FedSpam: Privacy Preserving SMS Spam Prediction

Jiten Sidhpura*, Parshwa Shah*, Rudresh Veerkhare*, and Anand Godbole

Sardar Patel Institute of Technology, Mumbai
{jiten.sidhpura, parshwa.shah, rudresh.veerkhare, anand_godbole}@spit.ac.in
†

Abstract. SMSes are a great way to communicate via short text. But, some people take advantage of this service by spamming innocent people. Deep learning models need more SMS data to be adaptive and accurate. But, SMS data being sensitive, should not leave the device premises. Therefore, we have proposed a federated learning approach in our research study. Initially, a distilBERT model having validation accuracy of 98% is transported to mobile clients. Mobile clients train this local model via the SMSes received and send their local model weights to server for aggregation. The process is done iteratively making the model robust and resistant to latest spam techniques. Model prediction analysis is done at server side using global model to check which words in message influence spam and ham. On-device training experiment is conducted on a client and it is observed that the losses of the global model converge after every iteration.

Keywords: Spam Detection · Federated Learning · Distil-BERT · SHAP Analysis · Android .

1 Introduction

Short Message Service (SMS) is one of the popular modes of communication. This service is provided by telecom operators and used by people and organizations. These organizations can be banks, governments, companies, startups, etc. Sensitive details like authentication OTPs, bank balance, verification codes, and booking details are shared via SMS which becomes a loophole in the system. Spammers can exploit this vulnerability by spamming and trapping innocent users using cheap tricks. Novice users blindly believe in the fake and lucrative rewards and benefits given by spammers and fall into the trap.

According to [1], there are more than 6 billion smartphone users currently, and will be 7 billion by 2025. This will allow spammers to bulk SMS spam on multiple smartphones with the hope that significant people get fooled. Spammers usually attract users by giving them fake rewards, lottery results, giveaways,

* These authors contributed equally to this work.

and promises of some free and affordable services. They ask users for PINs and passwords by gaining their trust. They also send hyperlinks that redirect them to some phishing page or it downloads some malware or spyware on the phone. In this way, they get unauthorized access to users' sensitive data. Users ignore these threats due to a lack of awareness and repent later. Thus, there is a clear need for an intelligent automated deep learning-based spam classifier that alerts mobile users about potential spam and ham messages.

Keeping in mind, the SMS data confidentiality, it will be more vulnerable if SMS data is sent to a centralized server to get spam classification. Also, bombarding the server with requests for each SMS by billion users at a time is not feasible as the server may crash. Deep learning models become robust if they are trained on large data. Smart spammers have multiple tricks to spam users. So, a system needs to be created that is adaptive to new spam techniques and at the same time classifies SMS spam on-device, thus respecting the privacy of users' sensitive SMS data. Federated Learning i.e. on-device privacy-preserving learning approach can solve all problems of SMS data privacy, server latency, and model robustness and adaptability.

The objective of this research is to apply federated learning to classify SMS spam messages. We have used a federated learning approach provided by the Flower framework to classify SMS spam on-device. The distilBERT deep learning model is used to classify whether the SMS is spam or ham. Initially, a global model is sent to each mobile device from a centralized server. This global model gets trained on the local mobile's SMS data on-device so every device will have different model weights. After these local models get trained, they will send their model weights and not the SMS data to a centralized server periodically at a certain interval. The centralized server will aggregate the weights received from each local model and send the final aggregated global model to each device. This happens iteratively and the model becomes more and more adaptive and robust and users' privacy is also not breached. Model prediction analysis is also done on the server-side to determine keywords in spam SMSes.

The structure of the paper is as follows: Section 2 describes the literature survey, where we discussed existing approaches. In Section 3, we have proposed our methodology. In Section 4 we have explained our experiments. In Section 5, we have presented our results. Finally, in section 6, we conclude our paper.

2 Literature Survey

Sandhya Mishra et al. [2] proposed a Smishing (SMS Phishing) detection system and evaluated it on real time datasets and they also performed a case study on Paytm smishing scam. Their proposed system consisted of Domain Checking Phase and SMS Classification Phase. Only the SMS containing URLs are fed into the domain checking phase where they verify the authenticity of the URL. Features such as domain name and query parameters are extracted and searched on Google. If the domain name is in top 5 results they marked them as legitimate else they carried second-level domain checking. In second-level domain checking,

source code of a website along with all URLS in that source code are also extracted. Domain names of all these URLS are compared and if they belong to the same domain then they are marked legitimate else it is passed to SMS Classification phase. In the final phase misspelled words, leet words, symbols, special characters, and smishing keywords are extracted. Leet words are the words in which some characters of the genuine word are replaced by digits and symbols to make it appear like a genuine word. Finally a neural network and classifications algorithms such as Naive Bayes, Decision Tree and Random Forests were used for the final prediction. Neural network gave the best performance among all and gave a 97.93% accuracy score.

M.Rubin Julis et al. [3] followed the Natural Language Processing (NLP) approaches for spam classification. They used the SMS Spam Corpus v.0.1 to apply their proposed methodology. They applied stemming on their dataset to change over the similar words to their base word format so that the number of unique words does not become too large. Eg: works, working would be converted to work. Spam SMS have the same looking words (homoglyphs) to trick the user to believe that it is a legitimate message. They converted these words into their genuine word format for better insights. Various classical machine learning algorithms such as Logistic Regression, Naive Bayes Classifier, K Nearest Neighbours, Support Vector Machines and Decision Tree Classifier were used. To compare the efficiency of models they also computed the prediction time. Support Vector Machine got the best accuracy score (98%) among all the models. Naive Bayes was fastest in terms of prediction and since it gave nearly the same results as the Support vector machine it was most effective.

Wael Hassan Gomaa et al. [4] proposed the impact of deep learning on SMS Spam filtering. They used the dataset from the UCI repository with 5574 English language emails containing 4827 non-spam and 747 spam emails. The dataset was converted to semantic word vectors using the Glove model. They used 6 machine learning classifiers and 8 deep learning techniques. The maximum accuracy they got from machine learning classifiers was 96.86% given by Gradient Boosted Trees. In deep learning, the maximum accuracy was given by RDML which is 99.26% accuracy.

X. Liu et al. [5] have compared various machine learning approaches for SMS spam detection. Neural network-based approaches performed better compared to classical machine learning algorithms. Further, the authors proposed a seq2seq transformers multi-head attention model for spam classification and it has outperformed other methods. To encode textual data, authors have used Glove embeddings for neural networks and TF-IDF representation for machine learning algorithms. SMS Spam Collection v1 and UtkMl's Twitter datasets are used for training and evaluation purposes.

Sergio Rojas–Galeano [6] aims to investigate the possible advantages of language models that are sensitive to the context of words. First, they performed a baseline analysis of a large database of spam messages using Google's BERT. Then, build a thesaurus of the vocabulary that contained these messages. The author has performed a Mad-lib attack experiment where he modified each mes-

sage with different rates of substitution. The classic models maintained a 93% balanced accuracy (BA) while the BERT model scored a 96%. The performance of the TFIDF and BoW encoders was also affected by the attacks. These results indicate that BERT models could provide a better alternative to address these types of attacks.

H. Brendan McMahan et al [11] introduced a distributed machine learning technique that keeps the data for the model training on the edge device such as mobile and develops a global model combined with the knowledge gained from each client by training models at edge devices. This new methodology is termed Federated Learning by the authors. The decentralized approach respects the privacy of the user-sensitive data and sends only the model weights to a centralized server through the network. They introduced the FederatedAveraging algorithm to combine the results of the local models gained after applying the Stochastic Gradient Descent (SGD) algorithm to the local models of all the clients. This algorithm executes on the server-side and performs model weights averaging. Their algorithm reduces the communication costs for transferring weights from clients to the centralized server by introducing more independent clients and performing extra computation on devices. Mobiles handling these computations is feasible since training dataset size is generally small, and mobile processors are modern and advanced. Later they conducted two experiments, one image classification task to identify the images that will be viewed frequently. The second task was to develop language models for predicting the next word, complete responses to messages. They achieved a test set accuracy of 99% for the image classification task with the help of the CNN model on the MNIST dataset with 1, 10, 20, 50, and 100 clients. In case of the language models, they used the LSTM model on a dataset created from The Complete Works of William Shakespeare with a total of 1146 clients and achieved a score of 54%. These results show the performance of federated learning and how one can use it to solve other data-sensitive problems with it.

Federated Learning has shown promising results in machine learning where the private-sensitive data of the user comes into the picture. However, implementing federated learning systems has many challenges due to scalability. Due to these difficulties involved while developing federated systems, researchers have to rely on simulation results. Daniel J. Beutel et al [15] introduced Flower, a federated learning open source framework that helps researchers test their federated systems in the real world on edge devices such as mobile. Their framework supports many deep learning frameworks and supports model training on many heterogeneous devices. The framework was capable to support a total of 15M clients with the help of a couple of GPUs only.

Andrew Hard et al [13] used a federated learning mechanism to train a recurrent neural network to train a federated learning system to predict the next word for a sentence from Google's keyboard application, GBoard. The authors of the paper chose federated learning because most of the datasets available generally have a different distribution than chatting between users. Their model was trained using Tensorflow Framework and used Tensorflow Lite for prediction

on mobile devices. Client devices needed to have a vocabulary of 10000 words, and after compression, a model of size 1.4 MB was placed in the mobile phones. The GBoard application gave three suggestions for the next word, so they used Top-3 recall and Top-1 recall as their evaluation metrics. Results of the federated learning approach were better than the server approach for real chat data.

3 METHODOLOGY

3.1 Dataset

We have merged three datasets which amount to 39792 rows in our proposed methodology. In total, we have 21768 ham messages and 18024 spam messages. In each dataset, we have two columns i.e. message body and label whether the message is spam or ham.

Enron Spam Dataset : Enron Spam dataset is a dataset created by [8]. There are a total of 33716 records and 4 columns in the dataset. The fields in this email dataset are the subject, message, spam/ham label, and date of the email. We have used message and label fields for our research and also discarded 371 rows as they were unlabelled. We have 33345 messages, out of which 16852 are spam and 16493 are ham.

UCI SMS Spam Collection Data Set : UCI SMS Spam collection dataset is a publicly available dataset made by [9] containing 5572 messages and 2 columns. The 2 columns are message and label. There are a total of 747 spam messages and 4825 ham messages.

British English SMS Corpora : British English SMS Corpora is a publicly available dataset made by [10] containing 875 messages with each message with a label i.e. spam or ham. It contains 425 spam messages and 450 ham messages.

Data preprocessing like removing special characters and punctuations are done on each dataset and then merged to create a final dataset.

3.2 DistilBERT

The DistilBERT [7] model is a distilled version of BERT. DistilBERT is a small, fast, cheap, and light Transformer model trained by distilling BERT base. It has 40% fewer parameters than bert-base-uncased and runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark. This Model is specifically calibrated for edge devices or under constrained computation conditions. Given this, DistilBERT is a suitable candidate for federated SPAM detection.

3.3 Federated Learning

In machine learning systems data is stored in a centralized format to train and improve machine learning models. Also, the input data is sent from devices over the network to compute inference. Sometimes the input data for the model is sensitive to the user, and in such scenarios, we can use Federated Learning[11]. Federated Learning is a modern distributed machine learning technique that utilizes sensitive data of the user present on mobile devices to train machine learning models directly on mobile phones. Clients then send the model weights and not the data over the network to a server. With the help of the FedAvg [11] algorithm, a weighted average of all the model parameters from various clients gets computed. Google has applied this technique in Google Assistant [12] and Google keyboard (Gboard) [13]. We have used this same Federated Learning technology to predict whether SMS received by the user is spam or ham.

Each part of our Federated Machine Learning system is described in following subsections:

Initial Global Model Training For initial global model training, we have fine-tuned the DistilBERT model for our task of SMS Spam detection. The input message is processed using the DistilBERT tokenizer and converted into input_ids and attention_masks. Then the classifier network as shown in Figure 1 after DistilBERT consists of 3 Dense Layers each having a Dropout with a rate of 0.2. 90% of the combined data is used for training and rest 10% for validation.

Model Training and Inference on Edge Devices - For training the Deep Learning model on an Edge device like android phones, our deep learning model has to be exported in multiple parts.

Base - This is the fine-tuned DistilBERT model, which would not get trained on the user's device, it will be only used to extract bottleneck features, and these features are used as an input to the head.

Head - This is the classifier part of the model, which comes after the base model. The Head will get trained on the user's device.

Scheduling of On Device Model Training The training model on mobile devices is a resource-intensive task, and the computing power of mobiles is also limited. We should not train a model when the device is operated to avoid a bad user experience, has a low battery, or has no internet connection. Due to these reasons, we need to train our model after the above constraints are followed, and to accomplish it we have used the WorkManager API. We have used this API to schedule training of models on recent SMS data available on the user's device once every 15 days.

Federated Cycle The server will wait for a specific number of clients to start the federated learning iterations. Once the server has connections from the required number of clients, the local model training on clients initiates, and the

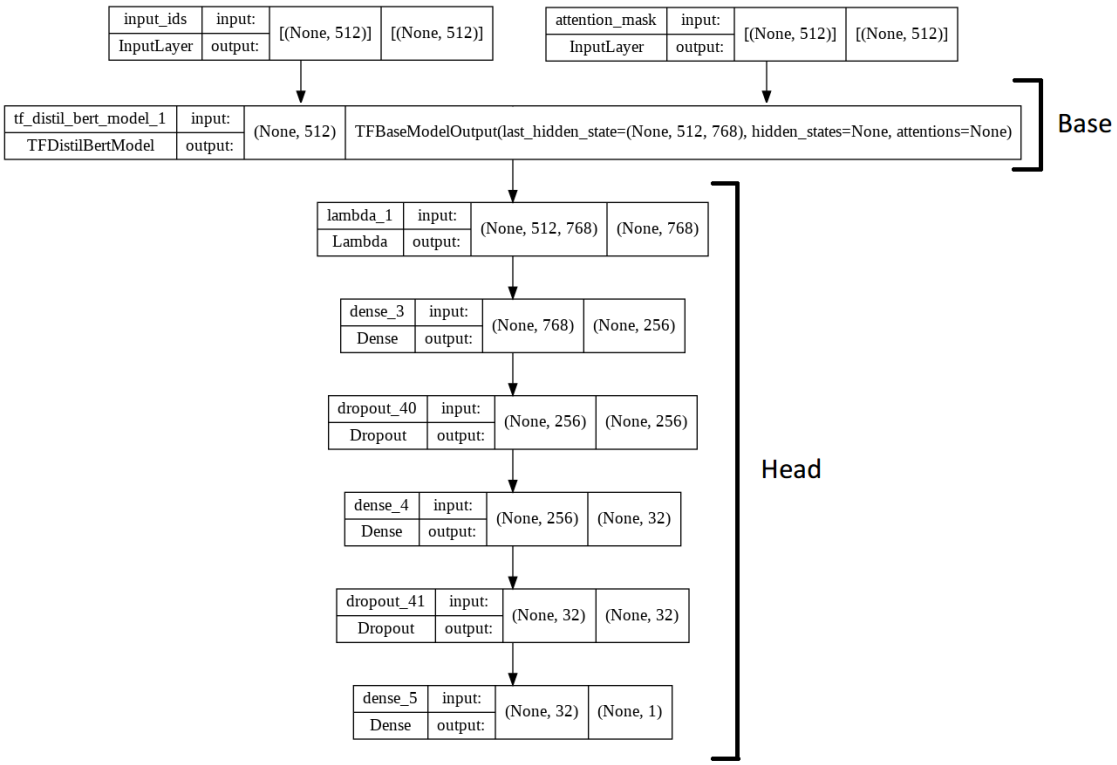

Fig. 1. Deep Learning Model Architecture

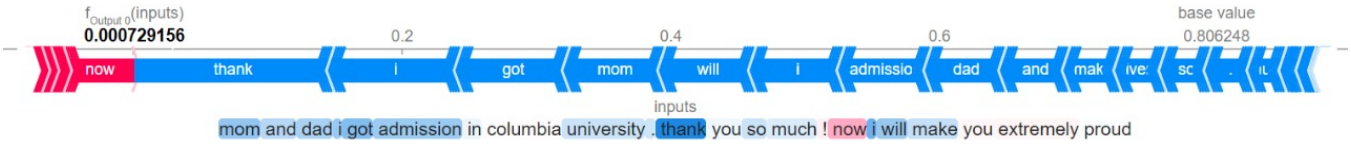

Fig. 2. Shap Text Plot for a Ham Message

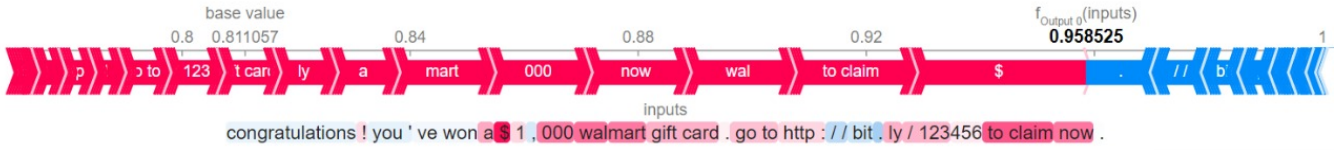

Fig. 3. Shap Text Plot for a Spam Message

weights are sent securely to the server. Later a global model is created from an aggregation of local model weights. This global model has knowledge of all the clients and is sent back to all the clients. In this way, every client gets the knowledge from all other clients without losing their privacy. We carry out analysis of the global model to understand its performance. This analysis is explained in detail in section 4.2.

3.4 System Flow

The FedSpam system flow is demonstrated in Figure 4. The steps mentioned in the flow diagram are explained below:

1. Initially, the centralized server will send the global model to all local mobile clients.
2. Mobile clients receive SMS messages from companies, operators, friends, etc.
3. All SMSes received during the sprint of the last 15 days stored in the mobile's local database are fetched and sent to the local model.
4. The local model will predict whether the messages are spam or ham and assign a label and store it in a local database.
5. Now, the user is given the option to rectify the label if the local model predicted wrong and writes the final label to the database. Then, the local model will train on these SMS data.
6. The local model weights are sent to the server for aggregation.
7. Secure Aggregation of all local model weights to make a global model.
8. Model Prediction Analysis of the global model to check which keywords influence messages towards spam or ham.

All these federated learning steps are scheduled periodically every 15 days to make model robust and adaptive.

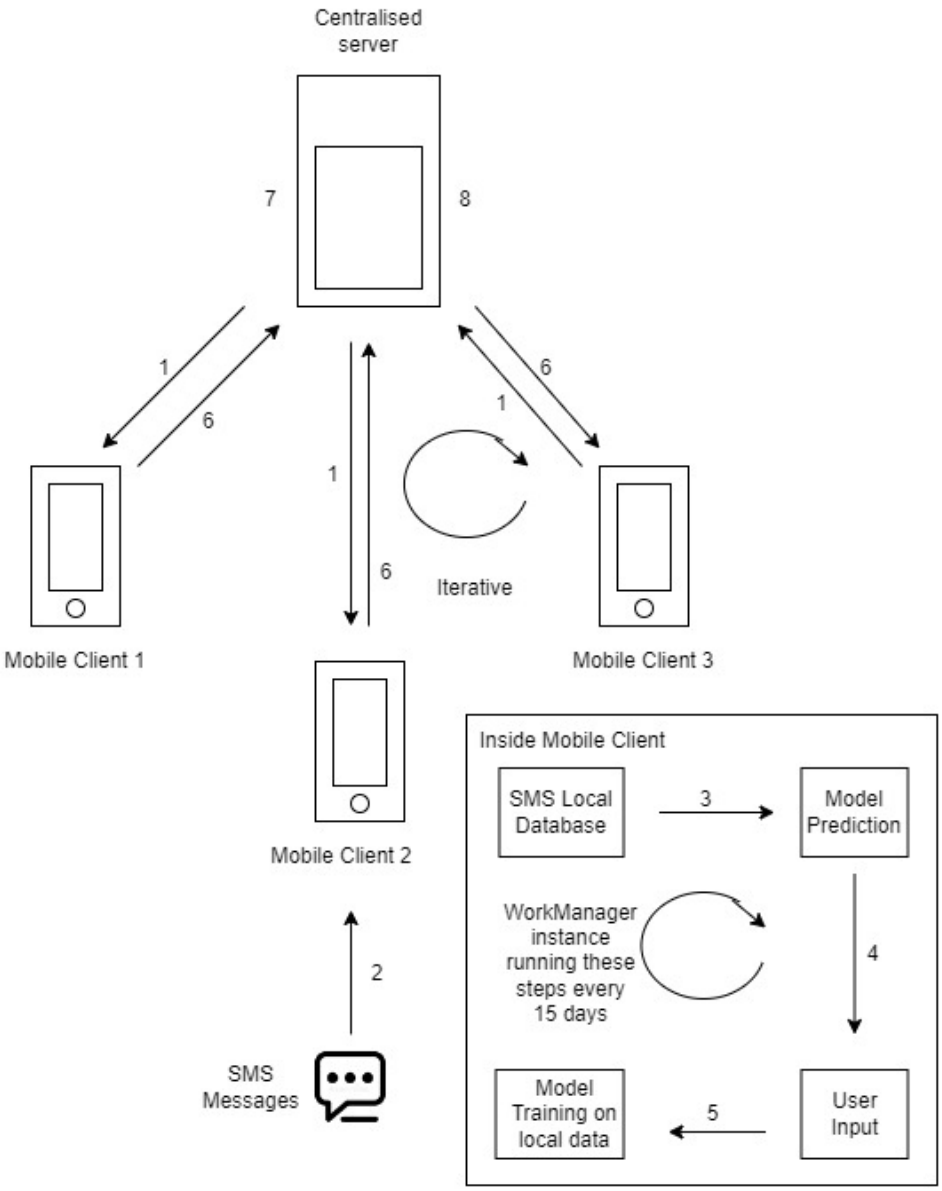

Fig. 4. FedSpam System Flow Diagram

4 Experiment

4.1 Experimental Setup

For training the initial global model, we used Google Colab with 12.69GB of RAM, and 107GB of ROM, having TPU runtime. Packages used are tensorflow 2.8.0, transformers 4.17.0, and flwr[15] 0.18.0. For federated learning setup, multiple android devices are used, all having android SDK version $\geq$ 28 and RAM $\geq$ 4GB.

4.2 Model Prediction Analysis

Machine Learning these days plays an important role for many businesses today. Businesses not only require a machine learning model with great accuracy but also require justification for the prediction. The importance of the model prediction justification is equal to its accuracy. We have used SHapley Additive exPlanations (SHAP)[14] for the global model to see which words or group of words present in the SMS made our model predict either the spam or ham class. In both figures 2 and 3, words that are marked with red color push the prediction towards class 1 (spam), and those marked with blue push the prediction towards class 0 (ham). From figure 2, it is clear that many words are blue, and their contributions towards prediction are much heavier than the few words marked in red. In the case of figure 3, many words are red, and their contributions are more dominant than those marked as blue.

4.3 On Device Resources Analysis

We used Xiaomi's Redmi Note 5 Pro (launched back in Feburary 2018) to understand the CPU and RAM consumption of our federated learning mobile app. Since we are training machine learning models on mobile phones, it becomes very important to analyze the CPU usage and RAM consumption to understand the impact of the application on mobile devices. We have used Android Studio's CPU Profiler to monitor the CPU and RAM consumption. After experimenting, we found that the CPU usage on average is around 12%, and in some cases, it can rise to 25% because of data loading and model training operations. Similarly, we found RAM consumption to be around 400 MB. With these requirements also, the application was working smoothly and did not cause any performance issues on the device. Due to the advancements in mobile processors, our proposed application will become more and more efficient on modern mobile devices.

4.4 Evaluation Metrics

We have used Categorical Cross Entropy mentioned in eq. 1 as the loss function for this classification problem

$$-\sum_{1}^{n} y_i . \log \widehat{y_i} \tag{1}$$

where n is the output size.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2}$$

where K is the number of classes. The last layer activation function is the softmax activation function mentioned in eq. 2. The optimizer we have used is Adam.

5 Results

Initial Global Model - We fine-tuned the DistilBERT on the combined dataset. There we achieved 98% accuracy on the validation set.

Figure 5 depicts how the federated loss of 2 clients converged after ten epochs. With just 32 samples used for training on both devices, the loss of the global model decreased. In the real world, we can use thousands of clients and then perform model weight averaging with the help of the FedAveraging algorithm.

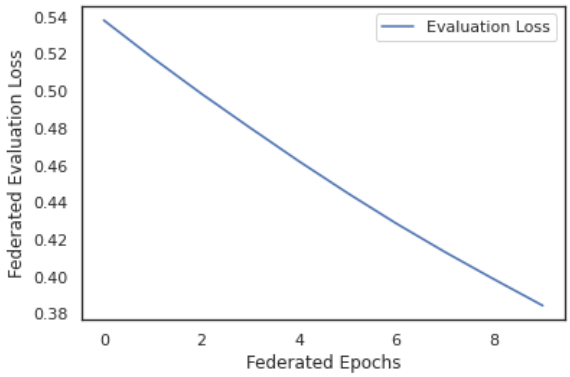

Fig. 5. Federated Loss Convergence

6 Conclusion & Future Scope

SMS Spamming is a problem billions of mobile users face. Spammers bulk spam a large number of people. Innocent people fall into the trap set by spammers by attracting them with exciting rewards and lotteries. In our research study, we have proposed a federated learning based on-device SMS Spam classification approach. We have used the DistilBERT model to classify our SMS. The validation accuracy we achieved is 98%. The SMS received on the user's mobile are sensitive so they will never leave the device premises. Only, the model will train on the user's SMS and its weights will be transferred to the centralized server. As a part of the future scope, we would like to extend this project to predict SMS spam in vernacular languages.

References

1. "HOW MANY SMARTPHONES ARE IN THE WORLD?" https://www.bankmycell.com/blog/how-many-phones-are-in-the-world, Last accessed 19 April, 2022
2. Mishra, S., Soni, D. DSmishSMS-A System to Detect Smishing SMS. Neural Comput Applic (2021).
3. Julis, M., S.Alagesan (2020). Spam Detection In Sms Using Machine Learning Through Text Mining. International Journal of Scientific Technology Research, 9, 498-503
4. Gomaa, Wael. (2020). The Impact of Deep Learning Techniques on SMS Spam Filtering. International Journal of Advanced Computer Science and Applications. 11. 10.14569/IJACSA.2020.0110167.
5. X. Liu, H. Lu and A. Nayak, "A Spam Transformer Model for SMS Spam Detection," in IEEE Access, vol. 9, pp. 80253-80263, 2021, doi: 10.1109/ACCESS.2021.3081479.
6. Rojas-Galeano, Sergio. (2021). Using BERT Encoding to Tackle the Mad-lib Attack in SMS Spam Detection.
7. Sanh, Victor Debut, Lysandre Chaumond, Julien Wolf, Thomas. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.
8. V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam Filtering with Naive Bayes - Which Naive Bayes?". Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006.
9. Almeida, T.A., GÃ³mez Hidalgo, J.M., Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.
10. M. T. Nuruzzaman, C. Lee and D. Choi, "Independent and Personal SMS Spam Filtering," 2011 IEEE 11th International Conference on Computer and Information Technology, 2011, pp. 429-435, doi: 10.1109/CIT.2011.23.
11. McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.
12. Your voice audio data stays private while Google Assistant improves, https://support.google.com/assistant/answer/10176224?hl=enzippy=. Last accessed 19 April, 2022
13. Hard, Andrew Rao, Kanishka Mathews, Rajiv Beaufays, Françoise Augenstein, Sean Eichner, Hubert Kiddon, Chloé Ramage, Daniel. (2018). Federated Learning for Mobile Keyboard Prediction.
14. Lundberg, S. M., Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.
15. Beutel, Daniel J., et al. "Flower: A friendly federated learning research framework." arXiv preprint arXiv:2007.14390 (2020).