reach sufficiently good accuracy to make reliable next word predictions. As expected, due to the differences in datasets described in Section III-A, language modeling on the Reddit dataset appears to be a more challenging task than on the data from Stackoverflow, which is supported by the significantly lower perplexity achieved for both types of baselines. For example, the perplexity reached with a LSTM-model on the Reddit dataset is 25.87, whereas the perplexity on the Stackoverflow dataset is only 3.51.

The differences in performance between the models with a GRU-layer and a LSTM-layer are minor. In the centralized setting, the LSTM-models reached a slightly lower perplexity – a difference of 0.61 for Reddit and 0.02 for Stackoverflow. For the locally trained baselines, LSTM reached a lower perplexity on the Reddit dataset and GRU on the Stackoverflow dataset. However, these differences are minor.

TABLE III: Local Baseline Performance

Dataset	Model	Mean Accuracy	Mean Perplexity
Reddit	GRU	3.51% (± 1.39)	249.63 (± 117.82)
	LSTM	3.81% (± 1.38)	260.1 (± 121.6)
Stackoverflow	GRU	4.60% (± 1.58)	16.30 (± 4.58)
	LSTM	5.02% (± 1.53)	16.44 (± 4.10)

B. Federated Learning

Table IV shows the perplexity and accuracy of the models after 2,500 rounds of federated learning, results are obtained by evaluating on the respective test set. Figures 3a to 3d show the accuracy throughout the training. The evaluation was performed every 10 rounds, and the resulting plots were smoothed using moving average with a window of size of 5.

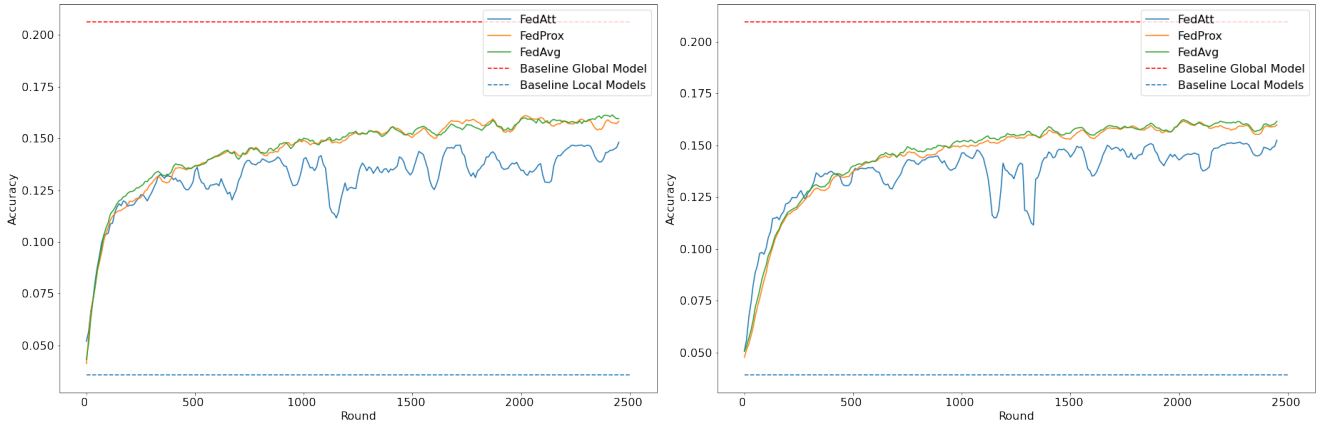
TABLE IV: Federated Learning Results after 2,500 rounds

Dataset	Model	Aggregation	Accuracy	Perplexity
Reddit	GRU	FedAvg	16.03%	38.51
		FedProx	16.15%	38.37
		FedAtt	15.05%	42.24
	LSTM	FedAvg	16.40%	38.60
		FedProx	16.26%	38.99
		FedAtt	15.69%	41.58
Stackoverflow	GRU	FedAvg	24.96%	5.33
		FedProx	24.99%	5.34
		FedAtt	24.42%	5.41
	LSTM	FedAvg	24.89%	5.39
		FedProx	24.96%	5.42
		FedAtt	24.30%	5.56

Federated Averaging achieved very stable results throughout the training – there were no significant drops in performance from round to round. The most accuracy improvements were achieved in the first 250 to 500 rounds of training. The models were able to reach an accuracy of 16.40% for the Reddit dataset, and 24.96% for the Stackoverflow dataset using FedAvg, meaning that the federated models are weaker than the global centralized baselines by 4.57 percentage points and 2.56 percentage points respectively. The centralized baselines reached a higher accuracy than the federated models after only one epoch on the Reddit dataset and eight epochs on the Stackoverflow dataset with the GRU model. However, the federated models clearly outperform all the local baselines.

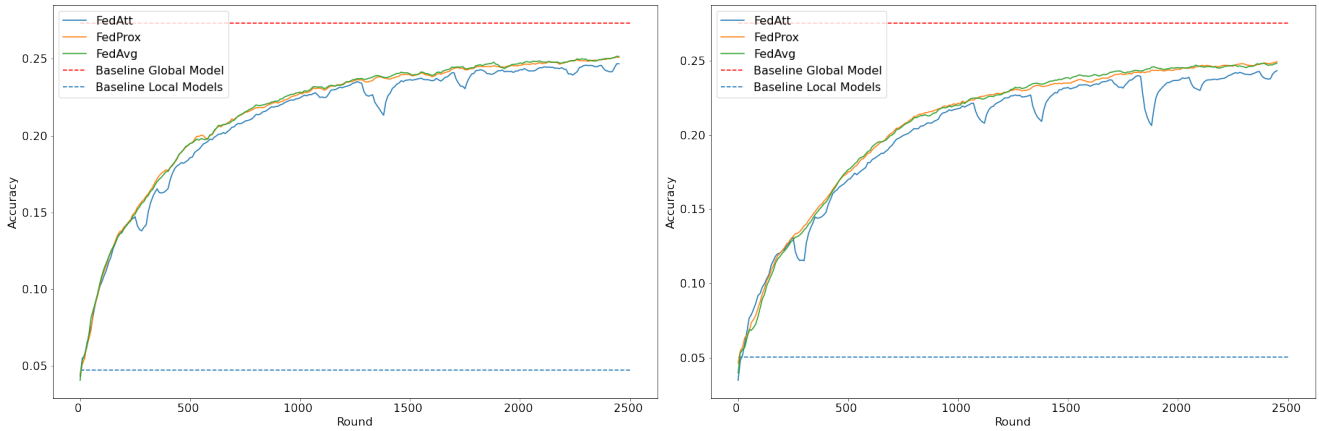
The choice of the RNN-Layer had little influence on the training behavior. The LSTM-models were slightly worse than the GRU-models in terms of perplexity: the difference was 0.09 for the Reddit dataset and 0.06 for the Stackoverflow dataset. However, as these differences are rather small, there is as well no clear trend in terms of accuracy: For the Reddit dataset the LSTM-model reached a higher accuracy with a margin of 0.37%, and for the Stackoverflow dataset the GRU-model with a margin of 0.07%.

The proximal term added in **FedProx** had no positive effect on performance and convergence behavior. The results are very similar to the ones from FedAvg without a proximal term, both in terms of convergence behavior and the achieved



(a) Accuracy of GRU-model on Reddit Dataset

(b) Accuracy of LSTM-model on Reddit Dataset



(c) Accuracy of GRU-model on Stackoverflow Dataset

(d) Accuracy of LSTM-model on Stackoverflow Dataset

Fig. 3: Accuracies of GRU and LSTM models, with three different aggregation methods. Comparison to baselines: *global model* denotes learning centralised, *local models* denotes the average of the only locally trained models.

performance metrics. On average, the difference between the accuracy of the two methods is only 0.2%. FedProx has reached a higher accuracy than FedAvg in three out of four settings, but managed to achieve a lower perplexity in only one of them.

In terms of the architectures, the results are also similar to FedAvg: the GRU models have lower perplexities than the LSTM models, but the difference is minor. Specifically, on the Reddit dataset, the difference is 0.23 and on the Stackoverflow dataset it is 0.08. The accuracy of the GRU-model is lower on the Reddit dataset and higher on the Stackoverflow dataset compared to their LSTM-counterparts.

To explore whether the proximal factor is more important in settings with a larger number of local epochs, an additional experiment was conducted. It was performed with 50 local epochs per client during local training, for 100 federated training rounds, with a client learning rate of 0.001 and proximal factors of 0 (which corresponds to FedAvg), 0.1, 0.3 and 1.0. The results are shown in Figure 4. They show that with the highest proximal factor the accuracy and loss are worse than with FedAvg, while factors of 0.1 and 0.3 lead to results that are very similar to FedAvg. At the 100th round,

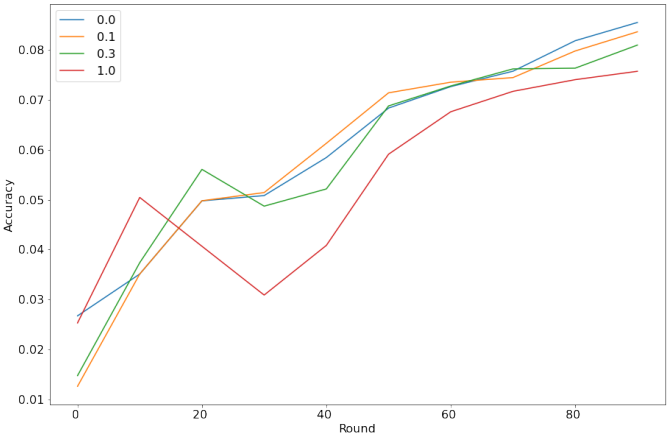


Fig. 4: Accuracy of FedProx with 50 Local Epochs

a parameter value of 0.0 achieved an accuracy of 8.58%, followed by 0.3 with 8.5%, 0.1 with 8.34% and finally 1.0 with 7.8%. The results indicate that adding a proximal term does not improve the results when no system heterogeneity is

present in the setup.

Federated Attention (FedAtt) generally showed the worst of effectiveness of the three tested aggregation methods, visible from around 250 rounds of the training process on. However, we now want to specifically investigate the performance during the early phase of training. Table V shows the performance of models trained using FedAvg and FedAtt after the first 50 and the first 100 rounds of training.

On the Reddit dataset, which is the one with more statistical heterogeneity, FedAtt performed consistently better within the first 100 rounds, but was then surpassed by FedAvg and FedProx for the rest of the training. The FedAtt accuracy after 50 rounds was higher than the FedAvg accuracy with a GRU-model by 0.78 percent points and 1.81 percent points with a LSTM-model. The perplexity of FedAtt was mostly lower than FedAvg the until around the 200th round for GRU and the 600th round for LSTM on the Reddit dataset.

For the Stackoverflow dataset, the FedAtt performance was slightly lower than the performance of FedAvg even at the 50th and 100th round, except for the LSTM-models at round 100. Interestingly, the FedAtt performance at the early rounds barely decreased (or even slightly increased) when using a LSTM instead of a GRU, as opposed to FedAvg, where the differences in performance were larger. For example, at the 50th round of FedAvg on the Reddit dataset, the perplexity of the GRU-model was lower by 5.14 than the perplexity of the LSTM-model. For FedAtt the LSTM-models perplexity was higher by 0.66.

These results do not support the claim of the authors that FedAtt converges faster and achieves a better perplexity than FedAvg. In experiments in the original paper [3] the perplexity of FedAvg and FedAtt is compared after only 50 communication rounds. In addition, the number of rounds it took to reach a perplexity of 90 is reported. Thus, it may be the case that FedAtt performs better than FedAvg only at the beginning of the training.

FedAtt might be more suitable for larger models and if there are strong limitations on the communication, so that a long training process might not be suitable. In such a scenario, one could use FedAtt to speed up the training at the beginning and then use a different aggregation method later on. This way, combinations of FedAvg and FedAtt could be used to exploit the strengths of both aggregation methods. A method to combine both was proposed in [22]. This method aggregates the updates by using either FedAvg or FedAtt depending on whether the difference in the training loss between the current and previous round is above or below a selected threshold. This method achieved both a lower perplexity and a faster convergence than FedAvg and FedAtt in the experiments.

Interestingly, after the initial 400 to 500 rounds, where FedAtt works fairly well, the performance of FedAtt seems to fluctuate from round to round with rather large decreases in accuracy in some rounds. This occurs both with the Reddit and the Stackoverflow dataset, but with the Reddit dataset, this behavior is more pronounced. For example, for the LSTM-model on the Reddit dataset, there are larger drops in performance around the 1,200th and the 1,400th round. In the first instance, the accuracy drops from 12.8% in the 1,170th round to 8.2% in the 1,190th round.

V. Conclusions and Future Work

Federated learning is an important technique for training language models, since it allows utilizing large quantities of real-life data without compromising the privacy of the data owners. Training a model that generalizes well in this setting is a challenging task due to the inherent statistical heterogeneity of the training data and due to the hardware limitations of private mobile devices. There are different approaches that address this issues through model selection, different aggregation and learning strategies, update compression and much more. In this paper, two possible model architectures, namely Long short-term memory (LSTM) and Gated recurrent unit (GRU), were evaluated in centralized and federated settings. For federated learning, the vanilla Federated Averaging algorithm and two extensions that address statistical heterogeneity were compared.

Our main findings are as follows:

- Throughout the experiments, models with GRU layers achieved slightly lower perplexity and very similar accuracy than those with an LSTM layer. The differences were especially significant at the first quarter of the training processes and became smaller trough-out the training. Thus, when the number of training rounds is limited, GRUs are more suitable, especially as the LSTM-models also require more parameters to be learned.
- Federated models achieved an effectiveness close to their centralized counterparts when trained for 2,500 rounds. The poor performance of the models that were trained locally on the data of only one user further shows the advantages of federated learning.
- Federated Attention did achieve a significantly worse performance than Federated Averaging by the end of the training process, and its performance fluctuated more. However, it consistently had better accuracy and perplexity for the Reddit dataset at the beginning of the training, making it a reasonable choice in scenarios with limited communication rounds. Also, it could be used in combination with aggregation methods that converge slower at the beginning but have a better performance later on.
- Adding a proximal term to the loss as in FedProx did not improve the results, both when training just one, or fifty local epochs.

Future work will pursue several strands of research. On the one hand, we will investigate other aggregation methods and their effects, such as *Probabilistic Federated Neural Matching* [23] as a way to deal with the problem that local models might learn the same pattern but have the respective groups of neurons at different positions, and its adaption to model architectures such as LSTMs [24], or FedPer [25], which aims to address statistical heterogeneity and improve the performance of local models. To this end, model layers

TABLE V: FedAvg and FedAtt performance at round 50 and round 100

			Round 50		Round 100	
Dataset	**Model**	**Aggregation**	**Accuracy**	**Perplexity**	**Accuracy**	**Perplexity**
Reddit	GRU	FedAvg	6.04%	105.56	10.04%	75.66
		FedAtt	6.82%	97.44	10.30%	72.23
	LSTM	FedAvg	5.09%	110.70	8.34%	85.90
		FedAtt	6.90%	96.78	10.00%	73.02
Stackoverflow	GRU	FedAvg	3.78%	12.35	9.15%	10.04
		FedAtt	5.95%	12.73	8.87%	10.18
	LSTM	FedAvg	3.78%	12.36	6.70%	10.65
		FedAtt	3.45%	12.64	7.98%	10.62

are split into base and personalization layers, with the latter being only trained locally. We will also aim at developing combinations of these methods.

Further, pre-training of language models on publicly available data and fine-tuning the models in a federated setting could utilise the advantages of both approaches and could greatly improve the results.

ACKNOWLEDGEMENTS

This work received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078 (FeatureCloud). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains. SBA Research (SBA-K1) is a COMET Center within the COMET - Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. The COMET Programme is managed by FFG.

REFERENCES

[1] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang, "Federated Learning Meets Natural Language Processing: A Survey," Jul. 2021, arXiv:2107.12603 [cs]. [Online]. Available: http://arxiv.org/abs/2107.12603

[2] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Recurrent Language Models," Feb. 2018, arXiv:1710.06963 [cs]. [Online]. Available: http://arxiv.org/abs/1710.06963

[3] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds. Fort Lauderdale, FL, USA: PMLR, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[5] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 429–450. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf

[6] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000, publisher: IEEE.

[7] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated Learning for Mobile Keyboard Prediction," Feb. 2019, arXiv:1811.03604 [cs]. [Online]. Available: http://arxiv.org/abs/1811.03604

[8] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, publisher: MIT Press.

[10] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, \. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and others, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[14] S. Reddi, Z. B. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and B. McMahan, "Adaptive Federated Optimization," in *International Conference on Learning Representations (ICLR)*, 2021, publication Title: International Conference on Learning Representations.

[15] J. Stremmel and A. Singh, "Pretraining Federated Text Models for Next Word Prediction," in *Advances in Information and Communication*, 2021.

[16] G. Kerrigan, D. Slack, and J. Tuyls, "Differentially Private Language Models Benefit from Public Pre-training," in *Proceedings of the Second Workshop on Privacy in NLP*. Online: Association for Computational Linguistics, Nov. 2020, pp. 39–45. [Online]. Available: https://aclanthology.org/2020.privatenlp-1.5

[17] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging Federated Learning by Local Adaptation," Mar. 2022, arXiv:2002.04758 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2002.04758

[18] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492

[19] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A Benchmark for Federated Settings," Dec. 2019, arXiv:1812.01097 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1812.01097

[20] D. Jurafsky and J. H. Martin, *Speech and language processing*, 3rd ed., 2023. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[21] S. F. Chen, D. Beeferman, and R. Rosenfeld, "Evaluation metrics for language models," 1998, publisher: Carnegie Mellon University.

[22] X. Wu, Z. Liang, and J. Wang, "FedMed: A Federated Learning Framework for Language Modeling," *Sensors*, vol. 20, no. 14, p. 4048, Jul. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/4048

[23] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian Nonparametric Federated Learning of Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, Jun. 2019, pp. 7252–7261. [Online]. Available: https://proceedings.mlr.press/v97/yurochkin19a.html

[24] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated Learning with Matched Averaging," in *International Conference on Learning Representations*, 2020.

[25] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019. [Online]. Available: http://arxiv.org/abs/1912.00818