Privacy–Preserving Online Content Moderation: A Federated Learning Use Case

Pantelitsa Leonidou
Cyprus University of Technology
Limassol, Cyprus
pl.leonidou@edu.cut.ac.cy

Nicolas Kourtellis
Telefonica Research
Barcelona, Spain
nicolas.kourtellis@telefonica.com

Nikos Salamanos
Cyprus University of Technology
Limassol, Cyprus
nik.salaman@cut.ac.cy

Michael Sirivianos
Cyprus University of Technology
Limassol, Cyprus
michael.sirivianos@cut.ac.cy

ABSTRACT

Users are exposed to a large volume of harmful content that appears daily on various social network platforms. One solution to users' protection is developing online moderation tools using Machine Learning (ML) techniques for automatic detection or content filtering. On the other hand, the processing of user data requires compliance with privacy policies. In this paper, we propose a framework for developing content moderation tools in a privacy-preserving manner where sensitive information stays on the users' device. For this purpose, we apply Differentially Private Federated Learning (DP–FL), where the training of ML models is performed locally on the users' devices, and only the model updates are shared with a central entity. To demonstrate the utility of our approach, we simulate harmful text classification on Twitter data in a distributed FL fashion– but the overall concept can be generalized to other types of misbehavior, data, and platforms. We show that the performance of the proposed FL framework can be close to the centralized approach – for both the DP–FL and non–DP FL. Moreover, it has a high performance even if a small number of clients (each with a small number of tweets) are available for the FL training. When reducing the number of clients (from fifty to ten) or the tweets per client (from 1K to 100), the classifier can still achieve ~81% AUC. Furthermore, we extend the evaluation to four other Twitter datasets that capture different types of user misbehavior and still obtain a promising performance (61% – 80% AUC). Finally, we explore the overhead on the users' devices during the FL training phase and show that the local training does not introduce excessive CPU utilization and memory consumption overhead.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

KEYWORDS

content moderation, federated learning, privacy

ACM Reference Format:
Pantelitsa Leonidou, Nicolas Kourtellis, Nikos Salamanos, and Michael Sirivianos. 2023. Privacy–Preserving Online Content Moderation: A Federated Learning Use Case. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543873.3587604

WWW '23 Companion, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9419-2/23/04.
https://doi.org/10.1145/3543873.3587604

1 INTRODUCTION

Users of all ages are exposed to a large volume of information from various Online Social Networks (OSNs). The content is often questionable or even harmful regardless of age, expressing abusive behavior, extreme sarcasm, cyberbullying, racism, and offensive or hate speech. Although mainstream OSN platforms claim they do their best to protect the users, harmful content is still present. The platforms' business model often dictates the applied rules and policies and, consequently, to what extent they monitor and control the content. Misbehavior can be profitable. Allowing users to be impulsive increases their engagement with the platform and the freshness of the available content, even if it is borderline harmful. Moreover, some platforms perform minimum content moderation to attract a specific audience (see 4chan).

Researchers and developers have made a great effort to develop automated content moderation tools mainly based on Machine Learning (ML) algorithms [5, 7, 11, 25, 31]. These state-of-the-art methods collect data, annotate them, and then train and test the models in a centralized approach. It is challenging to collect, process, and annotate large datasets suitable for deep learning training. The data come from millions of users, are multi-modal (text, video, and images or a combination of those), and change dynamically. The users' online data can be private and sensitive, so the EU has imposed strict policies to protect users' privacy (GDPR and accompanying national legislation).

In this paper, we propose a privacy–preserving Federated Learning (FL) framework for detecting harmful online content. We envision a system that gives power to the users to (i) control the moderation done in the system in a personalized fashion and (ii) protect their privacy. Specifically: **(a)** The platform may not be trusted to moderate inappropriate content. Some mainstream platforms, such as Twitter, provide some content moderation, whereas other more fringe, do not provide any moderation. **(b)** The way the platform does online moderation using its own labels – based on its definition of misbehavior – may not satisfy the users' needs. Therefore, we expect the users to label content they consider as "harmful". At the same time, they might not want to share these labels since they will reveal the type of content they read or receive. We readily admit the personalization inherent in the FL scheme possibly exacerbates the echo chambers effect in online social network platforms. On the other hand, the side benefit of personalized moderation is that it is tailored to the sensitivity of the users, and this makes moderation more acceptable and adoptable by the users.

(c) Even if some OSN data are publicly available, not all may be truly public. These data can be private posts, messages, or results of recommendation algorithms. Often the true users' identities can be actually hidden – they do not match their account usernames.

Our framework is on-purpose generic as it can be used for several types of content, such as text, audio, video, or images. However, in this paper, we focus on "harmful" text classification on Twitter – as a proof of concept demonstrating the utility of our approach. Although the FL paradigm complies, in theory, with the GDPR policies (since the raw data never leave the users' devices), privacy leakages can still occur. Prior studies have shown that FL is vulnerable to privacy attacks [17]. A proposed solution is Central Differential Privacy (CDP) – an adaptation of Differential Privacy (DP) [8, 10] for the FL framework. CDP provides privacy guarantees (at the user–level) against membership inference attacks [3, 13, 19]. This has been empirically verified in [21].

To answer our central research question, we bootstrap the ML text classifier presented in [12], and then incorporate the CDP model proposed in [3]. We evaluate it when trained in an FL fashion (with and without DP) on different text datasets from five studies of Twitter user misbehavior by generalizing the classification problem as detecting harmful or normal behavior. We compare the classifier's FL performance with the centralized version with access to all data. The implementation and the experiments serve as proof of concept, demonstrating that the proposed framework is feasible. Finally, we assess a typical user device's overhead while training the classifier locally to examine whether the FL approach slows down the device. This work makes the following **contributions**:

- We are the first to propose a framework where we apply privacy–preserving FL in the context of harmful content detection applicable in different OSN platforms. For this purpose, we instantiate this framework for the case of Twitter and provide a simulation process that utilizes existing Twitter datasets to test the performance of an FL framework.
- We show that the performance of the proposed FL framework can be close to the centralized approach – for both the DP and non–DP FL versions. The FL classification performance on a total of 50K tweets has only a 10% difference in AUC compared to the centralized approach. For instance, by training the classifier (without DP) for only 20 FL rounds on 50 clients, we achieve ∼83% AUC. Moreover, when reducing the number of clients (from 50 to 10) or the tweets per client (from 1K to 100), the classifier can still achieve ∼81% AUC. In other words, we can achieve high performance even if few clients (with few data points locally) are available.
- Our further evaluation of the classifier on four smaller Twitter datasets of other types of misbehavior shows promising performance, ranging from 61% to 80% AUC. This means that the classifier can generalize and detect different types of misbehavior.
- Finally, we show that the FL training process does not introduce excessive system overhead – in terms of CPU utilization and memory consumption - on the users' devices.
- The results, together with the experimental–evaluation code is publicly available[1].

[1]https://github.com/pleonidou01/FL-Online-content-moderation.git

2 RELATED WORK

2.1 Automatic Detection and Filtering of Harmful Content

Harmful content can be found in a text, visual (image, video), audio (songs, recordings) format, or a combination of those. We define any violent, abusive, sexual, disrespectful, hateful, illegal content, or any content that may harm the user as **"harmful"**. One solution to protect users from such content is adopting automatic detection or filtering using ML techniques in online moderation tools.

Several studies have investigated misbehavior on Twitter. [5] proposes a deep-learning architecture to classify various types of abusive behavior (bullying and aggression) on Twitter. Then, they applied the methodology to a large dataset of 1.6M tweets. [12] presents a unified deep learning classifier to detect abusive texts on Twitter. The authors tested the unified classifier with several abusive Twitter datasets and achieved high performance. One of the evaluation datasets was the one presented in [11] with 100K tweets labeled as "Abusive", "Hate", "Normal", and "Spam" using crowdsourcing annotation techniques. The unified classifier consists of two different classifiers whose results are combined to give the final result. One classifier is a text classification model, and the other treats domain-specific metadata (i.e., user's friend network, number of retweets, etc.). In this work, we adopt a simplified version of the proposed classifier by replicating the model for the text classification task – since we use no meta-data as training input but only text stored on a user's device.

Yenala et al. proposed a deep learning architecture for detecting inappropriate language in query completion suggestions in search engines and users' conversations in messengers [34]. They prove that the suggested architecture outperforms pattern-based and hand-crafted feature-based architectures. The authors in [2] collected a dataset of $\sim 4M$ records to assess the exposure of kids and adolescents to inappropriate comments on YouTube. They built a model consisting of five high-accuracy classifiers to classify the comments obtained into five age-inappropriate classes (Toxic, Obscene, Insult, Threat, Identity hate). The model acts as a binary classifier that classifies input as inappropriate if it falls into at least one of the five classes. Papadamou et al. built a deep learning classifier to detect videos with inappropriate content that targets toddlers on YouTube with high accuracy (84.3%) [22]. The authors in [27] created a dataset with three different categories of videos: "Original Videos", "Explicit Fake Videos", and "Violent Fake Videos". They trained a deep learning classifier to detect videos with content inappropriate for kids with an accuracy of more than 90%. Additionally, Papadamou et al. collected $\sim 7K$ YouTube videos related to pseudoscientific content and used the resulting dataset to train a deep learning classifier to detect misinformation videos on YouTube and achieved an accuracy of 79% [23]. These studies used video processing techniques to extract information from the videos but also collected other related information (e.g., video title, comments, caption, etc.).

2.2 Federated Learning and Differential Privacy

McMahan et al. introduced Federated Learning (FL) as a distributed approach for training machine learning models without sharing an

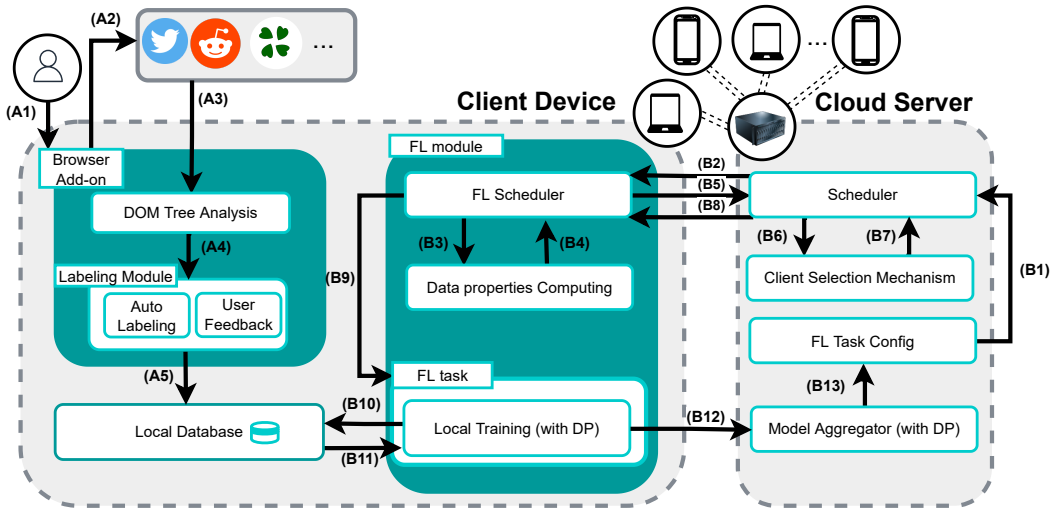


Figure 1: Framework for privacy-preserving online content moderation using differentially private Federated Learning – *A1 to A5:* The Client Device accesses an OSN application, it filters specific user's activity to store locally the data with a label defined after the aggregation of the automatically set and the user-provided label. *B1 to B13*: To initiate an FL learning task, the FL Task Config module sends the task description (i.e., the baseline model, criteria for participation, etc.) to the Scheduler that communicates with the available FL clients. The Data properties Computing module computes the metadata of the user's local dataset and device. The Client Selection Mechanism tells if the client will participate in the training or not based on the set of criteria defined in the task description. For the participating clients, the FL Task module executes the local training. By the end of the local training, the model's update is sent to the Model Aggregator which aggregates all the clients'updates, and applies the aggregated update to the global model. For more details see Appendix A.

individual's data with a central unit [18]. The idea is to train local models on clients' devices with their on-device available data and only share locally-computed updates with the central server. The server will collect the locally computed updates from the clients and aggregate them to update the global model. A client device in an FL setting can scale from a mobile device, a laptop, a desktop, or an IoT device to a company's data server.

Since the FL appearance, many studies have described FL applications in real settings. Gboard [33] uses FL for training, evaluating, and deploying a model for giving optimized web, GIFs, and Stickers query suggestions. Gboard also used FL to train a model for next-word prediction[14]. Next word prediction is used on the keyboard to suggest words for the user to type next based on the text already typed. In [6], the authors applied FL to train a neural network to learn out-of-vocabulary (OOV) words to minimize annoying users by auto-correcting the OOV words considering them as misspellings. FL is also used to train an image-classification model to decide whether a patient has the COVID-19 virus or not using x-ray images from several hospitals to preserve the patients' privacy in [32]. The performance obtained when training the models using FL was slightly worse than training using a centralized approach.

Several studies have shown that maintaining the raw data locally does not sufficiently protect the users' privacy in the FL framework [17]. An adversary with access to the FL-trained model's parameters can reveal private user information. The adversary can be (i) one of the other clients – or even the central aggregator – during the training phase and (ii) an external attacker who has access to the final trained model.

One solution to providing privacy guarantees to ML model's training tasks is the concept of Differential Privacy (DP). The DP was first introduced by [8–10] as a privacy–preserving technique for learning tasks on statistical databases. It can limit privacy leakages regarding the data records used for the learning phase. This means that an adversary, who has access to the model's parameters, cannot decide whether a data record is part of the model's training dataset. These privacy guarantees –at the record level– are achieved by adding noise to the learning process to limit the data records' influence on the algorithm's final output.

In the FL settings, a user's dataset may contain sensitive information. Therefore, it is important to provide privacy guarantees at the user level. This can be achieved by adapting the definition of DP with the notion of user-adjacent datasets, instead of record-adjacent datasets as proposed in [13, 19]. An FL training task with user-level DP guarantees ensures that an adversary cannot tell whether an individual's data is part of the total data used for training the model, i.e., limit a user dataset's influence on the output of the training task.

Two main variations of DP methodology have been incorporated into the FL framework toward privacy–preserving FL: the Central Differential Privacy (CDP) and the Local Differential Privacy (LDP) [21]; other hybrid approaches have also lately proposed [4]. In CDP, the agents send the model updates to the central server, which will perform the DP noise addition [3]. This implies that the

central server is a trusted system entity; it will not perform malicious inferences on the clients' data. In LDP, the DP noise addition is performed locally by the clients – before sending the updates to the central server [28]. In this context, no trusted entity is required.
Our contribution: In this paper, we propose a methodology of content moderation in a privacy-preserving fashion (using differentially private FL). We evaluate our approach on five Twitter datasets (with harmful content) using a variation of the text classifier proposed in [12]. The overall framework is easily applicable in other social media platforms (i.e., YouTube, Reddit, 4chan) and for different types of misbehavior. This can be achieved by incorporating ML algorithms from existing works [2, 5, 12, 22, 23, 27, 34].

3 CONCEPTUAL FRAMEWORK

To further explain the idea of applying the differentially private FL paradigm to online moderation tools, we present our proposed framework in Figure 1. Regarding the **threat model** we assume that the only trusted entity is the central aggregator. Under the Central Differential Private protocol [3] – that we use in this study – the central aggregator is responsible for adding the noise before aggregating the model updates that receives from the clients in an FL round to achieve user-level DP guarantees. This implies that the aggregator is a trusted entity, but the other participants may not be. Hence, possible adversaries are either some clients or an external entity that tries to reveal private user's information by performing membership inference attacks either during the training phase or through the final global model. In Appendix A there is a detailed description of the system components and the data-flow of our proposed framework.

4 FL SIMULATION PIPELINE

4.1 General Assumptions

Since we do not have access to the raw Twitter data from millions of users, the true distribution of harmful tweets to users is unknown. Thus, we have to simulate the users' browsing history somehow. For this purpose, we construct artificial clients by splitting a centralized Twitter dataset containing harmful tweets into a number of disjoint sets. Moreover, we study a homogeneous population of clients (with either IID or non–IID data), namely, all clients have the same number of total tweets with the same ratio harmful to normal (i.e., that same class ratio). Additionally, we assume that clients selected for FL training remain available during the whole FL process.

4.2 FL Training Simulation

We used TensorFlow Federated (TFF), an open–source framework for computations on decentralized data[2], to simulate the FL training process for our experiments. The FL algorithm we used for aggregating the client's model updates is the Federated Averaging [18]. TFF provides the implementation[3] of Central Differential Privacy that we use in our simulation to adopt a variation of the Federated Averaging algorithm that achieves user-level DP guarantees. Figure 2 presents our pipeline to simulate the FL training. We describe next the FL simulation pipeline's steps and main components.

[2]https://www.tensorflow.org/federated
[3]https://github.com/tensorflow/privacy

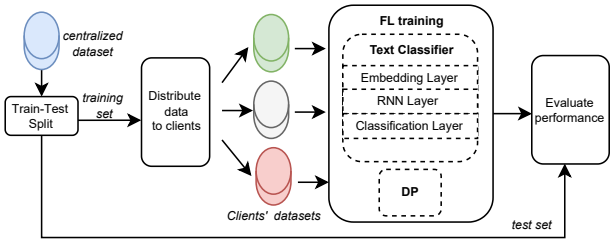


Figure 2: Federated Learning Simulation Pipeline

4.3 Text classifier

We use a simplified version of the unified classification model described in [12], where only the text-classification path is enabled. We used this classifier since it showed a high performance (~80% to ~93% AUC) across many harmful tweet datasets. We used this simplified version to give a lighter computational task to the user's device, and to use features readily available from the tweets (i.e., not relying on offline-computed features based on social network properties of users as in [12]). The input of the classifier is the tweets' text. We used TensorFlow Keras for the implementation of the classifier. The sequential ML pipeline starts with an Embedding layer, we use the GloVe embedding [24] with the highest dimension (200). A Recurrent Neural Network Layer follows with gated recurrent unit (GRU), 128 units, and a dropout of p=0.5. The output layer is a classification dense layer, with one neuron with the sigmoid activation function. TFF framework offers a function that wraps a Keras model[4] for its use in the federated training simulation.

4.4 Creating artificial clients for FL

We need a decentralized dataset with a sufficient number of harmful and normal texts to simulate the FL training of the text classifier. Since there is no such dataset fulfilling our criteria, we convert existing centralized datasets from past studies into artificial federated datasets. For this purpose, given a dataset with two classes of tweets (harmful and normal) and a sufficient number of harmful tweets, we do the following:

First, we create a test set with a size the 10% of the dataset, with the condition that 8% of the tweets in the test set are harmful. In other words, the class ratio harmful:normal in the test set is 8:92. We apply this percentage (8%) based on the results of previous studies [5, 11] that showed that the percentage of harmful content on Twitter is around ~8%. Then, we create the clients using the remaining 90% of the dataset. In our simulation, the clients are represented by sets of tweets (the clients' local data). To evaluate the FL on different populations of clients, we control the class ratio in clients' data, i.e., harmful:normal. In this way, we experiment on IID data (i.e., 50:50 ratio –balanced datasets) as well as emulate scenarios of non-IIDness (see Section 5.4.1) by having unbalanced datasets on FL clients devices (e.g., 10:90). We also set the total number of tweets per client. Finally, given the clients' class ratio and clients' data size, we compute the maximum number of clients we can construct.

[4]https://www.tensorflow.org/federated/api_docs/python/tff/learning/from_keras_model

5 EXPERIMENTAL EVALUATION

5.1 Training Setup

To address the research questions of this work, we conducted experiments having the following training setups:

FL training: For the FL training setup, we are following the method described in Section 4.4 – given the parameters (clients' data size, percentage of harmful tweets) – to construct the federated dataset. Then, we set the FL rounds and the number of participating clients in each round. Finally, we use the TFF framework to simulate the FL training. We refer to *Local training* as the training of the model on the client's device, using the client's whole dataset as the local training set.

Centralized training: This is the traditional ML training setup where the text classifier is trained with a single train set: this is the best-case scenario in which an OSN platform decides to apply content moderation. Regarding the train–test split, we construct the test set following the same procedure described in Section 4.4. That is, we initially split the dataset into a test set of 10% size with class ratio 8:92 (i.e., 8% harmful tweets). Then, from the remaining 90% of the dataset, we construct the train set. We set a class ratio and a training–set size, and then we randomly select a subset of tweets that satisfies these properties.

In both setups, we train the text classifier described in Section 4.3, and we compute the weighted classification metrics[5]. We set the parameters (epochs=7, batch size=10, Adams optimizer, learning rate=0.001) after experimenting with different values for tuning and applying early stopping. We run all the experiments on a server with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, and a 62GiB RAM except for the "overhead on client's device" (Section 5.7) which we run on a Dell laptop device with Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz and 8GB RAM.

5.2 Experimental questions

We experiment with different values of the simulation parameters to explore how they affect the FL classification performance. These will also give us insights into the effective client selection and FL–training strategy for online content moderation. For this purpose, we investigate the following research questions:

Q1: How many harmful tweets per client are needed for training–efficient FL?

We address this question by controlling the size of the harmful class on each client's dataset. We consider a homogeneous population with the same class ratio (harmful:normal). Generally, as studies showed, ∼8% of Twitter's online content is harmful [5, 11]. That said, there are often controversial topics where the users' behavior is highly polarized. For instance, COVID-19 vaccination, the Russian invasion of Ukraine, and several conspiracy theories. We expect that the browsing history of users interested in these topics will contain a higher number of harmful content.

Q2: How many data points per client are needed?

We address this question by controlling the client dataset size (i.e., the number of tweets on a client device). These tweets can represent either the user's browsing history or tweets posted, retweeted, etc., by the user.

[5]https://scikit-learn.org/

Q3: How many clients are needed?

We address this question by controlling the number of FL clients (i.e., the number of clients available for the FL training.

5.3 Datasets

We select the following datasets for the experimental evaluation based on past studies of misbehavior on Twitter. For all datasets, in order to keep the FL task lighter for the user device, we binarize the classification problem by merging the several harmful classes into a single "harmful" class. We report below the original classes together with the final binary ones.

Abusive Dataset [11] initially contains ∼100K tweets, labeled as "Abusive", "Hate", "Normal", and "Spam". We remove 14,030 tweets labeled as "Spam" – following the same methodology of [12] because there are more sophisticated techniques to handle spam profiles. The resulting dataset consists of ∼86K tweets with 31.6% "Abusive", 5.8% "Hate", and 62.6% "Normal" classes. Final binary classes: 37.4% "Harmful" and 62.6% "Normal". **Sarcastic Dataset** [25] contains ∼61K tweets text classified in two classes labeled as "Sarcastic"(10.5%), and "None"(89.5%). Final binary classes: 10.5% "Harmful" and 89.5% "Normal". **Hateful Dataset** [31] is a ∼16K tweets dataset. The tweets are categorized in "Racism"(12%), "Sexism"(20%), and "Normal"(68%) classes. Final binary classes: 32% "Harmful" and 68% "Normal". **Offensive Dataset** [7] consists of ∼25K tweets categorized in three classes: "Hate"(6%), "Offensive"(77%), and "Normal"(17%). Final binary classes: 83% "Harmful" and 17% "Normal". **Cyberbully Dataset** [5] is a smaller dataset, with ∼6K tweets distinguished the "Bully"(8.5%), "Aggressive"(5.5%), and "Normal"(86%) classes. Final binary classes: 14% "Harmful" and 86% "Normal".

We **preprocess the tweet texts** by removing tags, URLs, numbers, punctuation characters, non-ASCII characters, etc. Moreover, we convert the text to lowercase, all the white spaces into a single one. We also remove English stop words and words that appear only once in the dataset (in the case of misspelled words).

5.4 Non-DP FL online content moderation

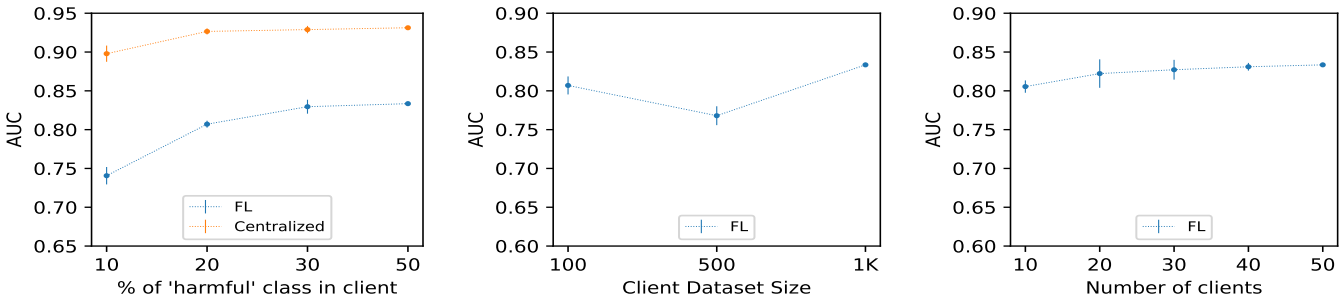
In the following experiments, we only evaluate the non–DP FL framework on the "Abusive" dataset. We chose this dataset because its size allowed experimentation with various FL simulation parameters.

5.4.1 How many harmful tweets per client are needed?

Here, we evaluate the FL classification when we vary the percent of "harmful" data in the clients' datasets using the values 10%, 20%, 30%, and 50%. For a given "%harmful" value, first, we randomly select 50 clients and then we train the classifier in these clients for 20 FL rounds. Each client dataset consists of 1K data. Finally, we repeat the experiment five times to acquire and report average scores and standard deviations.

We also ran experiments with the *Centralized training setup* by varying the percent of "harmful" text in the training set. Then, we randomly select 50K tweets as the training set. We chose the 50K samples to compare the centralized classification performance with the previously mentioned FL training. We repeated the training three times for each "%harmful" value.

Results Discussion: In Figure 3a, we present the average AUC values (test evaluation). We note that by increasing the examples



(a) Harmful Class Ratio (b) Client Dataset Size (c) Number of Clients

Figure 3: Evaluation of non-DP FL. (a) 50 clients, 1K data points per client; centralized –50K data points (b) 50 clients, balanced data per client (i.e., 50% harmful data); (c) 1K data points and balanced data per client

of the "harmful" class by five times (i.e., from 10% to 50%), we have ~9% increase in AUC (from 74% to 83%). These results show that balancing the data at the client side enabled the classifier to learn better both classes. In the case with a 10% harmful class size, we got a 95% score in precision, recall, and F1-score. Interestingly, in the case of 50% of harmful class size, we obtained precision (93%), recall (89%), F1–score (90%), which shows a decrease by ~1%, 6%, and 4% respectively. The training dataset is imbalanced when only 10% of clients' data is harmful. To understand this reduction in the model's performance, we calculated the metrics only on the harmful class (i.e., the minority class), where we observed a ~30% increase in recall but also a 40% negative impact on precision (with 10% of harmful class size we got a recall of 50%, and precision of 82%, with 50% we got a 77%, and a 40% respectively). This means having a balanced dataset (with 50% of harmful class size) impacts the recall of the harmful class: i.e., it helps the model learn the harmful class better. This is what drives AUC up as well (in the weighted metrics as well as in the harmful–only case).

In the centralized approach, the classifier shows high performance, with only a 3% AUC difference between the 10% and 50% of harmful class size (90%, and 93% AUC, respectively). Finally, we get the best FL classification performance for balanced clients datasets (only ~10% AUC difference with the centralized training).

5.4.2 How many data points per client are needed?

We assumed a homogeneous setting where all clients have the same dataset size. We evaluate the classifier performance for the client's dataset size of 100, 500, and 1K. We run the *FL training setup* for twenty FL rounds by using the same randomly selected fifty clients. Each client has a balanced dataset 50:50. We repeat the FL training twenty times for the training with 100 and 500 data, and five times for the 1K data. We present the average AUC metric in Figure 3b, with standard deviation as error bars.

Results discussion: Increasing client dataset size by ten times (from 100 to 1K data points) can lead to the overall improvement of performance metrics by ~3% in the AUC (from ~81% to 83%). We observed also a ~2% improvement in F1 score (from 88% to 90%), ~4% in accuracy (from 85% to 89%), recall (from 85% to 89%), and ~1% in precision (from 92% to 93%). The results show that increasing the data by five times did not significantly improve the performance, but the model performs similarly with the 100 data points per client.

Dataset		Accuracy	AUC	F1 Score
Abusive	FL	(0.85, 0.01)	(0.81, 0.01)	(0.88, 0.01)
	Centr.	(0.92, 0.01)	(0.92, <1e-3)	(0.94, 0.01)
Sarcastic	FL	(0.73, 0.01)	(0.66, 0.01)	(0.79, 0.01)
	Centr.	(0.76, 0.05)	(0.75, 0.03)	(0.83, 0.03)
Hateful	FL	(0.85, 0.02)	(0.61, 0.01)	(0.87, 0.01)
	Centr.	(0.79, 0.02)	(0.79, 0.01)	(0.85, 0.01)
Offensive	FL	(0.78, 0.02)	(0.78, 0.01)	(0.83, 0.02)
	Centr.	(0.92, 0.01)	(0.92, <1e-3)	(0.94, 0.01)
Cyberbully	FL	(0.94, <1e-3)	(0.80, 0.01)	(0.94, <1e-3)
	Centr.	(0.91, 0.03)	(0.91, 0.02)	(0.93, 0.02)

Table 1: Comparing FL and centralized approach. Average values (metric, std) over five repetitions for five different datasets. Each client has 100 data points and balanced data (i.e., 50% harmful class).

Therefore, the experiment shows that the FL training can build an effective model (~81% AUC) even with 100 data points per client.

5.4.3 How many clients are needed for a good FL model?

In this experiment, we run the *FL training setup* by varying the number of available clients, i.e., 10, 20, 30, 40, 50. Each client has a 1K balanced dataset, and the FL training runs for twenty rounds with the same randomly selected clients. We run the FL training five times for each value of the number of clients property, and we present the average test AUC in Figure 3c.

Results Discussion: Increasing the number of clients participating in FL training by five times (i.e., from 10 to 50) results in increasing the AUC by ~2% (from 81% to 83%). Additionally, the accuracy, precision, recall, and f1-score, increase by ~3%, 1%, 3%, and 2% respectively (from 86%, 92%, 86%, 88% to 89%, 93%, 89%, 90%). However, the interesting point is that even with ten users/clients, the system can build an efficient model. The model performs similarly well when varying the number of clients participating in the FL training.

5.5 Generalization on other Twitter datasets

Bootstrapping from the first round of experiments, we test the *FL training setup* with four other datasets (see datasets details in

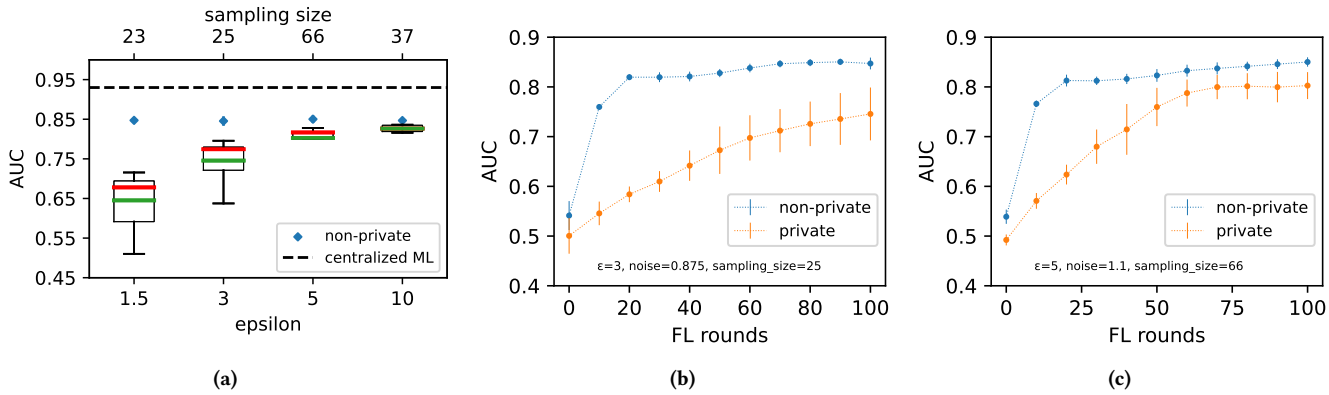


Figure 4: Comparing DP FL, non-DP FL, and centralized ML. Evaluation of (ε, δ)-DP FL for different ε values and $\delta = 10^{-3}$. Experiments with 628 total clients; 100 data points per client; 50% harmful-class (balanced data). For the non-DP FL, we perform client selection (per FL round) with the same sampling values used for the DP FL. Centralized ML on a balanced dataset with 50k data points.

Section 5.3) to explore the generalization of the classifier's utility. For each dataset, we run both the FL, and centralized training for five repetitions each, and then compare the average performances.

We run the FL training for 20 rounds, with the same clients participating in each round. Each client had a 100 tweets balanced dataset. We set the data size to 100 due to the datasets' size limitations and based on the previous experiments that 100 data points per client are sufficient for effective FL training. We randomly select 50 clients when the dataset size allowed us to do so. For small datasets, we build the maximum number of clients i.e., 37 and 16 clients for Offensive and Cyberbully datasets, respectively. For the Centralized training, we used a training set size= $\#clients \times 100$ to fit the total data used in the FL training for the corresponding dataset. We did not perform hyperparameter tuning to train the model with the different datasets. We present the average evaluation metrics (test phase) for both setups in Table 1.

Results Discussion: Across all five datasets, we observe an AUC performance >61%. We get the best AUC while training with the Abusive dataset (81%), and with the smallest, Cyberbully dataset, we achieved an AUC of 80%. Training with the Offensive, Sarcastic, and Hateful, we got an AUC performance of 78%, 66%, and 61%, respectively. Additionally, we can observe that the model's performance decreases by ~9% (the minimum) to ~18% (the maximum) when trained with the FL approach compared to the centralized one. However, the results show that the classifier can be generalized and achieve acceptable performance on different types of misbehavior, even without hyperparameter tuning.

5.6 DP FL online content moderation

We use the well-accepted concept of DP that has been shown in the literature, especially in the context of FL[3, 13, 19], as a way to provide user-level privacy guarantees against unwanted user's private information leakage.

We apply the concept of CDP to our FL training setup. Our implementation is based on the TensorFlow privacy library[6]. TensorFlow modifies the Federated Averaging algorithm to provide user-level DP guarantees, based on [3]. The variation of the algorithm implements the following: (i) each client clips the model's updates before transmitting them to the server adaptively and privately. Clipping bounds the influence of each client on the global model update in each FL round. (ii) the server, during the aggregation of the client's updates, adds Gaussian noise to the sum of the updates before averaging. TensorFlow privacy library provides an implementation that returns the necessary DP parameters (i.e., noise multiplier, sampling size) to achieve a specific (ε, δ)-DP for the FL training setup. This implementation is based on the Moment Accountant method [1, 20, 30], which assesses the (ε, δ)-DP of the model. Lower ε values indicates higher level of privacy i.e., we offer higher privacy to the clients participating in the FL training. The noise multiplier property defines the addition of noise to the sum of the model's updates, and the sampling size refers to randomly selecting a subset of the available clients to participate in each round. The sampling adds to the privacy guarantee of the training since we do not set a fixed number of clients participating in every round.

We run an experiment to assess the privacy guarantee and utility trade-off. For this experiment, we use the "Abusive" dataset, split and distribute the data to clients as described in Section 4.4. We run the *FL training setup* for 100 rounds, and each client has a 100-balanced dataset. These FL parameters give the maximum available number of clients, i.e, 628 clients. We use Poisson sampling, which gives a different number of clients to participate in each round, with a mean set to *sampling size* value.

We evaluate the DP-classifier with different ε values, while setting $\delta = 1e^{-3}$. We define $\delta = 1/|\text{total samples}|$ using the suggested formula in [1, 19]. For each ε value, we get the DP-parameters – necessary for achieving the given (ε, δ)-DP – using the TensorFlow privacy library mentioned before. So for ε value of 1.5, 3, 5, and 10, we get the following DP-parameters, i.e., *(sampling size, noise multiplier)* – (23, 1.15), (25, 0.875), (66, 1.1), and (37, 0.612) respectively. We repeated the simulations ten times for $\varepsilon = 1.5$ and $\varepsilon = 3$, and five times for $\varepsilon = 5$ and $\varepsilon = 10$. We present the average AUC

[6]https://github.com/tensorflow/privacy

achieved in Figure 4a (the green line shows the mean, and the red line the median AUC of all the repetitions).

To investigate the trade-off between utility and privacy, we run a set of experiments with the *FL training setup* using the same parameters mentioned before (i.e., clients dataset, sampling size, number of FL rounds) but without adding DP. In Figure 4a, we present the average AUC values (over five repetitions) for the non-DP model. We also depict in the same figure the AUC achieved by the model trained with the centralized approach with 50K data as the baseline AUC. We evaluated the model's performance every ten rounds of the FL training for both the non-DP model and DP model for $\varepsilon = 3$ (medium) and $\varepsilon = 5$ (medium-high). We present the average AUC values in Figure 4b, and 4c respectively.

Results Discussion: Figure 4a shows that adding DP with a strict privacy guarantee (i.e., $\varepsilon = 1.5$) causes a 20% decrease in AUC when compared to the non-DP model performance. Experimenting with lower ε values, we observed that we do not get a robust model with stable behavior (i.e., four out of ten repetitions gave a 10% to 30% AUC). We observed that the classifier could tolerate a noise multiplier near the value 1; adding more noise does not allow the classifier to learn during the training. With a medium DP level, ($\varepsilon = 3$) and ($\varepsilon = 5$), we get an average AUC of 75%, and 80%, approaching the non-DP model's performance. Figures 4b, 4c show that a DP-model training requires more FL rounds to converge (i.e., 100 rounds) while the non-DP model's performance shows a rapid increase, and reaches an acceptable AUC (i.e., 20-30 rounds). Additionally, the performance of the non-private model confirms our previous observations that altering the number of FL participants (i.e., sampling size) does not affect the model's performance. Finally, by training the model for 100 FL rounds, we get 85% AUC. In other words, the performance is improved by 4% from the case we present in Figure 3b — i.e., 50 clients with 100 balanced dataset each. In conclusion, we get 5% ($\varepsilon = 5$) to 10% ($\varepsilon = 3$) loss in AUC between private and non-private FL. We leave for future work the empirical investigation of the actual attack mitigation. Emiliano et al. have shown that CDP can quite effectively defend against membership attacks without significant loss in utility – for more details, see in [21] the Table 1, CDP and passive/active local attacker.

5.7 Overhead on Client's Device

We experiment to measure the extra overhead caused to the client's device when participating in the FL training. Specifically, we assess the overhead during the local training, which happens in one FL round on the client's device. Indeed, there is an extra user device overhead due to the communication between the client and the central aggregator [16, 26, 29]. Since we simulate the FL training, we don't have the information on the communication cost in real-world settings – we assume this overhead is constant.

We run the *Local training* on a laptop (see laptop properties in Section 5.1), using a client's dataset as the training set. Since the results of the experiments with 100 data per client showed that we can have a well–performing classifier, we set the client's dataset size to 100. While training the model locally, we monitor the machine resource utilization (memory consumption and CPU utilization) and collect the logs after every two seconds. We repeated the training ten times. We kept the CPU "idle" during the training by not running other applications. Figure 6 shows the device's CPU utilization (in %) and the memory consumption (in MB) during the local training after averaging the results of the experiment.

Results discussion: The average CPU utilization during the training across all repetitions is ~25.5%. The memory consumption varies between ~2300 to ~2600MB during the training, with an average of ~2560MB. See more details of the device resource consumption results in Appendix B. Overall, the results show that the local training, with a mean of the CPU utilization around ~64% and at a maximum of ~85%, occupies the device for a short time of ~14 seconds thus, it does not introduce a severe overhead for the client device.

6 CONCLUSION

In this work, we propose a framework that gives power to the users to contribute to the development of online content moderation tools. By applying Federated Learning (FL) with Differential Privacy (DP) guarantees, we provide user–level privacy guarantees that can be easily adapted to several social media platforms and types of misbehavior. Our experimental results – over five Twitter datasets – show that (i) for both the DP and non-DP FL variants, the text classification performance is close to the centralized approach; (ii) it has a high performance even if only a small number of clients (with small datasets) are available for the FL training; (iii) it does not affect the performance of user's device – in terms of CPU and memory consumption – during the FL training. Although we investigate the feasibility of our approach considering several factors, there are several directions for future research. We will conduct an empirical investigation of the effectiveness of CDP as a defense mechanism against membership inference attacks. Moreover, we will evaluate the model on the multiclass classification problem – classification of different types of misbehavior. Finally, several concerns need to be addressed, such as user incentivization, model bias and fairness issues, and potential data poisoning attacks by malicious clients.

ETHICAL CONSIDERATIONS

This work followed the principles and guidelines on executing ethical information research and using shared data [15]. The suggested methodology complies with the GDPR and ePrivacy regulations. We have not collected data from Twitter. We use existing Twitter datasets – that have already been published by other academic studies by requesting access from their publishers. For this reason, we will not publicly release any dataset used in this study. We did not use or present any identifiable user information from the datasets (e.g., Twitter user IDs). We applied text preprocessing to clean the tweets from any information that could identify specific Twitter accounts (see Section 5.3). Hence, the train data of the text classifier did not contain Twitter usernames. Finally, we implemented and executed the experiments locally – on our devices – without using any cloud computation services, so we did not upload any of the datasets to the cloud.

ACKNOWLEDGMENTS

This work has been funded by the EU H2020 projects CONCORDIA (Grant Agreement No. 830927), SPATIAL (Grant Agreement No. 101021808), and AERAS (Grant Agreement No. 872735).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. New York, NY, USA, 308–318.

[2] Sultan Alshamrani, Ahmed Abusnaina, Mohammed Abuhamad, Daehun Nyang, and David Mohaisen. 2021. Hate, Obscenity, and Insults: Measuring the Exposure of Children to Inappropriate Comments in YouTube. In *Companion Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 508–515. https://doi.org/10.1145/3442442.3452314

[3] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. 2021. Differentially Private Learning with Adaptive Clipping. In *Advances in Neural Information Processing Systems*, Vol. 34. 17455–17466.

[4] Varun Chandrasekaran, Suman Banerjee, Diego Perino, and Nicolas Kourtellis. 2022. Hierarchical Federated Learning with Privacy. (2022). Available at https://arxiv.org/abs/2206.05209.

[5] Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean Birds: Detecting Aggression and Bullying on Twitter. In *Proceedings of the 2017 ACM on Web Science Conference* (Troy, New York, USA) *(WebSci '17)*. New York, NY, USA, 13–22.

[6] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. 2019. Federated Learning Of Out-Of-Vocabulary Words. (2019). published by Google Research.

[7] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017*. AAAI Press, 512–515.

[8] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology (EUROCRYPT 2006)* (advances in cryptology (eurocrypt 2006) ed.) *(Lecture Notes in Computer Science, Vol. 4004)*. 486–503.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

[10] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (aug 2014), 211–407.

[11] Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *Proceedings of the International AAAI Conference on Web and Social Media* 12, 1 (Jun. 2018).

[12] Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2019. A Unified Deep Learning Architecture for Abuse Detection. In *Proceedings of the 10th ACM Conference on Web Science (WebSci '19)*. New York, NY, USA, 105–114.

[13] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. https://doi.org/10.48550/ARXIV.1712.07557

[14] 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* (2018).

[15] Erin Kenneally and David Dittrich. 2012. The menlo report: Ethical principles guiding information and communication technology research. (2012). Available at SSRN 2445102.

[16] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. 2020. FLaaS: Federated Learning as a Service. In *Proceedings of the 1st Workshop on Distributed Machine Learning* (Barcelona, Spain) *(DistributedML'20)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/3426745.3431337

[17] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. 2022. Privacy and Robustness in Federated Learning: Attacks and Defenses. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–21. https://doi.org/10.1109/TNNLS.2022.3216981

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Xiaojin (Jerry) Zhu (Eds.). PMLR, 1273–1282.

[19] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[20] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530* (2019).

[21] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. 2022. Local and Central Differential Privacy for Robustness and Privacy in Federated Learning. In *Proceedings of the 29th Network and Distributed System Security Symposium (NDSS 2022)*.

[22] Kostantinos Papadamou, Antonis Papasavva, Savvas Zannettou, Jeremy Blackburn, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, and Michael Sirivianos. 2020. Disturbed YouTube for Kids: Characterizing and Detecting Inappropriate Videos Targeting Young Children. *Proceedings of the International AAAI Conference on Web and Social Media* 14, 1 (May 2020), 522–533.

[23] Kostantinos Papadamou, Savvas Zannettou, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Michael Sirivianos. 2022. "It Is Just a Flu": Assessing the Effect of Watch History on YouTube's Pseudoscientific Video Recommendations. *Proceedings of the International AAAI Conference on Web and Social Media* 16, 1 (May 2022), 723–734.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[25] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm Detection on Twitter: A Behavioral Modeling Approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (Shanghai, China) *(WSDM '15)*. New York, NY, USA, 97–106.

[26] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3400–3413.

[27] Rashid Tahir, Faizan Ahmed, Hammas Saeed, Shiza Ali, Fareed Zaffar, and Christo Wilson. 2019. Bringing the Kid Back into YouTube Kids: Detecting Inappropriate Content on Video Streaming Platforms. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Vancouver, British Columbia, Canada) *(ASONAM '19)*. New York, NY, USA, 464–469.

[28] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated Learning with Local Differential Privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20)*. New York, NY, USA, 61–66.

[29] Luping WANG, Wei WANG, and Bo LI. 2019. CMFL: Mitigating Communication Overhead for Federated Learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 954–964.

[30] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled Renyi Differential Privacy and Analytical Moments Accountant. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 89)*, Kamalika Chaudhuri and Masashi Sugiyama (Eds.). 1226–1235.

[31] Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, San Diego, California, 88–93.

[32] Bingjie Yan, Jun Wang, Jieren Cheng, Yize Zhou, Yixian Zhang, Yifan Yang, Li Liu, Haojiang Zhao, Chunjuan Wang, and Boyi Liu. 2021. Experiments of Federated Learning for COVID-19 Chest X-ray Images. In *Advances in Artificial Intelligence and Security*, Xingming Sun, Xiaorui Zhang, Zhihua Xia, and Elisa Bertino (Eds.). Springer International Publishing, Cham, 41–53.

[33] 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.

[34] Harish Yenala, Ashish Jhanwar, Manoj K Chinnakotla, and Jay Goyal. 2018. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics* 6, 4 (2018), 273–286.

A CONCEPTUAL FRAMEWORK

A.1 System Components

Description of the proposed framework's components (Figure 5):

Client Device: The user's device (i.e., laptop, mobile, etc.) that accesses the OSN application (i.e., Twitter, Reddit, 4chan, etc.). We assume that the user may not trust the platform's content moderation.

Browser Add-On: filters the user's online activity, conducts DOM tree analysis, and sends the selected data (i.e., posts, chat messages, comments) to the Labeling Module.

Labeling Module: aggregates the labels obtained from the Auto-Labeling and User Feedback modules. *Auto-Labeling module* can use semi-supervised learning techniques to label the data automatically.

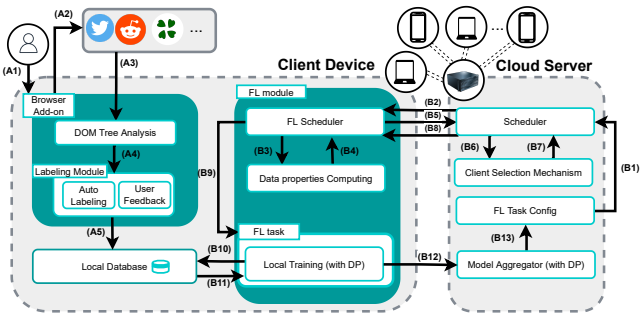


Figure 5: Proposed FL framework

The *User Feedback module* asks the user to label the data. Note that the labels the user gives can be considered sensitive information the user wants to protect.

Local Database: stores the labeled data locally on the user's device.

FL Module: schedules and executes FL tasks on the user device.

FL task defines and executes the **Local training**.

Data properties computing: the module that computes the meta-data of the user's dataset (i.e., size of data, etc.), accompanied by other device information (e.g., battery, internet connection type, device capabilities, etc.).

Cloud Server: a unit owned by a trusted party that coordinates the FL training.

FL Task Configuration: generates the FL Task description, which contains the baseline model for training – based on the specific learning task – the criteria for the clients to participate in this task, and the FL parameters (e.g., the number of FL rounds, the number of clients to participate, etc.).

Scheduler: advertise the FL task to the available clients and manage the communication with the clients.

Client Selection Mechanism: checks if the client's device complies with the criteria set by the FL Task Config module.

Model Aggregator: aggregates the clients' model updates and applies the aggregated update to the global model.

A.2 Data-Flow

Figure 5 shows the data flow of the proposed framework. Specifically: **(A1)** The user accesses the OSN application through the device's browser, **(A2)** and sends an HTTP request to it. **(A3)** The *Browser Add–On*'s *DOM Tree Analysis module* receives the social network's traffic from the user's interactions with the application, analyses the page DOM tree, filters specific user's activity (related to the learning task), and selects data for labeling. **(A4)** The *Labeling module* receives the data (e.g. a tweet text). The *Auto Labeling module* automatically labels the data. The *User Feedback module* asks the user to label it. **(A5)** Then, it aggregates the two {data, label} pairs (output of the two labeling methods), defines a final label for the data, and stores the labeled data in the *Local Database*.

When there is a pending FL task at the server, **(B1)** the *FL Task Config module* sends the task description to the *Scheduler*. **(B2)** The *Scheduler* sends the task description to the available clients. **(B3)** The *client's FL Scheduler* receives it, and forwards it to the *Data properties Computing module*, **(B4)** which sends the device's data properties back to it. **(B5)** The *FL scheduler* sends the properties to the *Scheduler*, **(B6)** which forwards them to the *Client Selection Mechanism* to tell if the client will participate in the training or not. **(B7)** The selection mechanism module sends its positive or negative decision to the *Scheduler*, **(B8)** which announces to the *client's FL scheduler* its participation in training with the global model to train or closes the connection with it.

For participating clients, **(B9)** the *FL Scheduler* sends the global model, and the task description to the *FL Task module*, and **(B10)** requests the local dataset. **(B11)** The *Local Database* sends the dataset, and starts the local ML training. Here, we apply the concept of Differential Privacy (DP) to achieve user-level privacy guarantees using the Adaptive Clipping DP methodology proposed in [3]. By the end of the local training, the local model's update is clipped, **(B12)** and sent to the *Model Aggregator*. The *Model Aggregator* adds Gaussian noise to the updates' sum, aggregates the updates, and applies the aggregated update to the global model. Finally, **(B13)** it sends the updated model to the *FL Task Config module* for its use in the next round of the FL training.

B CLIENT DEVICE OVERHEAD

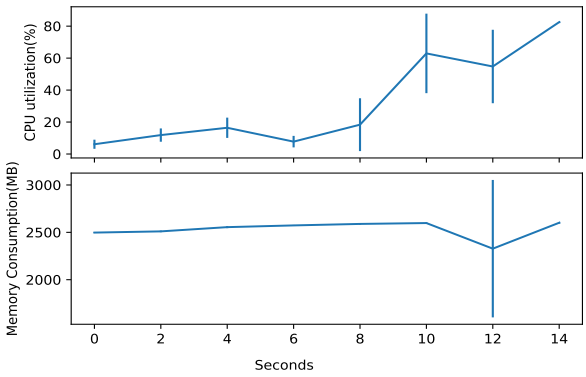


Figure 6: CPU and memory consumption (every 2 seconds) on client's device due to the FL training.

In Figure 6, we see that the total duration of the training phase is ~14 seconds. From seconds 0 to 8, the CPU utilization increases linearly from ~10% to 20%. Then, there is a rapid increase (from seconds 8 to 10) in which the CPU reaches ~70%. At the end of the training phase, there is a decrease to ~60%, and CPU utilization reaches a maximum of ~80%. The average CPU utilization during the training across all repetitions is ~25.5%.

The memory consumption varies between ~2300 to ~2600MB during the training, with an average of ~2560MB. There is a warm-up phase (from 0 to 10) (when the training phase begins) where the memory consumption increases by ~100MB. There is a decrease in memory consumption at 12 seconds (as also happens with CPU utilization), resulting from one of the repetitions completing the training faster than the rest. Overall, the results show that the local training, with a mean of the CPU utilization around ~64% and at a maximum of ~85%, occupies the device for a short time of ~14 seconds thus, it does not introduce a severe overhead for the client device.