Federated Learning with Noisy User Feedback

Rahul Sharma∗
rahul.sharma@usc.edu
Anil Ramakrishna∗
aniramak@amazon.com
Ansel MacLaughlin
ammaclau@amazon.com
Anna Rumshisky
arum@cs.uml.edu
Jimit Majmudar
mjimit@amazon.com
Clement Chung
chungcle@amazon.com
Salman Avestimehr
avestime@usc.edu
Rahul Gupta
gupra@amazon.com

Abstract

Machine Learning (ML) systems are getting increasingly popular, and drive more and more applications and services in our daily life. This has led to growing concerns over user privacy, since human interaction data typically needs to be transmitted to the cloud in order to train and improve such systems. Federated learning (FL) has recently emerged as a method for training ML models on edge devices using sensitive user data and is seen as a way to mitigate concerns over data privacy. However, since ML models are most commonly trained with label supervision, we need a way to extract labels on edge to make FL viable. In this work, we propose a strategy for training FL models using positive and negative user feedback. We also design a novel framework to study different noise patterns in user feedback, and explore how well standard noise-robust objectives can help mitigate this noise when training models in a federated setting. We evaluate our proposed training setup through detailed experiments on two text classification datasets and analyze the effects of varying levels of user reliability and feedback noise on model performance. We show that our method improves substantially over a self-training baseline, achieving performance closer to models trained with full supervision.

1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) have become increasingly common in modern society with applications ranging from simple standalone products to complex modules Kaplan and Haenlein (2019). However, this rise has also created growing privacy concerns Papernot et al. (2016); Yeom et al. (2018). Such concerns may affect user willingness the adapt new technologies Guhr et al. (2020). In response, many government agencies across the world have introduced regulations to protect the data-handling rights of their citizens, such as the European Union’s GDPR General Data Protection Regulation and California’s CCPA California Consumer Privacy Act.

Federated Learning (FL) is a step in this direction to improve consumer trust, where models are trained without moving data out of client devices. The typical FL approach is to iteratively train local models on edge devices and then propagate them back to a central node in order to update the global model. Most commonly, this is done using Federated Averaging (FedAvg) McMahan et al. (2017), where we take a simple average over the client model parameters. However, in order to update local models on the edge, this setup assumes the presence of labeled user data on each device, which is often not possible. Most prior works do not address this problem, but simulate fully-supervised federated learning by distributing existing labeled datasets across edge devices. In this work, we consider a more realistic scenario, where labels are not available on device. Rather than turning to unsupervised learning as seen in Hard et al. (2018), we instead propose a novel setup to leverage user feedback in order to train the FL model.

In many real world AI applications with direct or indirect human interaction, it is possible to collect either explicit user feedback (e.g., using a voice or a screen-based prompt) or implicit feedback in the form of behavioral cues. For an example of implicit feedback, consider a user interacting with a virtual AI assistant (such as Alexa), who asks to play the song ‘Bohemian Rhapsody’ from the band Queen. The virtual assistant would interpret the prompt and select a song from its library to play. If the user lets the music play without interruption, this can be viewed as positive feedback, suggesting that the underlying model interpreted the request correctly. On the other hand, if the user interrupts the music and makes a repeat (or different) request, this can be viewed as negative feedback, suggesting that the underlying model prediction was incorrect.
In this work, we propose to leverage such feedback in federated model training.

**Leveraging Positive and Negative Feedback** In our proposed setup, we first train a seed model on a small amount of labeled data on a central node. This mimics the real-world scenario where a small amount of data can be collected and annotated to bootstrap an initial model. We then propagate this weak seed model to each of the clients. On the edge, we use this seed model to make predictions for each user’s request. Since the model is trained with limited data, these predictions may be incorrect. To further improve this model performance, we leverage user feedback as an indirect indicator of the predicted label’s quality. Since positive user feedback likely indicates that a model prediction is correct, we label examples with positive feedback with the seed model’s prediction and add them to the training data. This mirrors the standard self-training setup Rosenberg et al. (2005), where weak models are further trained on a set of their own predictions. When a user gives negative feedback, however, we cannot assign a label to the example, since we only know that the seed model’s prediction is wrong. We instead treat these prediction as complementary labels Ishida et al. (2017); Yu et al. (2018) and extend federated model training to use such labels.

**Modeling Feedback Noise** In realistic human interactions, however, the user may not always provide consistent feedback, making user feedback signal noisy. In the virtual AI assistant example above, if the model predicts a different song from the same band, the user may choose to continue listening without interruption. Similarly, even if the model predicts the correct song, the user may change their mind once the song starts playing and interrupt with a new request. Such user behavior will introduce noise into the feedback signal. In order to assess typical levels of such noise in user feedback, we conduct a pilot study on Amazon Mechanical Turk (Mturk), and evaluate the accuracy of feedback from Mturk users on two different text classification problems. Based on this study, we define a model of user noise defined by two parameters that specify how likely they are to give accurate feedback on both correct and incorrect predictions by the seed model. With this model of user behaviour, we then study the effects of user noise on model performance. We conduct extensive experiments on two text classification datasets, training FL models on feedback data with varying amounts of user noise simulated using our model. We further experiment with various noise-robustness strategies to mitigate the effect of such noisy labels and present promising results.

The key contributions in this paper are as follows:

1. We propose a new framework for modeling and leveraging user feedback in FL, and present a strategy to train supervised FL models directly on positive and negative user feedback. We show that, under mild to moderate noise conditions, incorporating feedback improves model performance over self-supervised baselines.

2. We propose a novel model of user feedback noise and study the effects of varying levels of this noise on model performance.

3. We study the effectiveness of existing noise-robustness techniques to mitigate the effects of user-feedback noise and identify promising directions for future exploration.

**2 Related Work**

**2.1 Federated Learning**

Federated Learning McMahan et al. (2017) has recently seen a rise in popularity in a number of domains, including natural language processing (NLP) Yang et al. (2018); Ramaswamy et al. (2019); Hard et al. (2018). This is due to growing privacy concerns Papernot et al. (2016); Geyer et al. (2017); Truex et al. (2019), abundance of unlabeled data, and an increase in the computational capacity on edge devices. However, availability of labels on edge (or rather, lack thereof) limits the practical application of FL in most real-world use cases. In this work, we present an extension to FL and show improvements in federated model performance by leveraging user feedback. Recent works have also revealed risks of information leakage from gradients in federated learning, and several techniques have been developed to mitigate this risk (see Zhu et al. (2019), Lyu et al. (2020) and the references there in).

**2.2 Learning With User Feedback**

User feedback on model behavior provides learning signals which can be leveraged to continuously improve model performance. Using feedback signals
for model training provides robustness to concept and data drifts, as new data is always accompanied with new feedback labels from which to learn. Learning methods that leverage user feedback have been applied to a variety of tasks in NLP, such as semantic parsing [Iyer et al. (2017)], machine translation [Kreutzer et al. (2018)] and question answering [Kratzwald and Feuerriegel (2019)]. To our knowledge, however, our work is the first to build a parametric model of user feedback noise and to study how to train federated learning algorithms with noisy user feedback.

2.3 Negative Label Learning

Standard supervised learning operates on data labeled with their true classes. Feedback data from users, however, can also be negative, indicating that the predicted class is wrong. Since the correct class of examples with such negative-feedback is unknown, our proposed method must be able to handle such ambiguity during training. In our work, we label examples with negative feedback with a complementary label corresponding to the predicted class [Ishida et al., 2017]. Complementary labels simply specify the category that a given example does not belong to. In our work, we follow the setup of [Yu et al. (2018)], who propose loss functions to train neural models on biased complementary labels.

2.4 Noise-Robust Learning

When training models on labels derived from noisy user feedback signals, it is important to use a strategy to mitigate the effects of label noise on model performance. One straightforward approach is to use a noise-robust loss function, such as Reverse Cross Entropy [Wang et al. (2019)] or Mean Absolute Error [Ghosh et al. (2017)]. In our work we follow the noise-robust learning setup of [Ma et al. (2020)], who present a training strategy that combines two robust loss functions (one active, one passive) to better handle label noise.

3 Modeling User Feedback

We propose a general framework for leveraging user feedback in federated learning. We use text classification as an exemplar task to evaluate this framework, but the proposed setup can be easily applied to other tasks. We use two benchmark text classification datasets: the Stanford Sentiment Treebank dataset (sst2) and the 20 newsgroups dataset (20news). The sst2 dataset comprises of 11,855 phrases from movie reviews and the corresponding binary sentiment labels. The 20 newsgroup dataset 20news consists of 18k news articles and headers, organized into 20 classes.

3.1 Pilot Study: Real World User Feedback

To understand the dynamics of user feedback noise, we conduct a pilot study using Amazon Mechanical Turk (Mturk). We use text classification on the above two datasets, sst_2 and 20news, as the task of interest. For each dataset, we train a seed model on 1% of the training data, then run inference with this model to generate pseudo-labels on the remaining 99% of the training examples. We sample 2000 pseudo-labeled examples from this set, split them into disjoint groups of 40 examples each, and show them to 50 and 40 different Mturk workers for sst_2 and 20news, respectively. For each example, the worker is shown the text of the example and the corresponding predicted pseudo label. The workers are then asked to specify whether the pseudo label is accurate (positive feedback) or not (negative feedback) along with an option to mark ‘I Don’t Know’ in case they find it difficult to decide. Further details about the specific instructions used in our Mturk study can be found in Appendix C. We use the ground truth gold labels in sst_2 and 20news to evaluate the quality of user feedback by computing the proportions of times users gave positive feedback to correct pseudo labels and negative feedback to incorrect ones. The observed average error in feedback is 33.9% for 20news and 27.13% for sst2. The higher observed error for 20news is likely due to the fact that 20news is a 20-way topic classification problem with overlapping categories such as ‘autos’ and ‘motorcycles’. We further analyze the collected data using the noisy feedback model described next. Full data will be released with the final draft of the paper.

3.2 Feedback Noise Modeling

Motivated by the observed noisy user behavior above, we propose a parametric noise model using two user-specific Bernoulli random variables parameterized by γ and δ, as shown below.

\[
P(\text{positive feedback} | \text{correct prediction}) = \gamma \\
P(\text{negative feedback} | \text{incorrect prediction}) = \delta
\]

γ and δ capture the probability of the user accurately providing positive and negative feedback,
respectively. This scheme provides a powerful tool to model user noises of various types - by varying the values of these two parameters, we can simulate various user feedback behaviors. For instance, highly reliable users can be simulated by choosing \( \gamma \sim 1 \) and \( \delta \sim 1 \) while adversarial users can be simulated by choosing \( \gamma \sim 0 \) and \( \delta \sim 0 \). Similarly, users that provide consistently positive feedback can be simulated by selecting \( \gamma \sim 1 \) and \( \delta \sim 0 \), and vice versa.

![Figure 1: Distribution of noise parameters \( \gamma \) and \( \delta \) for annotators on Mturk for 20news and sst2 dataset.](image)

For each worker in our MTurk study, we estimate the noise parameters \( \gamma \) and \( \delta \) using the MLE estimators described in Appendix B. We show the distributions over the estimated noise parameters for each worker in Figure 1, which highlights several characteristics of the user noise. We find that parameters vary across users and across datasets. Many workers have high values for both \( \gamma \) and \( \delta \) (upper right quadrant in the plot), especially for the sst2 dataset. In such cases, user noise is relatively low. Some workers have \( \gamma \sim 1 \) and \( \delta \sim 0 \), suggesting that they provide positive feedback with very high probability, regardless of the correctness of the pseudo label. Similarly, we also observe some points with higher \( \delta \) but \( \gamma \) close to 0, suggesting that these workers provide negative feedback with high probability. Since we only recruited reliable worker from Mturk (95%+ approval rate and 5000+ approved HITs, see Appendix C), we do not see any workers in the adversarial or extremely-high noise scenarios (lower-left quadrant in the plot). Finally, we also observe that the workers in the sst2 dataset are more concentrated towards the top right corner while, for the 20news dataset, they are relatively spread out. This can be attributed to the inherent difficulty of the two datasets – sst2 is an easier 2-class sentiment classification dataset, while 20news is a more difficult news-classification dataset with 20 (sometimes overlapping) classes.

### 4 Approach

#### 4.1 Federated Self-Training

Given a training dataset \( D_1 = \{x_i, y_i\} \), we divide it into 3 parts: a training split \( D_s \subset D_1 : |D_s| = k|D_1|, k \ll 1 \), used to train the seed model; a validation split \( D_v \subset D_1 : |D_v| = v|D_1|, v < 1 \) and an unlabeled split \( D_u = D_1 - (D_s \cup D_v) \). We assume that the examples in \( D_s \) and \( D_v \) have gold labels available for training, which mimics the real-world situation where a small amount of data can be annotated to bootstrap the model training. We treat \( D_u \) as the unlabeled dataset which is available on edge. We initialize the seed model \( f_s(x) \) by training on \( D_s \) using standard cross entropy loss. After convergence, this model, \( f_s(x) \), is deployed to the edge devices to start federated training. In order to simulate a real-world federated learning setup, we distribute the examples from \( D_u \) among \( N \) edge clients following a skewed distribution, described in detail in §5. The dataset on each client \( n \) is labeled \( D_u^n \) where \( n \in [1, N] \). The seed model on device \( j \) then makes predictions on its corresponding client-specific dataset \( D_u^v \). Since the edge model does not have access to gold labels for training, there are only two potential sources of information it can learn from. First, its own predictions, \( \rho_i \), which we call pseudo labels:

\[
\rho_i = \arg \max(f_s(x_i)) : i \in D_u^v \tag{1}
\]

Labeling an example \( x_i \in D_u^v \) with \( \rho_i \) is typically referred to as self-training, a commonly-used semi-supervised training setup. However, in our setup, there is also a second source of information: user feedback to each \( \rho_i \). We assume that users give binary (positive or negative) feedback to each \( \rho_i \). We can thus use this feedback to validate or reject each \( \rho_i \), generating label \( \rho_i \) when the feedback is positive and \( \bar{\rho}_i \) when the feedback is negative. Then, with these new user-feedback-labeled datasets on each edge device, we can follow the standard FL
training, further training a copy of the initial global model on each edge device, then propagating each local model back to the global server for aggregation. Our overall setup used for federated learning with user feedback is shown in Figure 2.

4.3 Federated Learning with Feedback

For examples with positive user feedback, since we have user confirmation that the pseudo-label \( \rho_i \) is correct, we directly incorporate \( \rho_i \) into model training as if they were ground-truth. We use the standard categorical cross entropy (CCE) loss function similar to the seed model:

\[
\text{loss}_{\text{pos}} = - \sum_{x_i \in D^\text{pos}_u} \rho_i \log(f(x_i))
\]

where \( f(x_i) \) represents the posterior probability distribution for sample \( i \) and \( \rho_i \) is overloaded to depict the one-hot representation of the pseudo label for sample \( i \). On the other hand, negative feedback signifies that the sample does not belong to the class \( \rho_i \). Although the user feedback does not indicate which class these samples ought to be, we do acquire information for what the model should not do. Thus we can assume that these are complementary labels, denoted as \( \tilde{y}_i = \rho_i \). To incorporate these in our model training, we adapt the complementary learning methods introduced by Yu et al. (2018), in which the authors model the complementary posterior probability distribution, \( P(\tilde{Y} = d|X) \) as a function of true class posterior distribution, \( P(Y = c|X) \) and the transition probability matrix \( Q \), where \( q_{cd} \) is an entry in the matrix \( Q \) with \( c \) and \( d \) are class labels, following:

\[
q_{cd} = \begin{cases} 
P(\tilde{Y} = d|Y = c) & c \neq d \\
0 & c = d 
\end{cases}
\]

\[
P(\tilde{Y} = d|X) = \sum_{c \neq d} P(\tilde{Y} = d|Y = c)
\]

\[
P(Y = c|X)
\]

We estimate the transition probability matrix \( Q \) using the validation set \( D_v \) and the initial seed model \( f_i(x) \). To compute \( Q_{cd} \), the transition probability distribution for the class \( c \), we average the posterior probability of those samples with gold label \( c \), but are incorrectly predicted by the model. Given this, we set \( q_{cc} = 0 \) and renormalize the
vector as shown in Equation 5.

\[
Q_c = \frac{1}{K} \sum_{k \in D_c} \frac{f_k(x_k)}{1 - f_k(x_k)} : K = |D_c| \tag{5}
\]

\[
D_c \subset D_v : \text{argmax}(f(x_k)) \neq c; y_k = c
\]

Using the estimated transition matrix, and the posterior distribution for the true class, we estimate the posterior distribution for the complementary class, following Equation 3. We then use the pseudo labels as complementary labels and train the model with the objective function:

\[
\text{loss}_{\text{neg}} = - \sum_{i \in D_{\text{neg}}} \rho_i \log(Q_i f(x_i)) \tag{6}
\]

here, we overload \(\rho_i\) to depict the one-hot representation of the pseudo label for sample \(i\).

Our overall model is trained to jointly optimize the loss functions from positive feedback and negative feedback. Inspired by Kim and Kim (2020) where the contribution of negative learning is slowly increased during training, we use a scheduler to weigh the two loss functions, ensuring that the positive label learning component has higher weight in the early epochs, gradually increasing the weight for negative label learning as training proceeds. Specifically, at each client we optimize the following objective:

\[
\text{loss}_{\text{reg}} = (1 - \alpha) \cdot \text{loss}_{\text{pos}} + \alpha \cdot \text{loss}_{\text{neg}} \tag{7}
\]

where, \(\alpha = 1 - p^t\); \(t\) denotes the current epoch in the training process and \(p \in (0, 1)\), which was selected using a held out validation set.

4.4 Noise-Robust Loss Functions

Though user feedback provides a valuable learning signal to train our models on edge, it can be noisy. As noted in §3.2, we expect two behaviors from noisy users: if the user provides incorrect positive feedback, we have incorrect true labels in \(D_{\text{pos}}\). Similarly, if the user provides incorrect negative feedback, we have samples in the \(D_{\text{neg}}\) with incorrect complementary labels. Since we use cross entropy loss functions for training on both positive and negatively labeled data points, our model is prone to overfitting to noisy samples Zhang and Sabuncu (2018) since they would have lower posteriors (forcing the algorithm to put more emphasis on them during training). This necessitates some form of noise mitigation in our model training.

To mitigate label noise, we use the approach introduced by Ma et al. (2020), where they propose to add noise robustness to any given loss function by normalizing it across all labels. Ma et al. (2020) further improve convergence by presenting a combination loss function with active and passive loss components, to maximize the posterior for the true class and to minimize the posterior for complementary classes, respectively. In our experiments, we use a combination of Normalized Cross Entropy (NCE) Ma et al. (2020) and Reverse Cross Entropy (RCE) Wang et al. (2019) as the active and passive components, weighted in a ratio 1:2 selected using our validation set.

\[
\text{loss}_{\text{robust}} = \text{NCE} + 2 \cdot \text{RCE} \tag{8}
\]

The NCE and RCE functions are listed below.

\[
\text{NCE} = \frac{- \sum_{k=1}^{K} q(k|x) \log p(k|x)}{\sum_{j=1}^{K} \sum_{k=1}^{K} q(y = j|x) \log p(k|x)} \tag{9}
\]

\[
\text{RCE} = - \sum_{k=1}^{K} p(k|x) \log q(k|x) \tag{10}
\]

where \(K\) is the number of label classes, \(q(k|x)\) denotes the gold label distribution and \(p(k|x)\) denotes the posterior probability distribution.

5 Experiments

5.1 Implementation Details

We use the publicly-provided train and test splits for the sst2 and 20news datasets and further derive a validation split consisting of 20% (\(v = 0.2\)) of the train split \(D_t\), with uniform class distribution. We use another 1% (\(k = 0.01\)) of \(D_t\) as seed model set, \(D_s\). We choose a small value for \(k\) to mimic a real-world scenario where a larger volume of data may be un-annotated in an FL setup. The remaining unlabeled dataset \(D_u\) of the \(D_t\) is further divided among 15 mutually exclusive subsets \((D_u^n, n \in [1, 15])\), which simulates the data for 15 edge clients. While creating the clients’ partitions we ensure that all clients have data with a uniform class distribution which enables us to focus on our model performance in an idealized case. We use the DistilBERT Sanh et al. (2019) base model as the classifier for our tasks and follow the standard fine-tuning setup for text classification. To implement the federated learning pipeline we use the publicly-available FedNLP Lin et al. (2021) benchmark and apply the FedAvg McMahan et al. (2017)
Experimental settings | 20 news | sst2
--- | --- | ---
Initial model ($D_s$) | 59.14 | 77.37
Self training (no feedback) | 60.79 | 77.26
Positive feedback (noisy) | 62.10 | 79.79
All feedback (noisy) | 65.01 | 85.17
Positive feedback (noise robust) | 62.33 | 79.85
All feedback (noise robust) | 65.13 | 85.39
Positive feedback (noise free) | 70.44 | 83.80
All feedback (noise free) | 75.13 | 88.58
Full supervision | 82.12 | 89.12

Table 1: Accuracy of noise robust federated self training with user feedback against various baselines for 20news and sst2 datasets; *: all models using feedback (with and without noise robustness) are statistically significant against the self training baseline (without feedback), at $p < 0.05$.

We compare our model performance against three baselines:

**Initial model** This is the seed model $f_s(x)$ trained on just $D_s$ (1% of the training data).

**Self-training** We train this model using federated learning with pseudo labels, but do not utilize the user feedback. Hence, at each client, we only have the raw pseudo labels $\rho_i$ for each $x_i \in D^u_{pos}$ to train on. We use this setup as the primary baseline against which to compare the performance of our models trained with user feedback.

**Full supervision** We train a model in a federated setting using $D^u_{pos}$ and the true gold labels at each client. Although in a real-world setting, the clients will not have access to the gold labels, we establish this benchmark to set an upper bound.

We evaluate performance of our proposed strategy of leveraging user feedback in two settings:

**Positive feedback** At each client, we train the local version of the model using only the samples in $D^u_{pos}$, and corresponding pseudo labels $\rho_i$, i.e. only the samples which receive positive feedback. Since this baseline is trained using regular cross entropy, it provides a powerful yet computationally less-intensive alternative to training with both types of feedback, which is especially important in edge devices with low compute power.

**All feedback** We utilize all the data samples in $D^u_{pos}$ and $D^u_{neg}$ at each client and train using the loss function described in Section 4.3.

In both the positive and all feedback setups, we evaluate models with and without user feedback noise. For the noise-free scenario, we set $\gamma = 1$ and $\delta = 1$ while simulating the user feedback. This leads to perfectly accurate feedback, as discussed in §4.2. For the noisy feedback scenario, we use the noise parameters derived from the Mturk study. We obtain the following dataset-specific values of $\gamma$ and $\delta$ by averaging the estimates of $\gamma$ and $\delta$ across all annotators: ($\gamma = 0.79, \delta = 0.55)$ for 20news and ($\gamma = 0.76, \delta = 0.70)$ for sst2.

Table 1 reports the % accuracy for each of the experimental setups described above across both datasets. We observe that in both the noisy and noise-free settings, the introduction of positive user feedback shows a marked improvement in performance when compared to the self-training baseline. There is an additional performance gain when we add negative feedback (all feedback baseline), which signifies the importance of learning from complementary labels. As expected, the improvement is substantially larger in the noise free setting, suggesting the need for model robustness to mitigate the effect of noise. Note that for sst2, performance of the noise free model with all feedback is very close to that of full supervision, thanks to the fact that complementary labels in the case of binary classification provide same information as true labels. On the other hand, using perfect positive and negative feedback in 20news is still sub-optimal compared to full supervision, since a negative label in this dataset is less informative compared to sst2.

5.3 Noise Robustness

To mitigate the effects of noise, we replace the traditional cross-entropy loss function with the active-passive loss described in §4.4, using the same experimental setups presented earlier (positive only and all-feedback), with $\gamma$ and $\delta$ values from the Mturk study. However, as evident in Table 1, the robust loss functions only seem to confer marginal performance improvements in both datasets. This is likely due to the fact that the noise parameters extracted from Mturk belong to a moderate to low noise regime (Section 3.2), providing limited room for gains with noise robustness.

To further investigate this, we explore two ex-
Table 2: Performance analysis of noise robust loss functions trained on all feedback in different noise regimes for the 20news dataset; *: statistically significant against the adversarial model without robustness at $p < 0.05$.

| Noise level | Loss       | Accuracy |
|-------------|------------|----------|
| Low         | $loss_{\text{robust}}$ | 73.29    |
|             | $loss_{\text{reg}}$     | 74.30    |
| Adversarial | $loss_{\text{robust}}$ | 42.26*   |
|             | $loss_{\text{reg}}$     | 25.19    |

treme cases of user feedback noise for the 20news dataset: i) low noise, where we simulate user feedback with $\gamma \to 1$, $\delta \to 1$ for all the clients, which imitates clients providing correct feedback with very high probability, and ii) adversarial noise, with $\gamma \to 0$, $\delta \to 0$ for all the clients, which captures the possible risk of users deliberately providing incorrect feedback with high probability. In Table 2, we compare the performances of the all feedback model trained with and without noise robustness in these two scenarios. As seen in the table, when user noise is high, the noise-robust loss functions show a statistically significant improvement against the noisy model, highlighting the value of adding noise robustness. In the low noise regime, adding robustness seems to cause negligible degradation in accuracy, but within the bounds of statistical error. Given this, we recommend using noise robustness in all applications of this framework unless it is well known before hand that the feedback has very low noise. We defer the task of developing a noise robustness regime that works for all noise levels to future work.

5.4 Ablation Studies

5.4.1 Varying Degrees of Noise

As discussed in §4.4, the level of feedback noise has a substantial impact on model performance. In this section, we further investigate this effect, simulating user feedback across various noise parameters values, spanning $\gamma, \delta \in \{0.3, 0.5, 0.7\}$, to capture different points in the $\gamma - \delta$ space. Table 3 shows our results for each dataset with the noise robust loss function $8$. As expected, as $\gamma \to 0$ and/or $\delta \to 0$, model performance decreases on both datasets. At very low values of $\delta$ and $\gamma$, e.g. both $\leq 0.5$, training on the extremely noisy user feedback actually decreases model performance below the original seed model. This is not unexpected, since at $\delta = 0.5$ and $\gamma = 0.5$, user feedback is essentially random noise, and at lower values the feedback is adversarial. These results highlight the importance of evaluating the reliability of user feedback before using it to further train an ML system.

5.4.2 Non-Identical User Feedback Behavior

In previous sections, we used identical values of the noise parameters $\gamma$ and $\delta$ for all clients in the FL training setup. However, as observed in our Mturk study, real users have different feedback behaviors, with scores spread out over the $\gamma - \delta$ space. In this section, we simulate non-identical user feedback for four potential user behaviors:

1. Low noise users ($\gamma \to 1$, $\delta \to 1$)
2. Adversarial/high noise users ($\gamma \to 0$, $\delta \to 0$)
3. Positive users ($\gamma \to 1$, $\delta \to 0$) who provide consistently positive feedback, regardless of the correctness of the model prediction; and
4. Negative users ($\gamma \to 0$, $\delta \to 1$) who provide consistently negative feedback.

To simulate non-identical user feedback for the clients, we sample the noise parameters from an appropriately skewed $\beta(a, b)$ distribution. For example, in order to generate $\delta$ and $\gamma$ scores for setup four (negative users), each user needs $\gamma \approx 0, \delta \approx 1$. To generate these parameters, we sample $\gamma$ from $\beta(1, 10)$ and $\delta$ from $\beta(10, 1)$. Proceeding this way, we can simulate all four user behaviors listed above. Finally, we also conduct an experiment closer to the real-world scenario, where we randomly sample 15 workers each for both datasets from our Mturk study and use their estimated values of $\gamma$ and $\delta$ to simulate user feedback.
| User Behavior          | 20news  | sst2  |
|-----------------------|---------|-------|
| Low noise             | 73.67   | 88.35 |
| Adversarial           | 55.86   | 64.85 |
| Always positive       | 60.99   | 77.16 |
| Always negative       | 58.92   | 74.13 |
| Real world (mturk study) | 65.37   | 85.61 |

Table 4: Model performance at various user behaviors.

Table 4 shows our results across all five simulations for both datasets when trained with the noise robust loss function. As expected, the best model performance is achieved with the low-noise users, followed by the real-world users sampled from our MTurk study. In the three other simulations (adversarial, consistently positive, consistently negative), user feedback is highly noisy and unreliable, and the models show limited improvement over the initial seed model. Note that the performance in the positive feedback scenario is higher than negative feedback, which can be accredited to the fact that the initial seed model’s accuracy is greater than 50% for both datasets (Table 1). With > 50% accuracy, a majority of the pseudo-labels generated using the seed model will match the gold label. Hence, consistently positive feedback introduces less noise and in turn better performance compared to the all negative feedback model.

6 Conclusion

In this work, we propose a novel framework for federated learning which leverages noisy user feedback on the edge. Our framework eliminates the need for labeling edge data, in turn improving customer privacy since we no longer need to move data off of edge devices for annotation. In order to evaluate our framework, we propose a method to simulate user feedback on publicly-available datasets and a parametric model to simulate user noise in that feedback. Using that method, we conduct experiments on two benchmark text classification datasets and show that models trained with our framework significantly improve over self-training baselines.

Future work includes deploying our framework in a real world FL application and incorporating live user feedback in model training. We can also explore other noise-robustness strategies for low and medium label-noise scenarios. One such strategy would be to incorporate a measure of user reliability into the calculation of each new global model in FedAVG – e.g. the updated global model parameters could be computed as a weighted average of client models, with the weight capturing some measure of each client’s reliability.

7 Ethics Statement

Our Mturk data collection recruited annotators from across the globe without any constraints on user demographics. The annotators were compensated with above minimum wages and no personal information was collected from the annotators.

References

California Consumer Privacy Act. 2018. California consumer privacy act. https://oag.ca.gov/privacy/ccpa. Accessed: 2021-09-03.

General Data Protection Regulation. 2018. General data protection regulation. https://gdpr-info.eu/. Accessed: 2021-09-03.

Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS).

Aritra Ghosh, Himanshu Kumar, and P. Shanti Sastry. 2017. Robust loss functions under label noise for deep neural networks. In Proceedings of the AAAI conference.

Nadine Guhr, Oliver Werth, Philip Peter Hermann Blacha, and Michael H Breitner. 2020. Privacy concerns in the smart home context. SN Applied Sciences, 2(2):1–12.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604.

Takashi Ishida, Gang Niu, Weihua Hu, and Masashi Sugiyama. 2017. Learning from complementary labels. In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS).

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 963–973, Vancouver, Canada.

Andreas Kaplan and Michael Haenlein. 2019. Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. Business Horizons, 62(1):15–25.
Bernhard Kratzwald and Stefan Feuerriegel. 2019. Learning from on-line user feedback in neural question answering on the web. In The World Wide Web Conference, pages 906–916.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. 2018. Can neural machine translation be improved with user feedback? In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 92–105, New Orleans, Louisiana. Association for Computational Linguistics.

Bill Yuchen Lin, Chaoyang He, ZiHang Zeng, Hulin Wang, Yufen Huang, M. Soltanolkotabi, Xiang Ren, and S. Avestimehr. 2021. Fednlp: A research platform for federated learning in natural language processing. In arXiv cs.CL 2104.08815.

Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133.

Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romanu, Sarah Erfani, and James Bailey. 2020. Normalized loss functions for deep learning with noisy labels. In International Conference on Machine Learning, pages 6543–6553. PMLR.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agura y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), pages 1273–1282. PMLR.

Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. arXiv preprint arXiv:1906.04329.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In Proceedings of the seventh IEEE Workshop on Applications of Computer Vision (WACV/MOTION’05), volume 1, pages 29–36.

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

Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pages 1–11.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jingfeng Yi, and James Bailey. 2019. Symmetric cross entropy for robust learning with noisy labels. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 322–330.

Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. arXiv preprint arXiv:1812.02903.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In Proceedings of the 31st Computer Security Foundations Symposium (CSF), pages 268–282. IEEE.

Xiyu Yu, Tongliang Liu, Mingming Gong, and Dacheng Tao. 2018. Learning with biased complementary labels. In Proceedings of the European conference on computer vision (ECCV), pages 68–83.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS).

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS).

A Pseudocode

Algorithm 1 lists our training loop.

B Estimators for $\gamma$ and $\delta$

Let $X$ be the data set and $n$ be the total number of data points. For any data point $i \in [n]$, let $p_i$, $t_i$ and $f_i$ denote the model predicted label, ground truth label, and user feedback respectively. (Note that $p_i$ and $t_i$ take values from the set of labels and $f_i$ takes values from the set $\{pos, neg, idk\}$ representing feedbacks positive, negative, and ‘I don’t know’.) By definition, we have

$$
\gamma := \Pr(f_i = pos \mid p_i = t_i)
$$

$$
\delta := \Pr(f_i = neg \mid p_i \neq t_i)
$$

Let us also define

$$
\alpha := \Pr(f_i = neg \mid p_i = t_i)
$$

$$
\beta := \Pr(f_i = pos \mid p_i \neq t_i)
$$
Algorithm 1 Algorithm for simulating user feedback

INPUT: Client data: \( D_n^u = \{x_i, y_i\} \); Pseudo labels: \( \rho_i \)

OUTPUT: \( D_{pos}^n \) and \( D_{neg}^n \)

1: \( D_{pos}^n \leftarrow \{\} \), \( D_{neg}^n \leftarrow \{\} \)
2: for sample \( i \) in \( D_n^u \) do
3:  \( \text{if } y_i = = \rho_i, \text{ then} \) \( \triangleright \) correct model prediction
4:  \( D_{pos}^n \leftarrow \{D_{pos}^n \cup i\} \) with probability \( \gamma \)
5:  \( D_{neg}^n \leftarrow \{D_{neg}^n \cup i\} \) with probability \( 1 - \gamma \) \( \triangleright \) noise
6:  \( \text{else if } y_i! = \rho_i, \text{ then} \) \( \triangleright \) incorrect model prediction
7:  \( D_{neg}^n \leftarrow \{D_{neg}^n \cup i\} \) with probability \( \delta \)
8:  \( D_{pos}^n \leftarrow \{D_{pos}^n \cup i\} \) with probability \( 1 - \delta \) \( \triangleright \) noise
9: end if
10: end for
11: return \( D_{pos}^n \) and \( D_{neg}^n \)

Note that the above definitions imply that

\[
1 - \alpha - \gamma = \Pr(f_i = \text{idk} \mid p_i = t_i)
\]
\[
1 - \beta - \delta = \Pr(f_i = \text{idk} \mid p_i \neq t_i)
\]

Moreover, let \( a \) denote the accuracy of the labels predicted by the model defined as

\[
a := \Pr(p_i = t_i)
\]

Define sets \( \{S_j\}_{j \in [6]} \) such that

\[
S_1 := \{i \in [n] : f_i = \text{pos and } p_i = t_i\}
\]
\[
S_2 := \{i \in [n] : f_i = \text{pos and } p_i \neq t_i\}
\]
\[
S_3 := \{i \in [n] : f_i = \text{neg and } p_i = t_i\}
\]
\[
S_4 := \{i \in [n] : f_i = \text{neg and } p_i \neq t_i\}
\]
\[
S_5 := \{i \in [n] : f_i = \text{idk and } p_i = t_i\}
\]
\[
S_6 := \{i \in [n] : f_i = \text{idk and } p_i \neq t_i\}
\]

define \( n_j := |S_j| \). Note that \( \sum_{j \in [6]} n_j = n \).

Theorem 1. The maximum likelihood estimators for \( \gamma \) and \( \delta \) are \( n_1/(n_1 + n_3 + n_5) \) and \( n_4/(n_2 + n_4 + n_6) \) respectively.

Proof. Now for any data point \( i \in [n] \), we have

\[
\Pr(i \in S_1) = \Pr(f_i = \text{pos} \text{ and } p_i = t_i)
\]
\[
= \Pr(f_i = \text{pos} \mid p_i = t_i) \cdot \Pr(p_i = t_i)
\]
\[
= \gamma a.
\]

By a similar reasoning, we have

\[
\Pr(i \in S_2) = \beta(1 - a)
\]
\[
\Pr(i \in S_3) = \alpha a
\]
\[
\Pr(i \in S_4) = \delta(1 - a)
\]
\[
\Pr(i \in S_5) = (1 - \alpha - \gamma)a
\]
\[
\Pr(i \in S_6) = (1 - \beta - \delta)(1 - a)
\]

Therefore the likelihood function of the model is

\[
\mathcal{L}(\alpha, \beta, \gamma, \delta \mid X) = \frac{n!}{n_1! \ldots n_6!} (\gamma a)^{n_1} (\beta(1 - a))^{n_2} (\alpha a)^{n_3} (\delta(1 - a))^{n_4} ((1 - \alpha - \gamma)a)^{n_5} ((1 - \beta - \delta)(1 - a))^{n_6}
\]

and consequently the log-likelihood function is

\[
\log \mathcal{L}(\alpha, \beta, \gamma, \delta \mid X) = \log \left(\frac{n!}{n_1! \ldots n_6!}\right) + n_1 \log(\gamma a) + n_2 \log(\beta(1 - a)) + n_3 \log(\alpha a) + n_4 \log(\delta(1 - a)) + n_5 \log((1 - \alpha - \gamma)a) + n_6 \log((1 - \beta - \delta)(1 - a))
\]

(11)

To obtain MLE estimates of parameters \( \alpha, \beta, \gamma, \delta \), we wish to solve the following optimization problem

\[
\max_{(\alpha, \beta, \gamma, \delta) \in [0, 1]^4} \log \mathcal{L}(\alpha, \beta, \gamma, \delta \mid X)
\]

(12)
By Fermat’s theorem, the optimal solution to the above optimization problem lies at either a boundary point or a stationary point.

The boundary points of the set $[0, 1]^4$ are given by the set

$$B := \{(\alpha, \beta, \gamma, \delta) \in [0, 1]^4 : \alpha = 0 \text{ or } \alpha = 1 \text{ or } \beta = 0 \text{ or } \beta = 1 \text{ or } \gamma = 0 \text{ or } \gamma = 1 \text{ or } \delta = 0 \text{ or } \delta = 1\}$$

The value of the function $\log L(\alpha, \beta, \gamma, \delta)$ is negatively unbounded on the set $B$.

On the other hand, the stationary points can be determined by setting the gradient to be zero, i.e., by solving the equation

$$\nabla_{(\alpha, \beta, \gamma, \delta)} \log L(\alpha, \beta, \gamma, \delta | X) = 0.$$

Solving the above equation yields the stationary point $(\alpha^*, \beta^*, \gamma^*, \delta^*)$ given as

$$\begin{align*}
\alpha^* &= n_3/(n_1 + n_3 + n_5) \\
\beta^* &= n_2/(n_2 + n_4 + n_6) \\
\gamma^* &= n_1/(n_1 + n_3 + n_5) \\
\delta^* &= n_4/(n_2 + n_4 + n_6)
\end{align*}$$

The value of the log-likelihood function at the above critical point, i.e., $\log L(\alpha^*, \beta^*, \gamma^*, \delta^* | X)$ is positive which suggests that it is the optimizer of the optimization problem in (12).

C Details on MTurk study

Figure 3 shows the instruction page used to guide the workers on Mturk. Since our goal here was to simulate real user feedback for AI systems, we designed the prompt to mimic a situation where users provide their judgements on the accuracy of machine predictions on a given task. Each assignment page had 40 questions for the 20news task (50 for sst2), with an example question shown in Figure 4. For a real world application of this setting, we can imagine an email categorization model deployed on end-user email clients which automatically classifies incoming emails to a predefined class. The user would approve (select Accurate to above question) if the categorization was correct, reject or make correction (select Inaccurate to the question) or take no action. This closely follows the federated user feedback scenario described in our experiments with users explicitly providing positive or negative feedback.

We recruited highly reliable annotators on Mturk by selecting past approval rate as $95\%+$ and number of past approved tasks as $5000+$. The average time for each task was 30 minutes, and the annotators were paid $7 for completing the task which is above the US federal minimum hourly wage, given the average time for task completion. Note that we did not place any geographic restrictions on the annotators, nor reject any partial submissions, despite stating as such in the instruction sheet, as they were few in number.

In Figure 5, we show the error in user feedback computed against gold labels for all the users. We also show the distribution of positive and negative responses for all the users. As evident from the figure, users provide positive feedback in majority cases. This behavior is expected since the initial model’s accuracy for 20news is $59.14\%$ and for sst2 is $77.37\%$; since a the majority of the pseudo labels are correct predictions, we expect mostly positive feedback from the users.
Please read all instructions listed below, before you start the task.

- Our world is full of interactions between humans and AI systems. We interact with customer service chatbots, with search engines, with personal assistants, among many others. Such interactions happen in our homes, our cars, and pretty much everywhere. But machines don’t always get it right. In this task, we ask you to imagine that you requested the machine to assign an email body to a specific topic, such as gun politics, baseball, or motorcycles.
- We will then show you what the machine returned and ask to verify if it was an accurate prediction or not.
- There are 20 possible topics the machine can assign your email to, and they may sometimes be overlapping. Try to identify the most precise label that can be assigned to the email. We will regularly show you the 20 labels as you scroll through the questions to help with the task.
- There are 40 examples in this HIT. To get paid, you need to answer all questions before you hit the submit button.
- We wish to get answers from different Mturkers for different tasks without overlaps and we have created separate HITs for each of these. Please avoid answering more than one HIT.

Figure 3: Instruction page with guidelines for Mturk annotators.

| Input text | Lines: 3 I have 16MB of memory on my 386SX. I have been running Windows without a swap file for several months. Will Mathcad 4.0 be happy with this or insist on a swap file? |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Machine prediction | pc hardware |

What do you think about the machine prediction? Is it

- [ ] Accurate
- [ ] Inaccurate
- [ ] Don’t know

Figure 4: Example annotation task
Figure 5: User feedback behavior of clients in Mturk case study. A & C: Incorrect feedback(%) for all the clients for 20news and sst2. B & D: Distribution of negative and positive feedback for each client in 20news and sst2.