Now It Sounds Like You: Learning Personalized Vocabulary On Device

Ashish Shenoy*, Sid Wang*, Pierce Chuang, John Nguyen

yuwang2020@meta.com, ashishvs@meta.com, pichuang@meta.com, ngjhn@meta.com

Abstract

In recent years, Federated Learning (FL) has shown significant advancements in its ability to perform various natural language processing (NLP) tasks. This work focuses on applying personalized FL for on-device language modeling. Due to limitations of memory and latency, these models cannot support the complexity of sub-word tokenization or beam search decoding, resulting in the decision to deploy a closed-vocabulary language model. However, closed-vocabulary models are unable to handle out-of-vocabulary (OOV) words belonging to specific users. To address this issue, we propose a novel technique called “OOV expansion” that improves OOV coverage and increases model accuracy while minimizing the impact on memory and latency. This method introduces a personalized OOV adapter that effectively transfers knowledge from a central model and learns word embedding for personalized vocabulary. OOV expansion significantly outperforms standard FL personalization methods on a set of common FL benchmarks.

Introduction

Federated learning (FL) is a distributed machine learning paradigm that enables model training on decentralized datasets without exchanging or sharing sensitive data (McMahan et al. 2016). Personalized FL considers models unique to each client under heterogeneous data settings (Hanzely and Richtárik 2020). During personalization, the server sends a global model to the clients for local fine-tuning then the client uses that model for inference. Our work studies a class of on-device language models for next-word prediction task trained with personalized FL. The resource-constrained edge devices (Chen et al. 2019a; Qiu et al. 2022; Mathur et al. 2021; Yousefpour et al. 2023) limits the usage of subword-level tokenizers. On the one hand, subword tokenizers with large vocabulary require a large memory footprint, making deployment infeasible. On the other hand, a smaller vocabulary leads to longer tokenized sequences and increases generation latency. In order to satisfy these constraints, a closed vocabulary (Qin and Rudnicky 2013), a word-level vocabulary with a white-space tokenizer that treats all unknown words as a special token, must be used. Consequently, the model cannot handle out-of-vocabulary (OOV) words (Chen et al. 2019a), making it more challenging to understand the communication style of an individual user. Previous methods have demonstrated effectiveness of user specific adaptation using domain embeddings (Shenoy et al. 2021) or prompt tuning (Dingliwal et al. 2021). But the heavy tail of OOV (shown in Figure 4) motivates the need for personalized vocabulary which they do not address. In this work, we propose OOV Expansion to personalize on-device vocabulary, tailoring the model toward users’ unique wording habits and spelling patterns. OOV Expansion uses an adapter – an MLP with residual connections inserted to different submodules to help the model adapt to new knowledge domain (Houlsby et al. 2019) – to compute the embedding output of the OOV words. Our main contributions are as follows:

- We propose OOV Expansion, a novel personalized FL method, to extract useful features for user-specific vocabulary and address the long tail OOV issue.
- We perform comprehensive experiments, showing superior results over that of the baselines, across a set of standard FL datasets.
- We demonstrate that our technique significantly outperforms previous methods with up to 5.6% relative improvements on next-word prediction accuracy and reducing the averaged unknown-word-rate by at least 97%.

Related Work

Federated learning (McMahan et al. 2016) has achieved significant impact in the field of natural language processing (NLP) in recent years (Chen et al. 2019a,b; Wu, Liang, and Wang 2020; Lin et al. 2021; Hilmkil et al. 2021; Liu et al. 2021; Ro et al. 2022). Unlike non-federated language models where subword-based tokenizations such as WordPiece (Schuster and Nakajima 2012), Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2015) or SentencePiece (Kudo and Richardson 2018) have become major choices, word-level tokenizer with a closed vocabulary is the default choice for edge device settings due to its low capacity and real-time nature. This gives rise to the commonly known OOV issue (Chen et al. 2019c,b), which has been investigated by several preceding work in federated learning: (Chen et al. 2019a) trains a separate character-level generative model to sample new words from, but by design requires

*These authors contributed equally.
Due to memory and latency constraints on on-device modeling, we opt for a character-aware LSTM-based language model instead of transformers. This architecture is commonly used in on-device applications and consists of three components: (1) a character-level CNN (CharCNN) that computes word embeddings, (2) an LSTM encoder that provides input representations, and (3) an MLP decoder that outputs vocabulary scores. The CharCNN uses a UTF-8 embedding of shape $\mathbb{R}^{256 \times E}$, where $E$ is the embedding dimension and a CNN with $D$ channels and kernel size $K$. The LSTM encoder has $M$ layers with both input and output dimensions equal to $D$, while the decoder is a linear layer with input dimension $|\mathcal{V}|$, where $\mathcal{V}$ is the closed vocabulary.

**Baseline 1: OOV-as-UNK**

This is the traditional FL personalization (i.e. involving only the non-yellow stack in Figure 1). To be precise, the model is first pretrained on a general dataset using the standard “next-word prediction” task, and then sent to client devices for federated learning. Lastly, perform personalization without any additional treatments for OOVs rather than replacing them with the special token `[unk]`.

**Baseline 2: OOV-Oracle**

The second baseline assumes to have the knowledge of all users’ OOV on server. However, because of the tight memory budgets in practice we cannot afford full vocabulary training, we instead expand the OOV-as-UNK vocabulary by the $N$ most frequent OOV words. Upon expansion, the rest stages (including pretraining, FL, and personalization) proceed in exactly the same way as in Baseline 1.

**Our Method: OOV Expansion**

We start with the unpersonalized OOV-as-UNK and perform personalization as shown in Figure 1. First, we forward the input sentence using the character-aware language model to get the vocabulary likelihood. Then, we retrieve the client’s top $n$ OOV words, compute their representations using CharCNN, and pass them through an adapter (a randomly initialized residual MLP) with hidden dimensions $\mathbf{H} = (H_1, H_2, \ldots, H_L)$, where $L$ is the number of layers in the adapter. This gives us “adapted OOV word embeddings”. Next, we take the inner product between the adapter outputs and the LSTM encoder outputs to get OOV likelihood. Finally, we concatenate the OOV probabilities with the vocabulary probabilities to obtain the full likelihood over the client personalized vocabulary for next word prediction. The adapter helps separate the two tasks of providing inputs for the LSTM encoder and computing OOV embeddings, while adapting the prior knowledge of CharCNN to the new “OOV task”. Our design ensures that no sensitive OOV information leaves the client device, preserving privacy while allowing the model to learn OOV features.

**Experiments**

**Model Setup**

The model hyperparameters for our proposed model are chosen as $E = 100$, $D = 200$, $K = 4$, $M = 2$. To initialize OOV-as-UNK we extract the 5k most frequent words from a
centralized FL dataset to form a closed vocabulary, and pre-train the model on wikitext-103 (Merity et al. 2016) using next-word prediction task for 100 epochs and is evaluated by the exact match metric EMR. The OOV-oracle baseline follows the same configuration as OOV-as-UNK, except for having a 10k-sized vocabulary. In other words, OOV-oracle expands the vocabulary of OOV-as-UNK by N = 5000 most frequent OOVs. The OOV-as-UNK model has 1.76M (resp. 2.76M) parameters in total. During OOV expansion, we personalize up to 1000 most frequent out-of-vocabulary words from each decentralized dataset.

Datasets

We train, validate, and test our approach on two public benchmark datasets for federated learning: Reddit and Stack Overflow. The Reddit dataset contains preprocessed comments from the social network, while Stack Overflow consists of questions and answers from the platform. We conduct experiments on the next-word-prediction task using federated learning with natural non-IID partitionings of the datasets. Each dataset is split into training, validation, and test segments of client data. Equivalently, the model quality is measured by EMR on the centralized test segments of all test clients.

Evaluation Metric

The model quality is measured by how often the actual next word appears among the top K word predictions, denoted by EMR (Exact Match Rate). More precisely, given a dataset \( D \) of sentences,

\[
EMR_K(D) = \frac{\sum_{S \in D} \sum_{w \in S} I\{w \in \text{Top}_K \text{pred}(w)\}}{\sum_{S \in D} |S|}
\]

This is a natural metric for real-time language model production, where \( K \) next-word suggestions are surfaced when a user is typing. In our experiments we set \( K = 3 \). The final metric is given below:

\[
EMR_3 = \frac{\sum_{u \in U} \sum_{w \in D_u} \sum_{S \in S_u} I\{w \in \text{Top}_3 \text{pred}(w)\}}{\sum_{u \in U} \sum_{w \in D_u} |S|} \text{EMR}_3(D_u)
\]

where \( U \) is the set of all test clients and \( D_u \) stands for the test segment of client \( u \)’s dataset. Equivalently, the model quality is measured by EMR on the centralized test segments of all test clients.

Experiment Details

The training recipes reported below are the same for all datasets unless stated otherwise. At each FL training round, 96 clients are randomly selected to participate in local training, with local epochs set to be 1 and training batch size as 8. Each global training epoch consists of \#users/96 training rounds, where the total number of global training epochs are chosen to be 6 and 3 for Pushshift.io Reddit and Stackoverflow respectively. We use SGD without momentum as client optimizer, and FedAdam for server side optimizer with weight decay of \( 10^{-5} \), \( \beta_1 \) as 0.9, \( \beta_2 \) as 0.999, and \( \epsilon \) as \( 10^{-8} \). We do hyperparameter search on both client and server learning rates, with ranges \([10^{-6}, 0.05]\) and \([10^{-5}, 1]\) respectively. In Table 2 we specify the best choices for both baselines on each dataset. Every hyperparameter search contains 64 experiments (with 8 running in parallel at a time. See Appendix for the hyperparameters). For each set of hyperparameters, we run 10 local epochs that early stops when the validation EMR did not improve.

Figure 2: (a) and (b) showing EMR of OOV Expansion which are the two baselines on two different datasets. (c) and (d) showing EMR of two baselines before and after personalizations on each dataset.

Figure 3: EMR and KEMR of OOV expansion on 2 datasets, with and without adapter.
Results

Improves overall performance As shown in Figure 2, our method achieves 2.1% and 1.0% (resp. 5.6% and 2.5%) absolute (resp. relative) gains of EMR₃ compared to the standard approach (i.e. OOV-as-UNK) on Pushshift.io Reddit and Stackoverflow respectively. It shows competitive performance even when compared with the stronger baseline OOV-Oracle, seeing 3.7% and 1.7% relative EMR₃ gain on Pushshift.io Reddit and Stackoverflow respectively. The fact that OOV expansion yields better improvements on Pushshift.io Reddit relative to Stackoverflow partially attributes to the higher heterogeneity of Pushshift.io Reddit data, which is reflected by the larger OOV rates (see Table 1) and the worse long tail behavior (ref. Figure 4).

Increases word-coverage rates As Table 1 shows, the OOV expansion approach reduces the OOV rate of OOV-as-UNK (resp. OOV-oracle) by more than 97.7% and 99.9% (resp. 96.4% and 99.9%) on Pushshift.io Reddit and Stackoverflow respectively.

More parameter efficient As Table 1 indicates, OOV expansion uses 24% less parameters than the OOV-Oracle during personalization. In addition, our approach does not require any extra training parameters during pretraining or FL. Consequently, it is 36% more parameter-efficient in FL compared to OOV-Oracle.

Personalization is essential Taking Pushshift.io Reddit for instance, the standard personalization can already improve the global model accuracy by 6.5% relatively on EMR₃ (see Figure 2). With OOV expansion we can further boost this quality gain up to 12.5%.

Adapter is necessary As can be seen from the first plot in Figure 3, adapter yields 15% relative EMR₃ gain over trivial adapter (i.e. identity block) on Pushshift.io Reddit. For Stackoverflow dataset, inserting adapter achieves 2.6 times accuracy of the trivial adapter approach. In the second plot we present quality comparison under a new metric denoted by KEMRₖ which represents top K known word exact match rates, defined by formula (4).

From Figure 3, we see that adapter not only helps achieves better OOV-understanding (i.e. higher EMR), the fact that it consistently improves accuracy on known word implies the important role of adapter in suppressing the issue of forgetting the knowledge from pretraining and federated learning stage.

Conclusion

This paper proposes a personalized federated learning method that enables out-of-vocabulary words understanding for a class of on-device language models with closed vocabulary. By evaluating on two public benchmarks, we show that our method significantly outperforms the commonly used personalization approach in terms of next-word-prediction accuracy and drastically reduce the unknown-word rate on average while preserving user privacy and being parameter efficient.

Appendix

Hyperparameters for Personalization

During personalizations, we perform hyperparameter search at each client among the following grid (if applied):

\[
\begin{align*}
\text{lr} & \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10\}, \\
\sigma & \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1\}, \\
\mathbf{H} & \in \{(960,), (128, 256, 128), (256, 512, 256)\}.
\end{align*}
\]

Here lr denotes the learning rate for local training; \(\sigma\) stands for the standard deviation of Gaussian distribution used to initialize the adapter parameters (except for LayerNorm); \(\mathbf{H}\) represents the hidden dimensions of adapter. Note the grid only include \(\sigma\) and \(\mathbf{H}\) during OOV expansion.

<table>
<thead>
<tr>
<th>client</th>
<th>Pushshift.io Reddit</th>
<th>Stackoverflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>0.840 0.003 0.258 0.004</td>
<td>0.168 0.005 0.129 0.008</td>
</tr>
</tbody>
</table>

Table 2: Best learning rates for two baselines

Figure 4: The quantile plot of word frequency for top 10k words from 2 datasets that demonstrates the long-tail phenomenon of OOV.

Known Word Exact Match Rate

\[
\text{KEMR}_K(D) = \frac{\sum_{S \in D} \sum_{w \in S \cap V} 1\{w \in \text{Top}_K \text{Pred}(w)\}}{\sum_{S \in D} |S \cap V|} \tag{4}
\]

where \(V\) is the closed vocabulary.
References


Qin, L.; and Rudnicky, A. I. 2013. Finding recurrent out-of-vocabulary words. In INTERSPEECH.


