Classification of Toxic Comments using Knowledge Distillation

MSc Research Project
Data Analytics

Bijender Gupta
Student ID: x20142358

School of Computing
National College of Ireland

Supervisor: Prof. Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Bijender Gupta
Student ID:	x20142358
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Prof. Rejwanul Haque
Submission Due Date:	16/12/2020
Project Title:	Classification of Toxic Comments using Knowledge Distillation
Word Count:	7689
Page Count:	24

Signature:	Bijender Gupta
Date:	30th January 2022

Classification of Toxic Comments using Knowledge Distillation

Bijender Gupta
x20142358

Abstract

Any expression that attacks a group or individual based on qualities such as color of skin, sexuality, religion, nationality, or another aspect is considered toxic. It is also described as any form of communication (verbal, textual, or gestural) intended to elicit unpleasant sentiments in a group or person, which may result in acts of aggression, ego, or damage to society as a whole. To address this issue, machine learning and deep learning neural network models were utilized to efficiently classify harmful information on internet platforms. Building these models on massive datasets is challenging because it necessitates a significant amount of computing resources, and inference budgeting remains a challenge.We propose a knowledge distillation approach for categorizing toxic comments. Knowledge distillation is the process of distilling a model by instructing a smaller network, one step at a time, exactly what a larger trained network should do. The feature mappings produced by the bigger network after each convolution layer are referred to as "soft labels." By attempting to replicate the larger network's outputs at each level, the smaller network is eventually trained to mimic the larger network's activity. We begin by training a specifically developed teacher model, Bert-Base. The knowledge is then transferred from this teacher-model to Mobile-BERT (Sun et al.; 2020). It is a slimmed down form of BERT-LARGE, along with bottleneck design and a carefully crafted blend of self-attention and feed-forward-networks. After applying knowledge distillation method, we show that MobileBert is 60% quicker and significantly smaller in size than the Bert-Base model with accuracy of 98%, Precision 92% and Recall 88%. In terms of performance parameters such as precision, recall, and accuracy score, MobileBert outperformed the Bert-Base model, and it took less training time and inference time. We also demonstrate that our student model, MobileBert, outperforms standard ML algorithms such as Logistic-Regression, Decision-Tree-Classifier,Random-Forest-Classifier, and XG-Boost.

1 Introduction

Facebook, Twitter, YouTube, and other social networking sites are increasingly being used to express one's opinions and share information, and this trend is expected to continue. In spite of the fact that these platforms may be utilized for reasonable debate, they have become more popular for the transmission of obscene-language and the coordination of hate-motivated activities as a result of their portability and anonymity (Badjatiya et al.; 2017). The number of violent acts committed as a consequence of hate speech on the internet is steadily increasing. Following the Brexit vote and the attacks on the subway systems in London and Manchester, there has been an increase in anti-immigrant and anti-Muslim sentiment in the United Kingdom. Toxic comments and other illegal conduct have surged considerably in the United States as a result of Trump's election victory. These types of online violence not only damage the social atmosphere of the online groups that are exposed to them, but they may also lead to actual violence and significant damage. Therefore, governments and social media platforms should create methods for detecting violent conduct and hate speech.

Artificial intelligence (AI) has lately gained significant success in natural language processing (NLP) by using massive pre-trained-models with millions of para-meters. However, because of their large model sizes and significant latencies, these models are not suitable for use on portable devices with limited resources. A number of issues have been raised about the tendency toward larger models. The first is the environmental expense of exponentially growing the computing needs of these models, as discussed in Schwartz et al. (2019). While running these algorithms on-device in real time has the ability to reveal new and innovative language processing applications, the increasing computational and storage needs of such algorithms may make them unsuitable for widespread use.
Many scientific research papers have used Natural-Language-Processing(NLP) in collaboration with ML-algorithms and DL approaches to identify toxic comments or hate speech on the internet (Mehdad and Tetreault; 2016). Accordingly, although supervised machine learning methods may employ many different features, like sentiment classification, linguistic resources, user-based, knowledge-based-features and platform-specific meta-data (Davidson et al.; 2017), they all need an established strategy for the extraction of features.

The primary goal of this paper is to interpret, assess, train, and understand the text or online comments in a meaningful manner so that it can recognize remarks that spread hate in society via knowledge distillation. We propose a methodology to achieve comparable performance on a wide variety of downstream tasks using much smaller linguistic models that have been pre-trained with knowledge-distillation, potentially resulting in models that are lightweight and faster to infer, while also needing a lower computational training cost. Furthermore, we compare the model size of the teacher and student models, as well as the training duration. Lastly, we evaluate the performance of pre-trained bert models including teacher and student model with the fundamental machine learning models such as Logistic Regression, Random Forest, Decision Tree Classifier, and the xgboost algorithm.

1.1 Research Questions and Objective

1.1.1 Research Questions

RQ: To determine if student models can outperform the teacher model in recognizing toxic comments using a knowledge-distillation technique.

sub RQ1: Which machine learning approach is the most effective for detecting toxic comments, and which algorithm takes less amount of time to train and produces the best results in terms of accuracy, precision, recall, and F1 score? ?

sub RQ2:To compare the training and inference times as well as model size of the teacher and student models.

1.1.2 Objectives

- 1. Build a teacher model (Bert-Base) using training data and assess its performance using precision, recall, accuracy, and f1 scores on test data.
- 2. Create train data for the student model (Mobile Bert) and classify it using the teacher model.
- 3. Train student model on training data created in step 2.
- 4. Evaluate the student model against the held-out test set to determine the F1-score, recall, precision, and accuracy.
- 5. We also train and test data using fundamental machine learning algorithms such as Logistic-Regression (LR), Random-Forest-Classifier (RF), XG-Boost algorithm, Decision-Tree-Classifier, and assess their performance using precision-score, recall-score, F1-score, ROC-AUC curve, and accuracy.
- 6. Finally, we compare the inference time, training time (student versus teacher) and the number of parameters, as well as the performance in terms of precision, recall, accuracy, and f1-score, of these algorithms to that of other fundamental machine learning algorithms such as Random Forest Classifier, Decision Tree Classifier, and Logistic Regression.

1.2 Contribution

We present an approach for implementing a widely used method called knowledge distillation for identifying toxic comments. This method uses a teacher-student architecture to transmit information from complex to simple-networks by learning the dispersion of the teacher model's potential target (the labeled dispersion supplied by the teacher's outcome) instead of the ideal label. We use the knowledge distillation technique, in which a big model acts as a teacher and a smaller model learns to act as a student in imitation of the larger model. This is a model-agnostic technique that enables knowledge transfer between BERT and another neural framework, such as MobileBert. However, since model reduction often results in knowledge loss, the efficiency of the student-model is rarely equal to that of the teacher-model. To overcome the aforementioned problem, we may consider training our teacher model using a variety of alternative parameters and optimizers. In other words, we first train teacher models and then train a student model for the resulting test data generated by the teacher model after testing the model. Finally, the student model with the best inference time, accuracy, and other parameters is considered the final model. To our knowledge, this is the first time an unsupervised pre-trained model, MobileBERT (a thinned version of Bert-Base), has been utilized to transfer learning knowledge to low-resource toxic-comment classification, improve task performance, and shorten model training time.

While the majority of previous research has concentrated on using knowledge distillation to create task-specific models, we demonstrate that it is possible to downsize a BERT model while retaining 97% of accuracy for identifying toxic comments and running 60% quicker than Bert-Base while still accomplishing successful performance on well-known standards using knowledge distillation during the pre-training phase. We begin by training a specially constructed teacher model, the Bert-Base model with an inverted bottleneck, before going on to MobileBERT. Following that, we transmit information to MobileBERT from this teacher model. We propose a loss that combines language modeling and distillation in order to exploit the inductive biases learned during pre-training by larger models. We demonstrate the potential of our relatively small, quick, and lightweight model which take very less time in training and also outperform the base-line model.

Additionally, we show in this study that by using significantly smaller pre-trained language models via knowledge-distillation, we can achieve equivalent results on a variety of downstream tasks, resulting in models that are lighter and faster to infer, while also requiring a lower computational training cost. Our general-purpose pre-trained algorithms can be fine-tuned to behave very well on a range of downstream-tasks while retaining the flexibility of larger models.

The study piece is divided into sections and each section is explained in depth. Section 2 is mostly devoted to a review of the literature about previous research on the same subject. We learn about the study's methodology and experiments in the section 3. Section 4 discusses the design guidelines and architecture used in the project. Section 5 details the implementation, evaluation, models' execution and results of the algorithms and models. Section 6 discusses the models' performance, as well as the difficulties they experienced. Finally, in Section 7, the project comes to a close with a conclusion and future work.

2 Related Work

2.1 Using BERT to distill task-specific-knowledge into simple neural-networks

Tang et al. (2019) shows how to distill the information from BERT into a basic BiLSTM (Bi-directional long short term memory) based model using a standard approach. The distilled-model delivers outcomes that are equivalent to those obtained with ELMo i.e Embeddings from Language Model (Hu et al.; 2018) while using considerably fewer parameters and computation power. As a result of their findings, shallow BiLSTMs appear to be more descriptive of natural language activities than previously thought.This demonstrates that primitive, small neural networks might be considered effective without the need for new input characteristics, external training-data, or modifications to the network's framework. Specifically, they suggest distilling information from BERT, an advanced language representation model, into a single-layer-BiLSTM, including its siamese equivalent for task-based sentence-pair learning. We get equivalent results with ELMo across various datasets in summarizing, language processing interpretation, and sentimental analysis while utilizing around 100 times fewer parameters and 15 times less computation power.

2.2 Designing a Neuron Learning to Read Faster through knowledge distillation

The goal of this research Chatterjee (2019) is to find a solution to the Machine-Reading-Comprehension issue, in which queries must be addressed based on background text that is both computationally quick and outperforms well. There are two models that have been designed with this purpose in mind: the CONV-MODEL and the BERT-SMALL. The results of the experiments demonstrate that the CONV-MODEL is the quickest of the examined models and earns a relatively acceptable F1 score, whilst the BERT-SMALL model obtains the highest F1 score of all of the models studied. Also, it was discovered that using a multi-window-Convolution -Layer instead of a single-window-Convolution-layer enhanced the F1-score. The knowledge-distillation strategy significantly improved the F1-score, without the requirement for a larger capacity model to do this.

2.3 Reading Improves Student Learning: Pre-training Compact Models

Turc et al. (2019) demonstrates first that pre-training is still necessary in the case of shorter structures and that finetuning pretrained compact algorithms may be as effective as more complicated approaches suggested in contemporaneous research. They begin with pretrained small devices and then investigate how task information may be transferred from big, fine-tuned models using conventional knowledge-distillation. Pretrained distillation, the resultant basic but effective and generic approach, adds further benefits. They more broadly, via vast experimentation, investigate the relationship among pretraining and distillation using two previously unexplored variables: model scale and the features of unlabeled process input. One unexpected finding is that they have a cumulative impact even when applied consecutively to identical data.

2.4 Model Compression for Online Query Responding Systems using Multi-Task Knowledge Distillation

Deep pretraining and finetuning algorithms (like BERT) have exhibited outstanding performance in problem-solving domains. However, because of the large number of hyper-parameters, these models have a relatively poor inference time. The application of these complicated models to real-world business circumstances becomes a difficult but necessary task. Earlier work has often relied on model compression techniques to address this issue. However, these approaches often result in data loss throughout the model-compression operation, resulting in results that are inconsistent between the downsized and original models. To address this issue, Waseem et al. (2018) present a Multitask-Knowledge-Distillation (MKDM) Model for online Query Addressing Systems, which distills knowledge from many teacher models into a lightweight student-model. This allows for the transmission of more generic information. The experiment findings demonstrate that their strategy outperforms baseline methods and even achieves outcomes similar to the initial teacher models, while also greatly speeding up model interpretation.

2.5 Identification of Toxic Comments and Offensive Language on the Online Platform using basic Machine Learning algorithms

Hate speech on social media sites such as Twitter, Facebook, and YouTube has been the subject of research for several years. Aspects that distinguish conventional machine learning techniques from one another are the features that are employed in the approaches, and surface-level-features such as bag-of-words, n grams at the word level and at the character level, and other features have been shown to be the best predictive characteristics (Waseem and Hovy; 2016). Various algorithms, such as Support-Vector-Machines (SVM) , Naive-Bayes (NB), and Logistic-Regression (LR), among others, have been used for classification tasks in addition to the use of characteristics. Using a set of criteria drawn from Gender Research and Critical-Race-Theory, CRT Waseem and Hovy (2016) developed a test that can distinguish between tweets that are racist, sexist, or neither. The corpus of more than 16K Twitter posts was annotated as racist, gender bias, or none. They employed a logistic-regression (LR) model with multiple sets of characteristics, such as word-and-character n-grams up to 5, genders, size, and location, to categorize tweets and determine their classification. They discovered that the most suggestive characteristics produced by their best model are character n-grams, and that employing location or size is harmful to the model's performance. With the help of crowd sourcing, Davidson et al. (2017) compiled a 24K data set of tweets carrying hate speech keywords and classified the corpus as hate speech, verbal abuse, or none. They then extracted distinct characteristics from the corpus, like n grams, some Twitter post metadata, including the count of hashtags, retweets, mentions, and Web links, Part-Of-Speech tagging (POS), and so on. It was discovered via their research with many different multi-class classifiers that the Logistic-Regression (LR) with L-2 regularization outperforms all others in this challenge. When it comes to separating racist hate speech from ordinary vulgarity on social networking sites, Malmasi and Zampieri (2018) suggested an ensemblebased method that employs many linear SVM-classifiers simultaneously to do so. Djuric et al. (2015) proposed a two-step strategy that included a continuous bag-of-words model to retrieve paragraph-2-vec embed dings and a binary predictor skilled in addition to the word embedding to discriminate between offensive speech and clean material as one of the first efforts in neural network models. The authors of this paper Davidson et al. (2019) studied three deep learning architectures, namely Fast-Text, Long-Short-Term Memory, and CNN, in which they started the word-embeddings with either arbitrary or GloVe-embeddings before training the architectures. Gambäck and Sikdar (2017) proposed a hateful speech encoder based on what CNN learned about several feature-embeddings, including such word-embeddings as well as character-n-grams, to distinguish between hateful statements and other forms of speech. In their study, Zhang et al. (2018) employed a CNN (Convolutional neural network) + GRU (Gated Recurrent Unit network) neural network model, which was loaded with pretrained word-2-vec embeddings, to extract both word and character pairs (for example, n grams, sentences) and word and character relationships. By presenting a multitask learning approach to cope with data-sets over diverse labeling techniques, tags, or geographical and social impacts from sampling techniques, Waseem et al. (2018) provided a fresh perspective on offensive communication and abusive speech recognition tasks for the first time. Using actual words and domainspecific data from Tweets, Founta et al. (2019) developed an integrated categorization model that can effectively handle diverse sorts of verbal abuse such as online bullying, hatred, mockery, and other forms of sarcasm, among others. Furthermore, investigators have increasingly focused their attention on the prejudice generated from hateful speech training data. The researchers Davidson et al. (2019) discovered that there were consistent and significant race disparities in five standard Twitter datasets that were categorized for foul language identification. In addition, Wiegand et al. (2019) discovered that predictors learned on sets of data with a higher percentage of latent abuse instances (tweets including some harsh phrases) are more susceptible to biases than classifiers trained on data sources with a higher proportion of explicit abuse examples (sarcasm, jokes, and other forms of humor in tweets).

2.6 Conclusion

Machine learning and Natural Language Processing (NLP) algorithms have achieved a significant success in recognizing damaging or dishonest information on web portals, according to a review of the literature. For toxic comment identification problem statement, both machine learning and deep learning models have been extensively applied. In contrast, ML models are strongly reliant on feature engineering, while deep learning models (RNN and CNN) need access to large datasets. Overfitting occurs when the number of hidden layers in the DL model increases, if the dataset size is insufficient. Furthermore, these models do not account for the interdependencies across numerous phrases, even when the sentences are very extensive. By reducing the necessity for extracting features and big datasets, current transfer learning approaches solve the drawbacks of ML and DL models.

To efficiently categorize harmful content on online platforms, machine learning and DL neural network models were used. Building these models on huge datasets is difficult since it requires a large amount of computer resources, and inference budgeting remains an issue. For classifying toxic comments, we suggest a knowledge distillation technique. Given the present level of research for harmful comment detection, it is obvious that the knowledge distillation approach for Bert-Base and MobileBert models is unexpolored. Thus, for the job at hand, this study will employ the pre-trained model, BERT and MobileBert, as well as several fundamental ML algorithms including Logistic regression, Random Forest, Decision Tree, and XG-Boost, and compare their performance with the baseline model.

3 Methodology

In this paper, we propose a unique technique for identifying toxic comments on social networking sites by analyzing texts, the underlying meaning of comments, and predicting hatred using Natural-Language-Processing (NLP) and Machine-Learning (ML) algorithms. The aim is to design a model that is computationally efficient and has a shorter inference time. The BERT model, which is the state-of-the-art in many natural-language processing problems, was utilized to train a smaller model using the knowledge-distillation approach. The generated models are compared to other models developed for the same purpose. The preceding section 2 discussed some of the published evidence on toxic comments on social media platforms and knowledge distillation. Therefore, the purpose of this study is to categorize toxic comments as hateful or non-hateful. As a result, this part will concentrate on the method through which this study will be carried out. The Cross-Industry Process for Data Mining methodological approach (CRISP-DM) was used as the overall methodology in this paper.

3.1 CRISP-DM Modules

Six modules comprise the CRISP-DM method. Some modules contain two-way paths, allowing you to alter any previous step as needed. The six modules are as follows:

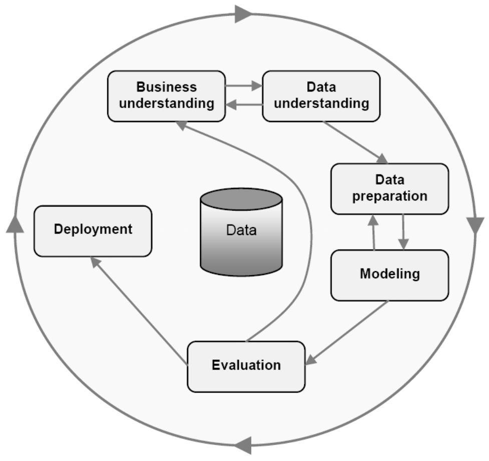


Figure 1: CRISP-DM Methodology

1. `Business Understanding:` The CRISP - DM method's first module outlined the thesis problem from a business perspective. This is used to prevent users from posting potentially toxic comments, to encourage more respectful discussions with others, and to study the toxicity of all other users' responses. This study contributes to the protection of individuals from being exposed to abusive language. There is one additional challenge: making the model lightweight and quicker to infer while needing fewer computational training resources. Additionally, we demonstrate that our compressed models are tiny enough to execute on the edge, for instance, on mobile devices and tablets.

2. `Data Understanding:` In this work, we utilized a dataset provided by Google Jigsaw in December 2017 as part of the Kaggle 'Toxic-Comment-Classification-Challenge. It is a freely available collection of labeled comments posted from Wikipedia-talk pages on Kaggle, which is a publicly accessible repository containing 159,571 labeled reader comments. Social rating agencies classified these statements as toxic, threatening, vulgar, and insulting. In this dataset, there is one column called "comment text" as an independent variable and another column called "toxic" as a dependent variable. The dependent variable is a binary classification in which we determine if a comment is toxic or not. A value of 1 indicates that the comment is poisonous, whereas a value of 0 indicates that the comment is not toxic.

3. `Data Preparation:` Data preparation is a data-mining methodology that relies on transforming raw data into a format that is understandable or acceptable. It cleans up raw data in preparation for future analysis. As user interaction on social media sites is often casual, the required English standards are rarely satisfied. As original data is inherently inconsistent, imprecise, and messy, it must be cleaned and transformed into a format that the classification model can comprehend. User comments include conventional English terms, acronyms, URLs, special symbols, slang, white-space, and emoticons, among other things. As a result, during our data processing and cleansing step, we developed a function utilizing the "re" library that eliminated stop-words, URL characters, meta tags, and special symbols such as punctuation marks. Stop-words are terms that appear often (and, it, is, etc.) but

are not required or significant for the model's classification phase. Moreover, we changed the comments to lowercase and deleted any whitespace. Also, we tokenized and encoded the training and validation sets. Each phrase should begin and conclude with special tokens. All sentences should be padded and truncated to a single fixed length. Utilize the "attention mask" to distinguish real tokens from padded tokens.

We used TF-IDF vectorization for fundamental machine learning algorithms such as logistic regression, decision tree classifiers, random forest classifiers, and xg-boost. This is a highly frequent approach for converting text to a meaningful numerical representation that is used to train ml algorithms for forecasting. It is a technique for deriving attributes from text data. Finally, we segmented the modeling dataset into training and validation sets, with 90% of the data being utilized to train the model and the remaining 10% being used to validate it. At Atlast, we test the model on 90% of the data that was utilized to train it. Using this method, we can assess our model's performance.

4. `Modelling:` To begin developing our model for categorizing toxic comments, we trained a specifically created teacher model, the Bert-Base model, with an inverted-bottleneck, before proceeding to MobileBERT. Following that, we used a knowledge distillation approach to communicate information to MobileBERT from this teacher model. Additionally, we show in this study that by using significantly smaller pre-trained language models (MobileBert) via knowledge distillation, we can accomplish similar outcomes on a wide range of downstream tasks, resulting in models that are lightweight, quicker to infer, and require less computational training. This is the first time, to our knowledge, that an unsupervised pre-trained model, MobileBERT (a thinner version of Bert-Base), has been used to transfer learning knowledge to low-resource toxic-comment categorization, increase task performance, and decrease model training time. Additionally, we employed ML algorithms such as Logistic-Regression, Decision-Tree-Classifier, Random- Forest-Classifier, and the XG-Boost algorithm. To train the model, we divided the dataset into a training set and a validation set, with the training set receiving 90% of the data and the validation set receiving 10%. At Atlast, we validate the model against 90% of the data used to train it. We may use this strategy to evaluate the performance of our model. Each model's required packages and libraries were installed and configured. We utilized the trained models to generate predictions on the test set. When we compared the results of each model, we discovered that it is possible to reduce the size of a BERT model by 43% while maintaining 97 percent accuracy in identifying toxic comments and running 60% faster than the Bert-Base while still achieving successful performance on well-known standards via knowledge distillation during the pre-training phase. In comparison to unsupervised state-of-the-art BERT models, other fundamental machine learning techniques obtained high accuracy but a low f1-score.

5. `Evaluation:` The execution outcome showed that our classifier was capable of simply and efficiently classifying words as toxic or non-toxic. The comparison stage revealed that the student model outperformed the teacher model and other fundamental machine learning algorithms such as the Random Forest classifier, Decision-Tree-Classifier, Logistic Regression, and xg-boost in terms of accuracy, f1-score, precision, and recall. Additionally, we evaluated our model using model training

time, model inference time, model weight, and a confusion matrix that showed the frequency of false positives and negatives. We will briefly describe it in the evaluation section.

6. `Deployment:` We trained our teacher (BertLarge) model using 90% of the training data, and it took around 2 hours and 30 minutes for two epochs, but our student model took 40% less time to train and yet achieved a higher accuracy and f1 score. We have built other basic ML algorithms, which are efficient but did not get a good f1-score.

4 Design Specification

This section describes the architecture that was utilized to create the state-of-the-art Bert as teacher model and its compressed model (student) MobileBert using a knowledge distillation approach for classifying toxic comments. Furthermore, it shows the implementation of supervised machine learning techniques for the purpose of identifying toxic comments. Figure 2 below illustrates the many actions that were taken. These stages involve importing the selected dataset into a Python environment using jupyter notebook, as well as pre-processing and cleaning the dataset. The next step was used to initialize the Bert models and train them; it involves the usage of sentiment polarity to validate the data's sentiment scores. The following models were constructed and trained in the section on building the model and training: BERT-Large, MobileBert, Logistic Regression, Decision Tree, Random Forest, and XG-Boost. The outcomes of all models will be compared in the next evaluation section, followed by a visual depiction of the final model's findings.

The next part will cover in depth the tools, characteristics, and functions used to develop the classification model.

5 Implementation

We employed the SimpleTransformer package, which is built on top of Hugging-Face, and simple machine learning methods to create the solution. The SimpleTransformer package offers a framework for quickly developing pre-trained transformer-based models for a variety of natural language processing applications. We employed knowledge distillation to categorize toxic comments from the Simple-Transformer library, with BERT-Base serving as the baseline model and MobileBERT serving as the student model. This section discusses the methods, tools, and libraries that were employed in our academic research. Additionally, the BERT-Base, MobileBERT, Logistic Regression, Random Forest, Decision Tree, and XG-Boost models' implementation details are discussed. The below figure 3 details the configuration used to develop this project.

5.1 Exploratory Data Analysis (EDA)

The toxic comment categorization dataset is taken from Kaggle and imported into a Kaggle notebook with GPU enabled. Python packages are used for exploratory data analysis and preprocessing. There are roughly 159,571 labeled user comments in the dataset, but relatively few examples from the toxic class (10.17%).

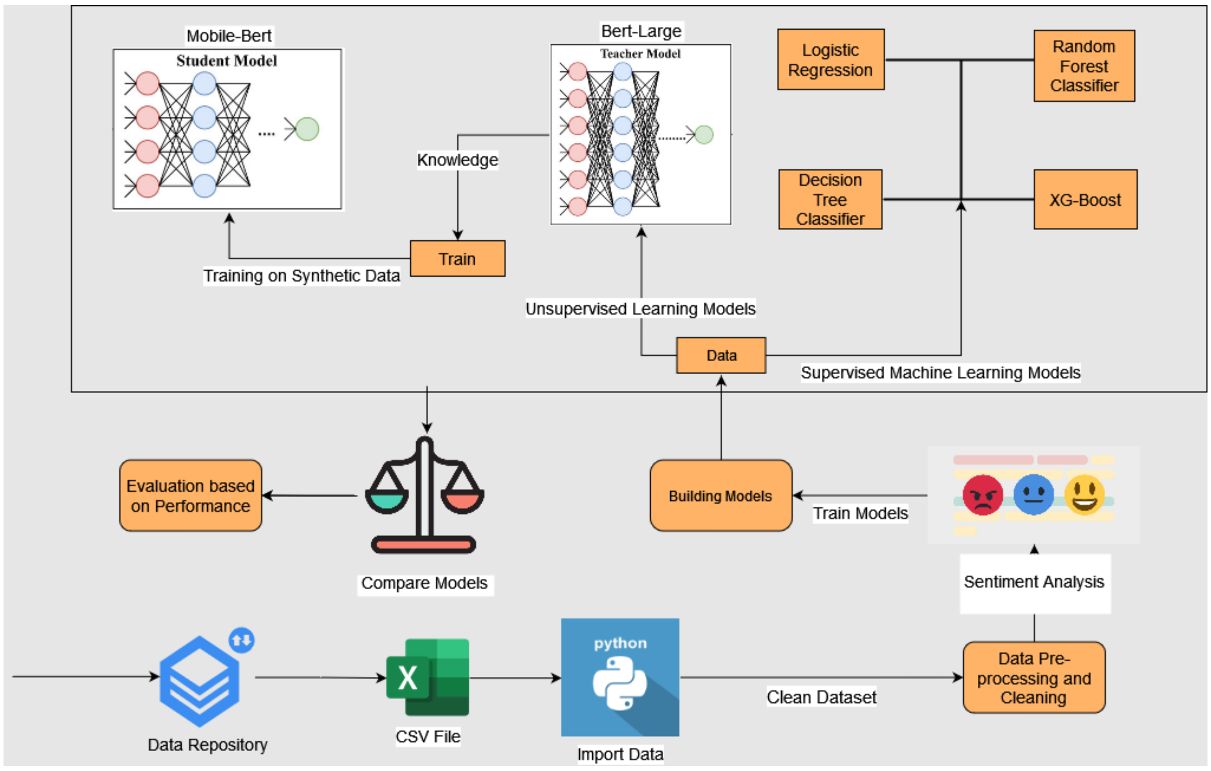
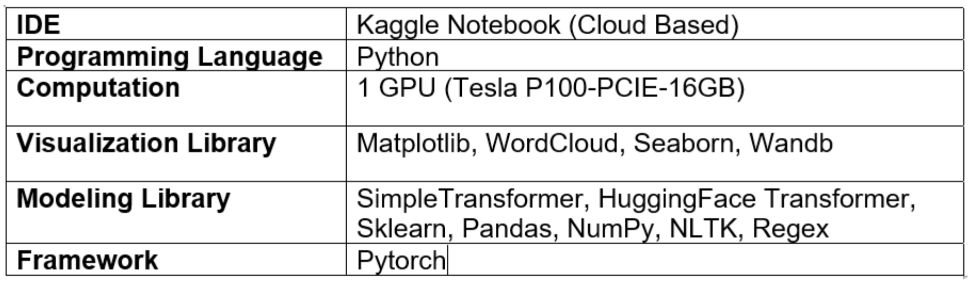


Figure 2: Toxic Comment Detection Proposed Framework



IDE	Kaggle Notebook (Cloud Based)
Programming Language	Python
Computation	1 GPU (Tesla P100-PCIE-16GB)
Visualization Library	Matplotlib, WordCloud, Seaborn, Wandb
Modeling Library	SimpleTransformer, HuggingFace Transformer, Sklearn, Pandas, NumPy, NLTK, Regex
Framework	Pytorch

Figure 3: Configuration

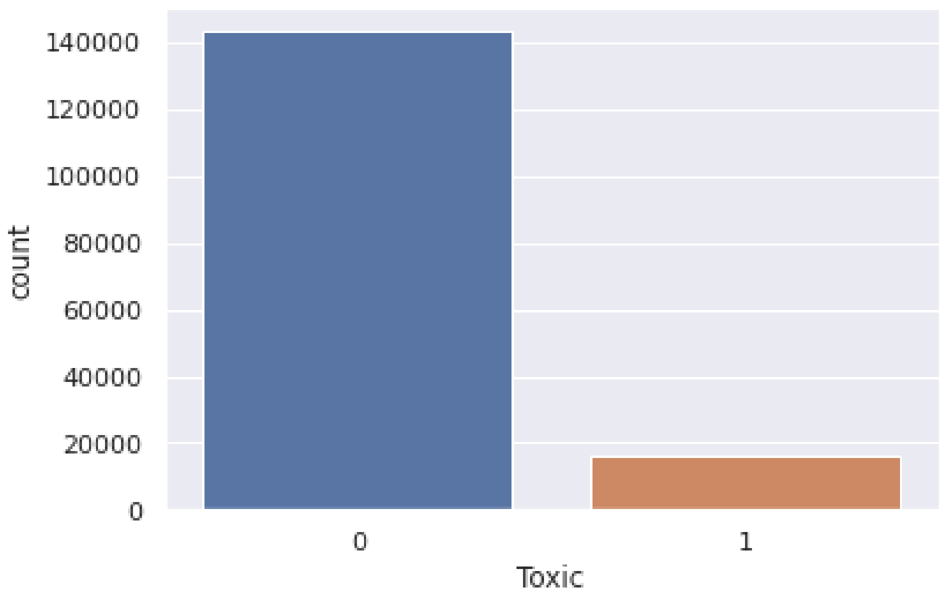


Figure 4: Count Plot for Toxic Comments

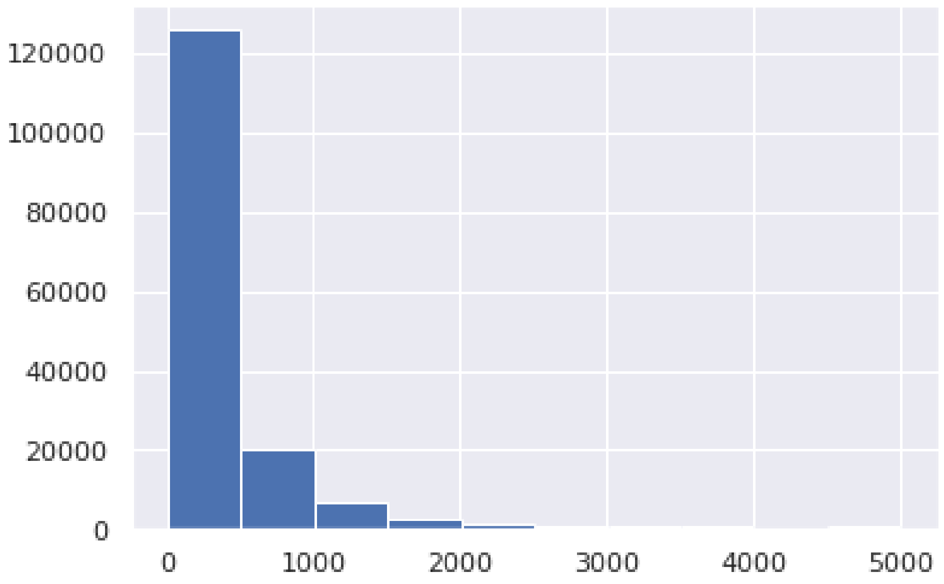


Figure 5: Character length per Comments

As seen in Figure 4, the total number of toxic comments is 16225, whereas the total number of non-toxic comments is 143346, indicating a highly skewed dataset.

In Figure 5, we represent the length of each comment text in the training data by the number of characters it contains. The majority of the comments are less than 500 characters long, with a few comments up to 5,000 characters.

5.2 Data Preprocessing and Cleaning

We replaced all instances of users, integers, tags, links, and regular emojis with the tokens <users>, <integers>, <tags>, <links>, and <emoticon>. Furthermore, we identified extended words and converted them to their regular short form; for example, we converted 'yeeees' to "yes". For example, we replaced tokens in hashtags with no spaces with their textual equivalents; for example, we changed the hashtag " notracism" to "no racism." We excluded all punctuation signs, unidentified unicodes, and other de-limiting characters, but we retained all stop-words since our model directly learns the order of words or phrases. Additionally, we transformed all comments to lowercase letters during word vectorization; hence, the same phrase would not be considered unique due to a case mismatch since they are case-sensitive. Finally, the text will be free of unwanted whitespace and punctuation.

The pre-trained models need identical-length input texts. To do this, a maximum length is specified based on the dataset being utilized and also to comprehend the distribution of evaluation lengths in the supplied dataset and to use computing resources efficiently. Each phrase begins and ends with specific tokens. All sentences are padded and truncated to a single, constant length and use the "attention mask" to distinguish genuine tokens from padded tokens. For tokenization in the student (MobileBERT) model, we employed the batch encode plus approach. For fundamental machine learning methods such as logistic regression, decision-tree-classifiers, random-forest-classifiers, and xg-boost, we employed TF-IDF vectorization. This is a very common technique for transforming text to a meaningful numerical representation for use in training machine learning algorithms for predicting. It is a method for extracting characteristics from unstructured text data.

5.3 Split training and test data

Segmentation of the dataset is accomplished using the Python module sklearn. Using the train-test split function, each sample of the dataset is divided into train-validate-test sets.

- For Teacher Model (BERT-Base): We segmented the data into 90 percent for model training and 10% for evaluation in the BERT-Base teacher model.We tested the model on 90% of the data used to train it.

- For Student Model (MobileBERT): We utilized the dataset generated during the teacher model testing process, termed the "synthetic dataset for student model".We employed 60% of the data for training the model, 20% for validating, and 20% for testing the built classifier.

- For basic ML Models: We utilized 80% of the data for model training and 20% for validation and testing of the developed classifier.

5.4 Knowledge Distillation

According to Hinton et al. (2015), Knowledge-Distillation is a method for compressing models in which a small model, the student, is taught to mimic the behavior of a bigger model, the teacher, or an ensemble of models. A classification model is generally trained in supervised learning to anticipate an instance class by optimizing the estimated probability of gold labels. Thus, a common training aim is to reduces the cross-entropy between the anticipated distribution of the model and the single-hot-empirical distribution of training labels. A classifier that performs well on the training dataset will anticipate an output distribution with a high probability for the actual class and almost no probability for the other classes. However, some of these "near-zero" probabilities are greater than others, and some reflect, in part, the model's generalization capability and its performance on the testing dataset.

Likewise, we used a knowledge distillation technique to categorize toxic comments. To begin, we trained the data for the teacher model across two epochs, with each epoch consisting of 4487 batches. The model achieved an average validation accuracy of 96.7 percent, an average precision of 80%, and a recall of 76% for both epochs, and it took approximately 2 hours and 15 minutes to train.In comparison, the student model ran on a synthetic dataset generated by the teacher model and ran for three epochs, each of which consisted of 2693 batches, and took approximately one hour and five minutes to train. This represents a nearly 60% reduction in training time and a higher performance score than the teacher model.

5.5 Model Training and Evaluation

Model performance is evaluated using the following metrics:

- Accuracy: This statistic is calculated by dividing the number of right predictions by the total number of samples in the test set.
- Precision and Recall: Precision and recall were developed to evaluate the model's ability to correctly label the toxic comments. Precision explains what fraction of harmful classed comments are genuinely toxic, whereas Recall assesses what fraction of dangerous remarks are correctly identified.
- F1 Score: Precision and recall are both critical for evaluating the model's performance. Implementing a more complex metric that combines Precision and Recall is, on the other hand, quite revealing and useful.
- Area Under Curve (AUC Score): This metric is one of the most extensively used to assess the effectiveness of binary classification models. It is the likelihood that a randomly chosen positive example will be ranked higher than a randomly chosen negative example by the classifier.
- Training Time: This metric shows the overall amount of time it takes a model to train.
- Model Size: It shows the size of the model.

5.5.1 BERT-Base (Teacher Model): Implementation and Results

To facilitate the creation of complicated transformer models, we utilized the Simple-Transformers library, which was developed on top of the Hugging-face library. The Classification Algorithm, which is specifically designed for binary-and multi-class text categorization, is utilized to create a classifier for toxic comment identification. BERT-base-uncased is used to fine-tune BERT for the specified dataset. To begin with, the BERT tokenizer tokenizes the comment text and converts the input text to tokens. The model building function is given a maximum sequence length of 300 as an input. The classifier is constructed across two epochs with a batch size of 32 for both training and evaluation. The smaller the batch size, the more computing power is required. As a result, we built a model using a batch size of 32. Furthermore, after building the classifier with a variety of alternative learning rates and evaluating the results, the rate of 5e-5 was chosen. The classifier is constructed using the NN cross-entropy loss function and the Adam-optimizer with an epsilon-value of 1e-8 is employed to minimize the losses.

We also constructed the "wandb project" parameter during training. It is compatible with the Weights and Biases framework, which is used to visualize model training. The framework generates dash-boards that assess the model's efficiency and facilitate the comparison of many trained models using gathered information. The visualizations give a clear view of how computing resources are being used. Calculate the amount of time that has passed and the train loss for each of the 20 batches. After each training period is complete, evaluate the model's performance on the validation-set. Running-time, training loss, validation loss, and validation accuracy are shown visually.

BERT has been fine-tuned to perform well on both portions of the datasets. 90% of the data is utilized for model training and 10% for model evaluation. We evaluated the toxic comments classifier on test data after model training using the predict function. We evaluated the classifier on 90% of the data used for training and added a column toxic predicted to serve as synthetic data for training the student model. During the test period, the dropout layers are deactivated. The dense and softmax-layers are implied in the classification-model.

Table 1 shows BERT's performance in identifying toxic comments across all assessment measures. This model requires two hours and fifteen minutes to train on two epochs, each of which has 4487 batches. The average validation loss is 0.10 percent and the average train loss is 0.05 percent, for a validation accuracy of 0.96 percent.

Table 1: BERT Results on Test Data (Teacher Model)

Evaluation Parameters	**Validation Data**	**Test Data**
Accuracy	0.96	0.98
Precision	0.80	0.98
Recall	0.76	0.79
F1 Score	0.78	0.88
AUC SCORE	0.98	0.9

ROC curves are used to measure the accuracy of classifications with various thresholds. AUC is a measure of the degree of separation, while ROC is a probability-curve that shows how likely it is that two groups can be distinguished. This metric shows the model's ability to distinguish between different classes. The higher the AUC, the more accurate the algorithm is in predicting 0 and 1 as 0 and 1, respectively. As an example, a

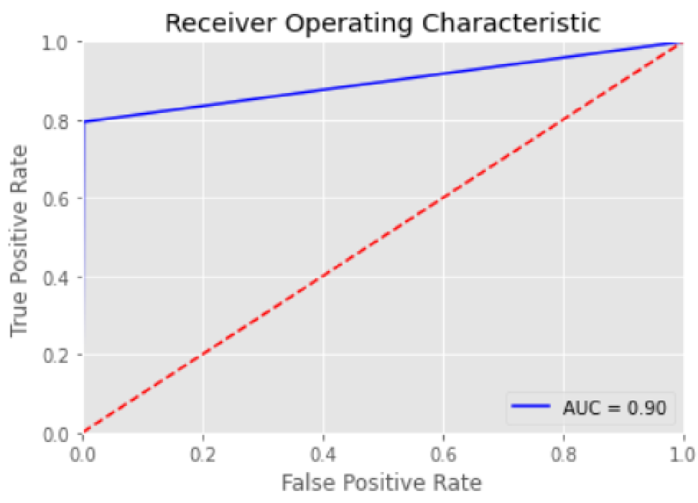


Figure 6: ROC-AUC Curve

larger AUC indicates a higher ability of the model to discriminate between harmful and non-toxic remarks. In order for a model to be considered good, the AUC must be near to 1. Figure 6 has an AUC value of 0.9, which is near to 1, which indicates that this model accurately recognized true harmful comments.Table 2 shows the training time for each epoch for BERT model. Figure 7 shows the confusion matrix of BERT model.

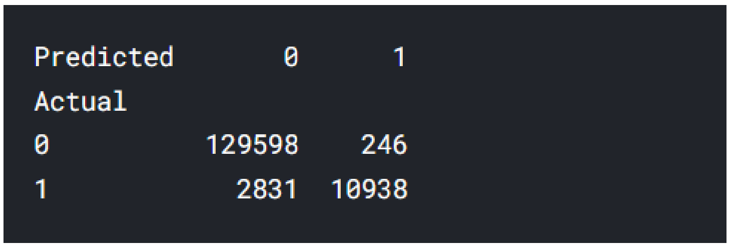
Table 2: Training Time of BERT model

Epoch	**Training Time (in mins)**
1	72.63
2	72.56

5.5.2 Mobile-BERT (Student Model): Implementation and Results

According to Sun et al. (2020), MobileBERT is intended to be as deep as BERT-LARGE, but each layer is substantially smaller due to the use of bottleneck architectures and the balancing of self-attention and feed-forward networks. MobileBERT is designed to be as deep as BERTLARGE while each layer is made much narrower via adopting bottleneck structures and balancing between self-attentions and feed-forward networks (Figure 8). The architecture of MobileBERT is illustrated in Figure 7(c). It is as deep as BERTLARGE, but each building block is made much smaller. As shown in Figure 9, the hidden dimension of each building block is only 128. On the other hand, we introduce two linear transformations for each building block to adjust its input and output dimensions to 512. To train Mobile-BERT, a deep and light model, we first train a specifically built teacher model, which, in our instance, is BERT-Base. Then we transfer knowledge from BERT-Base to MobileBERT. Training such a deep and thin network is difficult. To address the training problem, we first build a teacher model and train it until completion, then transmit knowledge from this teacher model (BERT-Base) to Mobile-BERT. We believe that this is far superior to training MobileBERT from the start.

The "google/mobileBERT-uncased" is used to fine-tune MobileBERT using the synthetic dataset generated by the teacher model. To begin, the batch encode plus function



```
Predicted       0      1
Actual
0          129598    246
1            2831  10938
```

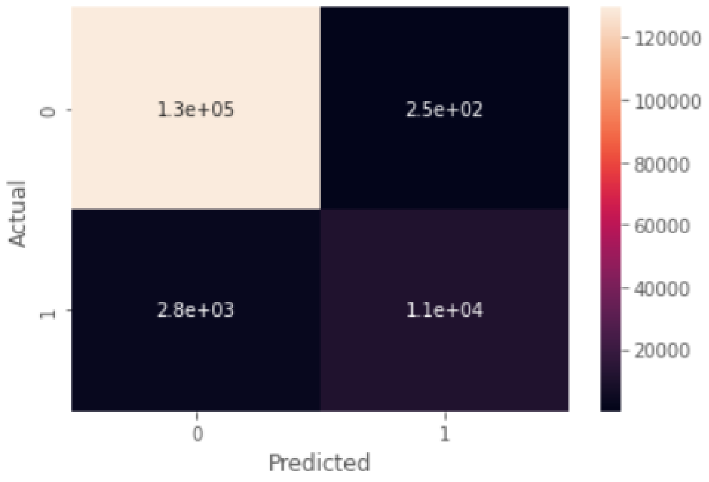


Figure 7: Confusion Matrix of BERT Model

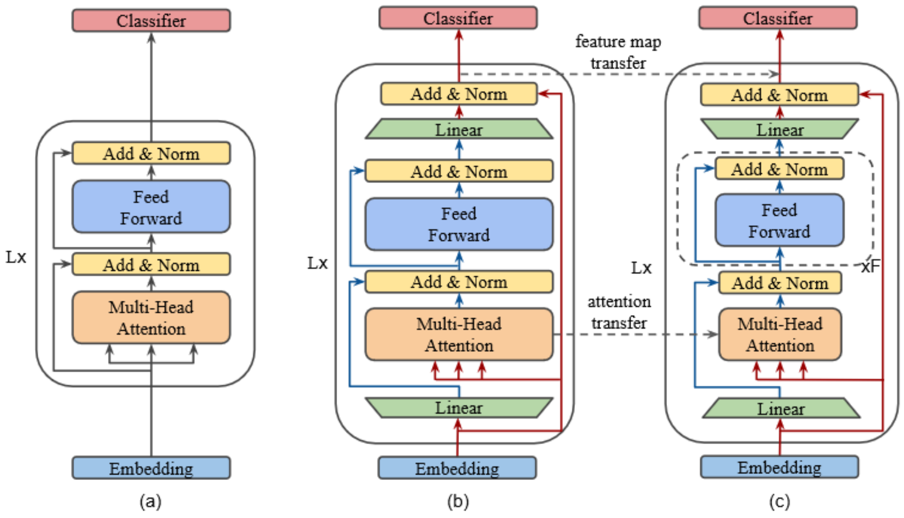
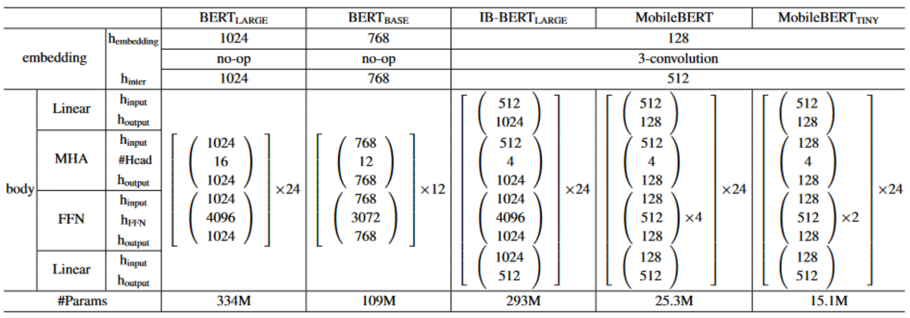


Figure 8: Illustration of three models: (a) BERT; (b) Inverted-Bottleneck BERT (IB-BERT); and (c) MobileBERT.In (b) and (c), red lines denote inter-block flows while blue lines intra-block flows. MobileBERT is trained by layer-to-layer imitating IB-BERT.



			BERT$_{LARGE}$	BERT$_{BASE}$	IB-BERT$_{LARGE}$	MobileBERT	MobileBERT$_{TINY}$
embedding		$h_{embedding}$	1024	768	128		
			no-op	no-op	3-convolution		
		h_{inter}	1024	768	512		
body	Linear	h_{input}			512	512	512
		h_{output}			1024	128	128
	MHA	h_{input}	1024	768	512	512	128
		#Head	16	12	4	4	4
		h_{output}	1024	768	1024	128	128
	FFN	h_{input}	1024	768	1024	128	128
		h_{FFN}	4096	3072	4096	512 ×4	512 ×2
		h_{output}	1024	768	1024	128	128
	Linear	h_{input}			1024	128	128
		h_{output}			512	512	512
			×24	×12	×24	×24	×24
#Params			334M	109M	293M	25.3M	15.1M

Figure 9: The detailed model settings of a few models. hinter, hFFN, hembedding, Head and Params denote the inter-block hidden size (feature map size), FFN intermediate size, embedding table size, the number of heads in multi-head attention, and the number of parameters, respectively.

was used to tokenize text into batches. We set the batch size to 32 so that the training model uses fewer computing resources. The tokenizer then tokenizes the comments, and the input text is turned into tokens. The model training algorithm receives a maximum sequence length of 200 as input. The model is trained for three epochs with a batch size of 32 for training and evaluation. In addition, after training the model with various learning rates and evaluating performance, the rate of 3e-5 is chosen. The classifier is constructed using the BCEWithLogitsLoss loss function, which combines LogSoftmax and NLLLoss into a single class, and the losses are optimized using the Adam-optimizer with an epsilon value of 1e-8. The product of [number-of-batches] and [number-of-epochs] equals the number of training steps. We saved a variety of parameters, including training and validation-loss, validation accuracy, and training time for every 20 batches, as well as cumulative training time for three epochs. As we get errors out of memory, we estimate the accuracy of test data for the first 1000 records. We also executed a forward-pass to assess the model on the training-batch and a backward pass to compute the gradients and reduce the gradient standard to 1, which helped avoid the ”exploding gradient” issue.

MobileBERT has been refined using both validation and testing datasets. 60% of the data is used for training the model, whereas 20% is used for model evaluation and testing. Using the predict algorithm, we assessed the toxic comments classifier on validation and test data after model training. MobileBERT performed three epochs in one hour and five minutes. Each epoch is composed of 2693 batches. We saved the model after training it and examined its size. The stored MobileBERT model is 98.79 MB, whereas the BERTBase model is around 1.3 GB. This demonstrates that Mobile BERT is much lighter than the teacher model (BERT-Base). Furthermore, MobileBERT takes almost 60% less time to train the model, making it significantly quicker. Table 3 displays MobileBERT’s performance in detecting toxic comments across all evaluation measures.

Figure 10 shows ROC-AUC curve of Student Model on Test Data. Figure 11 shows the training and validation accuracy as well as the training and validation loss plot for the MobileBERT model.

Table 4 show metrics such as training and validation loss, training and validation accuracy, and training and validation time for all 3 epochs. Figure 12 shows the confusion

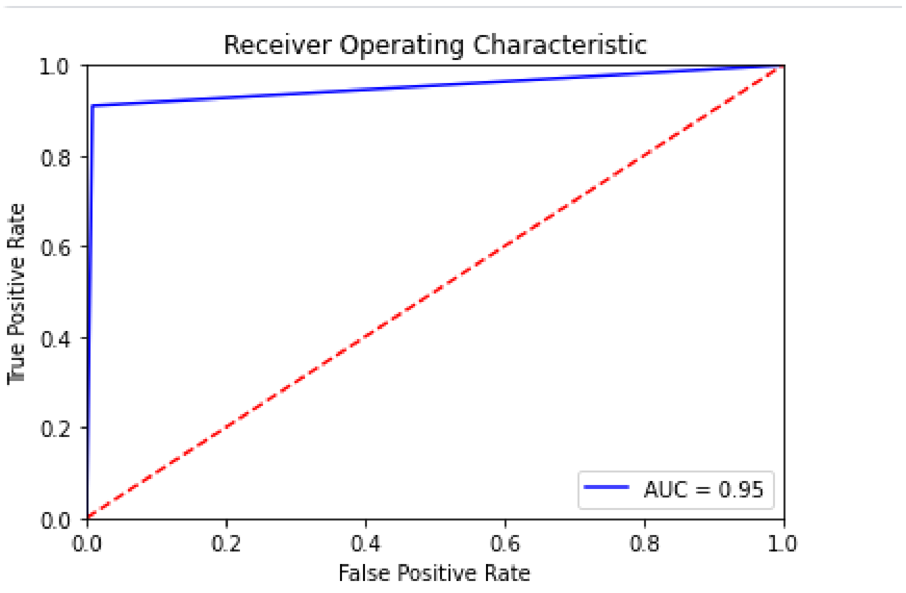


Figure 10: ROC-AUC Curve

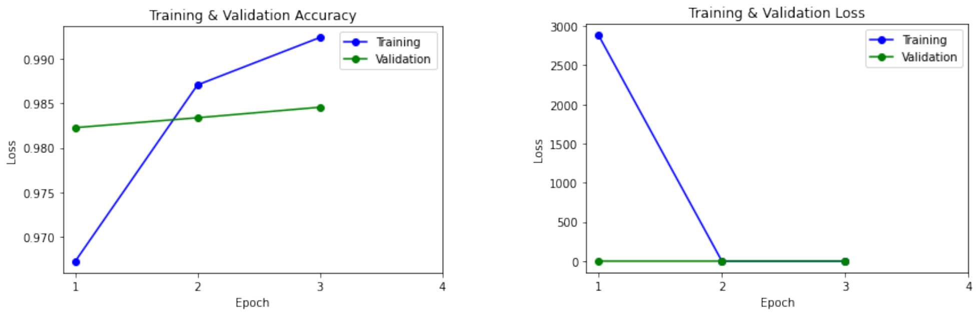


Figure 11: Training and Validation Comparison Plot

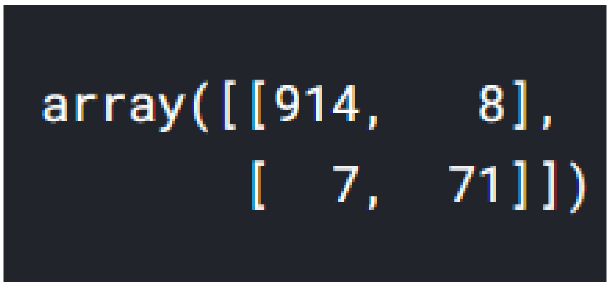
Table 3: MobileBERT Results (Student Model)

Evaluation Parameters	**Test Data**
Accuracy	0.98
Precision	0.92
Recall	0 0.88
F1 Score	0.9
AUC Score	0.95

Table 4: Training Parameters of MobileBERT

Epoch	**Train Loss**	**Valid Loss**	**Train Accu**	**Valid Accu**	**Train Time**	**Valid Time**
1	2880.5399	0.0735	0.9672	0.9823	00:20:42	00:01:27
2	0.04983	0.0512	0.9871	0.9834	00:20:32	00:01:27
3	0.03173	0.0658	0.9924	0.9846	00:20:24	00:01:27

matrix of MobileBERT model on test data.



```
array([[914,   8],
       [  7,  71]])
```

Figure 12: Confusion Matrix of MobileBERT

5.5.3 Basic Machine Learning Models: Implementation and Results

In this paper, We used a variety of basic machine learning models to classify toxic comments, including Logistic Regression, Random Forest Classifier, Decision Tree Classifier, and XG-Boost. We divided the data set into two parts: 80% is used to train the model, and the other 20% is used to test the model. Table 5 summarizes the results for implemented models.

6 Discussion

6.1 Performance Comparison of Developed Models

Observing the outcomes in Tables 1, 2, and 3, we discovered that both the Bert-Base and MobileBert models give state-of-the-art outcomes on testing data. We noticed that the Tacher model (Bert-Base) performed better on test data than on validation data. We retained 10% of the data for validation and evaluated the model on 90% of the data that had previously been used for training. Figure 6 shows that the TPR Rate (True Positive Rate) is near one and the FPR Rate (False Positive Rate) is near zero, indicating that

Table 5: Basic ML Models results on Test Data

Eval Parameters	**Logistic Regression**	**Random Forest**	**Decision Tree**	**XG-Boost**
Accuracy	0.96	0.96	0.94	0.95
Precision	0.92	0.86	0.72	0.92
Recall	0.62	0.70	0.69	0.60
F1 Score	0.74	0.77	0.70	0.73

this model correctly identified true toxic comments.On the other hand, our student model (MobileBert) outperformed the teacher model in terms of accuracy, precision score, recall, and f1 score. For classifying toxic comments, we also used fundamental machine learning methods such as logistic regression, decision trees, random forests, and XG-Boost. We evaluated these models on 20% of the test data. Hence, We received a high accuracy but a low recall score when compared to Bert models, indicating that these models performed less well than Bert models in accurately predicting toxic comments for all observations in the actual class. These models likewise have a lower f1 score than Bert models, as seen in the evaluation section. To conclude, our student model outperformed all other models using the knowledge distillation approach.Although we have assessed whether or not a model overfitting problem exists via the confusion matrix, we can be certain that additional parameters must be evaluated in order to determine whether or not a model overfitting problem exists.

6.2 Computational Time Comparison of Developed Models

We employed Bert models to classify toxic comments on a given dataset, and we trained the model on a GPU (Tesla P100-PCIE-16GB). We applied cutting-edge Bert-Base (Teacher) models for two epochs, with each epoch consisting of 4487 batches. We measured the model's training and found that it took 2 hours and 15 minutes for two epochs. After training the model, we saved it and checked the model size, which was around 1.3 GB. It demonstrates that it is a large model. We utilized batch size 32 to save computing resources. On the other hand, our student model (MobileBert) was trained for three epochs, with each epoch consisting of 2693 batches, and it took 1 hour and 5 minutes to train the model, which was about 60% less than the teacher model. The student model is just 98.79 MB in size, which is quite small compared to the teacher model.Also, over two epochs, we found that the teacher model (Bert-Base) had an average validation loss of 0.09 while the student model (MobileBert) had an average validation loss of 0.06. In terms of assessment measures, the student model beat the teacher model, although our training loss is significantly greater for MobileBert. LogisticRegression, DecisionTreeClassifier, RandomForest and XGBoost algorithms were implemented on the machine's CPU and evaluated on 20% of its data.

6.3 Model Comparisons Between Developed and Baseline Models

Bert-Base was built as a base line model that was assessed on a validation set and a test dataset. Although the assessment metrics score is high, it took a long time to train and the model size is large. To address this problem, we used knowledge distillation to implement the MobileBert model as a student model, which is even lighter than the

DistilBert version. The findings are rather impressive, since the student model performed better than the teacher model despite having less training time and a smaller model size. We also employed LogisticRegression, RandomForest, DecisionTree, and XG-Boost to categorize toxic comments, but we received less f1-scores and recall scores when compared to the baseline model. Thus, based on the findings of MobileBert, we can conclude that it outperformed the baseline model.

7 Conclusion and Future Work

The detection of toxic comments on internet portals is a constantly evolving field of study. Earlier, both ML and DL neural-network models were effectively used to categorize toxic content on online platforms. However, ML and DL models, respectively, rely on feature extraction and a significant amount of data. We show in this article that primitive, lightweight neural-networks can be effective and hence beneficial. We propose to distill knowledge from BERT, a state-of-the-art-language classification model, into a MobileBert, a task-independent compact variation of BERT. Additionally, we classified the toxic comments using the Logistic Regression, Decision Tree, Random Forest, and XG-Boost models. Transfer learning is possible in NLP using pre-trained models. The data for this analysis was collected from Kaggle and preprocessed in Python using user views expressed on Wikipedia-talk pages. Following that, we trained our teacher model for categorizing toxic comments, although it took a long time and the model is rather large. To address this problem, we built MobileBERT, a task-independent compact form of BERT. MobileBERT is similar to BERTBASE on major NLP benchmarks, but is much smaller and quicker. MobileBERT enables the rapid deployment of a variety of natural language processing applications on mobile devices. In this research, we demonstrate that MobileBert is 60% faster and significantly smaller in size than our teacher model (Bert-Base), which stores approximately 1.3 GB of trained models, whereas MobileBert stores approximately 98.79 MB of trained models and retains 98 percent accuracy in classifying toxic comments. The model hyper-parameters are carefully chosen, and the models are trained for three epochs using the preprocessed data (MobileBert). A pre-trained Bert-Base model is used as a baseline model. Due to the dataset's extreme imbalance, the major assessment metrics are accuracy, precision, recall, ROC-AUC Score, and f1-score. The model assessment findings indicate that MobileBert outperforms Bert-Base, LogisticRegression, RandomForest, DecisionTree, and XG-Boost models previously deployed. Our goal of providing state-of-the-art outcomes with little processing resources is achieved since MobileBert is 60% quicker and much smaller than the Baseline model and other created models, and beats them on assessment measures.

For future development, since MobileBERT is much smaller and quicker, it enables the easy deployment of diverse NLP applications on mobile devices, as per Sun et al. (2020). We propose that these compressed-models be used to classify hate speech and abusive comments on tiny devices such as mobile phones and tablets.

Acknowledgement

I would like to express my appreciation to Professor Rejwanul Haque for his constant assistance and support throughout my research. My ability to complete this course is

due to their constant effort and attention.

References

Badjatiya, P., Gupta, S., Gupta, M. and Varma, V. (2017). Deep learning for hate speech detection in tweets, *CoRR* **abs/1706.00188**.
URL: *http://arxiv.org/abs/1706.00188*

Chatterjee, D. (2019). Making neural machine reading comprehension faster, *CoRR* **abs/1904.00796**.
URL: *http://arxiv.org/abs/1904.00796*

Davidson, T., Bhattacharya, D. and Weber, I. (2019). Racial bias in hate speech and abusive language detection datasets, *CoRR* **abs/1905.12516**.
URL: *http://arxiv.org/abs/1905.12516*

Davidson, T., Warmsley, D., Macy, M. W. and Weber, I. (2017). Automated hate speech detection and the problem of offensive language, *CoRR* **abs/1703.04009**.
URL: *http://arxiv.org/abs/1703.04009*

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V. and Bhamidipati, N. (2015). Hate speech detection with comment embeddings, *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, Association for Computing Machinery, New York, NY, USA, p. 29–30.
URL: *https://doi.org/10.1145/2740908.2742760*

Founta, A. M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A. and Leontiadis, I. (2019). A unified deep learning architecture for abuse detection, *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, Association for Computing Machinery, New York, NY, USA, p. 105–114.
URL: *https://doi.org/10.1145/3292522.3326028*

Gambäck, B. and Sikdar, U. K. (2017). Using convolutional neural networks to classify hate-speech, *Proceedings of the First Workshop on Abusive Language Online*, Association for Computational Linguistics, Vancouver, BC, Canada, pp. 85–90.
URL: *https://aclanthology.org/W17-3013*

Hinton, G., Vinyals, O. and Dean, J. (2015). Distilling the knowledge in a neural network.

Hu, M., Peng, Y., Wei, F., Huang, Z., Li, D., Yang, N. and Zhou, M. (2018). Attention-guided answer distillation for machine reading comprehension, *CoRR* **abs/1808.07644**.
URL: *http://arxiv.org/abs/1808.07644*

Malmasi, S. and Zampieri, M. (2018). Challenges in discriminating profanity from hate speech, *CoRR* **abs/1803.05495**.
URL: *http://arxiv.org/abs/1803.05495*

Mehdad, Y. and Tetreault, J. (2016). Do characters abuse more than words?, *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Association for Computational Linguistics, Los Angeles, pp. 299–303.
URL: *https://aclanthology.org/W16-3638*

Schwartz, R., Dodge, J., Smith, N. A. and Etzioni, O. (2019). Green AI, *CoRR* **abs/1907.10597**.
URL: *http://arxiv.org/abs/1907.10597*

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y. and Zhou, D. (2020). Mobilebert: a compact task-agnostic bert for resource-limited devices.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O. and Lin, J. (2019). Distilling task-specific knowledge from BERT into simple neural networks, *CoRR* **abs/1903.12136**.
URL: *http://arxiv.org/abs/1903.12136*

Turc, I., Chang, M., Lee, K. and Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation, *CoRR* **abs/1908.08962**.
URL: *http://arxiv.org/abs/1908.08962*

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on Twitter, *Proceedings of the NAACL Student Research Workshop*, Association for Computational Linguistics, San Diego, California, pp. 88–93.
URL: *https://aclanthology.org/N16-2013*

Waseem, Z., Thorne, J. and Bingel, J. (2018). Bridging the gaps: Multi task learning for domain transfer of hate speech detection.

Wiegand, M., Ruppenhofer, J. and Kleinbauer, T. (2019). Detection of Abusive Language: the Problem of Biased Datasets, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 602–608.
URL: *https://aclanthology.org/N19-1060*

Zhang, Z., Robinson, D. and Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. © 2018 Springer International Publishing AG, part of Springer Nature. This is an author produced version of a paper subsequently published in Lecture Notes in Computer Science. Uploaded in accordance with the publisher's self-archiving policy.
URL: *https://eprints.whiterose.ac.uk/128405/*