Leveraging Similar Users for Personalized Language Modeling with Limited Data

Charles Welch[♣*] and **Chenxi Gu**[❀*] and **Jonathan K. Kummerfeld**[❀] and **Verónica Pérez-Rosas**[❀] and **Rada Mihalcea**[❀]

[♣] Conversational AI and Social Analytics (CAISA) Lab
Department of Mathematics and Computer Science, University of Marburg

[❀] Language and Information Technologies Lab (LIT)
Department of Computer Science and Engineering, University of Michigan

Abstract

Personalized language models are designed and trained to capture language patterns specific to individual users. This makes them more accurate at predicting what a user will write. However, when a new user joins a platform and not enough text is available, it is harder to build effective personalized language models. We propose a solution for this problem, using a model trained on users that are similar to a new user. In this paper, we explore strategies for finding the similarity between new users and existing ones and methods for using the data from existing users who are a good match. We further explore the trade-off between available data for new users and how well their language can be modeled.

1 Introduction

Recent work has suggested that there are several benefits to personalized models in natural language processing (NLP) over one-size-fits-all solutions: they are more accurate for individual users; they help us understand communities better; and they focus the attention of our evaluations on the end-user (Flek, 2020). Generation tasks in particular benefit from a personalized approach, for example, Dudy et al. (2021) argue that user intention is more often difficult to recover from the context alone.

We study personalization in language modeling, a core task in NLP. Direct applications of language models (LM) include predictive text, authorship attribution, and dialog systems used to model the style of an individual or profession (e.g., therapist, counselor). LMs are increasingly used as the backbone of models for a range of tasks in NLP, increasing the potential impact of personalization even further (Brown et al., 2020).

The standard non-personalized approach is to use pretrained models trained on a large volume of data written by many people. This approach does not take into account the differences between individuals and their language patterns. Given the same context, different people may act or write differently, but these general models cannot produce that type of variation. Approaches like fine-tuning can be used to tailor a pretrained model to an individual, but perform well only when enough data is available, which is often not the case.

Previous work on personalized and demographic word embeddings has seen successful application in downstream tasks. Garimella et al. (2017) look at location and gender and how they affect associations with words like "health" and many other stimulus words like "stack"– does it make you think of books or pancakes? Welch et al. (2020) discuss other associations, for instance, "embodying" an idea may more often refer to a religious or economic concept depending on your beliefs. Similarly, "wicked" may mean "evil" or may function as an intensifier depending on where you live (Bamman et al., 2014). These exemplify how personalized representations can help make distinctions in meaning, however, static representations have limitations. For example, Hofmann et al. (2021) find that in some contexts "testing" refers to seeing if a device works and "sanitation" refers to a pest control issue, while in another context both refer to conditions of the COVID-19 pandemic. Personalized LMs, or language models built to better predict what an individual will say, could better address these cases, as LMs learn dynamic encodings of words.

In this paper, we consider approaches to fine-tuning and interpolation that are novel in that they leverage data from similar users to boost personalized LM performance. We consider the case of users with a small number of available tokens and propose ways to (1) find similar users in our corpus and (2) leverage data from similar users to build a personalized LM for a new user. We explore the

*Authors contributed equally and work was performed while at the University of Michigan.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1742 - 1752

trade-offs between the amount of available data from existing users, the number of existing users and new users, and how our similarity metrics and methods scale. We then show an analysis to explore what types of words our method predicts more accurately and are thus more important to consider in personalization methods.

2 Related Work

Personalized Language Modeling. King and Cook (2020) examined methods for creating personalized LMs and their work is most similar to ours. They consider interpolating, fine-tuning, and priming LMs as methods of personalization, though they use these methods with a large generic model. In contrast, our work shows that performance can be improved by leveraging data from similar users. They also analyzed model adaptation for models trained on users with similar demographics, inspired by Lynn et al. (2017), who showed that these demographic factors could help model a variety of tasks, and found that personalized models perform better than those adapted from similar demographics. Shao et al. (2020) have also explored models for personalization but focused on handling OOV tokens.

Wu et al. (2020) proposed a framework to learn user embeddings from Reddit posts. Their user embeddings were built on the sentence embeddings generated by a BERT model. By using the learned user embeddings to predict gender, detect depression and classify MBTI personality, they concluded that their embeddings incorporate intrinsic attributes of users. In our work, user embeddings are learned in a different approach, and we focus on how to use similarity calculated from user embeddings to build better LMs.

Authorship Attribution. One of the tasks we consider as a means of computing similarity is authorship attribution, i.e., identifying the author of a document. Early work on this task used lexical features like word frequencies and word n-grams (Koppel et al., 2009; Stamatatos, 2009). As in Ge et al. (2016), we employ neural networks to model similarity between users and predict authorship.

Learning from Limited Data. Antonello et al. (2021) explored training a model to predict what data will be most informative for fine-tuning and select individual data points to improve language modeling. The similarity metrics that we derive are used to select data for fine-tuning in one of our methods of leveraging similar user data, however we consider indivisible sets of data grouped by author.

The cold start problem is a well-known problem in recommendation systems. A great amount of previous work addressed how to recommend items to new users, about whom the system has little or no history, often with a focus on matrix factorization methods (Zhou et al., 2011). Work from Huang et al. (2016) approached language modeling as a cold-start problem, in that they had no writing from a user, though they had a social network, from which they interpolated LMs from users linked in their social graph.

Language Models. We use a recently developed LM that has received widespread attention (Merity et al., 2018b). The LSTM-based model combines a number of regularization and optimization techniques explored in recent literature, including averaged SGD, embedding dropout, and recurrent dropout. Subsequent work has developed variations of the model with improved perplexity, but these take at least twice as much time to train (Gong et al., 2018), making them less practical for the user-specific experiments we consider.

Another direction of research has shown impressive results using extremely large models (Radford et al., 2019; Devlin et al., 2019). Using these as a basis for experiments could be an interesting direction, but fine-tuning models in low data settings is known to be difficult and highly variable (Dodge et al., 2020). Similar transformer models have been used for controlled generation. Zellers et al. (2019) developed a model for news generation that conditioned on meta-data including domain, date, authors, and headline. No ablation is performed, and though it would be interesting to compare to a transformer method that conditions on authors alone, we opted for a model that is faster and cheaper to train (Grover-Mega from Zellers et al. (2019) was trained for two weeks and cost around 25k USD). Additionally, when fine-tuning models for new users, little data is available. Contextualized embedding models often require a large amount of data to train effectively, though this type of comparison would be an interesting future direction to explore. Variations of the LSTM have consistently achieved state-of-the-art performance without massive compute resources and thus we chose this architecture for our experiments (Merity et al., 2018a; Melis et al., 2019; Merity, 2019; Li et al., 2020).

Rule	Example
(1) it contains more than 20 tokens but the average token length is less than 3	" i " " " " w " " i " " l " " l " " " " " n " " e " " v " " e " " r " " " " g " " i " " v " " e "
(2) it contains a long token whose length is greater than 30	COOLCOOLCOOLCOOLCOOLCOOLCOOLCOOL... (There is usually duplication inside this kind of post)
(3) it contains less than 8 tokens among which more than 3 are URLs	URL URL URL URL
(4) it contains more than 3 math related symbols, such as "|", "+" and "="	before humanity , maybe 2 +2 = 5 . no , before humanity 2 +2 = 4 did not exist .
(5) it contains symbols like "{", "}" and "( )" with only white spaces in the parentheses	we specialize in ( ) ( ) ( ) ( ) ( ) ( ) ( )
(6) it contains less than 5 tokens and the last token is "*"	(This kind of post is usually a spelling correction to a previous post)
(7) there are less than 4 unique tokens in every sequence of 8 adjacent tokens	w , w , w , w , would n't it be better if we just bend over and follow their rules ?
(8) it contains hashtags, indicated by: [ ] (/ / #	[** if i were a rich man ... **] (/ / #ggj)
(9) it is a duplicate of another post in the user's data	
(10) more than 60% of the characters are non-alphabetical.	=+=+= 1st =+=+= 2nd =+=+= End

Table 1: Examples of rules for filtering posts as described in Section 3.1.

3 Dataset

We examine a corpus of publicly available Reddit comments and select users active on Reddit between the years of 2007-2015 who have at least 60k tokens of text.[1] We refer to the existing users with at least 250k tokens of text as **anchor** users. These are users that are leveraged through interpolation or fine-tuning in order to improve performance on **new** users. Reddit posts are mostly in English.

We experiment with two settings: In the **small anchor** setting, there are 100 *anchor* users, with a 200k, 25k, 25k split for training, validation, and test, and 50 *new* users, with 2k tokens for training, and 25k for each of validation and test. In the **large anchor** setting, there are 10k *anchor* users and 100 *new* users, each having 2k tokens for training and validation and 20k for test.

Preprocessing Reddit data can be noisy, containing URLs, structured content (e.g., tables, lists), Subreddit-specific emoticons, generated, or deleted content. We first extract all posts for each user in our dataset. During this process we remove noisy posts, where a post is considered "noisy" if it matches one of ten rules. These rules and examples of each are shown in Table 1. After this filtering step, we remove markup for emojis and hyperlinks from the remaining posts (keeping the posts themselves). We take these steps to ensure that we capture language used by the authors, rather than reposts, collections of links, ASCII tables and art, equations, or code. Tokens that occur fewer than 5 times are replaced with ⟨UNK⟩, which results in a vocab size of 55k for the small anchor set and 167k for the larger one.

[1]Posts are retrieved from https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/ and we exclude known bots and do not include posts in the /r/counting subreddit in our dataset.

4 Experiments

Our method for constructing personalized LMs consists of a similarity metric and a method for leveraging similar user data to train a personalized LM. The similarity metric measures which anchor users are most similar to a new user. That is, given a set of users ($anchors$), a new user (n), and a similarity function (sim), we compute $z = sim(n, anchors); z \subset anchors$ to get a set of similar users z. We explore three similarity metrics and two methods of applying them to the construction of personalized models. Figure 1 shows how user data is used for each step.

4.1 Calculating User Similarity

We explore three methods for measuring the similarity between users. Two of them, authorship confusion and user embeddings, are derived from classifiers trained for other tasks, while the third, perplexity-based similarity, is obtained from the performance of LMs on the new user. The user embedding method results in a vector space where we can use cosine similarity to measure the distance between individuals. The perplexity directly gives a distance between each pair and the authorship confusion vectors can be treated as a vector of continuous values where each value represents the similarity to an anchor user.

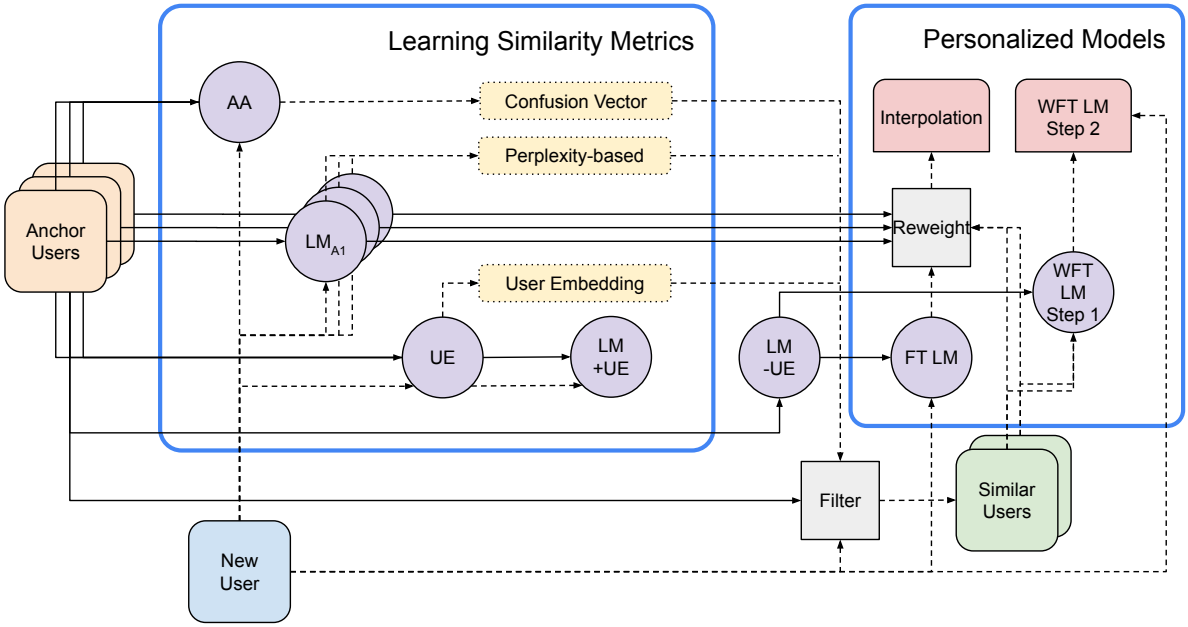

Figure 1: This diagram shows how data, models, and metrics are used in this paper. There are two main sections, a rectangle on the left showing how the three similarity metrics are computed, and a rectangle on the right showing our two methods of leveraging similar user data to create personalized models. The solid lines indicate the flow of anchor user data, while a dashed line indicates data from a new user. Anchor user data is used to create the authorship attribution model (AA), the individual user LMs for the perplexity-based metric (denoted as a set with the first as LM_{A1}), and the user embeddings (UE). The three metrics can be used to filter anchor user data to find similar users. With these users, we fine-tune a baseline LM (without UE, denoted LM-UE), which is then further fine-tuned with new user data for the weighted fine-tuning method (WFT LM). When interpolating, the individual anchor user LMs are reweighted, and combined with the predictions of an LM fine-tuned on new user data (FT LM).

Authorship Attribution Confusion (AA). Similarity can be measured from the confusion matrix of an authorship attribution model. This model takes a post as input and encodes it with an LSTM (Hochreiter and Schmidhuber, 1997). The final state is passed to a feed-forward layer and then a softmax to get a distribution over authors. We denote this model A, and $A(U)$ as the class distribution output by the model for a given utterance set. For a new user, we take their set of utterances, U_n and pass them to our model $A(U_n)$ which will give us a confusion vector of length K, one value for each author.

We train this model on the data from anchor users.[2] Embeddings are initialized with 200d GloVe vectors pretrained on 6 billion tokens from randomly sampled Reddit posts (Pennington et al., 2014). For $K = 100$ anchors the test accuracy is 42.88% and $K = 10,000$ the test accuracy is 2.42%. These accuracies are reasonably high given the difficulty of the task.[3] The classifier does not have to be high performing given our application to computing a user similarity metric.

We apply this model to each post in the training data from *new* users. The scores produced by the model for each new post indicate which of the anchor users has the most similar writing. The more frequently posts from a new user are predicted as coming from a specific anchor user, the more similar this anchor user is to the new user.

User Embeddings (UE). We first train an LM with a user embedding layer on the data from anchor users. The model is adapted from Merity et al. (2018b) with an added user embedding layer. This token embedding layer is initialized with our pretrained GloVe vectors and frozen during training. The output of the LSTM layer is concatenated to the user embedding at each time step based on the author of the token at that time step.[4] Note that this is then passed through another feed-forward layer before being used for prediction. Our optimizer starts with SGD and will switch to ASGD if there

[2] See Appendix A for hyperparameters

[3] Note that when $K = 10,000$ the majority class is 0.01%.

[4] See Appendix B for hyperparameters

is no improvement in validation loss in the past 5 epochs (Polyak and Juditsky, 1992). We removed continuous cache pointers (Grave et al., 2016) to speed up training. For $K = 100$, the validation perplexity converges to 59.06 and test perplexity is 58.86. When training with $K = 10,000$ the validation perplexity converges to 88.71 with test perplexity 88.54.

The embeddings of anchor users can be obtained from the user embedding layer in the trained model. To learn the embeddings of new users, we freeze all parameters of the trained model except the user embedding layer. We train the model on the data from each new user separately with the same training strategy. It takes 2 minutes to learn the embedding of each new user. The average test perplexity is 66.67 when $K = 100$ and 90.48 when $K = 10,000$. For each pair of new user and anchor user, we use the cosine similarity between two embeddings as the similarity.

Perplexity-Based (PPLB). Given N trained LMs, one for each user, we can then use the perplexity of one LM on another user's data as a measure of distance. We could compare the word-level distributions, though this would be very computationally expensive. In our experiments, we use the probability of the correct words only, or the perplexity of each model on each new user's data.

We take the large LM trained on all anchor users, as described in the user embedding section and fine-tune it for each anchor user. We then measure the perplexity of each model on the data of each new user. For this matrix of $new \times anchor$ perplexities, we turn each row, representing a new user, into a similarity vector by computing $1 - \frac{c - min(row)}{max(row)}$ for each cell, c. This step is expensive, taking close to 24 hours for $K = 100$ and intractable given our hardware constraints in the $K = 10,000$ setting.

5 Leveraging Similar Users

Our three similarity methods provide a way to identify anchor users with the most relevant data for a new user. In this section, we describe two methods to learn from that data to construct a personalized model.

5.1 Weighted Sample Fine-tuning

Users who speak in a similar style or about similar content may be harder to distinguish from each other and should then be more similar. For a given similarity metric, we compute similar users and use data from these users to fine-tune an LM before fine-tuning for the new user.

We compare to two baselines, (1) a model trained on all anchor users with no fine-tuning and (2) a model trained on all anchor users that is fine-tuned on the new user's data, as is done in standard fine-tuning. Our method of weighted sample fine-tuning has two steps. The first step is to fine-tune the model trained on all anchor users on a new set of similar users, as determined by our chosen similarity metric. Then we fine-tune as in the standard case, by tuning on the new user's data.

5.2 Interpolation Model

Our interpolation model is built from individual LMs constructed for each anchor user. It takes the predictions of each anchor user model and weights their predictions by that anchor's similarity to the new user. No model updates are done in this step, which makes it immediately applicable, without requiring further training, even if the aggregation of output from all anchor models is more resource intensive.

We also want to incorporate the predictions of the model fine-tuned on the new user data with the predictions of models trained on similar anchor users. We define a set of similar anchor users, σ, each of which has a similarity to the new user, n. We vary s for each similarity function. The weight to give the new user fine-tuned model is η, and we interpolate as follows for a given resulting probability p_r, of a word, w:

$$p_r(w|\cdot) = \eta p_n(w|\cdot) + (1-\eta) \sum_{i \in \sigma} s(\sigma_i, n) p_{\sigma_i}(w|\cdot)$$

The similarities are adjusted to the range $(0, 1)$ and normalized to sum to one.

6 Results

We divide our results into separate subsections for each of the anchor sets. On the small anchor set we were able to perform more exploration of the weighted fine-tuning method, as it does not scale as well to the large anchor set.

We present results using standard perplexity measurements as a function of the probability of a correct prediction of a token. We also present results with accuracy at N, where a prediction is counted as correct if the correct token occurs within the top N most probable words given by the model.

Method	#Sim. Users	Δ Perplexity UE	Δ Perplexity AA	Δ Perplexity PPLB	Δ Accuracy@1 UE	Δ Accuracy@1 AA	Δ Accuracy@1 PPLB
Weighted Fine-tuning	5	0.276	1.728	0.627	0.159	0.155	0.148
Interpolation	100	-2.055	-2.415	-1.992	0.249	**0.277**	0.223
Interpolation	50	-2.163	-2.415	-2.043	0.260	**0.277**	0.204
Interpolation	25	-2.242	-2.415	-2.022	0.248	**0.277**	0.232
Interpolation	10	-2.286	**-2.435**	-2.183	0.235	0.260	0.249

Table 2: Difference in perplexity for our interpolated model and weighted fine-tuning results on the **small anchor** set. The baseline metrics are subtracted from our model, meaning that more negative perplexity and more positive accuracy are better. The baseline Merity et al. (2018a) perplexity average is 64.3 for a model that uses standard fine-tuning and 67.6 without fine-tuning. Bold indicates best performance.

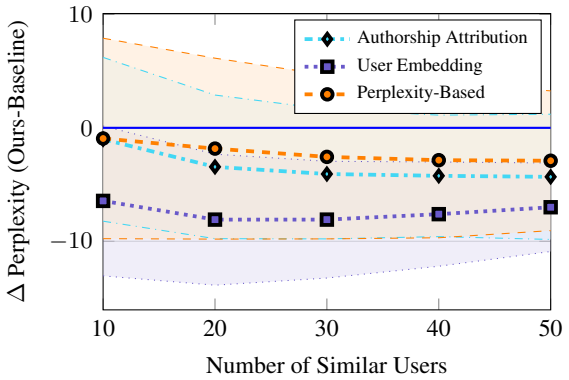

Figure 2: Change in perplexity for varying number of similar users considered in weighted fine-tuning for the three similarity metrics.

6.1 Small Anchor Set

In this section, we compare our weighted sample fine-tuning and interpolation approaches to the more standard fine-tuning, where a large pretrained model is fine-tuned only on the new user's data. With no fine-tuning our LM achieves a perplexity of 67.6 and when fine-tuning on the new user only, this perplexity drops to 64.3. For weighted fine-tuning, we attempt to fine-tune the large pretrained model on 100 anchors using our two step method, first fine-tuning on a million tokens from most similar users, and then fine-tuning on new user data. Through tuning the number of similar users, we found 5 worked best. For the interpolation model, we found more similar users improved accuracy, though perplexity was slightly higher for ten similar users. Our interpolation model combines predictions from similar anchor user LMs. We have an LM fine-tuned to each of our anchor users and for a given new user we predict words by weighting the predictions of the models representing the most similar users.

Results in Table 2 show that our weighted sample fine-tuning is not able to outperform the baseline for any of our three similarity metrics. Perplexity and accuracy results are reported averaged over the test set users. We also tried fine-tuning with random user's data and found that this performance was better than no fine-tuning but worse than fine-tuning on new user data only, showing that there is no added benefit from simply continuing to fine-tune on all data.

For the interpolation model, we tune η (see Section 5.2) on a held-out set and use a value of 0.7. The results show that the authorship attribution similarity performs best on both metrics. We find that as the number of similar users increases it has little effect past around ten similar users, as the similarity weights decrease and have a smaller impact.

Retraining with Similar User Data: It appears that having similar user data does not help the weighted fine-tuning model. To further investigate this we looked at settings where the amount of training data is fixed, but the source is either random, or a sample of similar user's data. For each new user, we build six datasets: a random dataset and five datasets consisting of data from top-k similar anchor users for this new user where k is in {10, 20, 30, 40, 50}. Each of these datasets has 2m tokens. The random dataset is comprised of 20k tokens from each anchor user. For the dataset built from the top-k similar users, we want the number of tokens selected from each anchor user to be proportional to the similarity between the new user and each anchor user. To do this, we normalize the three similarities by subtracting the minimum and dividing by the maximum such that they are between zero and one.

For a given set of k users and similarity metric, we sort all anchor users in descending order by their similarity to the new user and choose the top k anchor users. For the rank 1 anchor user a_1, we choose the following number of tokens from the

#Sim. Users	Δ Perplexity	Std.Dev.
Random 10	0.176	0.367
10	-0.354	0.659
20	-0.534	0.977
30	-0.673	1.080
40	-0.714	1.040
50	-0.803	1.127
100	-0.941	1.351
150	-0.986	1.560
200	-1.069	1.549

Table 3: Difference in perplexity for fine-tuning varying number of similar users on the **large anchor** set, first fine-tuning on similar users, and second on the new user's data, as compared to Merity et al. (2018a) fine-tuned on new user data only with perplexity 89.7. Each similar user has 2k tokens and each new user has 2k.

training data, where $s(\cdot, \cdot)$ is the similarity between a pair of users:

$$n_{a_1} = 2000k * \frac{s(newuser, a_1)}{\sum_{i=1}^{k} s(newuser, a_i)}$$

If $n_{a_1} > 200k$, we choose $n_{a_1} = 200k$. For the rank x anchor user a_x, we choose

$$n_{a_x} = (2000k - \sum_{j=1}^{x-1} n_{a_j}) * \frac{s(newuser, a_x)}{\sum_{i=x}^{k} s(newuser, a_i)}$$

tokens from their training data. If $n_{a_x} > 200k$, we choose $n_{a_x} = 200k$. We repeat this procedure until the rank k anchor user. The ratio of similarities in this equation enforces that the amount of data we select from each of the top-k similar users is proportional to their similarity.

We then train a separate model on each dataset. The architecture of the model is the same as what is described in Section 4.1 except that it does not have a user embedding layer. We then fine-tune the trained models on the training data of the new user.

For a chosen similarity metric and number k, we average the test perplexity of the fine-tuned models for all new users and subtract from it the average test perplexity of the fine-tuned models trained on random datasets, whose average perplexity is 111.0. The results are shown in Figure 2 with shaded areas indicating standard deviation. In the figure, the lower a point is, the better the datasets built using the corresponding similarity metric and number k is for training an LM for new users, which we infer is because the weighted sample datasets are closer to the data from new users.

We see that in terms of similarity metrics, the user embedding is the best while perplexity-based is the worst. As k increases, the performance first increases then decreases. The best performance is achieved when using the similarities calculated with user embeddings and using top 20 or 30 similar anchor users. After that, including more users has little effect, as their similarity weights continue to decrease. The main takeaway from this experiment is that although similar user data helps more than random data, the benefit does not transfer to the larger fine-tuning scenario. This area may be worth further exploring for fine-tuning strategies or for training data selection in applications where new models must be trained.

6.2 Large Anchor Set

In a set of only one hundred anchor users, it may be the case that existing users are not similar enough to the new user to benefit from our approach. To test this idea we ran experiments using the larger set of 10k anchor users and 100 new users.

Taking our most promising user embedding similarity metric from the weighted sample fine-tuning, we tested this method's performance varying the number of similar users. Our results in Table 3 show a reduction in perplexity of 0.94 at 100 similar users and over one point at 200 users. There is a logarithmic improvement with the number of similar users considered, as we would expect more dissimilar users to be less informative. The results in this table suggest that the anchor set must be diverse enough to contain similar users to new users, in order to benefit from this method.

We also try the interpolation model with a larger set of anchor users. Our base model is trained on 10k anchor users and 2k tokens from each anchor. Note that we are controlling for the total points from anchor users, using 100 times fewer points per user and 100 times more users. Scaling up these experiments to more points and users is computationally expensive but may be worth exploring in future work. We fine-tune this model to each similar anchor user for weighting predictions. On a held-out set we tune η and find that in this setting performance starts to drop after around 10 similar users. It is computationally expensive to run each of the 10k models on each new user. The perplexity similarity metric requires that all of these are run in order to determine similarity and thus is not scalable to the large anchor user setting. The user embedding metric scales better because similarity can be determined by tuning an existing LM on

#Sim. Users	2k per Anchor Δ PPL	Acc @1	Acc @3	Acc @5	Acc @10	6k per Anchor Δ PPL	Acc @1	Acc @3	Acc @5	Acc @10
10	-0.692	0.097	0.111	0.100	0.058	**-11.726**	**0.497**	**0.697**	**0.723**	**0.718**
5	-0.615	0.091	0.103	0.090	0.049	-11.463	0.491	0.656	0.694	0.705
4	-0.590	0.088	0.091	0.079	0.045	-11.287	0.486	0.650	0.677	0.684
3	-0.553	0.084	0.087	0.072	0.039	-11.001	0.457	0.622	0.657	0.654
2	-0.415	0.084	0.060	0.052	0.033	-10.604	0.439	0.588	0.602	0.617
1	-0.006	0.047	0.016	0.002	-0.002	-8.866	0.282	0.423	0.485	0.516

Table 4: Comparison of our interpolated user embedding similarity model on the **large anchor** set to a standard fine-tuned Merity et al. (2018a) baseline measured in perplexity and accuracy @N. We show results for 2k and 6k tokens per anchor user, showing improved performance when more data per anchor is available. Bold indicates best performance.

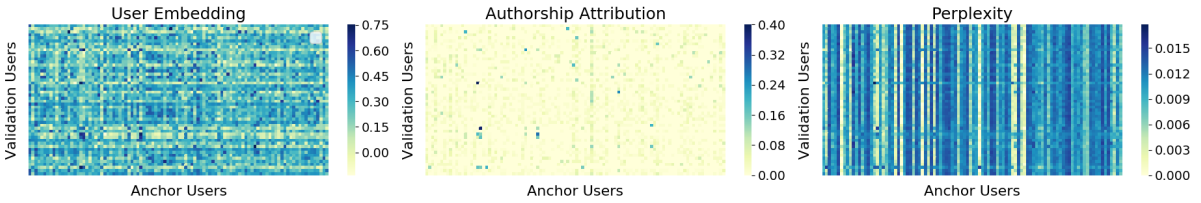

Figure 3: Heat maps showing normalized similarity for each metric on our 100 author anchor set.

Metric 1	Metric 2	Pearson's r	Spearman's ρ
UE	AA	0.360	0.362
UE	PPL	0.280	0.316
PPL	AA	0.073	0.025

Table 5: Spearman and Pearson correlation coefficients for each pair of similarity metrics (User Embeddings (UE), Authorship Attribution (AA), and Perplexity (PPL)) computed for each of our 100 anchor users similarity to each new user.

new user data. For ten similar users we require 1,000 times fewer computations than we would to weight all 10k users. We found that authorship attribution performed much worse in this setting, as the confusion matrix becomes very sparse.

The results for our best similarity metric, user embeddings, are shown in Table 4. On the left we see performance for our model on the larger set containing 2k tokens per anchor user. For this analysis of our best, scalable model, we include accuracy @N, a metric denoting the percentage of times the correct word was in the top-N most probable choices. This is comparable to Table 3, where we used the same amount of data for the weighted sample fine-tuning approach. On the right we see performance when the amount of data per anchor user is tripled. The baseline and fine-tuned models all benefit from this additional data, however we find that the difference in perplexity is much larger, as having additional data will allow the models to learn more accurate similarity metrics. We also find that when tuning η it tends toward 0.6 when there are 2k tokens per anchor user but 0.3 when there are 6k. As the amount of data from the anchor users increases, the optimal interpolation weights shift to weight the anchor user models more heavily than the model fine-tuned on the new user. How the tuning of η could be done on a per-user basis, rather than globally, is an interesting open question.

7 Analysis

7.1 Differences in Similarity Functions

We looked at the differences between our three similarity functions by computing the correlation coefficients for Spearman's ρ and Pearson's r in Table 5. Interestingly, the perplexity and authorship attribution metrics correlate much more strongly with the user embedding metric than with each other. It is possible that the user embedding metric performs best in our experiments because it contains more of the useful information from both of the other metrics. Additional heat maps for each metric are in Figure 3. In general, they show that the three metrics seem to capture different information about the relationships between users. The user embedding metric leads to more evenly distributed similarities, while the other two metrics have outlier anchor users that show stronger correlation with a subset of the new users.

fuels, qaeda, zealand, inte, al., antonio, facto, neutrality, kong, differ, olds, custody, cruise, obligation, arts, beck, guise, scrolls, vegas, mph, dame, conclusions, laden, pedestal, throne, ck, charm, occasions, disorders, correctness, disposal, capita, hominem, floyd, thrones, sarcastic, ghz, explorer, comprehension, standpoint, ambulance, noting, diego, accusations, cares, forth, enforcement, amp, nukem, convicted

Table 6: Top 50 words for which our best model outperforms the baseline based on the frequency of word correctly predicted normalized by the word's total frequency.

7.2 Personalized Words

We take the highest performing model using user embedding similarity trained on our large anchor user set and compare it to our baseline model to look at which words are more accurately predicted. By taking the number of times each word is correctly predicted by the best model when the baseline was wrong and dividing by the total number of occurrences of that word in our language modeling data, we can find words that have the highest normalized frequency of being improved by our model.

The top 50 words for which we see improvement are shown in Table 6. We see the second word of many two-word proper nouns in this set. Many names can start with "San" or "Las" and so we see "vegas", "diego", and "antonio", in this list. Similarly, "new" precedes "zealand" and other location names. The top word is "fuels", which occurs often in the data in conversation about "fossil fuels", though there are also many others that mention other kinds of fuels, or use "fuels" as a verb, as in "it fuels outrage". We also see that units such as "mph" or "ghz" are more accurately predicted. The units that one chooses may be more common depending on where one lives, or in the case of "ghz" it may depend more on the subject matter that a user is familiar with or tends to talk about. Other proper nouns such as "game of thrones", or "hong kong" vs. "donkey kong", contain common words, which individually may be hard to predict, but with knowledge of an individual's preferences could be predicted more accurately.

8 Ethical Considerations

Work on personalized LMs could be used for surveillance by detecting language from individuals or groups (Stamatatos, 2009). We recommend against such applications, as they threaten intellectual freedom and risk discrimination (Richards, 2013). There may be a risk in storing private data necessary to construct these models, as data may not be properly secured or used. Furthermore, a personalized model could reinforce incorrect language usage, which may be an issue for individuals learning to speak a new language, making it more difficult to learn. Learning personal language patterns in a given context and suggesting these patterns in other contexts may lead to potentially incorrect or offensive results and we recommend that if this type of personalization is deemed appropriate, users are made aware of how their data is being used and potential consequences.

9 Conclusions

In this paper, we addressed the issue of language modeling in a low data setting where a new user may not have enough data to train a personalized LM and presented a novel approach that leverages data from similar users. We considered three similarity metrics and two methods of leveraging data from similar *anchor* users to improve the performance of language modeling over a standard fine-tuning baseline, and showed how our results vary with the amount of data available for anchor users and the number of available anchor users. We found that the most easily scalable and highest performing method was to use user embedding similarity and to interpolate similar user fine-tuned models. Additionally, we provided an analysis of the kind of words that our personalized models are able to more accurately predict and further discussed limitations of our methods.

Acknowledgments

This material is based in part on work supported by the NSF (grant #1815291) and the John Templeton Foundation (grant #61156). Any opinions, findings, conclusions, or recommendations in this material are those of the authors and do not necessarily reflect the views of the NSF or the John Templeton Foundation. Clover icons taken from Freepik at flaticon.com.

References

Richard Antonello, Nicole Beckage, Javier Turek, and Alexander Huth. 2021. Selecting informative contexts improves language model fine-tuning. 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)*.

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Shiran Dudy, Steven Bedrick, and Bonnie Webber. 2021. Refocusing on relevance: Personalization in NLG. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Lucie Flek. 2020. Returning the N to NLP: Towards contextually personalized classification models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Aparna Garimella, Carmen Banea, and Rada Mihalcea. 2017. Demographic-aware word associations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Zhenhao Ge, Yufang Sun, and Mark J. T. Smith. 2016. Authorship Attribution Using a Neural Network Language Model. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: frequency-agnostic word representation. In *Advances in Neural Information Processing Systems*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Dynamic contextualized word embeddings. 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)*.

Yu-Yang Huang, Rui Yan, Tsung-Ting Kuo, and Shou-De Lin. 2016. Enriching cold start personalized language model using social network information. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 21, Number 1, June 2016*.

Milton King and Paul Cook. 2020. Evaluating approaches to personalizing language models. In *Proceedings of the 12th Language Resources and Evaluation Conference*.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26.

Yinqiao Li, Chi Hu, Yuhao Zhang, Nuo Xu, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and Changliang Li. 2020. Learning architectures from an extended search space for language modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2017. Human centered NLP with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Gábor Melis, Tomáš Kočiskỳ, and Phil Blunsom. 2019. Mogrifier LSTM. In *International Conference on Learning Representations*.

Stephen Merity. 2019. Single headed attention RNN: Stop thinking with your head. *arXiv preprint arXiv:1911.11423*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018a. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018b. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Neil M Richards. 2013. The dangers of surveillance. *Harv. L. Rev.*, 126.

Liqun Shao, Sahitya Mantravadi, Tom Manzini, Alejandro Buendia, Manon Knoertzer, Soundar Srinivasan, and Chris Quirk. 2020. Examination and extension of strategies for improving personalized language modeling via interpolation. In *Proceedings of the First Workshop on Natural Language Interfaces*.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Compositional demographic word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xiaodong Wu, Weizhe Lin, Zhilin Wang, and Elena Rastorgueva. 2020. Author2Vec: A Framework for Generating User Embedding. *arXiv e-prints*, page arXiv:2003.11627.

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

Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*.

A Hyperparameters for Authorship Attribution Model

- Bidirectional LSTM layers=3
- LSTM hidden dim=400
- output dropout=0.5
- fully-connected layer dim=$800 \times K$
- Adam optimizer
- cross-entropy loss
- learning rate=1e-3
- batch size=64
- early stopping if no improvement over 10 epochs

B Hyperparameters for User Embedding Model

- scalar dropout=0.1
- embedding dropout=0.2
- LSTM layers=3
- LSTM hidden dim=1,150
- recurrent dropout=0.2
- user embedding dim=50 (tried 20,50,100 but 50 worked best)
- cross-entropy loss
- early stopping if no improvement over 20 epochs
- sequence length=70
- batch size=20
- learning rate=3
- parameter clipping=0.25

C Running Times

Authorship attribution models are trained on an NVIDIA GeForce RTX-2080Ti GPU and take 2.5 hours for $K = 100$ anchors and 4 hours for $K = 10,000$ anchors.

Training a new language model for weighted fine-tuning as described in Section 6.1 takes about 2.5 hours to train a model on a dataset on an NVIDIA Tesla V100 GPU. Fine-tuning the trained models on the training data of the new user takes about one minute on average.

The user embedding models are trained on an NVIDIA GeForce RTX-2080Ti GPU. For $K = 100$ anchors, it took 132 hours. When training with $K = 10,000$, we reduced the hidden LSTM size to 500, which reduced training time to 112 hours.