Federated Hate-Speech Recognition

Ashkan Vedadi Gargary
University of California, Riverside
aveda002@ucr.edu

Aditya Mohan Gupta
University of California, Riverside
agupt166@ucr.edu

Abstract

Content warning: This paper contains unfiltered content reported by Hate Speech Datasets that may be offensive to readers.

Hate speech recognition is a critical task for maintaining healthy online communities. This paper explores the efficacy of Federated Learning (FL) in enhancing hate speech detection models compared to traditional centralized approaches. We evaluate several state-of-the-art centralized models, including BERT-based models, Random Forest, Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression. Additionally, we introduce two federated methods based on Neural Networks and TinyBERT, facilitating distributed training across multiple devices while preserving data privacy.

Our experimental results, benchmarked using F1 score and accuracy metrics, indicate that FL models outperform classical centralized models by a significant margin, achieving over 6% higher accuracy. Moreover, our FL models perform almost as well as the centralized TinyBERT-based model. These findings underscore the potential of Federated Learning to improve the robustness and performance of hate speech recognition systems, paving the way for more secure and effective content moderation strategies.

1 Introduction

Hate speech recognition is important in natural language processing (NLP) and machine learning (ML), driven by the need to foster safer online environments. The rise of social media has increased the volume of harmful content, challenging community standards and public safety. Effective hate speech detection systems are essential for mitigating harmful rhetoric, protecting vulnerable groups, and maintaining the integrity of online discourse.

Traditional centralized machine learning approaches have been widely used for hate speech detection, including BERT-based models, random forests, decision trees, K-Nearest Neighbors (KNN), and logistic regression. These models are typically trained on large datasets aggregated in a single location, raising concerns over data privacy and security.

In recent years, Federated Learning has become popular, with more people wanting machine learning models that work on many devices without sharing local files. This method allows clients to train models on their data and only share models update, not the actual data, with a central server. This keeps data private and facilitates the learning process.

Federated Learning is particularly beneficial for hate speech recognition. FL harnesses multiple devices' computational power and data diversity without compromising privacy. This decentralized approach reduces risks associated with centralized data collection and enhances the model's generalization ability across varied contexts.

Problem Definition We propose an FL-based model for Hate Speech recognition and show that it can outperform centralized models by achieving over 6% higher accuracy. We also try to compare different methods of Federated-based recognition with TinyBert Classifier. We mainly focus on the datasets that classify a prompt or message into two groups: (i) Hate-Full and (ii) Non-Hate-Full.

Beginning Objectives Our main objectives can be summarized into these groups:

- **Finding Related Datasets:** Hate speech encompasses various sub-problems, such as Racism, Sexism, and Cyberbullying. This work focuses primarily on general hate speech recognition, where content is classified as hateful or not, without focusing on specific subcategories. Our objectives include identifying related datasets and adapting them to our needs, ensuring we have sufficient data

from different sources for fair evaluation in the FL setup.

- **Data Pre-Processing:** Addressing class imbalance is crucial in hate speech detection. Data pre-processing is one of our main goals to ensure the data is suitable for our models. Additionally, for centralized models, we need to merge datasets from different sources.

- **Re-implementing State-of-the-Art Centralized Methods:** Many centralized models, such as Random Forest, KNN, Decision Trees, and Logistic Regression, have shown high accuracy in hate speech recognition. Recent research also utilizes BERT classifiers. Our objective is to re-implement these models to facilitate fair comparisons.

- **Federating Best Centralized Models:** To develop federated models, our first step is to adapt best current centralized models to a federated learning framework.

- **Reproducing Current Federated Models:** Another objective is to reproduce existing state-of-the-art federated hate speech recognition models.

- **Comparing All Models:** Ultimately, we aim to compare all models and discuss their performance fairly.

2 Related Work

Significant research and development have been conducted in the fields of centralized and federated learning, reflecting the evolving priorities in machine learning, particularly regarding data privacy and computational efficiency.

In centralized learning, data is aggregated into a single repository for model training. This traditional approach has facilitated breakthroughs in various domains. For instance, the development of convolutional neural networks (CNNs) by LeCun et al. (1989) revolutionized image recognition tasks, and subsequent improvements by Krizhevsky et al. (2012) with AlexNet demonstrated the power of deep learning on large-scale datasets. Additionally, centralized learning has proven effective in natural language processing with the introduction of the Transformer model by Vaswani et al. (2017), which underpins many modern NLP applications.

Conversely, federated learning (FL) emerged to address privacy and data distribution concerns inherent in centralized learning. Introduced by McMahan et al. (2017), FL enables model training across multiple decentralized devices, sharing only model updates rather than raw data. This approach is crucial in sensitive domains like healthcare, where Sheller et al. (2020) demonstrated the viability of FL in medical imaging, and in mobile applications, as explored by Hard et al. (2018) with Google's Gboard. Recent research has focused on optimizing communication efficiency (Konecny et al., 2016) and improving model aggregation methods (Bonawitz et al., 2019), addressing the unique challenges posed by non-IID data distributions (Zhao et al., 2018).

3 Background

Centralized learning, the traditional method of training machine learning models, involves collecting data from multiple sources into a single location. This centralized approach has the advantage of creating comprehensive datasets, enhancing model accuracy and robustness. However, it raises significant concerns about data privacy, security, and scalability. The requirement to transfer large amounts of data to a central server can lead to privacy breaches and increased vulnerability to cyberattacks.

On the other hand, Federated learning is a decentralized approach designed to mitigate these issues. By keeping data on local devices and only sharing model updates, FL preserves data privacy and reduces the risk of breaches. This method is particularly beneficial in contexts where data is sensitive or subject to strict privacy regulations, such as in healthcare or finance. FL also leverages the computational power of edge devices, potentially reducing the central server's load and enhancing scalability. However, FL introduces new challenges, such as ensuring efficient communication between devices, handling heterogeneous data, and achieving robust model aggregation.

The evolution from centralized to federated learning reflects a broader shift towards prioritizing data privacy and distributed computing power. Both approaches have their merits and are suited to different scenarios, complementing the broader landscape of machine learning research and application.

4 Data

We combine our dataset using the mentioned dataset proposed by (Fortuna et al., 2021).

4.1 Selected Dataset

We reuse 3 of the 9 datasets used by (Fortuna et al., 2021) to form our combine datasets. We perform a stratified split of all training data into training (70%) and testing (30%) sets.

4.1.1 Kaggle (1): Twitter Hate Speech (Agarwal, 2024)

One of the datasets from Twitter combines many different Twitter datasets by having 49159 rows. Their dataset only contains tweets and labels.

4.1.2 Kaggle (2): Cyberbullying (Elsafoury, 2020)

One of the largest datasets from Wikipedia contains 159571 rows. Their datasets contain many different things and provide different datasets for different classification problems. However, we only use *toxicity_parsed_dataset.csv*, which is used for the hate-speech classifier.

4.1.3 Davidson (Davidson et al., 2017)

It is one of the most well-used datasets among all of the existing datasets, including tweets from Twitter. Davidson dataset contains 24783 rows that contain three labels: (i) hate full, (ii) offensive language, and (iii) neither. Since our model's focus is only on hate speech recognition, we remove offensive language texts and only keep the hateful and non-hateful rows. After refactoring the datasets, it has 5593 rows.

4.2 Data Cleaning

We have two aspects of data cleaning. First, as mentioned in Section 4.1.3, we removed offensive language and focused only on hateful and non-hateful. Second, we cleaned each text by converting it to lowercase and removing unnecessary characters (e.g. @, Non-Alphanumeric, URLs).

4.3 Up-sampling

As mentioned in Figure 1, we have a huge class imbalance by checking the count of each label in each dataset. We address issues of extreme class imbalance by augmenting the hateful words and generating new texts on train sets based on them to balance the size of hateful and non-hateful texts. In

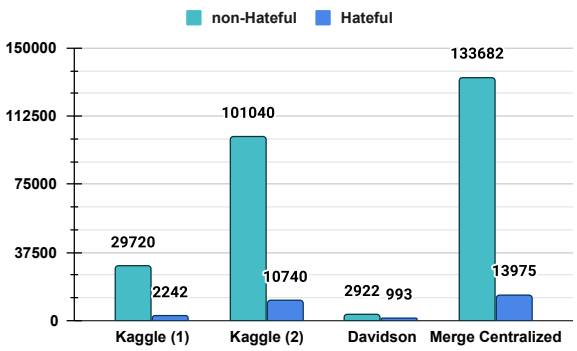


Figure 1: Data Imbalance Among Different Train Set of Each Dataset.

Appendix A, we present data visualization of different classes before and after data augmentation.

5 Centralized Methods

5.1 Decision Trees

The Decision Tree classifier uses a tree-like model of decisions based on the *entropy* criterion to split nodes, optimizing the classification process by maximizing information gain (Song and Ying, 2015). We used the `DecisionTreeClassifier` from the scikit-learn library for our decision tree implementation. To ensure reproducibility, the model was configured with the criterion set to *entropy* and a random state of 0. This setup allows the model to make splits based on the information gain, optimizing the decision-making process at each node.

5.2 Random Forrest

The Random Forest classifier employs an ensemble of many decision trees, using *entropy* as the criterion for splits, to enhance prediction accuracy and robustness through aggregation (Rigatti, 2017). We implemented a Random Forest classifier using scikit-learn's `RandomForestClassifier`. The model was initialized with 10 estimators, using *entropy* as the criterion for information gain calculation and a random state of 0. This ensemble method uses multiple decision trees to improve the robustness and accuracy of the predictions.

5.3 KNNs

The K-Nearest Neighbors classifier classifies data points based on the majority class among the nearest neighbors, utilizing the Minkowski distance metric with a Euclidean distance configuration (Peterson, 2009). We utilized scikit-learn's

`KNeighborsClassifier` for the K-Nearest Neighbors classifier. The model was set with 5 neighbors and employed the Minkowski distance metric with a power parameter (p) of 2, equivalent to the Euclidean distance. This method classifies a sample based on the majority class among its nearest neighbors.

5.4 Logistic Regression

Logistic Regression is a linear model used for binary classification, which predicts the probability of a class membership based on input features, serving as a baseline comparison in this study (LaValley, 2008). The Logistic Regression model was implemented using scikit-learn's `LogisticRegression`, with a random state of 0 to ensure consistent results across runs. This linear model is well-suited for binary classification tasks and was used as a baseline for comparison.

5.5 Tiny-Bert

For our BERT-based model, we employed Tiny-BERT, a compact version of the BERT model (Bhargava et al., 2021; Turc et al., 2019). Tiny-Bert has a total of 4 million parameters and 2 hidden layers with 128 hidden dimension. Tiny-Bert is $7x$ smaller and $9x$ faster than BERT while achieving 96% of its performance (Jiao et al., 2020). We used the `BertTokenizer` and `BertForSequenceClassification` from the Hugging Face library, pretrained on the *prajjwal1/bert-tiny* model with 2 output labels for binary classification. The model was trained using the Trainer API from Hugging Face's Transformers library, with a training configuration that included 5 epochs, a batch size of 16, 10 warmup steps, 0.01 as weight decay, and logging every 10 steps. The training was performed on a GPU.

6 Federated Methods

We propose two main methods for federated implementation. First, classic tokenization and neural network design are used for binary classification with FedProx as aggregated method. Second, we federated our Centralized TinyBert Classifier using PySyft with FedAVG as aggregate method for our training process.

6.1 Federated Learning Setup

Federated learning is employed to train the model across multiple clients without transferring the data to a central server. This approach ensures data privacy and security while leveraging distributed computational resources. Both implementations utilize the same setup, involving the following steps:

- **Client Initialization:** We initialize multiple virtual clients to simulate a federated learning environment. Each client represents a separate entity holding a portion of the training data.
- **Data Distribution:** The training dataset is partitioned and distributed among the clients. Each client receives a unique subset of the data to perform local training.
- **Model Training:** The model is trained locally on each client's data. The local models are updated based on the data subsets they hold.
- **Model Aggregation:** After local training, the updated model parameters are sent to a central server for aggregation. The server averages the parameters to create a global model. The Federated Averaging (FedAvg) algorithm is used to combine the model updates from all clients.
- **Iterative Process:** The training and aggregation steps are repeated iteratively over multiple epochs until the model converges or achieves satisfactory performance. We set the number of iterations to 10.

This federated learning setup ensures that the model can learn from a diverse dataset while preserving the privacy of each client's data. By keeping the data local and only sharing model updates, we mitigate the risks associated with data breaches and ensure compliance with privacy regulations.

6.2 Federated Neural Networks with FedProx

The FedProx algorithm is implemented to add a proximal term to the loss function. This term penalizes deviations from the global model parameters, thus stabilizing training in heterogeneous environments.

6.2.1 FedProx

The implemented model for hate speech detection is a simple neural network designed for binary classification. The architecture consists of:

- **Input Layer:** This layer accepts feature vectors derived from the preprocessed text data.

- **Hidden Layers:** One or more hidden layers with ReLU (Rectified Linear Unit) activation functions to introduce non-linearity and enable the model to learn complex patterns in the data.

- **Output Layer:** A single neuron with a sigmoid activation function to output the probability of the text being classified as hate speech. The sigmoid function is suitable for binary classification as it maps the output to a value between 0 and 1.

The choice of a simple neural network architecture is motivated by the need for efficient training and inference in a federated learning setting, where computational resources on client devices might be limited. The architecture is flexible enough to be extended with additional layers or neurons if required to handle more complex data or achieve higher performance.

6.3 Federated TinyBert with FedAVG

We utilize our best centralized model, TinyBERT, for our federated setup. By using virtual machines in the PySyft library, we simulate the federated environment. For our BERT-based model, we employed Tiny-BERT, a compact version of the BERT model (Bhargava et al., 2021; Turc et al., 2019). As mentioned earlier, Tiny-Bert has a total of 4 million parameters and 2 hidden layers with 128 hidden dimension. Tiny-Bert is $7x$ smaller and $9x$ faster than BERT while achieving 96% of its performance (Jiao et al., 2020). We used the `BertTokenizer` and `BertForSequenceClassification` from the Hugging Face library, pretrained on the *prajjwal1/bert-tiny* model with 2 output labels for binary classification. The model was trained using the Trainer API from Hugging Face's Transformers library, with a training configuration that included 5 epochs, a batch size of 16, 10 warmup steps, 0.01 as weight decay, and logging every 10 steps. The training was performed on a GPU.

7 Experiments and Results

This section presents and compares our proposed Federated and Centralized evaluation results with each other. All the source code and datasets are available online for results reproduction, mentioned in Section B.

Table 1: Data Information

	Train			Test
Data	**#Size**	**#non-Hateful**	**#Hateful**	**#Size**
Kaggle (1)	31962	29720	2242	17197
Kaggle (2)	111780	101040	10740	47905
Davidson	3915	2922	993	1678
Merged	147657	133682	13975	66780

7.1 Experiment Setup

In this section, we summarize the Hardware and Libraries that we utilized. We also review the datasets and comparison methodologies for both centralized and federated setups.

7.1.1 Hardware and Software

We utilized the free version of the Google Colab research system for our experimental results. We employed scikit-learn libraries to implement the centralized models, Hugging Face to fine-tune the necessary models, and the Syft library to incorporate federated learning into our code.

7.1.2 Dataset

As mentioned in Section 4, we have three different datasets: (i) Kaggle (1), (ii) Kaggle (2), and (iii) Davidson. By pre-processing and up-sampling the data, we have three training sets for both centralized and federated methods. We also create one additional dataset, which results from merging these three datasets, and call it (iv) the Merged dataset. More information on the dataset is summarized in Table 1.

7.1.3 Approaches

For centralized approaches, we compare Decision Tree (DT), Logistic Regression (LR), K-nearest neighbors (KNN), Random Forest (RF), and Tiny-Bert (TB) classifier. According to our resources, the TinyBert classifier is the best available mechanism. Then, we use the Bert classifier (TB) and compare it with our Federated Neural Network and Federated TinyBert model.

7.2 Centralized Methods

In this section, we focus on centralized methods to find the best-centralized methods. First, we compare the performance of different clients.

7.2.1 Each Dataset

As shown in Table 2, all models perform strongly on the validation set. This is mainly due to the upsampling applied during implementation, which,

Table 2: Compare the Train and Accuracy of each dataset.

Datasets	Model	Validation (20%)		Test (30%)	
		Accuracy	F1	Accuracy	F1
Kaggle (1)	DT	97.49%	97.40%	67.24%	79.89%
	LR	94.03%	93.90%	51.86%	66.62%
	KNN	96.57%	96.43%	67.24%	79.89%
	RF	98.67%	98.64%	71.03%	82.75%
	TB	**98.36%**	**98.36%**	**95.33%**	**95.47%**
Kaggle (2)	DT	46.70%	52.61%	51.82%	66.60%
	LR	90.83%	91.01%	78.67%	87.91%
	KNN	90.09%	89.18%	87.33%	93.22%
	RF	98.44%	98.42%	58.85%	73.06%
	TB	**97.73%**	**97.72%**	**94.00%**	**94.43%**
Davidson	DT	95.95%	95.80%	54.35%	67.90%
	LR	96.12%	96.00%	55.42%	68.52%
	KNN	85.29%	86.05%	54.35%	67.90%
	RF	96.01%	95.86%	54.89%	68.34%
	TB	**92.13%**	**92.13%**	**91.90%**	**91.96%**
Merge	DT	96.36%	96.24%	60.78%	74.59%
	LR	94.96%	97.26%	85.35%	88.04%
	KNN	91.46%	95.37%	60.78%	74.59%
	RF	98.12%	98.09%	73.87%	84.61%
	TB	**97.20%**	**97.20%**	**93.31%**	**93.96%**

while beneficial for training, tends to increase the risk of overfitting. Therefore, we also report test accuracy and F1 scores for a more balanced evaluation.

Random Forest (RF) achieves the best performance among classical machine learning models. However, the performance gap is substantial compared to the TinyBERT classifier, which outperforms all other classical machine learning models by a significant margin.

TinyBERT achieves over 91% accuracy across all three datasets. One notable exception is the Davidson dataset, which shows lower performance. This is because, after removing one of its classes to fit our binary classification problem (0 for non-hateful and 1 for hateful), the dataset is left with fewer instances, affecting the model's performance.

7.2.2 Merged Datasets

To compare our models using merged datasets, we combined all three datasets into a single dataset and reran all the experiments. We achieved an accuracy and F1 score of 93%. Notably, the validation and test accuracy and F1 scores are almost identical, indicating no signs of overfitting in the results.

One interesting result is the increase in the Accuracy and F1 score in classical ML due to merging the datasets.

We will further compare the federated methods with the Centralized TinyBERT classifier.

Table 3: Centralized and Federated Models Comparison

Training Model	Aggregation Method	Test Accuracy
Centralized DT	N/A	60.78%
Centralized RF	N/A	73.87%
Centralized LR	N/A	85.35%
Centralized KNN	N/A	60.78%
Centralized TinyBert	N/A	93.31%
Federated Neural Network	FedProx	90.35%
Federated TinyBert	FedAVG	90.80%

7.3 Federated Methods

In this section, we first compare various federated methods. Subsequently, we contrast the performance of the best federated setup with the TinyBERT classifier on merged datasets, representing the best-centralized method.

7.3.1 Federated Hate-Speech Recognition Comparison

We applied the Federated Neural Networks model with FedProx aggregation to binary hate speech detection datasets. The model was trained over several iterations (ranging from 1 to 10), with accuracy recorded after each iteration. The training process demonstrated that the FedProx algorithm effectively maintained high accuracy while preserving data privacy across multiple clients. The final accuracy achieved was satisfactory, confirming the efficacy of this approach.

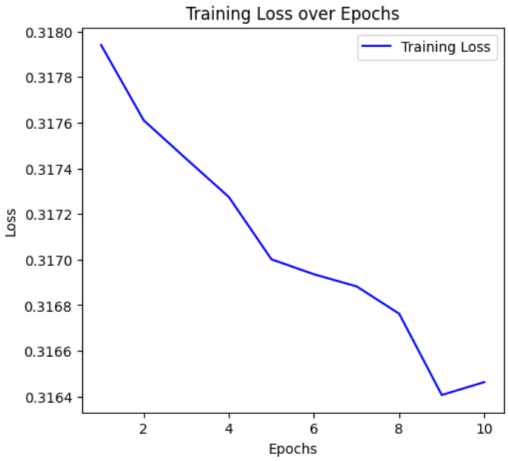
To visualize the training progress, Figure 2 shows the accuracy and loss over different iterations. The graph illustrates the improvement in model accuracy with each epoch, highlighting the stability and effectiveness of the FedProx algorithm in a federated setting.

In contrast, we applied Federated TinyBERT with FedAVG aggregation to the hate speech detection datasets over 10 iterations. Due to GPU resource limitations, we avoided upsampling. Figure 3 demonstrates the increase in accuracy over 10 iterations.

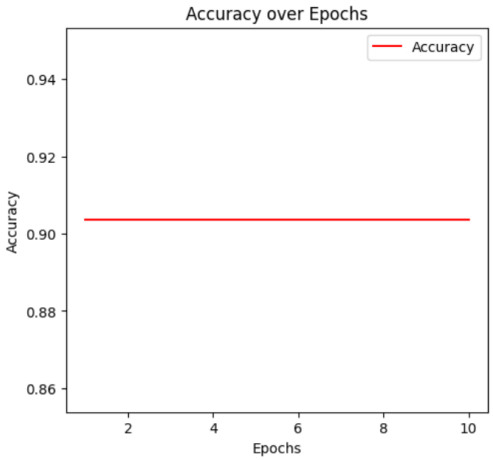
Both models achieved 92% accuracy on the test sets after 10 iterations.

7.3.2 Centralized Vs. Federated

As shown in Table 3, our implementation of both federated methods outperforms traditional centralized methods. Additionally, when compared to the centralized TinyBERT, our federated approaches achieve nearly the same accuracy after 10 iterations, while providing enhanced privacy.

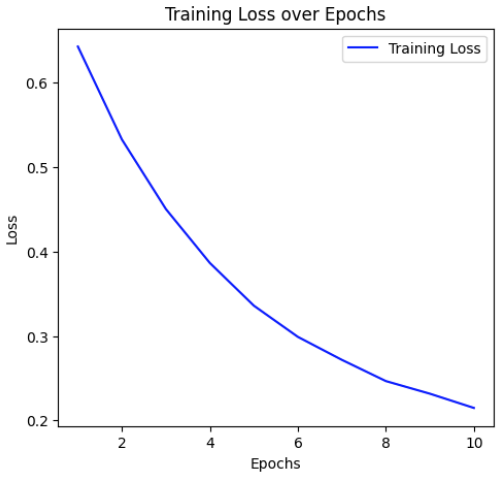


(a) Training Loss over 10 iterations

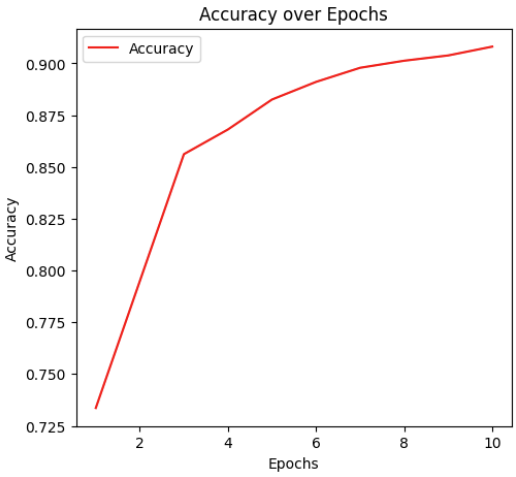


(b) Accuracy of Model over 10 iterations

Figure 2: Training Loss (a) and Accuracy (b) over 10 iterations of Federated Neural Networks Model



(a) Training Loss over 10 iterations



(b) Accuracy of Model over 10 iterations

Figure 3: Training Loss (a) and Accuracy (b) over 10 iterations of Federated TinyBert Classification

8 Conclusion & Discussion

In this section, we summarize the main results of our study and discuss the limitations and challenges we encountered.

8.1 Discussion

The results of this study highlight the significant advancements in the field of hate speech detection through the use of both centralized and federated learning models. Our experiments demonstrate that centralized models, particularly TinyBERT, achieve high accuracy across multiple datasets. However, the federated learning approaches offers a compelling alternative by maintaining high accuracy while ensuring data privacy and security across multiple clients. This dual approach showcases the potential for balancing performance with privacy, a critical consideration in the era of data-driven decision-making.

8.2 Revisiting Objectives

The initial objectives of this research were to identify relevant datasets, preprocess data to address the class imbalance, re-implement state-of-the-art centralized methods, adapt these models to a federated and centralized learning framework, reproduce current federated and centralized models, and compare all models fairly. Throughout the study, we successfully identified and utilized multiple datasets for hate speech detection, implemented several centralized models, including TinyBERT, and developed federated versions of these models. The comparative analysis provided insights into the trade-offs between centralized and federated approaches, fulfilling our primary research goals.

8.3 Implications

The implications of this study extend to both practical applications and future research directions. The successful implementation of federated learning models for hate speech detection suggests that organizations can deploy these models to improve content moderation while safeguarding user privacy. This is particularly relevant for social media platforms and other online communities. From a research perspective, the study underscores the need for further exploration of federated learning techniques, especially in handling non-IID data and enhancing model aggregation methods. The findings also highlight the importance of balancing model performance with ethical considerations such as privacy and security in machine learning applications.

8.4 Limitations and Challenges

During our implementation, we encountered several limitations, which are listed below.

- **Resource Limitation:** One of the main problems we encountered was related to resource limitations. First, the data size is very large, especially when we merged the datasets. Second, using TinyBERT requires significant GPU power. Since we used the free resources of Google Colab, which has a small amount of RAM and limited GPU time access, we could not experiment with more robust results using a larger number of clients and more epochs.
- **SOTA Comparison:** Due to time limitations, we could not compare other state-of-the-art methods and reproduce more SOTA papers.
- **Other NLP Models:** There are many other encoders for classifying each tweet, which we could not implement due to a lack of resources. Additionally, TinyBERT is a small model, and using the base BERT model, which is much larger, would provide a broader perspective for our work.
- **Server-Client Design FL:** Due to time limitations, we only considered a virtual machine federated setup without implementing the real server-client model transfer

8.5 Future Work

Working on Federated Hate-Speech Recognition is an interesting topic that has many open research problems.

- **Effect of Client Size:** The effect of the number of clients is important in federated settings. Due to the mentioned limitations, testing with a larger number of clients was not feasible.
- **Multi-Class Classification:** Our main focus is Federated Learning for binary classification. However, there are many centralized models for multi-class classification, and feeding different class types can also be a challenging problem.
- **Combining Generative Models for Rare Classes:** Hate-speech datasets suffer from imbalanced labels. Using deep generative models for each client can help generate rare labels.
- **Federated Generative Models:** Hate-speech recognition involves tabular data, and using Generative Models in a federated manner can be an interesting research problem for further exploration.
- **Privacy and Security:** Further research is needed on the possibility of integrity and privacy attacks when using hate-speech datasets in a federated learning manner. Comparing existing methods regarding privacy and security is crucial.
- **Different Aggregation Methods:** Currently, we only consider FedProx and FedAVG as our aggregation methods. Many other well-performing aggregation methods exist and can be helpful.
- **Non-IID Data:** Currently, we only consider IID data in our implementation. However, working with non-IID data is another possible area for exploration.
- **Flower:** Flower is also one of the famous Federated Learning frameworks with many different setups. Using their implementation is also recommended for future works.

References

Rahul Agarwal. 2024. Twitter hate speech data. https://www.kaggle.com/datasets/vkrahul/twitter-hate-speech/data. Kaggle.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.

Fatma Elsafoury. 2020. Cyberbullying datasets. https://doi.org/10.17632/jf4pzyvnpj.1. Mendeley Data, V1.

Paula Fortuna, Juan Soler-Company, and Leo Wanner. 2021. How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets? *Information Processing Management*, 58(3):102524.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding.

Michael P LaValley. 2008. Logistic regression. *Circulation*, 117(18):2395–2399.

Leif E Peterson. 2009. K-nearest neighbor. *Scholarpedia*, 4(2):1883.

Steven J Rigatti. 2017. Random forest. *Journal of Insurance Medicine*, 47(1):31–39.

Yan-Yan Song and LU Ying. 2015. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

A Data Visualization

One of the experiments that we did before data augmentation was data visualization for different setups before and after up-sampling. Figure 4, Figure 5, and Figure 6 are representative of each class data visualization based on different words.

B Provided Materials

We have provided our code on GitHub, which you can find at https://github.com/ashkanvg/Hate-Speech-Recognizer. Also, all of the datasets are available at Google Drive.

C Individual Efforts

In this section, we summarize the main contribution of each person in the group.

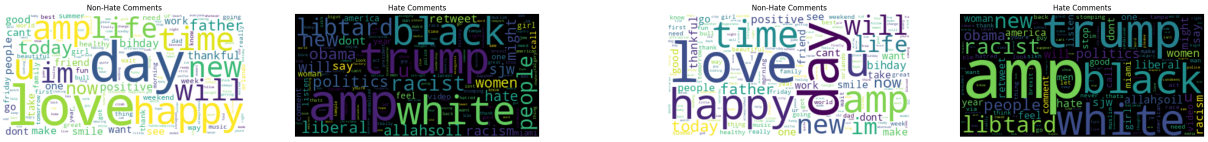
C.1 Ashkan Vedadi Gargary

My contributions to the project were primarily focused on the centralized learning aspect, emphasizing the exploration of different models and leveraging the BERT classifier. Additionally, I was responsible for federating the best-centralized model (TinyBERT) and evaluating its accuracy in a federated learning setup. Part of my role also involved reviewing related works to identify relevant datasets and models for further research.

Throughout the project, collaboration was key. We worked closely as a team, with each member making equal contributions to the project report and various parts of the implementation. We supported each other in overcoming challenges, ensuring a smooth and effective workflow.

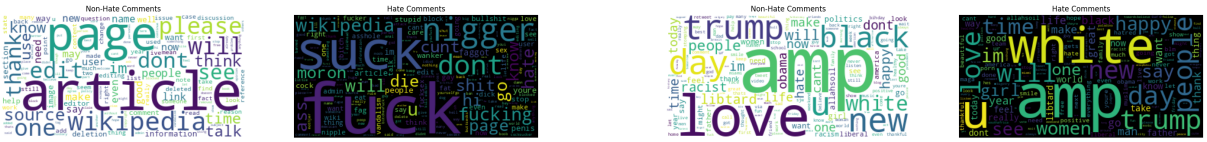
C.2 Aditya Mohan Gupta

Focused on the federated learning aspect of the project, with an emphasis on hate speech detection. Conducted an extensive review of various datasets suitable for federated learning applications. Undertook a thorough analysis of existing literature to comprehend the implementation and mechanics of federated learning frameworks. Developed and implemented the code for training and evaluating the federated learning model. Collaborated closely with Ashkan, who provided valuable assistance throughout the project. Contributed significantly to the composition of the project report and the preparation of the presentation materials.



(a) Before Up-Sampling (b) After Up-Sampling

Figure 4: Kaggle(1) Dataset Each Class Visualization



(a) Before Up-Sampling (b) After Up-Sampling

Figure 5: Kaggle(2) Dataset Each Class Visualization

(a) Before Up-Sampling (b) After Up-Sampling

Figure 6: Davidson Dataset Each Class Visualization