FedPerC: Federated Learning for Language Generation with Personal and Context Preference Embeddings

Andrew Silva*
Georgia Institute of Technology
School of Interactive Computing
Atlanta, GA
`andrew.silva@gatech.edu`

Pradyumna Tambwekar*
Georgia Institute of Technology
School of Interactive Computing
Atlanta, GA
`pradyumna.tambwekar@gatech.edu`

Matthew Gombolay
Georgia Institute of Technology
School of Interactive Computing
Atlanta, GA
`matthew.gombolay@cc.gatech.edu`

Abstract

Federated learning is a training paradigm that learns from multiple distributed users without aggregating data on a centralized server, promising the ability to deploy machine-learning to a diverse population of users without first collecting large, labeled datasets. As federated learning involves averaging gradient updates across a decentralized population, there is a growing need for personalization of federated learning systems (i.e. conversational agents must personalize to individual users and the context of an interaction). In this work, we propose a new direction for personalization research within federated learning, leveraging both personal embeddings and shared context embeddings. We also present an approach to predict these "preference" embeddings, enabling personalization without backpropagation. Compared to state-of-the-art personalization baselines, our approach achieves a 50% improvement in test-time perplexity using 0.001% of the memory required by baseline approaches, and achieving greater sample- and compute-efficiency.

1 Introduction

As conversational agents and dialog systems are deployed to real-world scenarios, these systems require data-efficient personalization paradigms such that language systems such as conversational agents can be effectively adapted on-device. The benefits of on-device optimization are two-fold; (1) Swift adaptation of model-behavior based on human-interactions (Dudy et al., 2021), (2) Privacy protection by means of retaining all data related to the user on-device (Li et al., 2020b). One of the prevailing paradigms for learning from and engaging with end-users is *federated learning*. Federated learning is an inherently decentralized learning paradigm that assumes no access to a large labeled dataset and instead leverages averaged parameter updates across all users of the system (McMahan et al., 2017). Such averaged updates invariably dilute individual preferences or deviations from the mean, resulting in a model that works well for the average user while failing to appropriately capture under-represented preferences or sub-groups within the data. In this work, we present a novel approach (FedPerC) to personalizing federated learning with personal and context embeddings (collectively called "preference embeddings"), adapting more efficiently and effectively than prior work with respect to both data and compute on-device.

We leverage the insight that a client's data distribution is informed by both individual preferences and additional contextual information. For example, while each user may have their own *individual* style, there may be more general *population-wide* trends that inform the style of personalized predictions (e.g., dialogue assistants helping patients with cognitive disorders, whereby agents can personalize to individual patients and broader condition-wide trends). While individual preferences may be unique to each client (e.g. a user's taste or affect), we can more accurately personalize to client preferences with the addition of context, as shared-context parameters carry beneficial stylistic information across clients (Dudy et al., 2021; Jones, 1999). Stylistic or situational context provides additional information to curate relevant language

---

[*] The authors contribute equally to this paper.

Findings of the Association for Computational Linguistics: EACL 2023, pages 869–882
May 2-6, 2023

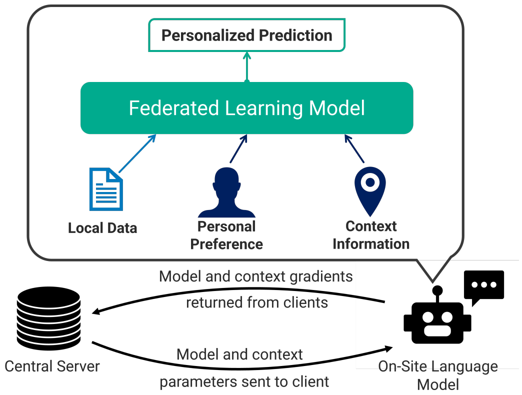

Figure 1: Overview of our personalized federated learning setup, FedPerC. Language models within client devices, such as individual agents deployed to communicate with people at hospitals, homes, or construction sites, pull down global model parameters and context embeddings. Local, on-device data is then paired with both personal and context embeddings to produce personalized predictions with global model parameters.

outputs that can be shared across users.

In this work, we contribute a new approach to personalized federated learning that is both easier to learn and more effective than prior work, and investigate the utility of personalization via individual preferences and contexts. While prior language generation approaches have developed personal or persona-based generative systems (Wu et al., 2021; Zhang et al., 2018) or context-based generative systems (Cheng et al., 2019; Lin et al., 2019a) individually, none have combined them to personalize outputs in a low-data setting under stylized preferences. We show that our approach is more sample-efficient than state-of-the-art baselines, while requiring less time to train. We additionally present an inference-only version of our approach, personalizing without backpropagation for new users. Finally, we directly test the potential for personalization with users who have been held-out from training (i.e., testing with new users). An overview of our approach is given in Figure 1.

2 Related Work

Federated learning enables machine-learning at-scale to a diverse population of end-users without first collecting a large, labeled dataset for all possible tasks. After the introduction of *federated averaging* (McMahan et al., 2017), focus has shifted to different ways of personalizing to individual users. Prior personalization approaches for federated learning have typically involved learning personal network heads and a shared global encoder (i.e., "split-learning" approaches (Gupta and Raskar, 2018)), or learning a separate local model from a global initialization (i.e., a "meta-learning" approach (Finn et al., 2017; Nichol et al., 2018)).

Learning Personal Model Heads The most prevalent approach to personalization in federated learning is through personalized model heads. Such approaches share gradient information to learn a global feature encoder, but retain user-specific classification-head gradients on-device. Approaches such as FedRep (Collins et al., 2021) solely separate out local and global gradients, while other methods such as PFedMe (Dinh et al., 2020) enforce constraints on model-divergence (such as via FedProx(Li et al., 2020a)). Other approaches, such as FedMD (Li and Wang, 2019), enable clients to adopt any desired architecture, sharing a common backbone but allowing for completely divergent model heads (Arivazhagan et al., 2019; Kim et al., 2021; Rudovic et al., 2021; Paulik et al., 2021). Finally, there has recently been increased effort on identifying clusters of related users to share model heads, such as with K-Means clustering in PFedKM (Tang et al., 2021) or through clustered personal embeddings in FedEmbed (Silva et al., 2022). Notably, there is no prior work which learns both personal *and* contextual model heads for personalization within federated learning.

Meta-Learning Global Models An alternate approach to personalizing federated learning models is through the adoption of meta-learning (Jiang et al., 2019; Fallah et al., 2020), for learning a global model prior to fine-tuning on client-data. After cloning the global model as an initialization from all client's updates, local, client-side models are permitted to diverge and fine-tune to a user's individual preferences or data distribution (Fallah et al., 2020; Deng et al., 2020; Hanzely and Richtárik, 2020; Hanzely et al., 2020; Lin et al., 2019b; Chen et al., 2022). However, computing and applying gradients for a full model often requires too much time, power, and memory. As such, expensive full-model gradients can often only be computed and applied when a device is not actively in-use. As in the split-learning literature, there are not meta-learning approaches for disentangling personal and contextual preferences within personalized federated learning.

Learning with Personal Embeddings Our work leverages the insight that personal preferences can be represented using a personalized embedding, allowing the model to condition output predictions on personal preferences without requiring completely re-trained classification heads or networks. Personal embeddings have been used in prior work to capture an individual's "style," often in imitation learning settings (Tamar et al., 2018; Hsiao et al., 2019; Paleja et al., 2020; Schrum et al., 2022a,b). Treating personal embeddings as neural network parameters that are updated on-device, these approaches learn to embed preferences and condition network output over both input data and preference embeddings. Most closely related to our work are FedNLG (Lu et al., 2021), which predicts "persona" parameters for users, and the Global+ model in FedEmbed (Silva et al., 2022), which learns a personal embedding for each user. However, FedNLG requires access to a user's entire history of language and demographic data in order to produce a "persona" for each user, informing the generation of a "persona" embedding. Such information is difficult to collect for large datasets, and may compromise privacy requirements in federated learning scenarios. Similarly, the Global+ model incorporates supervised style feedback, requiring labels that may be impractical to obtain in a private, federated setting. Finally, prior embedding-based approaches solely learn *personal* embeddings, neglecting stylization through context. In our work, we explore the utility of incorporating context in addition to personal preferences, and all preference embeddings are updated solely via a self-supervised language-modeling loss.

Personalization in Language Personalization for language generation systems seeks to produce grounded systems that can efficiently adapt to end-user needs (Yang and Flek, 2021; Dudy et al., 2021). One such approach to personalization is by learning a "persona" for each user and conditioning the language model on the embeddings or representation for the persona via a memory network (Zhang et al., 2018; Wu et al., 2021; Lu et al., 2021). "Personas" are generally short sequences of 5-6 sentences which contain information about an individual such as "I have blonde hair" or "My mom is a doctor." Similar approaches leverage Bayesian inference methods to infer context (Majumder et al., 2020) or persona (Kim et al., 2020), and then condition the language generation on the inferred context. However such approaches involve collecting and maintaining user-profiles on a central server which may violate user-confidentiality. Alternate approaches seek bypass this issue by enabling dynamic speaker modeling through context-based fine-tuning rather than conditioning on profile information (Cheng et al., 2019; Li and Liang, 2021). FedPerC leverages a similar design to dynamically learn personal and context embeddings through data from small datasets for a given user, while also preserving user-confidentiality via federated learning.

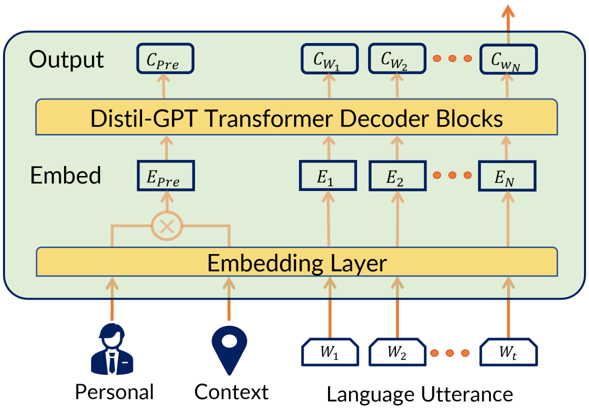

Figure 2: The FedPerC model architecture. Input data, such as on-device conversation data for a user, is passed into the language model in addition to personal and context labels specifying user's preference. The personal and context labels are embedded through a preference embedding layer to produce a single preference embedding. This preference embedding is combined with the word embeddings for the input sequence and passed into the DistilGPT2 model to predict the next word.

FedPerC represents a new direction in personalized federated learning research, enabling personal and stylized language generation with a fraction of the memory, data, and compute costs of prior approaches without requiring access to pre-made personal profiles or sequence labels.

3 Approach

In this section, we present our novel approach to personalization in federated learning with FedPerC. FedPerC produces personal and contextual preference embeddings either via backpropagation (i.e., learning preference embeddings), or by inference (i.e., predicting preference embeddings). A visual overview of our federated learning architecture is in Figure 2, and a step-by-step walk-through of our training algorithm is given in Algorithm 1.

3.1 Personalization via Embeddings

Personalization in FedPerC is achieved entirely through preference embeddings. Every input sample (e.g., an incomplete sentence) is accompanied by both a personal preference embedding, representing the user, and a contextual preference embedding, representing the context or style of the prediction. These two embeddings are combined via an element-wise multiplication to produce a single preference embedding that accompanies the input sample. By leveraging both personal and context embeddings, FedPerC considers the individual user *and* the broader context of an utterance, enabling personal, stylized prediction.

In the language-modeling domain, the unified preference embedding is prepended to the input utterance, providing a prefix for the model to consider (Li and Liang, 2021). The model then predicts the next token of the utterance, and a language-modeling loss is calculated by comparing the prediction to the user's actual next token. The next token is then appended to the sequence, and preference embeddings are again prepended to the new input sequence, and the process repeats. After completing a full utterance, preference embeddings may be updated, either through backpropagation or by using an embedding-generator to predict new personal and contextual embeddings for the client.

3.2 Federated Learning Algorithm

To begin, all clients initialize their own personal embedding on-device, and the server initializes a set of C context vectors for each relevant setting given the target task. We additionally assume that all data points on a client device have an associated context, c, being derived from the contextual information of the client device when the data point was captured (e.g., time of day, location, etc.).

Training begins by distributing all the requisite information to client-devices. Client devices pull down the global model parameters, θ, and the global context embedding parameters, ϕ, making local copies, θ_d and ϕ_d (line 6). Unlike the global model parameters and context embeddings, the personal embeddings, ψ_d do not need to be copied from the server as they are kept on client-devices.

Client devices then take K gradient steps using their own on-device data, where each input sample is paired with the client's on-device embedding, ψ_d, and the context embedding for the particular sample, $\phi_{d,c}$, assuming the data point was drawn under context $c \in C$. Gradients are calculated using a language-modeling objective, though any objective could theoretically be applied. If preference embeddings are being generated via forward-propagation rather than learned via backpropagation, contextual and personal preference embeddings will also be predicted by an embedding-generator at this stage (note: the parameters of the embedding-generator are shared globally, being a part of θ).

Gradients are applied to the shared-model parameters, θ_d, and are then used to update preference embeddings (line 9). If preference embeddings are being predicted, these gradient steps are also applied to the shared embedding-generator, and preference embeddings (i.e., context embeddings ϕ_d and personal embeddings ψ_d) are overwritten with their latest predicted values (lines 10-11). If preference embeddings are being learned via backpropagation, gradient steps are applied to ϕ_d and ψ_d using Equation 1 (lines 10-11).

After K steps, gradients for θ_d and ϕ_d are sent back to the server, while ψ_d remains on-device (lines 13 - 15). The server computes a single update for the global model and context embeddings by averaging across all clients (lines 17-18). The server applies the averaged update to θ and ϕ, and the process repeats (lines 19-21).

$$\begin{aligned} \phi_d &= \phi_d + \nabla_\phi \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d) \\ \psi_d &= \psi_d + \nabla_\psi \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d) \end{aligned} \tag{1}$$

In a typical federated averaging deployment, client devices will pull down global parameters, fine-tune on local datasets, and then test on held-out, local data. With FedPerC, the majority of the network's parameters, θ, are frozen, reflecting a federated-learning setup with a more constrained computational budget when deploying large language models. Using FedPerC, clients pull down and subsequently freeze global parameters, θ, and either generate preference embeddings from observation, or only compute and apply gradients to context embeddings, ϕ, and their local personal embedding ψ. Relying on forward-propagation calls rather than backpropagation, or by computing gradients over only these embeddings, we reduce the computational overhead of FedPerC while preserving or even improving upon accuracy relative to fine-tuning an entire model. When testing over local data, all updates to context embeddings ∇_ϕ are not sent to the central server. Rather, these gradients are directly applied to the context embeddings for

Algorithm 1 FedPerC Training Loop

```
Given: Training objective $\mathcal{L}$, Client devices $D$, # client steps, $K$, # global steps, $N$
Initialize: Global model $\theta$, Context embeds $\phi$
Initialize: Personal embeddings on-device $\psi$
for $n \in N$ do
  for $d \in D$ do
    $\theta_d = \theta, \phi_d = \phi$
    for $k \in K$ do
      Sample $B_d$ from user's on-device data
      $\theta_d \leftarrow \theta_d + \nabla_\theta \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d)$
      $\phi_d \leftarrow \phi_d + \nabla_{\phi_d}$
      $\psi_d \leftarrow \psi_d + \nabla_{\psi_d}$
    end for
    $\nabla_{\theta_d} \leftarrow \theta - \theta_d$
    $\nabla_{\phi_d} \leftarrow \phi - \phi_d$
    Return $\nabla_{\theta_d}$ and $\nabla_{\phi_d}$ to the server
  end for
  $\nabla_\theta \leftarrow \frac{1}{D} \sum_d^D \nabla_{\theta_d}$
  $\nabla_\phi \leftarrow \frac{1}{D} \sum_d^D \nabla_{\phi_d}$
  $\theta \leftarrow \theta + \nabla_\theta$
  $\phi \leftarrow \phi + \nabla_\phi$
end for
```

the current user, and then discarded. When instantiating a new embedding for a previously unseen user, we set the user's embedding to the noisy-average of all known user embeddings.

3.2.1 Generating Preference Embeddings

To generate embeddings, we adopt a similar procedure to HyperNetworks (Ha et al., 2016; Shamsian et al., 2021), in which a neural network is trained to predict parameters of another network. In FedPerC, an embedding-generator is trained to predict the parameters of preference embeddings (either personal or context). To generate embeddings, we apply an additional transformer decoder block (Vaswani et al., 2017), that uses a randomly-initialized personal embedding and a known context embedding as the queries, along with the word embeddings for the utterance as the keys and values to update the given preference embeddings. We utilize separate generators to predict the personal embedding, ψ_d, and the context embedding, ϕ_d. Specific training details for the embedding-generator applied to language-modeling are given in the appendix.

While the embedding-generator must be learned from scratch during training, this method of predicting preference embeddings allows us to generate personal embeddings for previously *unseen* users when testing. By predicting preference embeddings, we circumvent the need for expensive gradient calculation and on-device learning. Instead, new users can quickly reap the benefits of personalized predictions via a trained preference prediction module (i.e., the embedding generator), as opposed to conventional personalized federated learning methods that require slow and sample-inefficient on-device learning.

4 Experiments

We conduct several experiments to evaluate the sample efficiency, generalization, and runtime of our approach relative to baseline federated learning frameworks. In our experiments, we compare:

- FedPerC – Learning personal and context embeddings jointly with a global feature encoder, and performing local fine-tuning of personal and context embeddings on-device.
- FedPerC (Frozen) – As above but without local fine-tuning for preference embeddings.
- FedPerC (Generated) – Learning an embedding generator and global feature encoder, and then using only generated embeddings at test-time (i.e., not directly learning embeddings).
- Split-Learning – Learning personal and context-specific model-heads jointly with a global feature encoder, and performing local fine-tuning of the personal and context-specific model heads on-device (Dinh et al., 2020; Collins et al., 2021).
- Meta-Learning – Learning a single global model for all users and contexts, and fine-tuning the shared model-head on-device (Finn et al., 2017; Nichol et al., 2018).

Because our experimental datasets do not contain labeled personas for all users, we do not compare directly to prior works that assume access to such information (e.g., FedNLG (Lu et al., 2021)).

We conduct two sets of experiments to compare the above approaches on both sample efficiency and runtime efficiency. For the sample efficiency experiments, we present perplexity numbers for all methods across two versions of the dataset: known users and withheld users. For our known user experiments, all users are present in the training and testing set. For our withheld user experiments, a subset of users from each dataset is withheld entirely from training, and performance results are presented only for the held-out users. Perplexity is calculated over unseen utterances with the first three tokens of each utterance given as a prompt. Finally, we present qualitative results from our method, demonstrating the power of stylized generation for individual users.

All models are initialized with the DistilGPT2 pre-trained model (Wolf et al., 2019), with all layers frozen. We note that the use of large language models for federated language generation is a significant improvement over prior work (Lu et al., 2021) which instead learned Seq2Seq models from scratch. For our Split-Learning and Meta-Learning baselines, the last layer of the model is unfrozen. Training details are in the appendix.

4.1 Datasets

We conduct our experiments using two datasets, a smaller dataset of TV Show scripts ("Friends" (Chen and Choi, 2016) and "Game of Thrones" (Koirala, 2019)) and a larger dataset of Reddit posts (Chang et al., 2020). Each dataset has a diverse set of individuals as well as clearly defined contexts/styles (i.e., TV shows or subreddits). These properties enable us to not only compare our approach to baseline approaches for personalized predictions, but they also enable us to move users between contexts or styles (e.g., producing text for a "Friends" character under a "Game of Thrones" context). By generating sequences for different users under new styles, we demonstrate the power of FedPerC for personal, stylized prediction. FedPerC is the first work to experiment on a dataset consisting of language data from real-world users, and not just movie scripts or dialogues. Additional information about the datasets used in this work is given in the appendix.

For both datasets, we treat each sentence from a speaker (i.e., TV Show character or Reddit user) as an independent utterance and we only consider utterances with at least three tokens. For experiments on known users, we perform a 60/20/20 Train/Validation/Test data split. For experiments on novel, unseen users, we perform a 70/15/15 split of Reddit users, and we manually select the "Friends" and "Game of Thones" users to include in each data fold. For both sets of experiments, all contexts are seen during training.

4.2 Results and Discussion

All experiments are repeated fifteen times, with different random seeds for each run, and means and standard deviations for performance and runtime results are presented in Tables 1, 2, and 3. Tables 1 and 2 show that our approach is able to generate sensible language for both held-out user instances and known users. Both embedding-based approaches presented in this paper (i.e., FedPerC with generated or learned embeddings) show drastic improvements over baselines in terms of both sample- and runtime-efficiency, and are more suitable for real-world on-device language models.

Summary With known users, FedPerC achieves perplexity as low as 46.7 and 100.3, on the TV Show and Reddit datasets, respectively, compared to the best baseline perplexities of 82.1 and 233.2 (a 45-50% improvement). For unknown users, FedPerC achieves perplexities of 52.3 and 97.6, respectively, compared to baselines at 96.7 and 212.7 (a 45-55% improvement). FedPerC training times are between 25-400% faster than baseline training times. Finally, FedPerC uses 0.001% of the memory that baseline methods use for stylized personalization.

Memory Costs FedPerC incurs a significantly lower memory cost than prior Split-Learning based approaches (Li and Wang, 2019; Collins et al., 2021; Dinh et al., 2020; Tang et al., 2021; Rudovic et al., 2021; Gupta and Raskar, 2018). The Split-Learning baselines require maintaining a model-head for each user and context present in the dataset, and the size of these model heads is proportional to the size of the vocabulary. On each client-device, a user's personal model head and all context heads need to be stored in memory and used in forward passes. In our work, every GPT model head is approximately 154 MB (being 768×50257 parameters). To update the model on-device, one would need to store a model head corresponding to every possible context. Our Reddit dataset involves 57 contexts, totalling an additional $\sim$ 8GB of data in memory. This memory requirement for personalized heads could become infeasible for real-world tasks, particularly for on-device inference or backpropagation on mobile devices. Using FedPerC, which only requires the addition of a drastically smaller preference embedding, the total amount of memory required on device to store the embeddings is only $\sim$171 KB (0.001% of the memory

Table 1: Perplexity Showing Sample Efficiency Across All Methods for Known Users. Lower is Better.

	# Samples	FedPerC	FedPerC (Frozen)	FedPerC (Generated)	Split-Learning	Meta-Learning
Reddit	1	219.5 ± 35.7	146.2 ± 2.3	**120.2 ± 1.4**	1297.5 ± 21.9	226.2 ± 3.7
	5	131.6 ± 10.1	136.9 ± 3.4	**123.3 ± 2.8**	994.3 ± 27.8	234.7 ± 5.1
	15	**111.4 ± 3.5**	132.6 ± 4.7	120.0 ± 3.3	691.3 ± 34.3	227.1 ± 8.4
	All	189.5 ± 6.7	167.9 ± 2.0	**124.9 ± 1.3**	930.4 ± 30.9	241.4 ± 2.1
TV Shows	1	57.2 ± 3.6	**50.3 ± 1.6**	51.6 ± 1.3	359.4 ± 28.2	111.7 ± 4.6
	5	51.5± 1.5	**50.7 ± 2.1**	51.7 ± 2.0	244.5 ± 15.1	110.0 ± 6.5
	15	**48.8 ± 1.7**	51.0 ± 2.1	51.7 ± 2.0	167.7 ± 8.6	111.9 ± 6.1
	All	**46.7 ± 1.7**	51.2 ± 2.0	52.1 ± 2.6	82.1 ± 3.3	113.0 ± 4.7

Table 2: Perplexity Showing Sample Efficiency Across All Methods for Withheld Users. Lower is Better

	# Samples	FedPerC	FedPerC (Frozen)	FedPerC (Generated)	Split-Learning	Meta-Learning
Reddit	1	594.3 ± 973.8	202.0 ± 5.9	**117.3 ± 1.8**	922.9 ± 27.8	213.9 ± 6.0
	5	139.4 ± 4.4	202.9 ± 10.9	**117.5 ± 2.7**	655.9 ± 18.8	212.2 ± 5.4
	15	117.4 ± 1.9	203.6 ± 11.2	**116.6 ± 2.6**	449.2 ± 11.4	211.7 ± 3.7
	All	**101.1± 2.2**	202.2 ± 7.6	117.9 ± 2.8	309.3 ± 8.3	212.8 ± 5.2
TV Shows	1	205.1 ± 292.2	96.4 ± 10.4	**68.7 ± 5.9**	283.6 ± 30.9	113.5 ± 13.1
	5	68.6± 5.6	90.1 ± 4.9	**66.7 ± 6.3**	220.7 ± 29.2	111.4 ± 13.3
	15	**62.1 ± 5.0**	97.6 ± 6.8	66.1 ± 5.5	158.1 ± 20.0	117.3 ± 10.5
	All	**52.3 ± 3.3**	98.2 ± 9.5	68.6 ± 5.1	96.7 ± 14.5	114.2 ± 17.0

required by separate model heads).

Sample Efficiency FedPerC is able to outperform Split-Learning and Meta-Learning models with significantly fewer samples across both experiments and both datasets. This trend is reflected regardless of whether embeddings are generated or learned through backpropagation. When embeddings are learned, FedPerC improves with online data to more effectively model the given user's style as more data is made available to the model. Conversely, while the generated embeddings exhibit greater sample performance with a single sample, they are unable to improve with more data. For both known and with-held users, FedPerC with generated embeddings is unable to effectively update the preference embedding to improve generation performance. Finally, we see an increase in perplexity for Reddit users with all available data when using FedPerC. This result suggests that it is possible to *overfit* preference embeddings, as we see an increase in perplexity from 15 to "All" samples (Table 1).

We observe no improvement for the Meta-Learning baseline, regardless of how much data is available for each user. This lack of improvement suggests that the model is not capable of rapidly personalizing to a single user or context with only a handful of available samples. Only updating the model head may be insufficient when the base, shared model head must generalize across all possible contexts and characters.

The Split-Learning baseline, on the other hand, does show significant improvement with increasing amounts of data for withheld and known users. In our known user experiments, all personal model heads should have already been well-tuned to personal preferences. Our result therefore suggests that context-specific model heads are over-generalized to their respective contexts, and must be refined to better-align with individual users.

Runtime Efficiency FedPerC incurs significantly lower training costs than both Split-Learning and Meta-Learning approaches to personalization. While Meta-Learning baseline does not have the memory-constraints of the split-learning model in terms of storing *additional* model heads, training the Meta-Learning baseline still involves computing gradients over all 768×50257 parameters in the shared output layer. As we show in Table 3, this leads to a significantly more costly training time for each user. Similarly, the Split-Learning baseline must update *at least* two model heads for

Table 3: Training and Testing run-time for FedPerC and our baselines, in milliseconds. Lower is better.

Method	FedPerC	FedPerC (Frozen)	FedPerC (Generated)	Split-Learning	Meta-Learning
Train Pass Time	88.18 ± 24.104	**43.57 ± 11.99**	55.96 ± 12.41	222.08 ± 37.55	111.81 ± 22.33
Test Pass Time	40.37 ± 11.76	40.25 ± 12.10	47.02 ± 12.63	65.42 ± 16.49	**36.77 ± 8.95**

Table 4: Generated Examples using Arya, from "Game of Thrones" (GoT) and Chandler, from "Friends".

Character	Show	"We Must"	"I think"
Chandler	Friends	be careful! I'm not going to get a divorce.	I'll be able to do this.
		be a little bit more relaxed than we're here.	I'm a good man
		be the one who's the one who's the one...	I'm a big fan of you
Chandler	GoT	be honest with you.	I'm going to be a little more serious
		be very nervous about the possibility of a bomb attack.	I'm going to be a little bit of a jerk
		be a little nervous about the situation	I'm going to have a big secret.
Arya	GoT	be a little more careful.	you can't help me
		be careful about the dangers of the sea.	of the people
		be wary of the possibility of a coup.	you're not going to be a hero?
Arya	Friends	be a thief	I'm not a bad person
		be a hero.	I can do it
		be a little girl.	I should have a chance to do something

each backward pass, requiring gradient computation for $2 \times 768 \times 50257$ parameters. If a user is active in multiple contexts, then additional context model-heads must be used, further exacerbating the training cost of the Split-Learning approach. The Split-Learning approach must also leverage these additional context model-heads at test-time, resulting in the slowest forward-passes of any baseline.

In contrast to prior approaches, training for FedPerC only requires updating 2×768 parameters. This reduced computation results in significantly lower training times. When we train an embedding-generator, there is an increase in training times reflecting the added cost of computing gradients for the embedding generator. Additionally, there is a test-time penalty incurred by the added forward-pass parameters. When running inference with any version of FedPerC, preference embeddings are combined and then prepended to the input utterance. This process results in marginally slower test times with FedPerC relative to the Meta-Learning baseline, though the differences are not significant.

Qualitative Results Our qualitative results in Table 4 demonstrate the power of FedPerC, and justify the need for personal *and* context embeddings. Not only is our model able to complete sequences for a character in their "home" context (i.e., the context from which all of their data is drawn), but we are also able to stylize generation for characters, bringing them into *new* contexts. We present generated samples from a "Game of Thrones" (GoT) character (Arya) with a "Friends" context embedding and a GoT context embedding. We see that Arya's generated sequences are distinct under the two different contexts. Under the GoT context, Arya's utterances match the theme of the show, suggesting danger and revolution. Under the "Friends" context, Arya's utterances change to instead reflect more mundane, modern language while still preserving personal attributes of the character.

Across all of our experiments, particularly the novel experimental evaluation on held-out user-instances, our results provide evidence that embedding-conditioned personalization within federated learning can be effectively applied to real-world use-cases. FedPerC offers a promising avenue of future work towards on-device language models, capable of efficient language generation with respect to compute-power and data.

5 Conclusion

We present FedPerC, a new approach to personalized federated learning, enabling efficient and high-performance personalization to client devices by leveraging individual and shared preference embeddings. Combining shared contexts with individual personal preferences, FedPerC outperforms baselines even when allotted a lower computational budget, and is the first federated language generation approach to build on large language models rather than training sequence generation models from scratch. We also provide a method of generating preference embeddings through inference alone, providing personalization with no on-device gradient computation, and we show comparable performance to FedPerC using learned embeddings.

We presented experiments on two datasets, TV Show scripts and Reddit user data, presenting empirical evidence of the utility of FedPerC towards personalizing to unseen users in a federated learning setting, i.e. a 50% improvement in terms of runtime *and* perplexity when fine-tuning on with new users. We also demonstrated qualitative results, showing the power of separate personal and context embeddings and enabling stylization of users in new contexts. Our results show that FedPerC offers a promising path forward for personalization within federated learning, achieving superior quantitative results and requiring significantly less training time and data relative to baseline approaches.

Limitations

Firstly, although our embedding-generator offers a promising avenue of personalizing without any on-device gradient computation, our generator is currently unable to improve on its generated embeddings given more examples for a given user. As shown in our results from Sec 4.2, while the model can generate an effective preference embedding for a user with a single sample, it is unable to improve with more data. In future work, we hope to explore approaches to facilitate a generator which can effectively modify embeddings given additional data.

Secondly, our approach caters to confidentiality by ensuring that user-data and embeddings remain on-device, however we have not incorporated differential privacy in our experiments (Li et al., 2020b). Future work may apply differential privacy to guarantee user privacy while personalizing and contributing feature encoder information to a central server. Finally, it is important to note that FedPerC does not solve all problems within the scope of language generation models. As FedPerC offers a path forward to facilitate privacy protection and efficient on-device learning for large language models, future work may extend FedPerC to additional problems (e.g., language summarization or turn-based dialogue generation).

Ethics Statement

Federated learning systems promise the ability to learn useful models without needing access to private, protected data on user's devices. By contributing improvements to personalization and contextualization within the federated learning paradigm, FedPerC takes a step towards improving fairness of federated learning systems, which otherwise struggle with fitting to data distributions that are not common in training populations. However, it is important to note that FedPerC works to maximize the likelihood of the observed data, which may reinforce existing societal biases and stereotypes–there are no protections or safeguards in place to ensure responsible generation or unbiased preference learning (May et al., 2019; Nadeem et al., 2021; Silva et al., 2021). While this problem is certainly not unique to FedPerC, it is important to consider the safety and fairness implications of improved language generation, and future work must address biases inherent to large language models (Schick et al., 2021; Ravfogel et al., 2020). Another important ethical consideration is the potential misuse of our generative modelling approach for malicious impersonation. In our federated setup, personal embeddings would be kept on-device, meaning that an individual's style is not accessible to others. However, this does not prevent users from manually impersonating other individuals (e.g., celebrities). Future work must explore additional mechanisms for the prevention of misuse at all stages of the personalization pipeline, including protections against impersonation of other individuals.

6 Acknowledgements

This work was supported by the Office of Naval Research (ONR) under award N00014-19-1-2076, the National Science Foundation under awards NSF CPS-2219755 and NSF IIS-2112633. Andrew Silva was supported by the Apple Scholars in AI/ML PhD fellowship.

References

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*.

Jonathan P. Chang, Caleb Chiam, Liye Fu, Andrew Z. Wang, Justine Zhang, and Cristian Danescu-Niculescu-Mizil. 2020. Convokit: A toolkit for the analysis of conversations. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2020, 1st virtual meeting, July 1-3, 2020*, pages 57–60. Association for Computational Linguistics.

Letian Chen, Sravan Jayanthi, Rohan Paleja, Daniel Martin, Viacheslav Zakharov, and Matthew Gombolay. 2022. Fast lifelong adaptive inverse reinforcement learning from demonstrations. In *Proceedings of the 6th Conference on Robot Learning (CoRL), 2022*.

Yu-Hsin Chen and Jinho D Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in tv shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 90–100.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2019. A dynamic speaker model for conversational interactions. 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 2772–2785.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2089–2099.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*.

Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. 2020. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*.

Shiran Dudy, Steven Bedrick, and Bonnie Webber. 2021. Refocusing on relevance: Personalization in nlg. *arXiv preprint arXiv:2109.05140*.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135.

Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. 2020. Lower bounds and optimal algorithms for personalized federated learning. *arXiv preprint arXiv:2010.02372*.

Filip Hanzely and Peter Richtárik. 2020. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*.

Fang-I Hsiao, Jui-Hsuan Kuo, and Min Sun. 2019. Learning a multi-modal policy via imitating demonstrations with mixed behaviors. *arXiv preprint arXiv:1903.10304*.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*.

Karen Sparck Jones. 1999. Automatic summarizing: factors immarizing: factors and directions. *Advances in automatic text summarization*, page 1.

Hyunwoo Kim, Byeongchang Kim, and Gunhee Kim. 2020. Will i sound like me? improving persona consistency in dialogues through pragmatic self-consciousness. *arXiv preprint arXiv:2004.05816*.

Joongheon Kim, Seunghoon Park, Soyi Jung, and Seehwan Yoo. 2021. Spatio-temporal split learning. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pages 11–12. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Shekhar Koirala. 2019. Game_of_thrones. `github.com/shekharkoirala/Game_of_Thrones`.

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

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

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

Yiwei Li, Tsung-Hui Chang, and Chong-Yung Chi. 2020b. Secure federated averaging algorithm with differential privacy. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019a. Moel: Mixture of empathetic listeners. *arXiv preprint arXiv:1908.07687*.

Zhaojiang Lin, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019b. Personalizing dialogue agents via meta-learning. *arXiv preprint arXiv:1905.10033*.

Yujie Lu, Chao Huang, Huanli Zhan, and Yong Zhuang. 2021. Federated natural language generation for personalized dialogue system. *arXiv preprint arXiv:2110.06419*.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. *arXiv preprint arXiv:2010.03205*.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. 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 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Rohan Paleja, Andrew Silva, Letian Chen, and Matthew Gombolay. 2020. Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations. In *Advances in Neural Information Processing Systems*, volume 33, pages 6417–6428. Curran Associates, Inc.

Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. 2021. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503*.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Ognjen Rudovic, Nicolas Tobis, Sebastian Kaltwang, Björn Schuller, Daniel Rueckert, Jeffrey F Cohn, and Rosalind W Picard. 2021. Personalized federated deep learning for pain estimation from face images. *arXiv preprint arXiv:2101.04800*.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Mariah L Schrum, Erin Hedlund-Botti, and Matthew Gombolay. 2022a. Reciprocal mind meld: Improving learning from demonstration via personalized, reciprocal teaching. In *Proceedings of the 6th Conference on Robot Learning (CoRL), 2022*.

Mariah L Schrum, Erin Hedlund-Botti, Nina Moorman, and Matthew C Gombolay. 2022b. Mind meld: Personalized meta-learning for robot-centric imitation learning. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 157–165.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR.

Andrew Silva, Katherine Metcalf, Nicholas Apostoloff, and Barry-John Theobald. 2022. Fedembed: Personalized private federated learning. *arXiv preprint arXiv:2202.09472*.

Andrew Silva, Pradyumna Tambwekar, and Matthew Gombolay. 2021. Towards a comprehensive understanding and accurate evaluation of societal biases in pre-trained transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2383–2389, Online. Association for Computational Linguistics.

Aviv Tamar, Khashayar Rohanimanesh, Yinlam Chow, Chris Vigorito, Ben Goodrich, Michael Kahane, and Derik Pridmore. 2018. Imitation learning from visual data with multiple intentions. In *International Conference on Learning Representations*.

Xueyang Tang, Song Guo, and Jingcai Guo. 2021. Personalized federated learning with clustered generalization. *arXiv preprint arXiv:2106.13044*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yuwei Wu, Xuezhe Ma, and Diyi Yang. 2021. Personalized response generation via generative split memory network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1956–1970.

Diyi Yang and Lucie Flek. 2021. Towards user-centric text-to-text generation: A survey. In *International Conference on Text, Speech, and Dialogue*, pages 3–22. Springer.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

A Generation Algorithm

At each time-step during inference, the embeddings are updated by the following equations.

$$e_t = Multi\_Head\_Attn(W_{<t}, LN(e_{t-1}), W_{<t})$$
$$e_t = LN(LN(e_t) + LN(e_{t-1}))$$
$$e_t = LN(FFN(e_t) + LN(e_{t-1}))$$

For the first timestep, e_{t-1} is initialized as ϕ or ψ for personal and context embeddings respectively, and LN represents a layer normalization function. We apply future-masking to prevent any future-information in the sequence from leaking forward into the rest of the model. After processing the entire utterance, the generated embedding is updated to the final value of e_t, which can then be stored on-device for future processing. An updated algorithm which applies the generator to predict preference embeddings can be found in Alg 2.

B Training Details

All models are initialized with the DistilGPT2 pretrained model from Huggingface (Wolf et al., 2019). All layers of the model are frozen, and FedPC only backpropagates error to personal and context preference embeddings. For our Meta-Learning baseline, the last layer is unfrozen and all users jointly update this final output layer (note: there is no dedicated context head in this approach). Our Split-Learning baseline assigns a unique model head to each user and to each context, and each user only updates their own model head and the contexts that they use.

All models are trained for 55 epochs over their training datasets using the Adam optimizer (Kingma and Ba, 2014) for global updates (learning rate = 1) and local updates (learning rate = 0.001). Each client (character or Reddit user) makes 10 local updates before passing their pooled gradient information back to the server. During training, each client samples 15 data points per training pass. For local fine-tuning updates at test-time, each user makes 15 updates using a small portion of the test data (the data used for fine-tuning is not used for testing).

All models use a frozen DistilGPT2 model from HuggingFace as their initialization. After empirical experimentation, we opted to freeze the majority of the DistilGPT2 parameters by default. This freezing helped to save on computational and memory costs as well as improving generalization performance across diverse users. As a result of this freezing, shared learning and personalization updates will only affect model heads, shared embeddings, and/or personal embeddings.

FedPC leverages a standard federated averaging training procedure (FedAvg) (McMahan et al., 2017) with the addition of a FedProx penalty term (Li et al., 2020a) to regularize on-device client updates back to the globally-averaged model. Empirically, FedProx improved performance for all methods. We fix the FedProx μ parameter to 1.

Training was carried out on an NVIDIA A40 GPU with 48GB of memory. Due to limitations of the GPU, not all context-heads could be stored in memory at once for our Split Learning baseline when working with the Reddit dataset. The GPU could only accommodate 14 model heads in addition to the DistilGPT2 model, but the dataset featured 57 unique subreddits. To work around this limitation, 13 context heads were active at all times, and the parameters of those heads were saved and overwritten as necessary to ensure that each user had access to their required context heads.

C Dataset Information

The TV Show dataset is constructed by merging scripts from two shows, “Friends” and “Game of Thrones.” We use ConvoKit (Chang et al., 2020) to gather the “Friends” Corpus (Chen and Choi, 2016), and retain the six main characters. We use a set of “Game of Thrones” scripts (Koirala, 2019) to query for the thirteen characters with the highest utterance-count. Our merged dataset has 19 characters, 60650 utterances, and two contexts. The average utterance count for each character is 3370, with “Friends” characters having more utterances than “Game of Thrones” characters.

Algorithm 2 Personalized Federated Learning Loop with Generated Embeddings

```
Given: Training objective, $\mathcal{L}$, Client devices $D$
Given: Number of client steps, $K$
Given: Number of global steps, $N$
Initialize: Global model, $\theta$, Context embeddings $\phi$, Context Generator $\Gamma$, Client Generator $\nu$
Initialize: Personal embeddings on-device $\psi$
for $n \in N$ do
    for $d \in D$ do
        $\theta_d = \theta, \phi_d = \phi, \Gamma_d = \Gamma, \nu_d = \nu$
        for $k \in K$ do
            Sample $B_d$ from client's on-device data
            $\theta_d \leftarrow \theta_d + \nabla_\theta \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d)$ // Fine-tune global model with local data
            $\phi_d \leftarrow \nu_d(\theta_d, \phi_{d,c}, B_d)$ // Generate context embedding from local data
            $\psi_d \leftarrow \Gamma_d(\theta_d, \psi_d, B_d)$ // Generate personal embedding from local data
            $\nu_d \leftarrow \nu_d + \nabla_\nu \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d)$ // Update client Generator
            $\Gamma_d \leftarrow \Gamma_d + \nabla_\Gamma \mathcal{L}(\theta_d, \phi_{d,c}, \psi_d, B_d)$ // Update context Generator
        end for
        $\nabla_{\theta_d} \leftarrow \theta - \theta_d$ // compute final client $\theta$ gradients
        $\nabla_{\Gamma_d} \leftarrow \Gamma - \Gamma_d$ // compute final client $\Gamma$ gradients
        $\nabla_{\nu_d} \leftarrow \nu - \nu_d$ // compute final client $\nu$ gradients
        Return $\nabla_{\theta_d}$, $\nabla_{\nu_d}$ and $\nabla_{\Gamma_d}$ to the server
    end for
    $\nabla_\theta \leftarrow \frac{1}{D} \sum_d^D \nabla_{\theta_d}$ // calculate average $\theta$ gradients
    $\nabla_\Gamma \leftarrow \frac{1}{D} \sum_d^D \nabla_{\Gamma_d}$ // calculate average $\Gamma$ gradients
    $\nabla_\nu \leftarrow \frac{1}{D} \sum_d^D \nabla_{\nu_d}$ // calculate average $\nu$ gradients
    $\theta \leftarrow \theta + \nabla_\theta$
    $\phi \leftarrow \phi + \nabla_\phi$
end for
```

Our Reddit experiments use the “reddit-corpus-small“ dataset from ConvoKit (Chang et al., 2020), which includes posts from the top-100 subreddits over a set period of time. We filter the dataset to only include users with at least 50 utterances and contexts (subreddits) with at least 150 utterances. The resulting dataset has 326 characters, 30260 utterances, and 57 contexts.