Project/Thesis No.: CSER-24-47

CSE 4000: Thesis/ Project

A Comprehensive Study on Transformer-based Architectures and Their Application in Hate Speech Classification from Textual Data

By

Bokhtiar Adil Prottoy

Roll: 1807028

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

February, 2024

A Comprehensive Study on Transformer-based Architectures and Their Application in Hate Speech Classification from Textual Data

By

Bokhtiar Adil Prottoy

Roll: 1807028

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

Supervisor:

Md Nazirulhasan Shawon

Lecturer,

Department of Computer Science and Engineering,

Khulna University of Engineering & Technology,

Khulna, Bangladesh.

_____

Signature

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

February, 2024

Acknowledgement

All the praise to the Almighty Allah, whose blessing and mercy succeeded me in completing this work fairly. I gratefully acknowledge the valuable suggestion, advice, and sincere cooperation of my honorable supervisor, Md Nazirulhasan Shawon, without whom this work might not have reached this stage. I would also like to thank all the researchers, teachers, authors, and writers in various online platforms, whose explanations and provided resources in this field made it possible for me to dive this far into the study.

Author

Abstract

Due to worldwide expansion and easy access, the internet has become an active breeding ground for hate speech and offensive comments. Especially in social media, such activities have created legitimate concerns among specialists. To tackle this, a robust hate speech classification mechanism is needed. In this paper, to classify hate speech from textual data, an investigation of different types of transformer-based architectures was performed. Several large language models were extensively studied. Four major tasks were performed throughout the thesis. A cross-dataset analysis was presented using six existing datasets and three transformer-based model architectures - ALBERT, DistilBERT, and XLM-RoBERTa to measure the generalization ability of these datasets. Through the cross-dataset analysis, it was found that the size and the class balance of these datasets didn't have any noteworthy impact on the generalization ability of the models trained on them. Results showed that the models trained on the OLID dataset performed nearly consistently better in all test scenarios compared to the other models indicating its superior generalization ability. Results also indicate that the innate randomness and diversity of hate speech make it too hard to build a generalizable dataset. After that, two types of ensemble networks were prepared using different types of models on each dataset. They were hard-voting ensemble and weighted ensemble. The latter was found to be the best-performing approach. Then, using syntactic manipulation, several types of adversarial samples were generated using the HASOC2020 dataset and the OLID dataset. These augmented datasets were experimented with different models to improve the result. The investigation showed improvements in the result of the previously underperformed models on these datasets. Finally, a contrastive loss was introduced using the 'pull and push' mechanism in the training process on the augmented datasets to optimize the representations of the sentences in the embedding space. This method considerably improved the performances of ALBERT and XLM-RoBERTa models on both datasets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Internet communication is an integral part of the current technology-based world. Almost in every part of our lives, we are compelled to take part in interactions based on the internet. Especially the rise of social media has completely reshaped the entire humanity. It has opened a new horizon in expanding relations across continents. Unfortunately, it has also unraveled a new threat and unleashed a new devil in the wild. And that is the alarming acceleration in the use of hate speech and toxicity. These verbal attacks and abuses have victimized millions of people and traumatized thousands of innocent kind-hearted souls. Particularly, religious and ethnic minorities have become the main targets of most of these abuses.

The definition of hate speech is a fairly disputed term. Different sources define it differently. Different organizations have different policies on managing hate speech. This difference of opinion in defining hate speech has a significant impact on annotating real-world data. Here is a summary of different definitions of hate speech in contemporary sources -

- A complete and comprehensive definition of Hate Speech (HS) is given by General Policy Recommendation no. 15 of the European Commission and defines it as ”the advocacy, promotion or incitement, in any form, of the denigration, hatred or vilification of a person or group of persons, as well as any harassment, insult, negative stereotyping, stigmatization or threat in respect of such a person or group of persons and the justification of all the preceding types of expression, on the ground of race, color, descent, national or ethnic origin, age, disability, language, religion or belief, sex, gender, gender identity, sexual orientation and other personal characteristics or status.” [1]

- Encyclopedia of the American Constitution: “Hate speech is speech that attacks a person or group on the basis of attributes such as race, religion, ethnic origin, national origin, sex, disability, sexual orientation, or gender identity.” [2]
- Facebook: “We define hate speech as a direct attack against people – rather than concepts or institutions – on the basis of what we call protected characteristics: race, ethnicity, national origin, disability, religious affiliation, caste, sexual orientation, sex, gender identity, and serious disease. We define attacks as violent or dehumanizing speech, harmful stereotypes, statements of inferiority, expressions of contempt, disgust or dismissal, cursing, and calls for exclusion or segregation. We also prohibit the use of harmful stereotypes, which we define as dehumanizing comparisons that have historically been used to attack, intimidate, or exclude specific groups, and that are often linked with offline violence. We consider age a protected characteristic when referenced along with another protected characteristic.” [3]
- Twitter: ”You may not directly attack other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease.” [4]

Regardless of the definition, there is no denying the fact detecting hate speech in an automated fashion on online platforms is necessary. Over the years, many scientists have tried to crack the way to detect hate speech efficiently. Many different kinds of methods have been proposed. From applying classical methods of machine learning to utilizing relatively new large language models, so many great approaches have been adopted by them. Hate speech detection is nonetheless a specific example of a sentence classification task. It’s also a kind of sentiment analysis. However, many of these methods have succumbed to different types of weaknesses.

In the last few years, transformer-based architectures and large language models have completely flooded the entire Natural Language Processing (NLP) world. Like all other NLP tasks, they are being used in the sentence classification process too. Leveraging this architecture, several large language models have been developed. These models are trained on billions of instances and contain very deep computational layers. BERT, ALBERT, DistilBERT, XLM-RoBERTa, GPT, etc are some of the prominent large language models. These models have immense potential and opened up creative opportunities to implement an efficient hate speech classification model.

The adversarial attack is a way to mimic real-life data. This process can be adapted to teach the model to deal with different types of noise in a controlled environment. Even the mighty transformer-based architectures or the large language models are susceptible to noisy data and distorted forms of expression. Adversarial attack-based learning can be a big help to make the model more robust. The contrastive learning method is also

another way to teach the model to understand the distortion of a particular sample. This is a fairly new concept in the hate speech classification domain.

This research performs an extensive study on the mechanism of transformer-based architectures and large language models. Then, it investigates their application in the hate speech classification field. This study analyzes six different existing hate speech datasets using three different types of transform-based model architectures. To improve the result, two different types of ensemble modeling approaches are applied. Then the effect of adversarial samples and contrastive learning are investigated. The purpose of this study is to utilize transformer-based architectures most effectively to build a robust and reliable hate-speech classification model.

1.2 Background

The alarming rise of hate speech and abusive remarks has affected all sectors of our online interactions. From social media to online gaming platforms, every corner of the internet has been infected by this plague. Scientists have tried to counter this issue by developing different models and architectures that can detect hate speech on a real-time basis. Over the years they have developed many keyword-based approaches, rule-based approaches, classical machine learning-based approaches, emotion modeling approaches, etc. A detailed discussion of these approaches can be found in chapter 3.

1.2.1 Importance of Context

Many of the approaches invented so far suffer from a fatal limitation which is failing to capture the context of the hate speech. For example, consider these sentences -

- That tough ba***d was not killed in the war
- Kill that Asian ba***d

Both of them contain the infamous b-word and the word 'kill' which has a negative connotation. But, the context of the first sentence suggests that the narrator is commending a tough warrior whereas in the second sentence, it is clear that the narrator is expressing his utter hatred for Asian people, which is a form of racism. So, capturing the right context is important.

Also, the proper context may not be present in the nearby words. For example,

- "That filthy Asian guy, who came to visit our home with many gifts and presents, I hope his people rot in their disgusting country"

This sentence contains multiple racist elements residing at the two farthest parts of the

sentence. However, it also contains two positive words 'gifts' and 'presents' in the middle. Here, not only the distance between the two parts of the sentence context is long, but the middle portion contains positively connotative words that can easily confuse any model. Classical approaches can not capture such long-distance contexts. Neural network (NN) based models such as CNN, RNN, LSTM, BiLSTM, etc can capture context up to a degree, but when the sequence is too long, the chance of failure for them gets very high. A long test sample with a lot of words with characteristics like the above example may cause their failure.

Context is also affected by the domain and the demography of the narrators. It can have many types too such as racist, sexist, anti-Semitic, anti-religious, etc. The time, place, genre of the medium, recent events, and many other factors can also influence the context of hate speech. Depending on the attributes, the same sentences can be interpreted as both hate speech at some conditions and non-hate speech at other conditions. Some researchers, therefore, introduced some other classes such as 'offensive', 'toxic', 'aggression', 'insult' etc. However, as we have already discussed, there are no universally accepted clear definitions of the distinguishing factors between these classes. As a result, these classes introduce an unwanted bias to the datasets that can negatively affect the models.

1.2.2 Dataset Generalizability

Hate speech is innately random. Depending on the use case, the same sequence of words can convey opposite meanings. For a model to perform well in tackling such varying contexts and randomness, it needs to have a good understanding of language as well as a sound generalization ability. Even with the existence of the transformer-based massive pre-trained models with tremendous computational abilities, the importance of a well-crafted dataset can not be ignored during fine-tuning tasks as the generalization ability greatly depends on it. The inherent nature of hate speech makes it easily affected by time, place, events, etc. Even the type of online platforms changes the genre of the hate speech. With such a vast variety of how hate speech is expressed in real life, models trained on only a single domain or demography can fail miserably when deployed.

A good dataset should introduce the models with such randomness, without any bias, during training most effectively. Many hate speech datasets have been developed over the years. However, most of these datasets contain some sort of demographic or other types of bias that can not be alleviated. This greatly hampers the generalization abilities of the models that are trained on them. [5] showed that the nature of the data is more important than the model architecture in the hate speech classification domain. Even though the transformer architectures are already pre-trained on huge amounts of multi-domain data, the dataset nature used during fine-tuning plays a crucial role in developing the generalization ability of the models.

1.2.3 Dealing with Randomness

One way to confront the randomness issue of hate speech is to teach the model the possible defects and noise in the data. This can be done by deliberately modifying the actual data and feeding it to the model. This way the model can be taught to learn the possible altercations of a sample in a controlled environment. This is called the adversarial attack. This process augments the dataset using multiple defective and modified versions of the actual sample. All these defective samples will have the same label as they essentially are the same thing. Contrastive learning adds another layer to this adversarial attack approach. It uses a pull and push mechanism to maximize the similarity between the original and the defective samples in the embedding space while minimizing the similarity with other samples. This makes it more suitable for the model to understand the nature of the randomness of the hate speech dataset.

The purpose of this study is to address all these issues and investigate the way to solve them by utilizing transformer-based architectures along with other techniques to come up with a robust hate speech classification model.

1.3 Objectives

The objectives of this study are -

- Perform a comprehensive study on transformer-based architectures and large language models.
- Perform a cross-dataset analysis on several different existing hate speech datasets using multiple types of transformer-based architectures to test their generalization abilities and find the best-performing model.
- Apply different types of ensemble modeling to improve the result achieved by the cross-dataset analysis.
- Produce adversarial samples from the clean dataset, train different model architectures using it, and investigate its effect on the model performance.
- Implement a contrastive learning mechanism and attach it to the model architectures to find out how well it improves the model's performance.

1.4 Scope

This study needs to analyze textual data, which is highly computationally expensive. A good computer with a powerful CPU, GPU, RAM, and SSD is needed. A good internet connection is also needed. Modern data science-related tools such as Notebooks, kernels,

IDE, etc are needed. For coding purposes, Python provides a great library, PyTorch. This is a specialized library for the usage of data science-related activities. For IDE, Kaggle Notebook, Google Colaboratory, Visual Studio Code, Spyder IDE, etc will be used.

1.5 Unfamiliarity of the Problem

Hate speech detection is considered a unique and valuable research idea. It has a great social impact. It bears immense ethical responsibility. Cultural, contextual diversity, political perspective, limited resources, privacy and freedom of speech balance, etc are the main concerns in this field. An automated hate speech detection model can help to prevent the spread of negativity in online interactions. However, because of the randomness of the nature of hate speech, it is necessary to investigate the possible options to improve the performance of an automated hate speech detection model. Recent inventions such as adversarial attacks and contrastive learning methods need to be explored and implemented. Dataset generalizability is another concern that needs to be addressed. The existing datasets require to be analyzed thoroughly. Many studies have discussed different aspects of these issues but it is hard to find a single study that systematically researches all these domains together.

1.6 Project Planning

The entire project planning timeline is shown in the following Gantt charts.

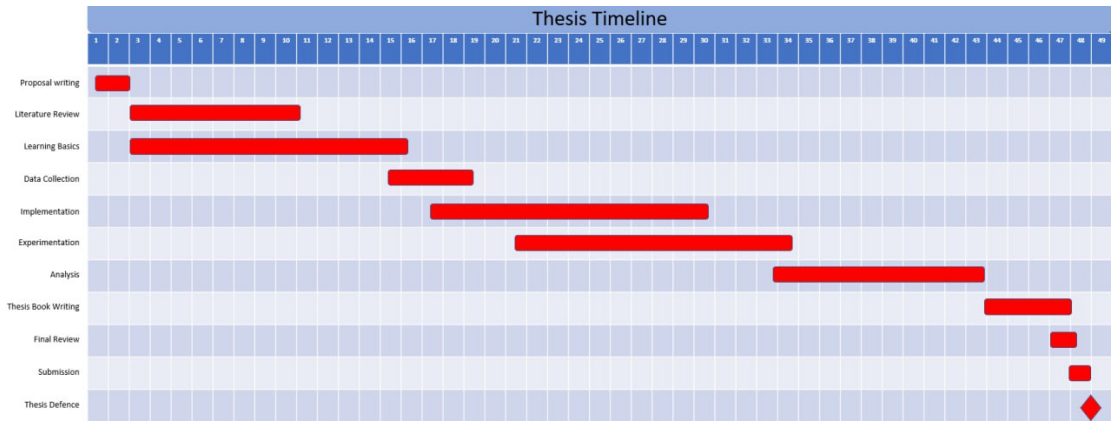


Figure 1.1: Gantt chart of the thesis timeline (in weeks)

1.7 Application of the Work

There are many possible ways the result of this work can be applied.

- Social Media Platforms: Many social media platforms face challenges with hate speech proliferating within their user-generated content. A transformer-based hate speech detection model can be integrated into these platforms to automatically flag hateful content to maintain a safer and more positive online environment.
- Content Moderation Systems: Online forums, comment sections, and discussion boards often rely on content moderation to ensure that inappropriate or harmful content is not published. A robust automated model can assist moderators by automatically identifying hate speech. Hence it can reduce the manual effort required for content moderation.
- News Websites: Hate speech can often appear in user comments sections on news websites. This leads to toxic discussions and reduces the quality of the content. By employing a hate speech classification model, news websites can filter out hateful comments to foster a more constructive and respectful online community.
- Chat Applications: Chat applications, both for personal and professional use, can benefit from hate speech detection models to prevent the spread of harmful language and maintain a positive communication environment.
- Online Gaming Communities: Online gaming communities frequently encounter instances of hate speech and toxic behavior among players. Integrating a hate speech detection model into gaming platforms can help identify and address such behavior and promote a more inclusive and enjoyable gaming experience for all users.
- Educational Platforms: Online educational platforms may also face challenges with hate speech among students or participants. By utilizing a hate speech detection model, these platforms can ensure that discussions remain respectful and conducive to learning. Thus, it can help to foster a supportive educational environment.

These are some of the most common domains to which a hate speech classification model can be applied. There are many other grounds too.

1.8 Organization of the Thesis

Chapter 1 is the introduction. Chapter 2 contains the literature review conducted for this research. Chapter 3 discusses some theoretical considerations that are required for this

study. Chapter 4 provides an elaborate discussion of the methodology adopted in this research. Chapter 5 contains the implementation details and a discussion of the result. Chapter 6 discusses the societal, health, environmental, safety, ethical legal, and cultural issues regarding this research. Chapter 7 addresses the complex engineering problems and activities related to the hate speech classification system. Chapter 8 concludes the book.

Chapter 2

Literature Review

2.1 Introduction

For a long time, scientists have worked on various keyword-based, rule-based, and standard machine learning algorithm-based approaches. However, their inability to detect context properly would eventually lead them to use neural network-based approaches. CNN, RNN, LSTM, Bi-LSTM, etc are some of the hot topics in this genre. Since the arrival of transformer-based architectures and various large language models, a visible paradigm shift has occurred in the hate speech classification domain. Most of the recent studies have shifted their attention to this field. Adversarial attacks and contrastive learning are relatively new concepts in the text classification domain.

2.2 Literature Review

[1] proposed a multi-task learning model using a monolingual transformer-based model. They used a BERT encoder. They also integrated polarity and emotion knowledge for hate speech detection. They used four different datasets consisting of Spanish tweets for training achieving an f1-score of 78% and 86% on two of them. Their model significantly outperformed single-task learning models. They stated that the correlated effect of affective knowledge and hate speech can provide a new way for natural language processing to be more efficient. [6] investigated comparatively less visible hate speech in the discussions of controversial topics such as immigration, skin color, religion, etc. They explored a semi-automatic way to find pages that discuss sensitive topics. Then they built a new framework to cluster the posts and detect the highly discussed topics that generate hate speech. Initially, they collected 11.8 thousand Facebook posts containing more than 644 thousand comments with 1.7M+ sentences. They used four

stages - discovery stage, sensitive social data Collection stage, sentiment and emotion analysis stage, and finally clustering stage. In the discovery stage, using 9 influential personas who like to talk about sensitive issues and the graph API of Facebook more than 17 thousand pages and 46 thousand liked IDs were reached. Among them, using Betweenness Centrality, the 50 most influentials were kept. In the sensitive social data collection stage, two dictionary-based filters were used to distinguish hate speech - one for swearing and the other one for potential hate speech triggers. In sentiment and emotion analysis stage, Negativity ratio, average negative score, and average compound score were computed using VADER and JAMMIN scores. Custom definitions were used to determine threshold values to filter out non-negative posts. In the clustering stage, they vectorized data using TF-IDF grouped by post. Then K-means clustering was used. Best clusters were generated when the maximum Document Frequency (DF) was reduced a bit, meaning that the resulting clusters had fewer topic-specific stop words and probably more topic-related words. Also, the worst results were produced by using the smallest maximum DF as it returned rare words that didn't represent the topic well.

[7] worked on Bangla language and proposed an encoder-decoder-based machine learning model. It considered categories - hate speech, aggressive comment, religious hatred, ethnical attack, political comment, religious comment and suicidal comment. They grouped them into three classes - hateful, non-hateful and both. Data were preprocessed using some preprocessing methods. Information was extracted from emoticons and emojis to detect the type of speech. Bangla natural language tokenization was used to split sentences into words. Features were extracted using TF-IDF vectorization. Word embedding was also used. Then, classification approaches such as CNN, Bidirectional LSTM, and GRU were applied to compare their performance. They further applied a Recurrent neural network (RNN) with an Attention Mechanism for text classification. Finally, the performances of all the classification approaches was analyzed and compared. The attention-based model achieved 78% f1-score outperforming CNN, LSTM and GRU-based models. [8] used LSTM and SVM on an Italian dataset having more than 17 thousand samples. This dataset contained three classes - no hate, weak hate and strong hate categorized into the following categories - Religion, Physical and/or mental handicap, Socio-economical status, Politics, Race, Sex and Gender issues, and Others. They used Fleiss' kappa inter-annotator agreement, morpho syntactically tagged texts, sentiment polarity, word embedding lexicons, and a 10-fold cross-validation process.

[2] used two-view SVM. Instead of combining all features into a single feature vector, each view classifier learned to classify the sentence based on only one type of feature allowing them to pick up different aspects of pattern individually. This process achieved over 80% f1-score on the Stormfront dataset and over 53% on the TRAC dataset. Outperformed top-ranked models in TRAC and came out as second best in Stormfront, after BERT. But, underperformed in Hatebase Twitter and HatEval. It noted that without

societal context, systems cannot generalize sufficiently. Varying definitions of hate speech among entities toughen the detection. They also talked about how user intention and context are important in detecting hate speech with examples such as the "The Nazi organization was great" sentence. It also discussed the shortcomings of different available datasets. They stated that the individuals are aware of the process of hate speech detection, and they actively try to evade detection. [9] used deep learning ensemble model approach with 10 different models on two Twitter classification datasets - 'abusive speech' and 'SemEval 2013 sentiment analysis dataset. They took the soft-max results from each underlying model, summed them together, and then took the average of the sum of soft-max results. The Highest one was declared as the winner. They used four batch sizes (10,25,50,100), three epochs (3,5,10), 10-fold cross-validation, a max-pooling layer, and a convolution layer with a single 3 token window and 150 filters. They achieved a 2% improvement in f1-score compared to the baseline. The greatest improvements were made with smaller batch sizes and a larger number of epochs. They discussed about how the ensemble method, having a lower variance, worked better than single models 98% of the time. The ensemble approach appears to leverage the high variance as an advantage for final classification via the simple method of averaging soft-max output. They also urged that non-reported weight initialization info causes difficulty to ensemble models as it lessens reproducibility.

[10] presented a survey on different models in hate speech detection. It reviewed the following approaches: keyword based, rule-based, classical learning based, deep learning and hybrid and compared different papers on them covering multiple languages. In their study, they found out that the keyboard-based approach is the main approach for explicit emotion recognition. Rule based methods can detect implicit emotions but they are affected by the quality of the text i.e. grammatical mistakes, informal style, etc. But, if any emotion is expressed but no listed keyword is present, this method fails. Classical learning-based methods need efficient features but human engineered features do not cover all the cases of how emotions are expressed. Deep learning methods don't need feature extraction but they need a large quantity of training data. Hybrid approaches can improve the results but they can also inherit the limitations. Utilizing word embeddings and deep neural networks enhances the performance. They suggested that the compression-based approach and constraint Optimization approach should be further investigated. Also, combining features extracted by deep learning models and word embeddings, as well as handcrafted features, has a promising research avenue. [11] provided a systematic overview over researches conducted in this field. They found out that the majority of the research papers in this field use databases of 1,000 to 10,000 instances. The best results were achieved when deep learning was used. They concluded that the evolution of social phenomena and language makes it difficult to track all insults. The use of sarcasm makes it even harder. [12] provided a short, comprehensive and structured overview of automatic hate speech detection and outlines the existing

approaches systematically. They found out that hate speech displays a high degree of negative polarity. Using dependency relationships significantly improves performance. Most of the datasets are imbalanced in the ratio of hate speech and non-hate speech. Skewed datasets lack the effectiveness.

[13] proposed a multi-model hate speech detection system that classified six classes - happiness, sadness, anger, fear, surprise, and disgust. It worked on 2100 sentences containing more than 65 thousand words and gained an accuracy of 75%. [14] tried to detect hate speech in the Indonesian language using an artificial neural network method optimized with a backpropagation algorithm. They achieved an accuracy of 89% over 1.2k tweets. [15] investigated the application of deep learning in this regard. It worked on a dataset of 16k tweets having three classes - racist, sexist and neither. They performed experiment and comparison of results produced by different methods such as SVM, LSTM, CNN, Random Forest, Logistic Regression etc. For neural networks, word embeddings were initialized with either random embeddings or GLoVe embeddings. The combination of LSTM + Random Embedding + GBDT achieved the highest f1-score of 93%. They found out that deep neural network models worked better than non-DNN models. [16] undertook a deep learning based fusion approach. They used ELMo, BERT and CNN. Four types of fusing method was taken - voting, meaning, maximizing and production. They used a dataset of 13k tweets having two classes - hate and non-hate. They achieved a f1-score of 71%. They concluded that whether by designing several different classifiers or using the same type of classifier with different parameters, the fusion of different classifier results can improve the classification accuracy and f1-score.

[17] provided a dataset in Bangla of 30 thousand sentences of seven categories of Facebook comments - sports, entertainment, crime, religion, politics, celebrity, TikTok, and meme. 50 annotators annotated 3 times according to the setup rules. Used three-word embedding models - Word2Vec, FastText and BengFast. SVM, LSTM and Bi-LSTM were used as classifier models. SVM outperformed all with an accuracy of 87.5% and f1-score of 91.1%. However, the provided dataset is heavily biased towards not-hate speech. There are lots of misspellings, grammatical errors, cryptic meme language, emoji, etc in the dataset which makes it vastly different from the existing corpora created from newspapers and blog articles. They mentioned that there is no proper word-embedding for the Bengali language used in social media. They found out an interesting fact - average hate speech tends to be shorter than average not-hate speech. They concluded that standard deep learning models are not sufficient for extracting the necessary information from an unbalanced dataset to predict the minority class with reasonable accuracy. BERT, XLM, RoBERTa can be of great use in this regard.

[18] provided a framework for deep CNN-based model development, reducing overhead for manual feature extraction. For baseline models - Logistic Regression, Naive Bayes, Random Forest, Support Vector Machine, Decision Tree, Gradient Boosting and K-

nearest neighbour were used. For the deep learning model, the DCNN, LSTM and C-LSTM model were used. They used GLoVe embedding. They worked on an imbalanced dataset of over 31 thousand tweets. Among the proposed models, DCNN with k-fold achieved the best result with f1-score of 92%. In their experiment, the perceptron model worked with one-hot encoded input and hence did not preserve the semantics of the tweets, which lead to a high rate of misclassification. They talked about the limitations of TF-IDF. [19] introduced a hybrid approach by combining a machine learning approach with a novel rule-based approach to perform sentiment analysis on user reviews by classifying their polarities. They started with a machine learning approach and then integrated the intermediate outcome with a newly generated rule-based approach to produce the final result. They applied the quantification of the polarity method. They also developed a novel dataset manually containing 1600 reviews of two categories - good and bad. They achieved over 95% accuracy.

[20] prepared a comprehensive review of 150 deep learning-based models for text classification and their performance review on 16 different benchmarks. They also presented a summary of 40 different datasets. Analyzing all those papers, they concluded that (the feature extractor + classifier) approach has several limitations. For example, reliance on hand-crafted features requires tedious feature engineering and analysis to obtain good performance. In addition, the strong dependence on domain knowledge for designing features makes the method difficult to generalize to new tasks. Finally, these models cannot take full advantage of large amounts of training data, because the features (or feature templates) are pre-defined. They found out that vanilla RNN underperforms feed-forward neural networks. One of the computational bottlenecks suffered by RNNs is the sequential processing of text. Although CNNs are less sequential than RNNs, the computational cost to capture relationships between words in a sentence also grows with the increasing length of the sentence. By offering more parallelism, transformer architecture overcomes those hurdles. Their analysis revealed that feed-forward neural networks view text as a bag of words. RNNs can capture word orders. CNNs are good at recognizing patterns such as key phrases. Attention mechanisms are effective to identify correlated words in text. GNNs can be a good choice if graph structures of natural language (e.g., parse trees) are useful for the target task. Lack of interpretability is the biggest problem in deep neural networks.

[21] proposed a deeply shared private multi-task learning (SP-MTL) framework. Used word2vec embedding and skip-gram model. They Prepared four deep neural networks - CNN, LSTM, CNN + GRU, and CNN(modified) + GRU. They experimented on 5 separate datasets and found a result outperforming state-of-the-models by a range of 10%-27% in the f1-score. They noted that removing stop words has little or no impact on classification performance. Better not remove them as doing so might result in losing useful info. They stated the shortcoming of a single dataset is its small training samples

and is related to a particular domain. RNN is very suitable for sequence learning, but as it suffers from vanishing gradient and exploding gradient it does not perform well for the long-range dependency problem. [22] created four CNN models and compared them with a baseline logistic regression model. These four CNN models utilized random vectors, word2vec (skip-gram), character 4-grams, and the combination of word2vec and character 4-grams. All the CNN models outperformed the baseline in the f1-score. [23] used BERT transformer model on two different sub-tasks. Subtask 'a' was to detect hate-offensive and non-hate-offensive. Subtask 'b' was to detect hate speech, offensive, and profane examples. They workd with multilingual datasets containing texts in English, German and Hindi. In their experiment in three languages, BERT showed an increase of 5-7% in classification metrics compared to ELMO and SVM models. In English language, BERT achieved the highest f1-score of more than 88%. Multilingual BERT did not work well with German and Hindi. [24] applied deep context-aware embedding in two main modules - deep hybrid contextual word representation and BiLSTM with an attention mechanism. They concatenated word2vec, ELMo, SenticNet and BiLSTM with attention mechanism. They used three multiclass datasets with a total of over 60 thousand samples and gained an f1-score of 85.5,92.3 and 73.6 in them respectively. They handled issues of polysemy, semantics, sentiment, OOV words, and syntax within each tweet.

[25] applied logistic regression with l1 regularization and with l2 regularization in a dataset of 24 thousand tweets. Their dataset was further used by many other researchers. They also applied SVM, decision trees, random forest, linear SVM, and aive bayes. Their best performing model had an overall f1 score of 90%. In their experiment, racist tweets are more likely to be classified as hate speech but sexist tweets are generally classified as offensive. Tweets without explicit hate keywords were more difficult to classify. Social biases and mislabeling caused errors. They said that Lexical methods are effective ways to identify potentially offensive terms but are inaccurate at identifying hate speech. [26] created a dataset of 35 thousand tweets. This dataset was highly imbalanced with 5 times more negative samples than the positive ones. They ignored the context and analyzed the text alone. They divided the samples into the following sub-categories - the very worst, threats, hate speech, directed harassment, potentially offensive, and non-harrassing. The last two were labeled as "non-harrassing" in the final corpus. Each tweet was labeled by two coders. If they agreed on labeling a sample the same, it was added to the final corpus. If they disagreed, a third coder was brought in to break the tie. Inter-rate agreement measured by Cohen's Kappa was 0.84 among the initial coders. [27] created another dataset of 16.9 thousand tweets. Their dataset had three classes - racist, sexist, and neither. They used Twitter API. Inter-annotator agreement was 0.84. They removed all stop words except 'not'. They also applied character n-grams that outperformed their word n-grams by at least 5 f1-score points.

[28] developed HateNet and t-HateNet. HateNet was a deep neural network architecture which was capable of creating task-specific word and sentence embeddings, without the need for expensive hand-crafted features. t-HateNet connected the HateNet architecture with transfer learning methods that allow leveraging several smaller, unrelated datasets to construct a general-purpose hate speech embedding. They also prepared a Map of Hate which is an interpretable 2D visualization of hateful content, capable of separating different types of hateful content and explaining what makes text hateful. Their word embedding was a combination of GLoVe and ELMo pre-trained embeddings. They also applied a bi-LSTM mechanism. The entire processing pipeline is shared among all learning tasks apart from the final classification units. This pipeline included a pre-processing unit, a word embedding unit, a bi-LSTM layer, a max-pooling layer, and a hate classification unit. They used the datasets from [27] and [25]. Their model outperformed the baseline models from those two papers. t-HateNet also outperformed HateNet in both datasets. t-HateNet confuses the Hate class with Offensive in 41% of the cases. It was particularly difficult even for a human to differentiate between purposely hateful language (with a particular target in mind) and generally offensive texts (without a target). They concluded that the advantage of jointly leveraging multiple data sets emerges when only limited amounts of labeled data are available. Prediction on Waseem dataset [27] improved with the help of Davidson dataset [25]. However, the inverse didn't work. Interestingly, their map of hate makes false positive, sampling bias easily detectable.

[29] used BERT to classify hate speech and also proposed four fine tuning techniques for so. They inserted nonlinear layer, bi-LSTM layer and CNN layer to fine tune BERT. They also used datasets from [27] and [25]. (BERTbase + CNN) i.e. 'Insert CNN layer' fine-tuning method outperforms all with f1 scores of 88 and 92 on both datasets, respectively. They claimed that this BERT-based classifier can discriminate tweets in which neither implicit hatred content exists. They said that the many errors were caused by the biases from data collection and rules of annotation and not the classifier itself. [30] investigated a transformer-based method and tested it on a publicly available multi-class hate speech corpus. In their work, the DistilBERT transformer method was compared against attention-based recurrent neural networks and other transformer baselines for hate speech detection in Twitter documents. They also used the dataset from [25]. Compared with BERT, XLNet, RoBERTa and attention-based LSTM, DistilBERT outperformed all in f1-score and accuracy. Comparative results based on five different metrics from this work show that the transformer models consistently outperformed the LSTM with attention. Transfer learning is particularly beneficial to imbalanced datasets with few instances of hate speech. The performance of the LSTM drops as sequence length increases beyond thirty words [31].

[32] investigated Bi-LSTM with regard to explainability using IG and error analysis

for the T5 model. They gained insights into the autoregressive data augmentation technique and the argument for more credible data annotation. They used Bi-LSTM, CNN, RoBERTa (RoBERTa-base), and T5 large language models. For BiLSTM they used glove embedding of dim 100, dropout, 2 Bi-LSTM layer of size 20, and fully conn layer. It had a total of 1,317,721 parameters. For CNN, 3 convolution layers with 100 filters in each having filter size n*100, word embedding of 100 dimensions were used. This one had 1,386,201 parameters. They used two data augmentation techniques. The first one was a simple token-level deletion of the start and end tokens for each sample. This was less effective and dropped. The second one was autoregressive text generation using the model checkpoint. They used the HASOC-2021 dataset with around 5 thousand samples and the OLID dataset [33] with 14,200 annotated English tweets. In their experiment, transformer-based models outperform both the Bi-LSTM and CNN models. The 'T5 model with augmented data' method showed improved scores in both tasks of HASOC-2021 when compared with the plain T5. The T5 model is more stable with predictions when fine-tuned with the target. labels of numbers, explicitly type-cast as a string. Otherwise, some predictions during fine-tuning can be an empty string or expressions from the training set, especially in the early epochs of the training. However, due to the imbalanced dataset, model performed worse on detecting 'not hate'.

[34] proposed DeepHateExplainer, an explainable approach for hate speech detection from the under-resourced Bengali language. They preprocessed Bengali texts and fed them to neural ensemble method of transformer-based neural architectures. They performed sensitivity analysis and layer-wise relevance propagation to get the explanations for hate speech detection. They used two types of baseline models. For machine learning baselines, logistic regression, SVM, KNN, naive bayes, random forest, and GBT were used. For DNN based baselines, CNN, Bi-LSTM, and Conv-LSTM were used. They compared monolingual Bangla BERT-base, mBERT (cased and uncased), and XLM-RoBERTa with the baselines. An extended Bangla Hate Speech Dataset with 5 thousand extra instances were used here. XLM-RoBERTa outperformed others with an f1 score of 87%. An MCC value of more than 0.77 signifies that predictions are strongly correlated with ground truths and BERT variants are more effective compared to ML or DNN baseline models. A serious drawback of many existing approaches is that the outputs can neither be traced back to the inputs, nor it is clear why outputs are transformed in a certain way. They said that the ensemble prediction is effective at minimizing confusion. They suspected that the model might have overfitted due to limited labeled data.

[35] fine-tuned multi-lingual transformer models that are trained on large corpus. They inserted a special start of sequence (SOS) token at the start and a special end of sequence (EOS) token at the end. They only considered the first token for the classification task. Three models were produced from two model classes - multilingual BERT and XLM-RoBERTa. They utilized 6 different datasets of different types such as sentiment

analysis, news classification, etc. Some of them contained texts in Bangla, English and romanized Bangla. In their experiment XLM-RoBERTa-large models outperformed others in almost all cases. They found out that the transformer models performed better on Bangla words written in English texts compared with pure Bangla data. XLM-RoBERTa performed better due to the availability of a larger vocabulary for Bangla in the model and better generalization ability. Subword embedding performed better on noisy user-generated texts. [36] worked on Spanish texts to research which individual features are the most effective for hate speech detection and how. Their pipeline included a dataset selector, a text cleaner, a dataset splitter, a feature generator, a feature selector, a model resolver, a hyperparameter selector and a classification report module. They used 4 differnet datasets with over 24 thousand texts in multiple languages. BETO outperformed all other models in their experiment. They said that the word and sentence embeddings from the pre-trained models have the drawback that they do not take into account polysemy. So words have a unique representation regardless of their context.

[37] used emoji2vec to embed emojis. Then they concatenated clean texts, hashtag embeddings, and emoji embeds, which were later fed to XLM-RoBERTa. Finally, the samples passed through a 2-layer mlp. Instead of early stopping, they considered the change in validation performance at the end of each training iteration. If the validation performance went down across an iteration, they reused the previous model weights and scaled-down learning rate. They also used Google Perspective API features. They used datasets of three languages - English, German, and Hindi. They claimed that the usage of hashtags as well as emojis add valuable information to the classification head. [38] used a 5-fold ensemble training method using the RoBERTA model and additional fine-tuning. They divided the dataset into 5 parts to the ensemble and computed the ensemble by averaging predictions. They used HASOC data with more than 6 thousand tweets. They found no effect of emojis and hashtags scores. They used BERT and mBERT along with two sequence classification heads - one for Hate/offensive speech and another for Emotion recognition. They also designed a multitask dataloader with BERT/mBERT and used hate speech logits and emotion logits to generate two separate outputs. They used the Davidson dataset [25] and the GoEmotion dataset. Multi-task joint learning models outperformed single-task models and exhibited fewer false positive errors, indicating that emotional knowledge helped to improve the classification.

[39] addressed a multi-task joint learning approach that combined external emotional features extracted from another corpus in dealing with the imbalance and scarcity of labeled datasets. Shared representations between several related tasks regarding the auxiliary dataset to solve the problem of scarcity in labeled data. [40] performed ensemble learning of BERT models and other DL algo models. They used BERT, BiLSTM, BiGRU, and BiLSTM. For ensemble calculation, they adopted average prediction and weighted average prediction techniques. They also concatenated FastText character

n-gram-based (300dim) and GloVe word-based (100dim) embeddings and randomly initialized representations for unseen words. They used the Jigsaw dataset with over 1.5 million entries. The ensemble of all 4 models achieved the highest ROC-AUC score. [41] tried to locate toxic text spans in a given text and utilized additional post-processing steps to refine the boundaries. They labeled character offsets between consecutive toxic tokens as toxic and assigned a toxic label to words that have at least one token labeled as toxic. They used a combination of BERTbase and spaCy NER model. Then stored mapping of each token to its relative character offsets in the original string. all intermittent tokens between two consecutive toxic tokes were also labeled as toxic. If at least one token of a word was predicted toxic by the model, all constituent tokens were assigned a toxic label. They used a Civil comment dataset containing more than 10 thousand samples. Their experiment showed that data augmentation and ensemble modeling strategies did not outperform the BERT-standalone model. The models had an inductive bias to predict shorter toxic spans. Their findings suggested that different data sources and annotation guidelines can introduce noise that hurts the performance of models. Dataset size did not appear to be the limiting factor affecting the performance of BERT in this task. They also claimed that BERT lacked nuance in understanding the use of offensive words in neutral contexts and encountered boundary detection issues when faced with noisy ground truth annotations.

[42] performed a large-scale cross-dataset comparison. They discussed how some datasets are more generalizable than others and highlighted deficiencies in generalization across datasets. They also investigated how combining hate speech detection datasets can contribute to improving results. They unified the datasets by standardizing their format and combining the available content into two settings - binary hate speech classification and a multiclass classification task including the target group. They considered 7 classes - Racism, Sexism, Disability, Sexual orientation, Religion, Other, and Not-Hate. Eventually, they considered all ”hate” subclasses as one. Four models were used in the analysis - BERT-base, RoBERTa-base, BERTweet, and TimeLMs-21. For each language model, they trained on each dataset training set independently, and in the combination of all dataset-specific training sets. Results were averaged across all specific test sets and an independent test set. An equally sized training set was extracted from all available data while enforcing a balanced distribution between hate and not-hate tweets. They used a total of 13 datasets containing more than 80 thousand tweets. They found that the relatively limited scope of the individual datasets hindered the potential capabilities of models. Hence, the amount of available training data remained an important factor even for the transformer models. Even when information about a class is available in the training data, language models may fail to distinguish and utilize it. Limitations arise when the data collection is done by utilizing specific keywords. Some tweets have hateful undertones, they may not be necessarily hate speech without considering them in their broader context.

[5] studied five recent model architectures and presented a simple evasion method that completely broke the models. A comparative analysis of different models and datasets was also presented. They applied multiple adversary models i.e. evasion techniques such as word changes, inserting typos, leetspeak, word-boundary changes, inserting whitespaces, removing whitespaces, word appending, etc. Then they adopted several mitigation techniques such as spell checkers, whitespace addition/removal, etc. They used the Wikipedia Detox Proj dataset, the Davidson dataset [25], the Waseem [27], and the Zhang dataset. They noticed three main deficiencies in the models - lack of effective transferability across datasets, the conflation of hate speech and offensive ordinary speech, and susceptibility to simple text modification attacks. They found out that the word-based models were the most affected by tokenization changes and character-based models by word appending. Their experiment suggested that for successful hate speech detection, model architecture is less important than the type of data and labeling criteria. Features indicative of hate speech are not consistent across different datasets. They concluded that hate speech detection is highly context-dependent and transfers poorly across datasets. [43] developed TOXIGEN, a new large-scale and machine-generated dataset of 274 thousand toxic and benign statements about 13 minority groups. They designed a demonstration-based prompting framework along with an adversarial 'classifier in the loop' decoding method, ALICE. ALICE elaborated as Adversarial Language Imitation with Constrained Exemplars. The example statements were passed to a large language model, encouraging it to produce a similar, but distinct, statement. They manually collected some demonstrations, passed them to a large language model, combed through many responses, and added the best examples to a growing set. This way they adopted 20-50 demonstration sentences per group.

[44] analyzed the effect of varying pre-processing steps and the format for making data publicly available in making the datasets highly varying which makes an objective comparison between studies difficult and unfair. Then they compared the attributes of existing datasets for hate speech detection. They outlined their limitations and recommended approaches for future research. [45] worked with implicit hate speeches and proposed a method, ImpCon. This method utilizes contrastive learning to pull an implication and its corresponding posts close in representation space.

[46] proposed a dual contrastive learning approach and label-aware data augmentation technique. While producing the adversarial samples, they encoded the label of the original to that sample. The target sample with a "positive" class served as the anchor, and there was a positive sample having the same class label and a negative sample having a different class label. The dual contrastive loss was used to optimize both the parameters of the model and the representations of sentences in embedding space. [47] jointly optimized the self-supervised and the supervised contrastive learning loss for capturing span-level information beyond the token-level emotional semantics used in

existing models, particularly detecting speech containing abusive and insulting words. Proposed a dual contrastive learning framework for hate speech detection, particularly addressing the detection of hate speech containing insulting words by mining context information of data beyond the token level emotional semantics. They used the SemEval-2019 Task-5 (SE) and the Davidson Dataset [25] dataset. Their baseline models included SVM, LSTM, GRU, Bi-LSTM, CNN-GRU, BERT, and SKS. They claimed that the subtle differences in data distributions can significantly affect the detection performance. They saw that the subtle differences in data distributions can significantly affect the detection performance.

[48] generated adversarial examples by perturbing the word embedding matrix of the model during fine-tuning. Then they performed contrastive learning on clean and adversarial examples to teach the model to learn noise invariant representations. Instead of applying perturbation to word embeddings, they applied it to the word embedding matrix of Transformer encoders. They leveraged contrastive learning as an additional regularizer during the fine-tuning process. They used three datasets and the RoBERTa model model along with a contrastive loss. [49] performed causal contrastive analysis. They proposed a 2-layer pipeline to find out the causality of a sample i.e. the most important word in the sentence without which the sentence may have a different label. They masked each word in the sentence, generated some new sentences, analyzed their predicted labels, and found out which word disturbed the classifier the most. This way they found out the causing factor of the sentence. Then, they made causally positive and causally negative pairs to perform contrastive learning. They performed this method for multiple tasks along with sentence classification.

2.3 Discussion

Table 2.1 provides a summary of the literature review. It can be seen here that the research papers referenced here contributed to separate parts of the domain. Many of them used transformer-based architectures. Some performed cross-dataset analysis. Some did ensemble modeling of different types of models. A few experimented with constrastive loss.

On the other hand, this study encompassed a wide range of experiments and covered a broad field of study. It performed a cross-dataset analysis on six datasets using three transformer-based model architectures. Then, it used the best-performing models from the previous experiment to form two kinds of ensemble models - hard-voting ensemble and weighted ensemble. The result of the process was investigated. Then adversarial samples were generated from two of the datasets. The augmented datasets were used to train some new models. Finally, a contrastive loss was introduced and experimented with. Improvement and other changes of model performances were analyzed.

Table 2.1: A brief overview of the literature review

Paper	Approach	Comment
Das et al. [7]	CNN, LSTM, GRU, Attention	Proposed encoder–decoder-based ML model
Vigna et al. [8]	LSTM, SVM	over 17 thousand Italian data
Macaveney et al. [2]	SVM	Used Stormfront, TRAC, HatEval, and HateBaseTwitter datasets
Zimmerman et al. [9]	DNN ensemble	Used 2 Twitter datasets
Chuang et al. [13]	Keyword-based	Six types of emotions
Setyadi et al. [14]	ANN-based	Used Indonesian data
Badjatiya et al. [15]	SVM, LSTM, CNN, RF, LR, etc	Compares different models and their combination
Zhou et al. [16]	ELMo, CNN, BERT, Ensemble	Performs ensemble operation
Roy et al. [18]	CNN	Provided a good framework
Gamback et al. [22]	CNN, word2vec, 4-gram	Combined the models
Dowlagar et al. [23]	BERT, SVM, ELMo	BERT defeated others
Naseem et al. [24]	word2vec, ELMo, BiLSTM	Used 3 datasets
Yuan et al. [28]	BiLSTM	Provided HateNet, t-HatenNet, and Map of hate
Mozafari et al. [29]	BERT	Proposed 4 fine-tuning techniques for BERT
Adoma et al. [50]	BERT, RoBERTa, DistilBERT, XLNet	RoBERTa outperformed others in emotion recognition
Mutanga et al. [30]	DistilBERT	DistilBERT outperformed BERT, XLNet, RoBERTa, and attention-based LSTM
Sabry et al. [32]	BiLSTM, CNN, RoBERTa, T5	Utilized 2 data augmentation techniques on HASOC2020 and OLID datasets
Alam et al. [35]	XLM-RoBERTa, mBERT	6 different datasets of different tasks
Garcia et al. [36]	BERT, RoBERTa, BETO	BETO outperforms all
Alonso et al. [38]	5-fold ensemble	Used HASOC and 2020 Offenseval data
Mnassri et al. [39]	BERT, mBERT	Used two classification heads
Mazari et al. [40]	BERT, BiLSTM, BiGRU, BiLSTM, BiGRU, Ensemble	Ensemble brought the best result
Antypass et al. [42]	BERT, RoBERTa, BERTweet, TimeLMs-21	Cross-dataset analysis on 13 datasets
Chen et al. [46]	BERT	Dual contrastive learning and label-aware data augmentation
Lu et al. [47]	SVM, LSTM, CNN, BERT	Used dual contrastive learning

Chapter 3

Theoretical Considerations

3.1 Introduction

Hate speech classification is a subfield of sentence classification task which is a field within the Natural Language Processing domain. Hence, over the years, many sentence classification methods have been used to detect and classify hate speech. These approaches include rule-based approaches, keyword-based approaches, emotion modeling approaches, classical machine learning-based approaches, neural network-based approaches, etc. The use of transformer architectures in this field started fairly recently. This chapter provides a brief discussion in this regard.

3.2 Rule-based Approach

Among various methods for sentiment analysis, the rule-based approach offers a structured way to extract sentiments from text using predefined rules and lexicons. It relies on a predefined set of rules and lexical resources to determine sentiment. Key components include sentiment lexicons or dictionaries, polarity scoring, rules, and heuristics, and handling ambiguity. Sentiment lexicons are lists of words and phrases associated with different sentiments. Each word or phrase is assigned a sentiment polarity, such as positive, negative, or neutral. Sentiment analysis systems compute sentiment scores by summing or averaging the polarity scores of individual words present in the text. Grammatical rules, syntactic patterns, and heuristics are used to enhance accuracy. For instance, negation can reverse sentiment polarity. Fig 3.1 displays the general steps for this technique.

Rule-based systems are transparent due to their reliance on explicit rules and lexicons. Users can adapt sentiment lexicons and rules to specific domains or applications.

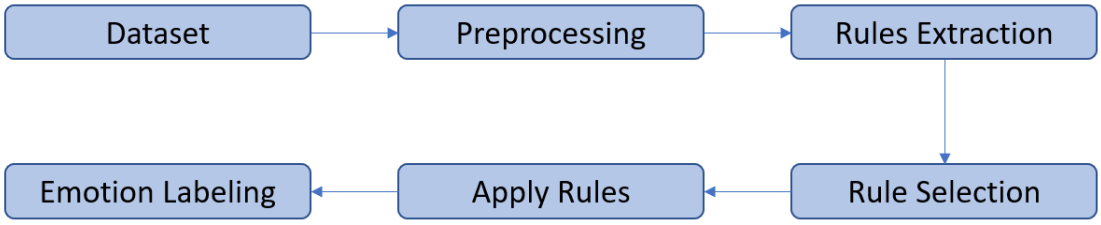


Figure 3.1: General steps in a rule-based approach

However, Rule-based approaches struggle with nuanced expressions and sarcasm. It's generally annoyingly blind towards the context of the sentences. As lexicons grow, systems may become complex and need frequent updates.

3.3 Keyword-based Approach

The keyword-based approach harnesses a predetermined set of keywords, also known as sentiment lexicons or dictionaries. These lexicons encompass a collection of words and phrases that are intrinsically associated with particular sentiments. For instance, words like "happy," "excellent," and "delightful" might be associated with positivity, while "sad," "disappointing," and "frustrating" could be linked to negativity. The keyword-based method scans the input text to identify instances of keywords present in the sentiment lexicon. By counting the number of positive and negative keywords, the approach computes a sentiment score for the given text. Based on the sentiment score and predefined thresholds, the text is classified into multiple classes. Fig 3.2 displays the general steps for this technique.

However, just like the rule-based approach, the keyword-based method also struggles to capture nuanced sentiments and detect sarcasm or subtleties. Words with double meaning, polysemy, or others are hardly detected by this approach.

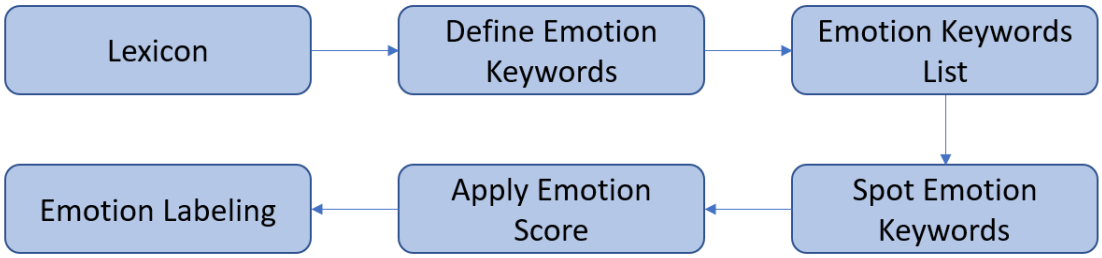


Figure 3.2: General steps in a keywords-based approach

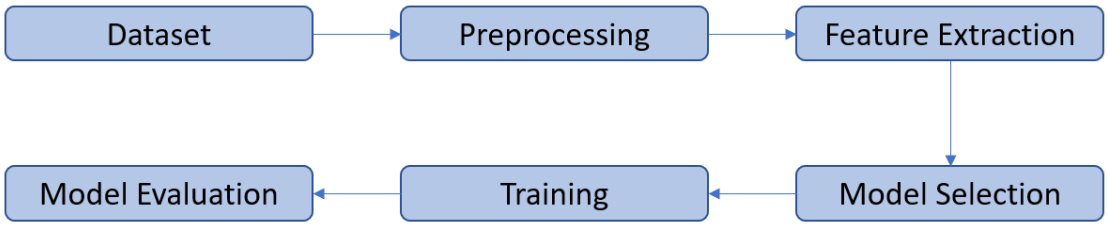


Figure 3.3: Classical machine learning algorithm-based approach

3.4 Emotion modeling approaches

Dominant emotion modeling approaches are - categorical, dimensional, and componential.

- Categorical approach: This approach is based on the idea that there exists a small number of emotions that are basic and universally recognized. The most commonly used model in emotion recognition research is that of Paul Ekman, which involves six basic emotions: happiness, sadness, anger, fear, surprise, and disgust.
- Dimensional approach: This approach is based on the idea that emotional states are not independent but are systematically related to each other. This approach covers emotion variability in three dimensions - valence, arousal, and power. Valence refers to how positive or negative an emotion is. Arousal refers to how excited or apathetic an emotion is. Power refers to the degree of strength i.e. intensity of the expression.
- Appraisal-based approach: This approach can be considered as an extension of the dimensional approach. It uses componential models of emotion based on appraisal theory. This theory states that emotion is extracted via an evaluation of events and the result is based on a person's experience, goals, and opportunities for action.

3.5 Classical Learning Based Approach

Classical learning-based models for sentence classification involve using traditional machine learning algorithms to classify sentences into different predefined categories or classes. These models typically rely on manually engineered features and well-established algorithms. Support Vector Machine, Logistic Regression, Decision Tree, Random Forest, and K-nearest neighbors are some of the most widely used machine learning algorithms in this field. Most of these are generally supervised approaches. A

lot of research papers have worked with these methods. For a long time, this approach dominated this domain. Fig 3.3 displays the general steps for this technique.

3.6 Neural Network Based Models

Scientists were aware of the limitations of the above-mentioned models in failing to detect the context of the sentences properly. So, eventually, they resorted to neural network-based models. CNN, RNN, LSTM, BiLSTM, etc were developed. These neural networks performed way better than the previous old-styled models. They revolutionized the field of sentence classification. Convolutional Neural Network i.e. CNN applies 1D convolution operation on text embeddings to capture the context of the words. It typically performs better than RNN (Recurrent Neural Network). The BiLSTM method analyzes the sentence from both directions to understand the context of the words.

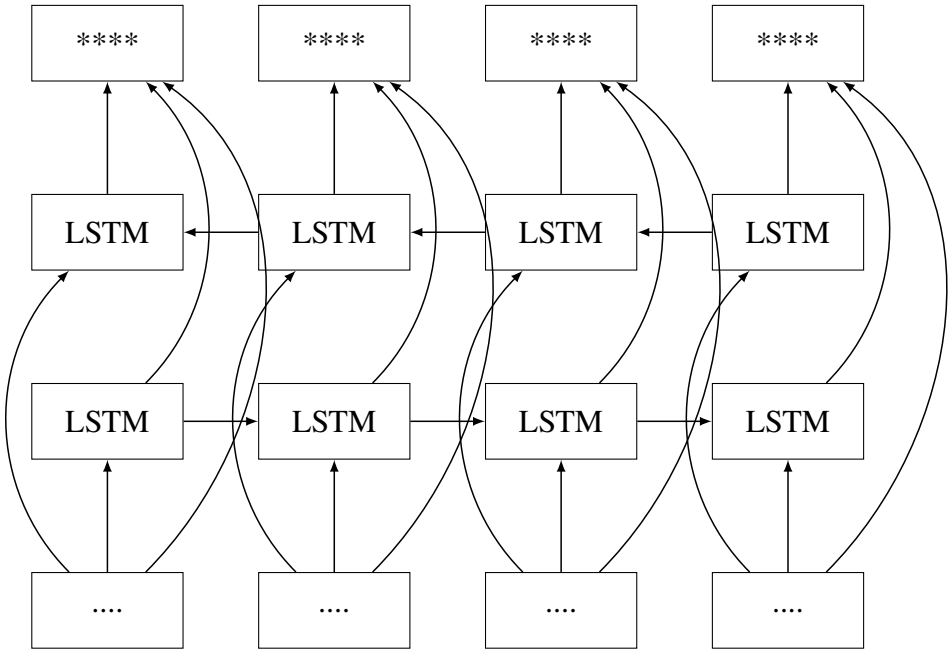


Figure 3.4: ELMo biLSTM unit

ELMo i.e. Embeddings from Language Models, proposed by [51], is a variant of BiLSTM. ELMo embeddings are based on deep bidirectional language models (biLMs) that are trained on large amounts of text data, approximately 1 billion words. These models learn to predict the next word in a sequence given the previous words, and they do so in both forward and backward directions. Many BiLSTM units work together to anlyze the whole sequence. Fig 3.4 shows a simplistic view of the internal biLSTM units of ELMo. The resulting biLMs can capture complex language patterns and dependencies.

This can be used to generate contextualized embeddings for individual words. To generate an ELMo embedding for a given word in a sentence, the biLM is first run on the sentence to obtain a sequence of hidden states i.e. layer representations for each position in the sentence. These hidden states are then combined using a weighted sum to create a context-dependent representation of the word. The weights for the sum are learned during training, and they allow the model to adaptively combine information from different parts of the sentence depending on the context. ELMo embeddings have been shown to outperform traditional word embeddings in tasks like hate speech classification.

However, the biggest weakness of BiLSTM or ELMo is that, if the sequence gets too longg, context relating to very far in the past gets lost in the calculation. In other words, these methods can capture the context up to a certain degree. If the context length is too long, their output gets faulty too.

3.7 Transformer-based Architectures

In 2017, Vaswani et al. [52] proposed a transformer architecture that leveraged the attention mechanism that could find out the proper context of a word from a very long sequence of words. Fig 3.5 shows their proposed architecture. For a given sequence of words, this mechanism would analyze the effect of every other word on a particular word to compute its mathematical representation. It is an encoder-decoder-based architecture that utilizes self-attention and cross-attention. The encoder consists of a multi-head attention module and a feed-forward layer with layer normalization and residual connection at both ends. The decoder consists of the same modules with an extra masked multi-head attention module. The encoder computes the proper embedding space representation of the input sentence and the decoder uses that embedding to produce proper output.

The input sentence is first tokenized. Then a positional encoding is added to the input sample. This positional encoding carries the positional values and information of each word in the sentence. Then the encoded input is fed to the encoder. For each input word, the encoder initializes three random values - query (q), key (k), and value (v). Then using these three vectors, the encoder performs self-attention to compute the proper embedding space representation of the input sentence.

$$Q = qW^{\mathrm{Q}} \tag{3.1}$$

$$K = kW^{\mathrm{K}} \tag{3.2}$$

$$V = vW^{\mathrm{V}} \tag{3.3}$$

$$Attention\Big(Q, K, V\Big) = softmax\Big((QK^{\mathrm{T}})/sqrt(d_k)\Big)V \tag{3.4}$$

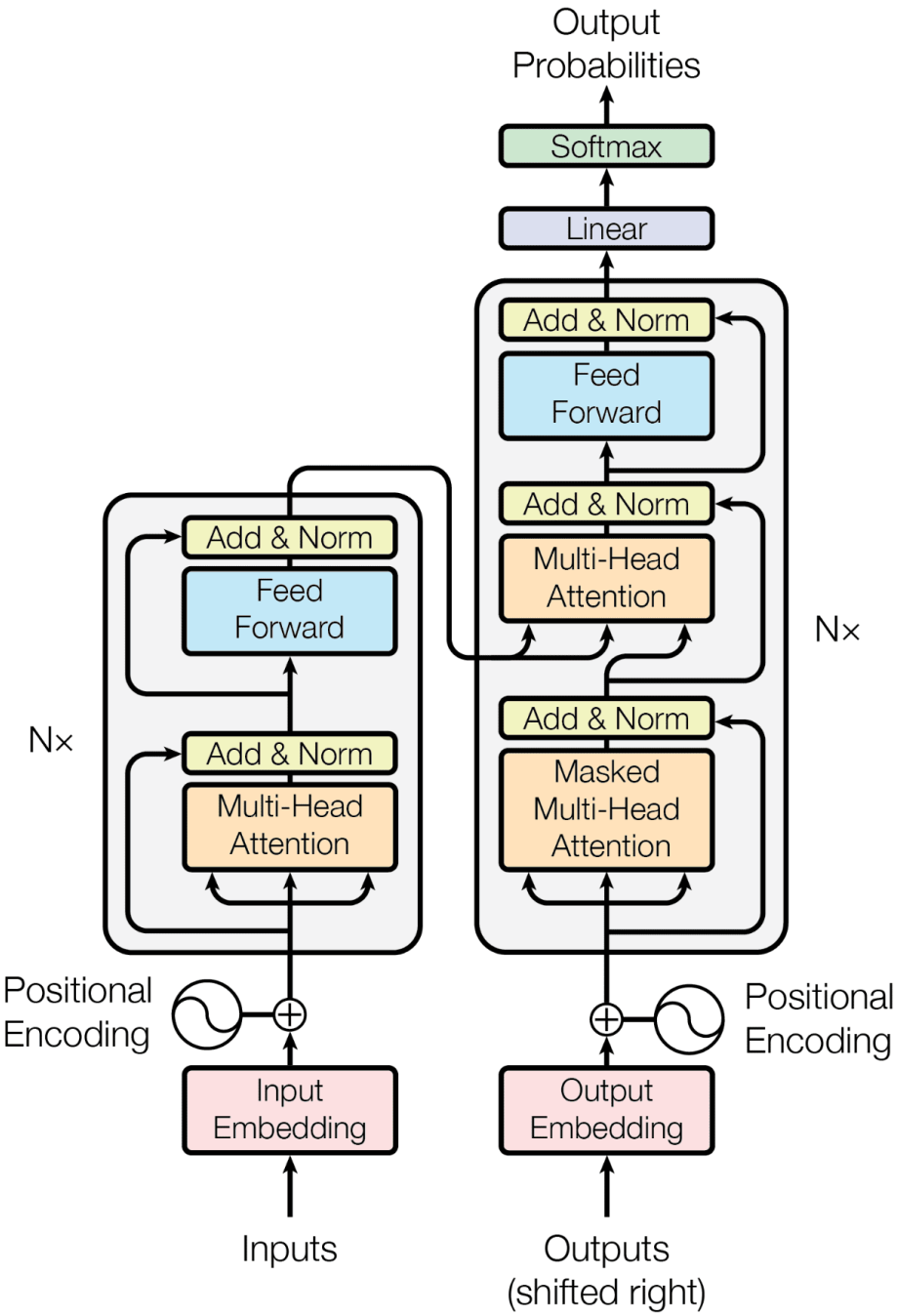


Figure 3.5: Transformer architecture

Self-attention used here can be also called scaled dot product attention. Let's consider the sentence "I live in Bangladesh". For each word in this sentence, the encoder first randomly initializes q, k, and v vectors. Then it does linear projections using (eq 3.1), (eq 3.2), and (eq 3.3) respectively to compute Q, K, and V. Then the encoder takes the query vector of the current word and performs the dot product with the key vectors of all other words. Then it scales it using the $sqrt(d_k)$ where d_k is the dimension of the query and key vectors. Then it softmaxes all the results and multiplies the value vector of all the words with the corresponding scores, which gives a probabilistic score for each word (eq 3.4). Finally, it sums them up to compute the result. The paper uses multi-head attention, which means they divide the dimension of the input (d_{model}) into h attention modules (eq 3.5). Each Q and K vector has the same dimension d_k where $d_k = d_{model}/h$ (eq 3.6). The decoder then takes this embedding representation and performs a cross-attention operation. In this case, it takes the query vector of a word from encoder outputs and then performs attention using the key and value vector of decoder inputs. Thus, it generates the output.

$$MultiHead\Big(Q, K, V\Big) = Concat\Big(head_1, head_2, ..., head_h\Big) \tag{3.5}$$

$$head_i = Attention(QW_i^{\mathrm{Q}}, KW_i^{\mathrm{K}}, VW_i^{\mathrm{V}}) \tag{3.6}$$

The greatest advantage of the transformer mechanism is that it can capture the context of a word from a very long sequence. Attention score means how much effect a particular word has over another word. So, if the softmax value for a word w_i is smaller than another word w_j when the current word is w_k, it means w_i has less correlation with w_k than that of w_j. This way, no matter how long the sequence is, the attention mechanism can capture the proper interconnection between different words. This advantage allowed this mechanism to be used for a wide range of tasks such as sentence classification, text generation, machine translation, etc. However, this process needs a huge amount of training data to bring out its true potential. It is computationally havier than deep networks too.

3.8 Large Language Models

Using the transformer mechanism, many large language models (LLM) have been developed in the last few years. These large language models are trained with a massive amount of data. They have two parts in their training process - pre-training and fine-tuning. During the pre-training stage, millions, if not billions, of sentences are fed into the model to train the parameters. Usually, this stage takes millions of continuous working hours on thousands of GPUs. Pre-training produces a generalized model that can be used as a basis for any other downstream task. Training for any specific downstream

task is called fine-tuning. Now, fine-tuning can be done either by freezing the pre-trained layer and only training an additional classifier layer or by re-training the entire network. During fine-tuning, only task-specific dataset is provided. Any downstream task can be implemented using these large language models such as sentiment analysis, text classification, text generation, machine translation, etc.

Some of the prominent large language models are BERT, GPT, ALBERT, DistilBERT, XLM-RoBERTa, T5, LLAMA, etc. The pre-trained weights of these large language models make them familiar with the nature of the language and provide a vast knowledge of vocabulary. As a result, during fine-tuning, they can correctly infer words and their context and hence, represent them in embedding space more effectively.

However, these massive models have some problems too. As these models are aggressively resource-hungry, hardware memory limitations have become a severe issue. Without a budget of millions of dollars, it is not possible to assemble such a large structure to pre-train a large language model. The attention mechanism needs a lot of data to function properly. Without a huge amount of data, attention mechanisms can even perform worse than standard machine learning-based models. This makes the attention mechanism inherently computationally expensive. As these large language models use gigabytes of data, training speed has also become a significant issue. Pre-training normally requires several days to finish even though thousands of GPUs are used. This also brings out the communication overhead. Also, the layer representations inside the architecture lack interpretability.

Chapter 4

Methodology

4.1 Introduction

This thesis was divided into four major tasks. The first task was a cross-dataset analysis on six different datasets using three different model architectures. The second one was an ensemble modeling of different models for each dataset. Thirdly, adversarial sample generation and training of different models with augmented datasets. Finally, a constrastive learning-based analysis where a new loss was introduced. Three transformer-based architectures were used. They were ALBERT-base, DistilBERT-base, and XLM-RoBERTa-base. The used datasets were the Agarwal dataset, the MHS dataset, the HASOC2020 dataset, the Vidgen dataset, the Jigsaw dataset, and the OLID dataset. This chapter will explain the entire methodology step by step.

4.2 Detailed Methodology

4.2.1 BERT

Bidirectional Encoder Representations from Transformers i.e. BERT is one of the most popular large language models. [53] proposed this encoder-only architecture back in 2018. It leverages the attention mechanism and is a fine-tuning-based approach. It uses WordPiece embeddings with a vocabulary of 30,000 tokens. The input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings. It adopts two unsupervised tasks - masked language modeling and next-sentence prediction. BERT considers a "sentence" as an arbitrary span of contiguous text, rather than an actual linguistic sentence. And, a "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. The first token of every sequence is always a special token [CLS]. This particular token can

be used for classification tasks in fine-tuning stage. Sentence pairs are packed together into a single sequence. [SEP] token separates them and segment embedding tells which sentence a token belongs to. For a given token, input representation is constructed by summing the corresponding token, segment, and position embeddings (fig-4.2).

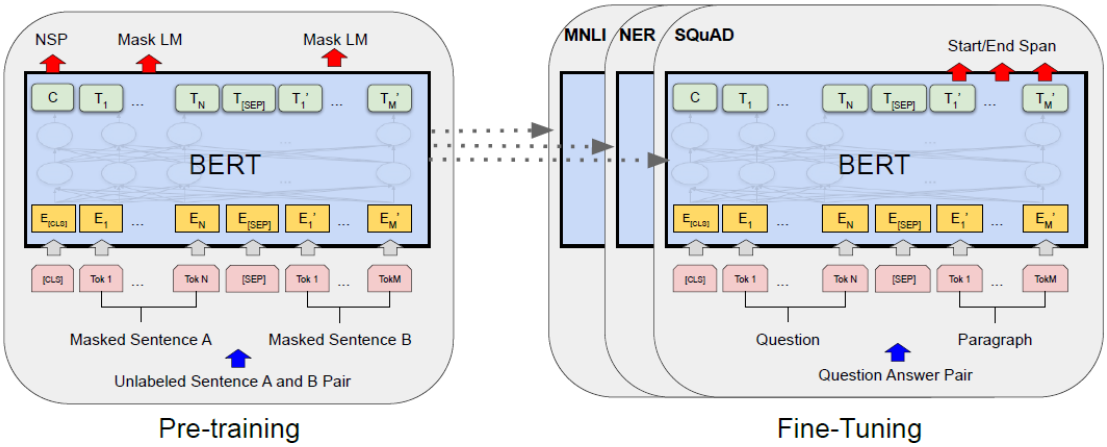


Figure 4.1: BERT training stages

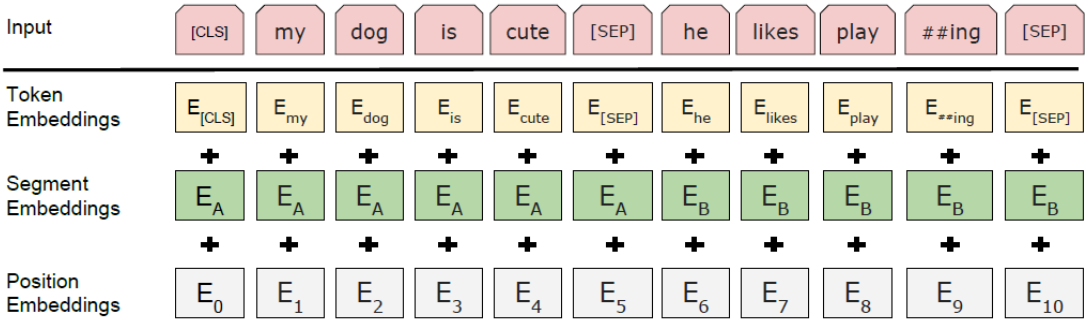


Figure 4.2: BERT input sequence representation

During masked language modeling, some percentage of the input tokens were masked at random using the token [MASK], and then the model predicted those masked tokens. The final hidden vectors corresponding to the mask tokens were fed into an output softmax over the vocabulary. During the next sentence prediction stage, the model saw the first sentence in every sequence and predicted what the next sentence would be. Weights are updated while performing these two tasks. These weights are the starting point for the fine-tuning process. These weights make it more effective to converge on the right embeddings. To fine-tune the model, we need to plug in the task-specific inputs and outputs into BERT and train all the parameters end-to-end. After flowing through

all the layers, the token representations can be used for further token-level tasks such as sentiment analysis, text classification, tagging, etc (fig-4.1).

Two versions of BERT were developed - BERT-base and BERT-large. Table **??** holds an overview of all the necessary information on different versions of BERT and its variants. To pre-train, BERT used BooksCorpus and English Wikipedia corpus with 800 million words and 2,500 million words respectively.

4.2.2 ALBERT

ALBERT i.e. 'A Light BERT' is a variant of BERT architecture proposed by [54]. An ALBERT configuration similar to BERT-large has 18x fewer parameters and can be trained about 1.7x faster. It introduced two parameter reduction techniques - factorized embedding parameterization and cross-layer parameter sharing. Along with reducing parameters, these techniques also acted as a form of regularization, helped with generalization, lowered memory consumption, and increased the training speed of BERT. These techniques are -

- Factorized Embedding Parameterization: The large vocabulary embedding matrix was decomposed into two small matrices. The size of the hidden layers from the size of the vocabulary embedding was separated. As a result, the hidden size was increased without significantly increasing the parameter size of the vocabulary embeddings. Instead of projecting the one-hot vectors directly into the hidden space of size H, they first projected them into a lower dimensional embedding space of size E and then projected it to the hidden space. By using this decomposition, reduced the embedding parameters from O(V * H) to O(V * E + E * H).
- Cross-layer Parameter Sharing: All layers shared the same parameters. It prevented the parameters from growing with the depth of the network. As a result, the transitions from layer to layer became much smoother for ALBERT than for BERT. This weight-sharing affected stabilizing network parameters too.

ALBERT also introduced a new self-supervised loss that focused on modeling inter-sentence coherence. This is called sentence order prediction loss (SOP). The original BERT used a next-sentence prediction loss (NSP). NSP conflated topic prediction and coherence prediction in a single task. However, topic prediction was easier to learn compared to coherence prediction. Another problem was that the learning done by topic prediction overlapped more with what was learned using the masked language modeling loss (MLM). This lack of difficulty as a task and overlapping situation made NSP ineffective. That's why, ALBERT used a sentence-order prediction (SOP) loss, which avoided topic prediction and instead focused on modeling inter-sentence coherence. It used the right order of sentence pairs in a sequence as positive examples and the wrong order as negative examples. The task was to predict the correct order of sentences in a

sequence. This forced the model to learn finer-grained distinctions about discourse-level coherence properties. SOP could solve the NSP task to a reasonable degree but the vice versa was not true. This loss consistently helped downstream tasks with multi-sentence inputs.

Due to all these design changes, ALBERT has a much smaller parameter size compared to the corresponding BERT models. Four versions were designed - ALBERT-base, ALBERT-large, ALBERT-xlarge, and ALBERT-xxlarge. Table **??** shows an overview of different versions of ALBERT and the comparison of them with the corresponding BERT models. ALBERT used 16GB of uncompressed data for pre-training. With only around 70% of BERT-large's parameters, ALBERT-xxlarge achieved significant improvements over BERT-large. ALBERT models have higher data throughput compared to their corresponding BERT models. ALBERTlarge is 1.7 times faster and ALBERTxxlarge is 3 times slower than BERTlarge. ALBERTxxlarge is computationally more expensive than BERTlarge due to its larger structure despite having fewer parameters. However, After training for roughly the same amount of time, ALBERT-xxlarge performed significantly better than BERT-large (on average, 1.5% better).

4.2.3 DistilBERT

DistilBERT architecture was proposed by [55]. It leveraged the knowledge distillation mechanism during the pre-training phase to build a better version of BERT. It showed that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and being 60% faster at inference time. It was lighter, faster, and smaller in computational budget. It could even run on mobile devices.

Knowledge distillation [56] [57] is a compression technique in which a compact model, the student, is trained to reproduce the behavior of a larger model, the teacher, or an ensemble of models. A model performing well on the training set will predict an output distribution with a high probability on the correct class and with near-zero probabilities on other classes. However, some of these "near-zero" probabilities are larger than others. These partially reflect the generalization abilities of the model and how well it will perform on the test set. Knowledge distillation walks on this narrow road to make the model more reliable in representation space. In DistilBERT architecture, the student is trained with a distillation loss over the soft target probabilities of the teacher. The normal loss would compare the prediction with the ground truth while the distillation loss would compare it with the teacher. The student would have to produce output that is closer to both the ground truth and the teacher. Fig 4.3 shows a simplified demonstration of knowledge distillation.

$$L_{distillation} = \Sigma_i t_i * log\left(s_i\right) \tag{4.1}$$

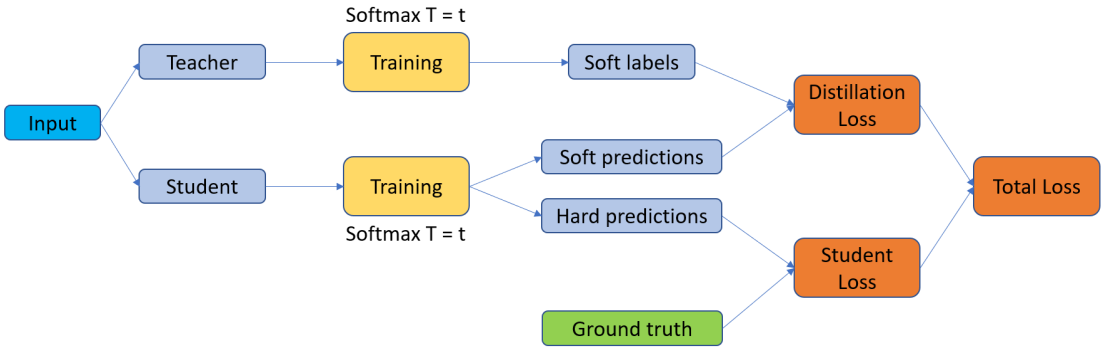


Figure 4.3: Simplified demonstration of knowledge distillation

Here, in eq 4.1, t_i and s_i are the probabilities estimated by the teacher and the student respectively. This objective resulted in a rich training signal by leveraging the full teacher distribution. A softmax temperature was also used to connect the teacher and student more accurately.

$$p_i = \frac{exp\left(\frac{r_i}{T}\right)}{\Sigma_j exp\left(\frac{r_j}{T}\right)} \tag{4.2}$$

Here, in eq 4.2, T controls the smoothness of the output distribution and r_i is the model score for the class i. The same temperature T is applied to the student and the teacher at training time. At inference, T is set to 1 to recover a standard softmax. Another loss, called cosine embedding loss L_{cos}, was added to align the directions of the student's and the teacher's hidden state vectors. In the end, a total of three losses were considered while training the student model. It is a linear combination of distillation loss $L_{distillation}$, masked language modeling loss L_{mlm}, and a cosine embedding loss L_{cos}. The last two are depicted as the 'student loss' in fig 4.3.

$$L = L_{distillation} + L_{mlm} + L_{cos} \tag{4.3}$$

Here, the teacher is the BERT model and the student model is the resulting DistilBERT model. It had the same general architecture as BERT. However, The token-type embeddings and the pooler were removed while the number of layers was reduced by a factor of 2. The student was initialized from the teacher by taking one layer out of two. It was distilled on very large batches leveraging gradient accumulation (up to 4 thousand examples per batch) using dynamic masking and without the next sentence prediction objective. It used a general-purpose pre-training distillation rather than a task-specific distillation. English Wikipedia and Toronto Book Corpus were used during training.

Leveraging the teacher's knowledge with initialization and additional losses led to substantial gains. DistilBERT retained 97% of the performance with 40% fewer parameters at 60% faster inference time than BERT. In the iPhone 7 mobile device test, DistilBERT is 71% faster than BERT. The whole model weighs 207 MB, which could be further reduced with quantization.

4.2.4 RoBERTa and XLM-RoBERTa

RoBERTa stands for 'Robustly optimized BERT approach'. It is a replication study of BERT pre-training that carefully measured the impact of many key hyperparameters and training data size. It was presented by [58]. They found that BERT was significantly undertrained and proposed an improved recipe for training BERT models, RoBERTa, that can match or exceed the performance of all of the post-BERT methods. They trained the BERT model longer with bigger batches over more data and longer sequences. Removed the next sentence prediction objective. Also, dynamically changed the masking pattern applied to the training data. They also investigated the data used for pre-training and the number of training passes through the data.

Dynamic masking is the process of generating the masking pattern every time a sequence is fed to the model. BERT used to place [MASK] tokens on fixed positions for all epochs. But, RoBERTa changed the masked sequence in 10 different ways over the 40 epochs of training. Each input was packed with full sentences sampled contiguously from one or more documents, such that the total length was at most 512 tokens. Removing the NSP loss matched or slightly improved downstream task performance. They also restricted sequences to come from a single document (DOC-SENTENCES). Training with very large mini-batches, while increasing the learning rate appropriately, improved optimization speed and end-task performance.

RoBERTa utilized the byte-pair encoding technique. Five English language corpora of varying sizes and domains were used to pre-train RoBERTa, totaling over 160GB of uncompressed text. It included BookCorpus of 16 GB, CC News corpus of 78 GB, OpenWebText corpus of 38 GB, and Stories corpus of 31 GB. Outperformed BERTlarge in all setups and XLNetlarge at (160GB data + 300k, 500k steps) setup. Also achieved state-of-the-art results on GLUE, RACE, and SQuAD, without multi-task fine-tuning for GLUE or additional data for SQuAD.

XLM-RoBERTa [59] a transformer-based multilingual masked language model pre-trained on 2.5 TB of newly created clean CommonCrawl data in 100 languages. They used a monolingual transformer architecture at first. Then they scaled it to 100 different languages. The designers had to consider the transfer dilution trade-off, the curse of multilinguality, the high resource vs low resource trade-off, the importance of capacity and vocabulary, etc.

Table 4.1: Comparison of large language models

Model	Versions	Parameters	Layers	Hidden	Embedding
BERT	Base	110M	12	768	768
	Large	340M	24	1024	1024
ALBERT	Base	12M	12	768	128
	Large	18M	24	1024	128
	X-Large	60M	24	2048	128
	XX-Large	100M	12	4096	128
DistilBERT	Base	66M	6	768	768
RoBERTa	Base	110M	12	768	768
XLM-RoBERTa	Base	270M	12	768	768
	Large	550M	24	1024	1024

4.2.5 Preprocessing

Text samples normally contain a lot of noises and unwanted characters in them. These noises can severely harm the model performance as they force the model to misrepresent the text in embedding space. This, in the end, produces a faulty output. So, a text cleaning process is a much-needed part of any kind of text analysis. For text preprocessing, the following tasks were carried out -

- Removal of numerical and special symbols
- Removal of extra spaces
- Removal of useless substring
- Contraction expansion
- Accented character removal
- Handling special characters and punctuation marks
- Lowercasing
- Tokenization

The datasets contained many useless words that didn't contribute to the dataset's values much. Rather, keeping them in the experiment would have hampered the result. So, these words were removed from the dataset. Most common was the word 'URL', 'user', 'RT', etc. When the datasets were created, every URL in the text samples was converted into the word "URL", the commenter's name was transformed into 'user', The retweeting was transformed into 'RT', etc. They conveyed no impact on the sentences at all. So, they were carefully excluded from the samples.

In English, often more than one word is combined into a single word such as "I'm", "You're", "got'em", etc. These contracted words sometimes can hide meaningful or impactful words in them, which can hinder analysis results. Furthermore, as the datasets contained tweets and comments from online platforms, they contained a lot of such examples. So, such contracted words were expanded to their full form. For example, "I'm" became "I am", "You're" became "You are", "got'em" became "got them" and so on.

Many of the characters in the dataset had some sort of accent over them such as - $\widetilde{a}$ or $\widetilde{w}$. These types of accented characters are very common on social platforms. Accented characters are usually used for aesthetic purposes. There are very few instances where they affect the context. People use them as some sort of styling or emphasizing factors such as - '$\widetilde{A}$merica'. However, instead of using accented characters, the words remain the same. But, while analyzing, the model will consider them as different words because, to the computer, their spellings are different. The computer can't understand that "America" and '$\widetilde{A}$merica' are the same words. To avoid this complexity, such accented characters were removed from the dataset and replaced with corresponding Unicode (ASCII, utf-8) characters.

Special symbols often have no meaningful impact on the comments. However, some punctuation marks are necessary to properly convey the message. The attention mechanism can correctly capture the context if some forms of punctuation marks are present in the samples. So, a few instances of (.,!) were kept. Excessive marks were omitted to reduce the complexity. Spaces were kept only to separate words. So, multiple consecutive spaces were removed. As the attention mechanism prefers raw information more, stop words were not removed. Removing stop words often has unusual consequences.

Fig 4.4 shows the entire proprocessing pipeline. Finally, the clean data was sent to the tokenization stages. As different models needed different tokenization functions because of their differences in architectures, several different pipelines were made to handle such situations.

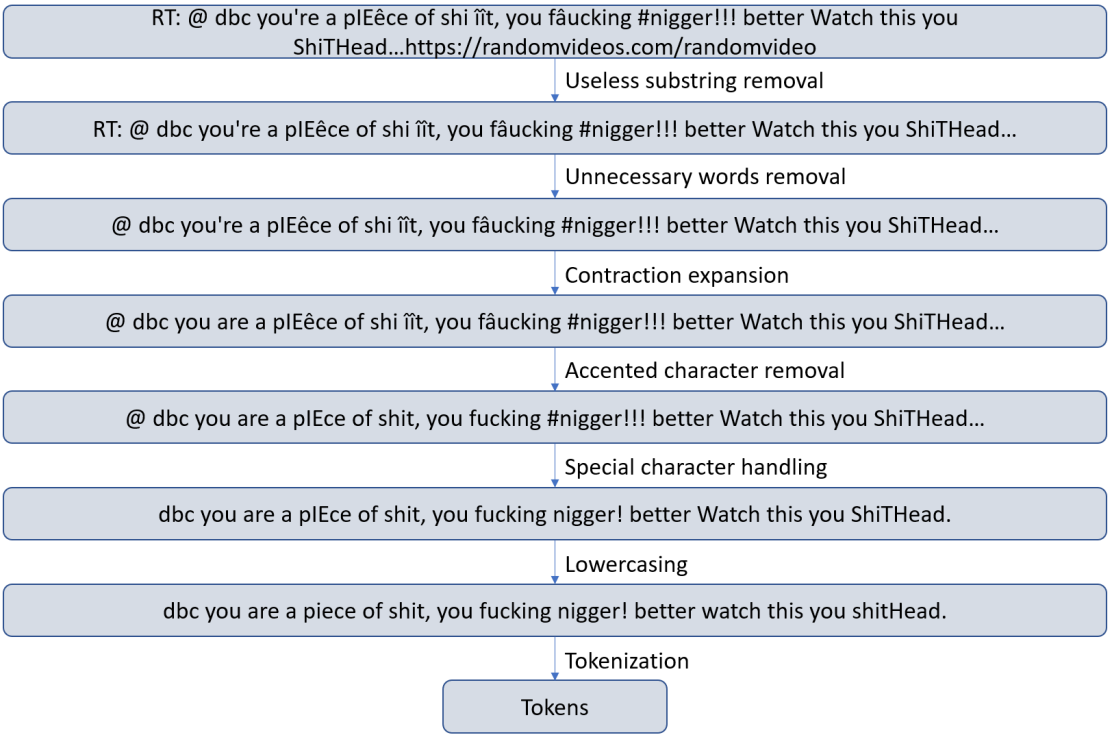


Figure 4.4: Preprocessing pipeline

4.2.6 Cross-dataset Analysis

The purpose of this experiment was to find out the generalizability of hate speech datasets across models and how much a model architecture gets affected by the newly encountered data. To do so, first, a model was trained on a single dataset. Then that dataset was tested on individual test sets from six separate datasets. If a model had good generalizable ability, this change in the dataset would not affect the model to a great extent. In this experiment, ALBERT, DistilBERT, and XLM-RoBERTa were applied. Due to computational complexity and resource constraints, the base versions of these models were used i.e. ALBERT-base, DistilBERT-base, and XLM-RoBERTa-base. The datasets included the Agarwal dataset, the MHS dataset, the HASOC2020 dataset, the Vidgen dataset, the Jigsaw dataset, and the OLID dataset. A total of 18 models were designed and trained. Then, each of these models was tested on all six test sets. Their result and performance were recorded. Fig 4.5 demonstrates an overview of the experiment process. As the loss function, binary cross-entropy loss as depicted in eq 4.7 was used.

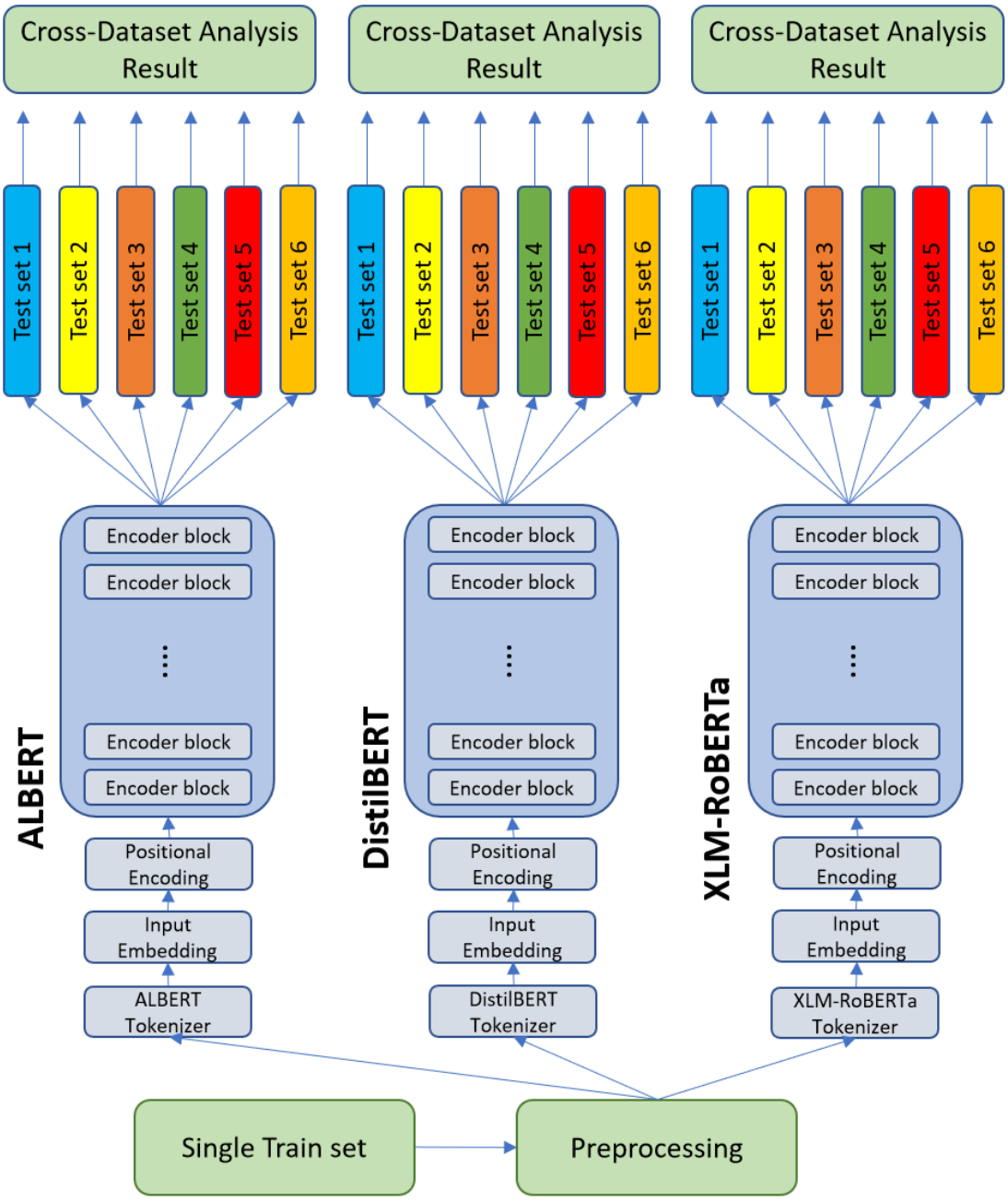


Figure 4.5: Cross dataset analysis

4.2.7 Ensemble Modeling

After performing cross-dataset analysis, a total of 18 models were designed. These models were the best performers of their respective architectures trained on corresponding

datasets. They had differences in their performances. They were not trained for the same amount of epochs either. So, the next step was to perform ensemble modeling to get a better result. There are multiple types of ensemble learning procedures. Among them, a hard-voting ensemble and a weighted ensemble were used in this experiment. In a hard-voting ensemble, the output generated by most of the contesting models was taken as the ensemble output. Fig 4.6 shows a simplified block diagram of the hard-voting ensemble. On the contrary, in a weighted ensemble network, all the outputs of all the models were passed through a trainable neural network to calculate the ensemble output.

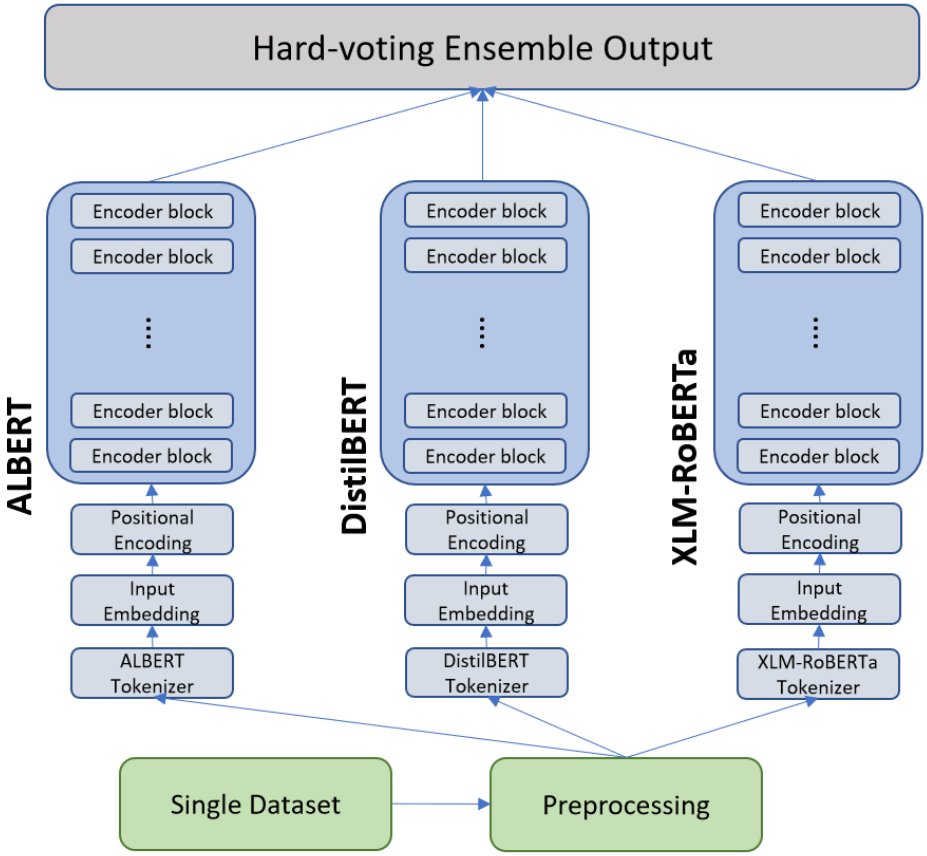


Figure 4.6: Hard voting ensemble

For each dataset, the best-performing models from each of the three architectures were collected. In this way, a total of 12 ensemble models were designed. Then an ensemble network was established between them. Among them, six were hard-voted ensembles and the other six were weighted ensemble networks. In the hard-voting ensemble system, no neural network was needed. Each of the models gave independent predictions. Then the ensemble block counted which class got the most votes, and made that as the final output.

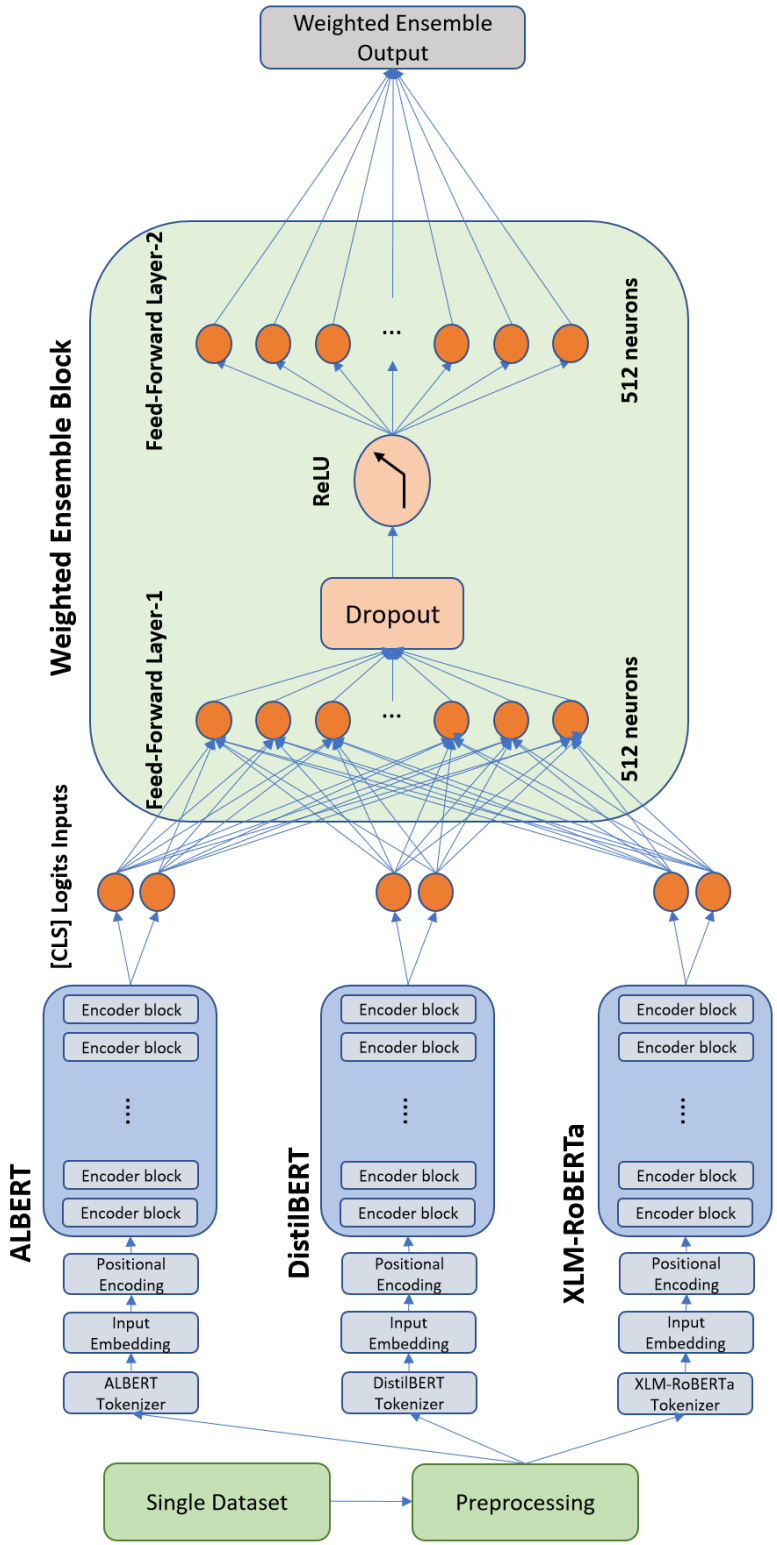


Figure 4.7: Weighted ensemble network

On the other hand, in the weighted ensemble network, the values of [CLS] tokens from each model's last layer representations were taken. These values are called 'logits'. For a single ensemble network, there were 3 models. This means, 6 logits were concatenated and fed into the ensemble network as input. The ensemble network was independently trained too.

$$\hat{p_{cl}} = Concat\Big(\hat{p_{al}}, \hat{p_{db}}, \hat{p_{xr}} \Big) \tag{4.4}$$

Here, eq 4.4 shows the concatenation of three sets of logits $\hat{p_{al}}$ $\hat{p_{db}}$ and $\hat{p_{xr}}$ from three different models.

Then, a 2-layer feed-forward network was designed. Each of these layers contained 512 neurons. To introduce non-linearity, ReLU and GELU were used. Dropout layers were added with a dropout value of 0.4. Several combinations of linear layers, dropouts, and different activations were designed. Two of them are shown in eq 4.5 and eq 4.6. Fig 4.7 shows the weighted ensemble architecture with ReLU and dropout. Then, all of them were trained. Only the best-performing one was kept as the final model. For six datasets, a total of 12 ensemble models were designed.

$$\hat{p} = FF\Big(ReLU\Big(Dropout\big(FF(\hat{p_{cl}})\big)\Big)\Big) \tag{4.5}$$

$$\hat{p} = FF\Big(GELU\Big(Dropout\big(FF(\hat{p_{cl}})\big)\Big)\Big) \tag{4.6}$$

Here, in eq 4.5 and eq 4.6, the concatenated logit $\hat{p_{cl}}$ was passed through different layers inside the ensemble block and finally produced class prediction $\hat{p}$.

$$L_{cls} = -\sum_{k=1}^{n} \Big(l_k * log\big(\hat{p_k}\big) + \big(1 - l_k\big) * log\big(1 - \hat{p_k}\big) \Big) \tag{4.7}$$

As the loss function, the binary cross-entropy loss was used as depicted in eq 4.7. Here, s is the set of samples, l is the set of corresponding labels, θ is the set of parameters of the model, and $\hat{p_k}$ is the prediction of the k-the sample among a total of n samples.

4.2.8 Adversarial Attack

For adversarial attack training, deliberately modified and defective versions of the samples from the dataset were produced. Each sample went through three types of modification -leetspeak generation, misspelling introduction, and special character insertion.

A leetspeak generator would change some random characters with similar-looking numbers or special characters. The changed characters were:

- 'i' became '1' or '!'
- 'o' became '0'
- 'e' became '3'
- 'l' became '1' or '!'
- 'a' became '@'
- 's' became '5'

These are some of the common character modifications seen in the real-life samples. For each sample, at first, the number of the above-mentioned characters was counted. Then, half of them were selected randomly and changed as stated above. This created a new sample, augmented from an actual sample. This augmented sample was added to the dataset.

In the misspelling introduction phase, at first, the number of total words in a sample was counted. Then, a random amount of them were deliberately modified into a wrong spelling. This new augmented sample containing misspellings would then be added to the new dataset. And, in the special character insertion phase, random special characters such as !,?¿*&%$#@()[]$@'" were inserted into the sample. Also, the spacing was modified by inserting random spaces at random positions. The augmented sample was later added to the dataset. Table 4.2 shows an example of this process.

Table 4.2: Adversarial sample generation example

Original	"look, i am never shutting up about this"
Leetspeak	"!0ok, ! am n3v3r shutting up about this"
Misspelling	"look, i aml neuer shuttxng up aboutv this"
Insertion	"loo!k[, i am -never shut$ting up a/bou*t th is"

In this way, from one sample three new samples were generated. As no words were introduced to or redacted from the samples, all three of them were assigned the same labels. This new augmented dataset with adversarial examples was used to train three different models using ALBERT, DistilBERT, and XLM-RoBERTa model architectures. They were tested on both the modified test set and the original test set.

4.2.9 Contrastive Learning

In the previous experiment, multiple different versions of the same sample were produced for adversarial training purposes. Now, in the contrastive learning process, these new samples were considered contrastively positive i.e. 'positive pairs' to each other and to the original. All other samples would be contrastively negative to them. The objective of contrastive learning is to place the representation of positive pairs in embedding space as close i.e. similar to each other as possible and also, the negative pairs as far or dissimilar as possible. Keeping the positive pairs together worked like 'attraction' or 'pull'. On the other hand, keeping the negative pairs far from each other acted like 'repulsion' or 'push'. Fig 4.8 shows a simplistic demonstration of contrastive learning concept.

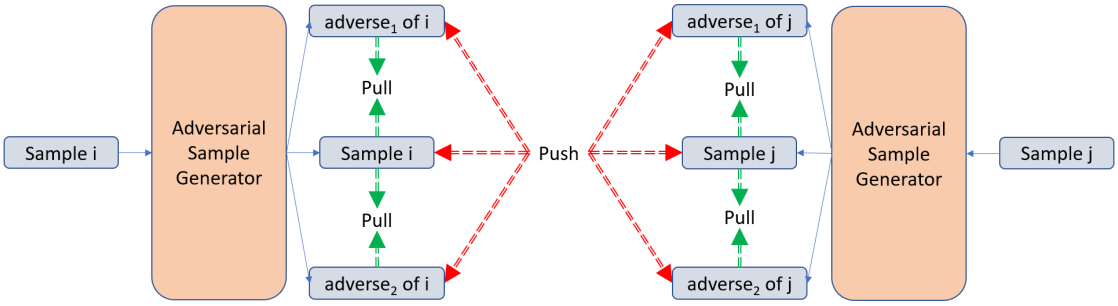


Figure 4.8: Contrastive learning

This 'push and pull' mechanism was implemented by the following loss functions -

$$L_{pull} = -log\left(\frac{exp\left(sim(r_i r_j)\right)}{\Sigma_{k\neq i} exp\left(sim(r_i r_k)\right)}\right) \tag{4.8}$$

$$L_{push} = -log\left(\frac{exp\left(sim(r_i r_j)\right)}{\Sigma_{k\neq i} exp\left(sim(r_i r_k)\right)}\right) \tag{4.9}$$

$$L_{cl} = L_{push} - L_{pull} \tag{4.10}$$

Here, in eq 4.8, for each positive pair representation r_i and r_j, the similarity was calculated. This similarity value was needed to be as large as possible. This is 'pull', denoted as L_{pull}. Then, in eq 4.9, the similarity of each negative pair representation r_i and r_j was calculated. This similarity value should be as little as possible. This was called 'push', denoted as L_{push}. For similarity measurement, cosine similarity was used. In both of the cases, the similarities were put through a softmax function. The logarithms

of these softmax results were later used in eq 4.10 to compute the total contrastive loss, denoted as L_{cl}.

$$L_{total} = L_{cls} + L_{cl} \tag{4.11}$$

So, as in eq 4.11, the total loss would be the summation of the classification loss, L_{cls} and contrastive loss, L_{cl}.

Classification loss tried to teach the model to learn how to label a sample properly by comparing the predicted label and the actual label. On the other hand, the contrastive loss attempted to train the model to group similar samples together by analyzing the respective representations produced by the model. As we know the large language models such as ALBERT, DistilBERT, and XLM-RoBERTa added an extra [CLS] token at the start of each sentence. This token value produced logits after being updated in every layer. These logits determined the class prediction and were used by classification loss, L_{cls}. The rest of the tokens formed the representation of that sentence in embedding space after going through all layers. L_{cl} used these values to compute the similarity and therefore, the contrastive loss. Fig 4.9 shows a figurative demonstration of this process.

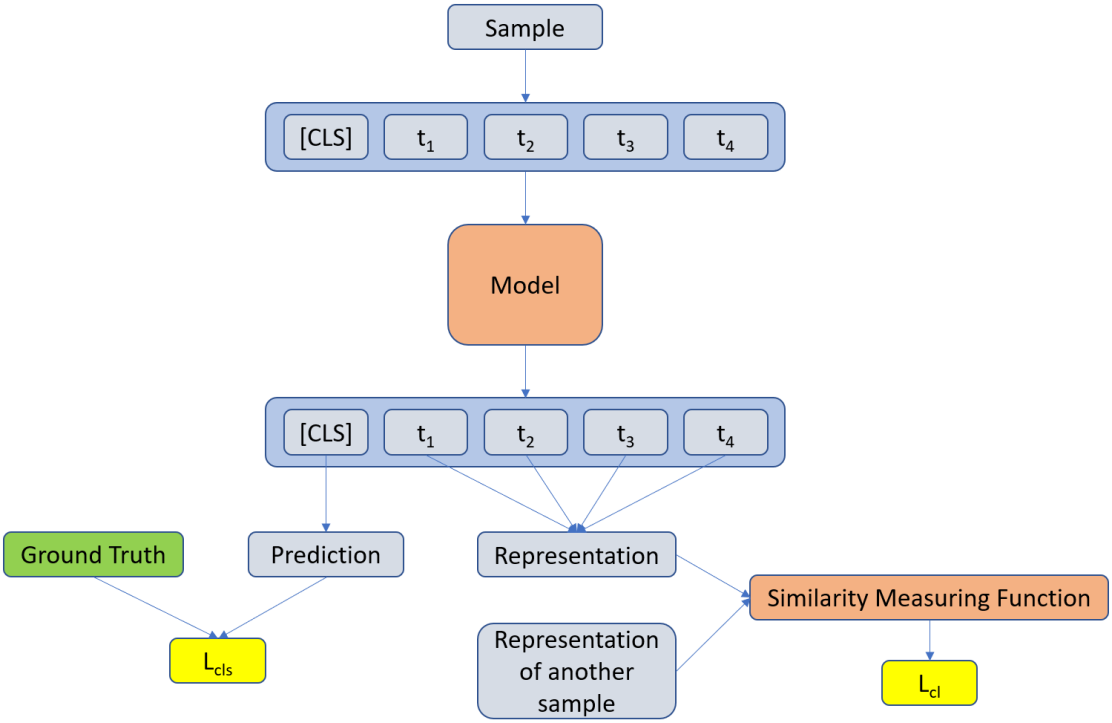


Figure 4.9: Classification and contrastive loss calculation process

After generating the contrastive adversarial samples of the HASOC2020 dataset and the OLID dataset, three models were created using ALBERT, DistilBERT, and XLM-

RoBERTa for each of the two. After that, their results were compared with each other and their previous non-contrastive versions to find out the differences between them.

Chapter 5

Implementation, Results and Discussion

5.1 Introduction

The experiment had four steps. At first, a cross-dataset analysis was conducted on six datasets. Then, their performance was measured. Secondly, two types of ensemble modeling were designed - hard-voting ensemble and weighted ensemble. Then their results were compared. In the third step, adversarial samples were generated and augmented datasets were used for retraining models on two datasets. FInally, a contrastive analysis was performed. A detailed explanation and discussion of all these experiments can be found in the later sections.

5.2 Experimental Setup

The entire work process was developed using PyTorch. PyTorch is a standard Python library designed for the usage of data science. Each dataset was first preprocessed. Then they were split with a ratio of 90-10. Maximum sequence length was set to 128 and 256. Both of the combinations were experimented with. As optimizers, Adam and AdamW were used. AdamW is a specialized version of Adam designed for transformer-based architectures. As a loss function, binary cross entropy was used. Different numbers of epochs were tried to find out the best result. 10% steps were set as warm-up steps. The learning rate was initially set to 0.00002. However, the Adam optimizer uses a variable learning rate. For ensemble modeling, the neural network was trained for 10 epochs. ReLU and GELU were used as non-linear functions. As ensemble dropout, 0.4 was set as the value. Comprehensive details of the experimental setups can be found in table 5.8.

In adversarial attack training and contrastive learning, each model was trained for 5 or 10 epochs.

Table 5.1: Experimental Setup

Parameter	**Value**
Dataset Split ratio	90-10
Batch size	32, 64, 100
Maximum sequence length	128, 256
Optimizer	Adam, AdamW
Loss function	Cross-entropy
Epochs	5, 10, 15
Warm-up period	10%
Learning rate	0.00002
Ensemble epochs	10
Ensemble activation	ReLU, GELU
Ensemble dropout	0.4
Framework	PyTorch

5.3 Evaluation Metrics

Standard evaluation metrics were used for evaluation purposes. These included accuracy, precision, recall, and f1-sore. However, most of the datasets used in this experiment were imbalanced. Accuracy is not a good measure when an imbalanced dataset is analyzed. F1-score can correctly measure the actual performance of the model.

If the model predicts a positive sample as positive, it is called a true positive. If the positively predicted sample is negative, it is called a false positive. If the prediction of a negative sample is positive, it is called a false positive. Otherwise, it's a true negative. Here,

- TP = True Positive
- TN = True Negative

- FP = False Positive
- FN = False Negative

So the formula for the metrics would be -

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1score = \frac{2 * Precision * Recall}{Precaision + Recall} \tag{5.4}$$

These metrics, as depicted in eq 5.1, eq 5.2, eq 5.3, and eq 5.4 were used extensively in each model training and testing session to judge their performances.

5.4 Dataset

One of the objectives of this research was to analyze different existing hate speech datasets. A cross-dataset analysis was also done. So, using one single dataset was not enough. Instead, a total of six datasets were used in this experiment. They were -

- Agarwal Dataset
- Measuring Hate Speech (MHS) Dataset
- HASOC2020 Dataset
- Vidgen Dataset
- Jigsaw Dataset
- OLID Dataset

These datasets varied in their characteristics, sizes, attributes, genres, labels, and other factors. Different research papers introduced these datasets. Some of these were fairly new compared to others. Some of these datasets categorized hate speech into several other classes such as 'offensive', 'toxic', 'aggression', 'insult', 'racist', sexist', etc. However, there are no universally accepted clear definitions of the distinguishing factors between these classes. The same comment can be viewed as 'hate speech' to somebody and 'non-hate but offensive' to others. Without any objective definition, this purely subjective division of hate speech raises new complexities. An unwanted bias gets introduced to the datasets that will negatively affect the models. This bias mimics the

personalities and the mentalities of the annotators. [60] addressed this issue and showed that the diverse and inconsistent definitions and annotations of multiple classes disturb the classifier performance.

This is why, it is safer and wise not to divide hate speech classes into multiple sub-classes i.e. categories. So, in this experiment, all the variants of hate speech classes were transformed into one single 'HSO' (hate-speech and offensive) super-class in all datasets.

Table 5.2: Details of different datasets used in the experiment

Dataset	Samples	HSO	Ave Length	Domain
Agarwal	31960	7%	72	Twitter
MHS	135556	41%	151	Social Media
HASOC2020	3707	50%	76	Twitter, Facebook
Vidgen	41144	54%	134	Internet
Jigsaw	159352	11%	375	Wikipedia
OLID	13240	33%	107	Twitter

A brief overview of the datasets can be found in the table 5.2. A detailed discussion can be found in the later sections.

5.4.1 Agarwal Dataset

This dataset was used by [18]. It is a publicly available dataset. It can be found on Kaggle[1]. The samples were collected from Twitter. It has 31960 samples in it. It has two classes labeled as '1' i.e. hate speech and '0' i.e. non-hate speech. It is an imbalanced dataset with only 7% of its samples being of the HSO class. The average sample length was 72. So, it is a medium-sized dataset with short sentences. Some samples from this dataset in its original and preprocessed form can be found in the table 5.3.

[1] https://www.kaggle.com/datasets/vkrahul/twitter-hate-speech

Table 5.3: Samples from Agarwal dataset

Original	**Preprocessed**
@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. run
@user @user thanks for #lyft credit i can't use cause they donǵ offer wheelchair vans in pdx. #disapointed #getthanked	thanks for lyft credit i cannot use because they do not offer wheelchair vans in pdx. disapointed getthanked
@user camping tomorrow @user @user @user @user @user @user @user	camping tomorrow
the next school year is the year for exams. canǵ think about that #school #exams #hate #imagine #actorslife #revolutionschool #girl	the next school year is the year for exams. cannot think about that school exams hate imagine actorslife revolutionschool girl
we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers	we won! love the land! allin cavs champions cleveland clevelandcavaliers

5.4.2 Measuring Hate Speech (MHS) Dataset

'Measuring Hate Speech' dataset is commonly known as the MHS dataset. It was prepared by [61] and [62]. It is also publicly available and can be found in Kaggle[2].

It consists of 39,565 comments annotated by 7,912 annotators, for 135,556 combined rows i.e. samples. The primary outcome variable is the "hate speech score" but the 10 constituent ordinal labels (sentiment, (dis)respect, insult, humiliation, inferior status, violence, dehumanization, genocide, attack/defense, hate speech benchmark) can also be treated as outcomes. It includes 8 target identity groups (race/ethnicity, religion, national origin/citizenship, gender, sexual orientation, age, disability, political ideology) and 42 target identity subgroups, as well as 6 annotator demographics and 40 subgroups. The hate speech score incorporates an IRT adjustment by estimating variation in the annotator's interpretation of the labeling guidelines.

It is a nearly balanced dataset with only 41% of its samples being of the HSO class. The average sample length is 151. So, it is a large dataset with short sentences. Some

[2] https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech

Table 5.4: Samples from MHS dataset

Original	**Preprocessed**	**Label**
@odysseuslahori @IqShoaib Hang till death all religious pimps !!	hang till death all religious pimps !	1
Well this explains all the Muslim people I saw gathering in a field this morning.	well this explains all the muslim people i saw gathering in a field this morning.	0
Saudi should attack Iran and see what happens to these barbaric cowards! America should stay out of the fight.	saudi should attack iran and see what happens to these barbaric cowards! america should stay out of the fight.	1
@4248 This is the influence of ”godly people” that you’re bragging about? It would seem to me that racists, demagogues, homophobes, transphobes, serial killers, bigots and misogynistic pieces of shit hide behind ”muh religion” to preach their hatred or anyone not a straight white male?	this is the influence of godly people that you are bragging about? it would seem to me that racists demagogues homophobes transphobes serial killers bigots and misogynistic pieces of shit hide behind muh religion to preach their hatred or anyone, not a straight white male?	0
get the fuck out of my country before i bash you beta cunt.	get the fuck out of my country before i bash you beta cunt.	1

samples from this dataset in its original and preprocessed form can be found in the table 5.4.

5.4.3 HASOC2020 Dataset

This dataset was introduced by [23]. It can be found in Kaggle[3]. It was the smallest dataset used here. It was perfectly balanced and mostly consisted of short sentences. A total of 3707 samples were in it. Exactly 50% of its samples were of the HSO class. The average sample length was 76. Some samples from this dataset in its original and preprocessed form can be found in the table 5.5.

Table 5.5: Samples from HASOC2020 dataset

Original	**Preprocessed**	**Label**
RT @mr_c3p0: Which is why u dumb bums need to stop getting your lashes done from any random lash âtechâ. Make sure theyâre certified at leaâ	which is why you dumb bums need to stop getting your lashes done from any random lash tech. make sure they are certified at lea.	1
RT @airjunebug: When youŕe from the Bay but youŕe really a NY nigga at heart. W/ @supportcaleon https://t.co/mZ8BAYlnlf	when you are from the bay but you are really a ny nigga at heart. w	1
RT @DonaldJTrumpJr: Dear Democrats: The American people arenât stupid, they know what spying is and no amount of gaslighting will change thâ	dear democrats the american people are not stupid, they know what spying is and no amount of gaslighting will change th.	0
RT @dreddxn: in my dream, youŕe with me	in my dream, you are with me	0
RT @TavianJordan: Summer â19 Iâm coming for you ! No boring shit ! Beach days, road trips, kickbacks and HOT DAYS ! Iâm ready Iâm ready Iâmâ	summer i am coming for you ! no boring shit ! beach days, road trips, kickbacks and hot days ! i am ready i am ready i am.	0

[3] https://github.com/suman101112/hasoc-fire-2020

5.4.4 Vidgen Dataset

Vidgen dataset was prepared by [63]. The dynamically generated this dataset using a human-in-the-loop approach. It is publicly available at Kaggle[4]. They divided hate speech into 6 types - derogation, animosity, threatening, support for hateful entities, and dehumanization. They focused on 29 target identities that are the most affected by targeted hate speech. Using 2 experts and 20 annotators they performed 4 rounds in the process. Every round had a target model that was trained on previous round data. For each new target model, they identified the best sampling ratio of data in previous rounds using the dev sets. Each round added new attacks to the weaknesses of the models. These attacks were designed using adversarial training. They used the RoBERTa model architecture in all rounds.

It contained 41114 entries. Where 54% of data were of the HSO class. The samples had an average sequence length of 134. These were collected from various sources on the internet. Compared with other datasets used in this experiment, this one had the least noise in the samples. Table 5.6 presents some samples from this dataset.

Table 5.6: Samples from Vidgen dataset

Original	**Preprocessed**	**Label**
Muslims are nothing like terrorists!	muslims are nothing like terrorists!	0
being chinese is a fate worse than death	being chinese is a fate worse than death	1
Chinese people aren't welcome around here	chinese people are not welcome around here	1
Muslims are murderers in disguise.	muslims are murderers in disguise.	1
Mental health is not a fucking disability	mental health is not a fucking disability	0
@BorisJohnson , fuck off!, Give it up you total twat!	, fuck off!, give it up you total twat!	0

5.4.5 Jigsaw Dataset

This publicly available dataset consists of Wikipedia comments labeled by human annotators. It originally had 7 classes. This is the largest dataset used in this experiment. It had about 159352 samples. However, this dataset was highly imbalanced. It had only 11% of samples that can be labeled as the HSO class. Most of the sentences in this

[4]https://github.com/bvidgen/Dynamically-Generated-Hate-Speech-Dataset

Table 5.7: Samples from Jigsaw dataset

Orignal	Preprocessed	Label
Could you please stop creating these problems? Indeed, Keith Briffa is notable largely because of his role in the ClimateGate and related events that are important for the big picture of the climate science. This is just a fact. That's why it's absolutely unacceptable for you to try to vandalize our article again.	could you please stop creating these problems? indeed, keith briffa is notable largely because of his role in the climategate and related events that are important for the big picture of the climate science. this is just a fact. that is why it is absolutely unacceptable for you to try to vandalize our article again.	0
"Ontario maps I notice you're the creator of a number of maps of Ontario, all apparently derived from a common source. Most, if not all, of them appear to have been uploaded without source information, and many of them have been deleted or are in danger of being deleted soon. à"	"ontario maps i notice you are the creator of a number of maps of ontario, all apparently derived from a common source. most, if not all, of them appear to have been uploaded without source information, and many of them have been deleted or are in danger of being deleted soon. "	0

dataset were longer than those of other datasets. They had an average sample length of 375. The source of the data was Wikipedia. Table 5.7 displays some example samples from this dataset.

5.4.6 OLID Dataset

Following a three layer annotation scheme, [33] prepared Offensive Language Identification Dataset (OLID). This dataset contained a total of 13240 samples. 33% of these samples were of the HSO class. This means it's not a balanced dataset. The average sample length is 107. This indicates most of the sentences are medium-sized. Table 5.8 contains some examples from this dataset.

Among the three layers, the first layer performed the task of offensive language detection. It discriminated the samples into two types - offensive and not offensive. Then comes the categorization of offensive language. Detecting whether the offensive sample is targeted or not targeted was the purpose of this layer. Then at the last layer, the targets of the offensive samples were extracted from the samples. They used Twitter API and keyword-based searching to collect the data. They used SVM, BiLSTM, and CNN as model architectures in the three layers.

Table 5.8: Samples from OLID dataset

Original	Preprocessed	Label
@USER @USER Why didn't Kavanaugh need terms? Guilty people: those are the ones that need terms. Let me clue you in: they're worried that she is going to make a terrible impression and will not be believable. They're trying to manage that concern.	why did not kavanaugh need terms? guilty people those are the ones that need terms. let me clue you in they are worried that she is going to make a terrible impression and will not be believable. they are trying to manage that concern.	0
@USER @USER @USER donât perpetuate colonialism. If you are going to talk about #puertorico you must have Puerto Ricans in the panel! Jeezuz.fucking.christ! How difficult can that be? #diÃsporaenresistencia URL	do not perpetuate colonialism. if you are going to talk about puertorico you must have puerto ricans in the panel! jeezuz.fucking.christ! how difficult can that be? diasporaenresistencia	0
An Anonymous message to #Antifa #Q #QAnon #TheGreatAwakening #CommieClowns #ResistanceRises #Resistance URL	an anonymous message to antifa q qanon thegreatawakening commieclowns resistancerises resistance	0
@USER Antifa has no place in society!	antifa has no place in society!	0
@USER Does anyone care what that dirtbag says???	does anyone care what that dirtbag says?	1

5.5 Implementation and Results

5.5.1 Cross-dataset Analysis

In the result of the cross-dataset analysis, some interesting scenarios were seen. In Table 5.11 the f1-scores of the corresponding models are shown. If any model achieves an f1-score of 70% or higher, it can be considered a 'good' performance. And, if the obtained f1-score is within 65% to 70%, it can be considered as a 'moderate' performance. Now, let's break down the results one by one. Table 5.9 holds the f1 scores and the accuracies of the models on the test sets of the respective train datasets. Table 5.10 holds the precisions and recalls of the models on the test sets of the respective

train datasets. Table 5.11 contains the result of the cross-dataset analysis. Let's break down the results one by one.

Table 5.9: F1 scores and accuracy of the models on corresponding datasets

Datasets	ALBERT		DistilBERT		XLM-RoBERTa	
	F1-score	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Accuracy**
Agarwal	74.35	96.31	72.77	96.56	**77.07**	**96.81**
MHS	76.86	81.26	77.21	81.63	**77.27**	**81.91**
HASOC2020	**92.78**	**92.45**	92.03	91.64	90.86	90.30
Vidgen	83.36	81.70	81.54	80.53	**83.96**	**82.99**
Jigsaw	82.31	**96.71**	82.12	96.32	**82.37**	96.46
OLID	72.52	79.68	72.05	**80.66**	**73.04**	78.93

Table 5.10: Precision and recall of the models on corresponding datasets

Datasets	ALBERT		DistilBERT		XLM-RoBERTa	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Agarwal	78.72	65.49	82.58	65.04	81.31	71.24
MHS	76.08	77.66	76.79	77.64	77.27	77.85
HASOC2020	94.74	90.91	93.72	90.40	91.33	90.40
Vidgen	79.71	87.36	81.15	81.94	83.08	84.85
Jigsaw	86.84	78.22	78.39	86.23	80.41	84.43
OLID	70.02	75.21	74.32	69.92	67.14	80.08

Agarwal Dataset

The Agarwal dataset was one of the most imbalanced datasets used in this experiment with only 7% of its sample being the HSO type. We can see that, among the three model architectures, XLM-RoBERTa achieved the highest f1 score on this dataset. It achieved

a f1-score of 77.07, which was about 3% higher than its nearest competitor, ALBERT. The same model also gained the highest accuracy, 96.81%. We can see in table 5.11 that every model that was trained on the Agarwal dataset completely broke when faced with a different test set, not from its dataset. The XLM-RoBERTa model, which gained the highest f1 score in this dataset, could not even cross the value of 50 in any other dataset. The states of the other two models were even poorer. It indicates that the Agarwal model has a very poor generalization ability. Models trained on this dataset hardly learn any useful info that can be exploited when encountered with an unknown scenario.

MHS Dataset

For the MHS dataset, the three models had a very close competition. The XLM-RoBERTa won the medal with the highest f1 score of 77.27, which is about 0.5% higher than the closest competitor, the DistilBERT model. The ALBERT model did not perform too badly either. The MHS dataset was the second-largest dataset in the experiment with about 135 thousand examples. Normally, it is expected to get better performance with a large dataset. However, we can see that the MHS dataset did not perform well in cross-dataset analysis. In most of the cases, it completely broke down. In the Jigsaw dataset, it barely passed the barrier of 50%. Other than that, it had no notable result in any other dataset. This means that the MHS dataset, although large, did not possess any generalization ability at all. This dataset was large but very sparsely distributed across different domains and genres. This sparse distribution made this dataset so weak to asses any new sample. Hence the disastrous result.

HASOC2020 Dataset

The HASOC2020 dataset showed a different behavior. The best model performed on it was not the XLM-RoBERTa, unlike the previous two. Instead, ALBERT was the best model with an f1-score of 92.78. It also showed some promising performance in the cross-dataset analysis. The ALBERT model trained on the HASOC2020 dataset performed well when it was tested on the Jigsaw test set. It scored a f1 score of over 70. On several other occasions, different models trained on this set achieved f1 scores of more than 65. This implies that the HASOC2020 dataset had a better generalization ability. It was the smallest dataset with over three thousand samples. However, it was perfectly balanced. Another nature was that this dataset mostly consisted of shorter sentences. Possibly, the smaller size and the shorter sentences made it less noisy than the previous two datasets. Also, the balanced ratio might have contributed to achieving a good performance.

Table 5.11: F1-scores of different models in cross-dataset analysis

Model	Train Dataset	Agarwal	MHS	HASOC2020	Vidgen	Jigsaw	OLID
ALBERT	Agarwal	74.35	21.87	10.48	31.21	13.49	15.16
	MHS	17.24	76.86	34.43	32.48	48.63	26.98
	HASOC2020	23.87	63.44	92.78	30.65	70.06	56.06
	Vidgen	42.07	63.03	28.91	83.36	23.47	24.43
	Jigsaw	18.23	71.67	90.77	39.75	82.31	62.53
	OLID	28.42	67.00	89.64	53.59	68.62	72.52
DistilBERT	Agarwal	72.77	36.21	12.73	33.87	20.12	14.9
	MHS	13.38	77.21	26.72	30.50	52.09	21.17
	HASOC2020	25.61	69.20	92.03	35.44	69.95	62.62
	Vidgen	18.89	66.08	21.85	81.54	25.19	23.09
	Jigsaw	21.96	71.56	90.64	43.75	82.12	67.90
	OLID	27.41	69.24	90.20	50.45	71.29	72.05
XLM-RoBERTa	Agarwal	77.07	40.28	18.10	39.07	25.17	22.02
	MHS	14.04	77.27	25.11	31.61	49.18	18.22
	HASOC2020	23.74	67.54	90.86	41.93	69.80	61.12
	Vidgen	18.54	64.02	21.49	83.96	25.73	22.15
	Jigsaw	19.12	71.30	90.77	46.55	82.37	67.40
	OLID	26.37	66.13	89.69	57.12	67.67	73.04

Vidgen Dataset

For the Vidgen dataset, XLM-RoBERTa came out as the winner again with an f1 score of 83.96. This model also achieved the highest accuracy. However, this dataset showed a poor performance in cross-dataset analysis. Like the first two, all the models trained on it completely broke down in all other scenarios except in the MHS dataset. All three models gained f1 scores of more than 60% in the MHS dataset. This indicates that these two datasets hold some sort of similar attributes. The types and the genre of the sentences in these two sets might have overlapped. Even though the Vidgen dataset was prepared with four consecutive rounds of rigorous analysis, its poor performance in cross-dataset analysis could not be avoided.

Jigsaw Dataset

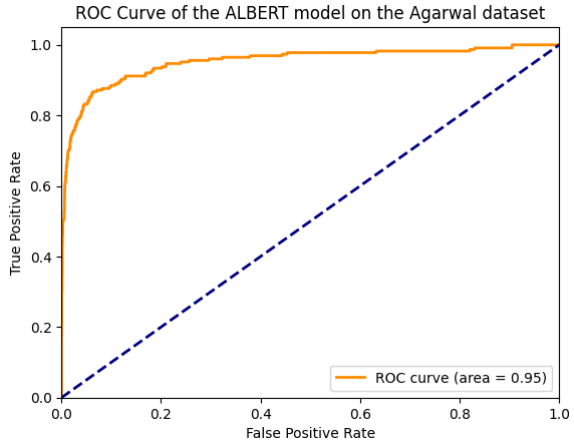
The XLM-RoBERTa model performed the best on the Jigsaw dataset too. It achieved an f1 score of 82.37%. However, it showed an odd behavior. All three models trained on this set achieved f1 scores of more than 90 in the HASOC2020 test set. In other words, they performed better on the HASOC2020 test set than on their own test set. These models also consistently gained f1 scores of more than 70 in the MHS dataset and f2 scores of more than 60 in the OLID dataset. As the several models performed well on several datasets, we can say that the Jigsaw model had very good generalization ability compared with the previous ones.

OLID Dataset

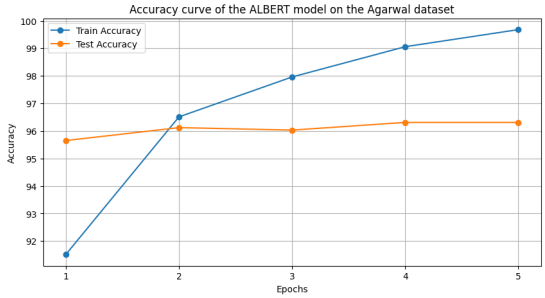
For the OLID dataset, again XLM-RoBERTa became the winner. It achieved an f1 score of 73.04%. However, the competition of the three models was very close with the difference being less than 1%. It also acted like the Jigsaw dataset. All the models trained on this dataset achieved higher f1 scores (about 90) on the HASOC2020 test set than their own test set. Models trained on this dataset showed a consistent performance across all test sets. They performed moderately well with the MHS dataset and the Jigsaw dataset. Only in the Agarwal dataset, they went haywire. By far, the OLID dataset showed the best performance in cross-dataset analysis.

The output characteristics i.e. accuracy curves, loss curves, ROC curves, and confusion matrices of all these models are shown below.

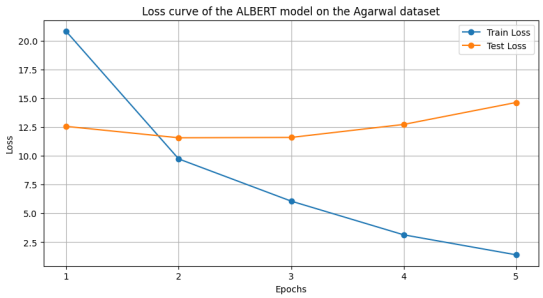
Model: ALBERT, Dataset: Agarwal



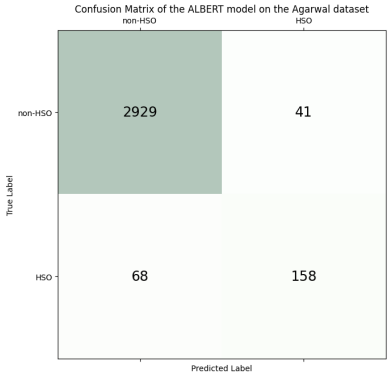
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

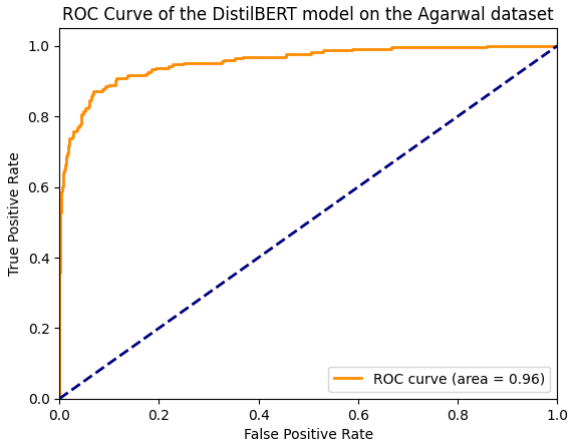


(d) Confusion matrix

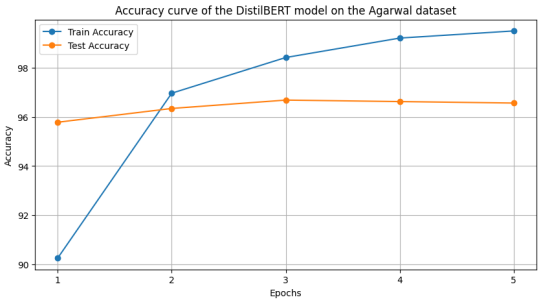
Figure 5.1: Performance of the ALBERT model on the Agarwal dataset

Fig 5.1 displays output characteristics curves and confusion matrix of the ALBERT model on the Agarwal dataset. This model was trained for five epochs. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

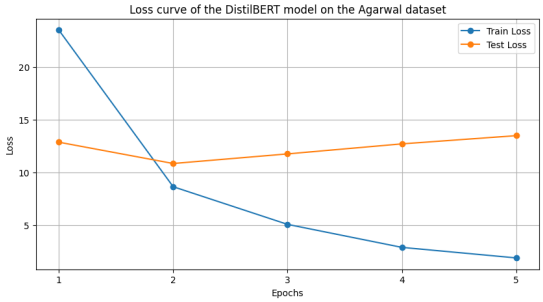
Model: DistilBERT, Dataset: Agarwal



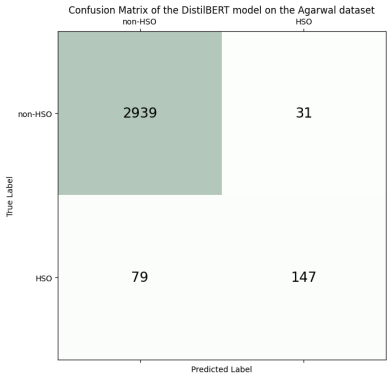
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

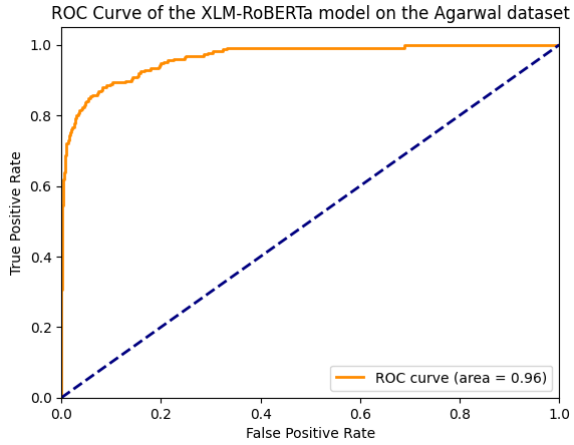


(d) Confusion matrix

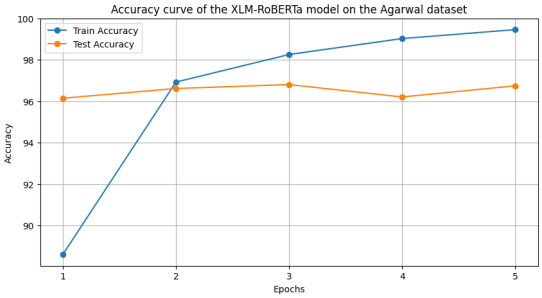
Figure 5.2: Performance of the DistilBERT model on the Agarwal dataset

Fig 5.2 displays output characteristics curves and confusion matrix of the DistilBERT model on the Agarwal dataset. This model was trained for five epochs. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

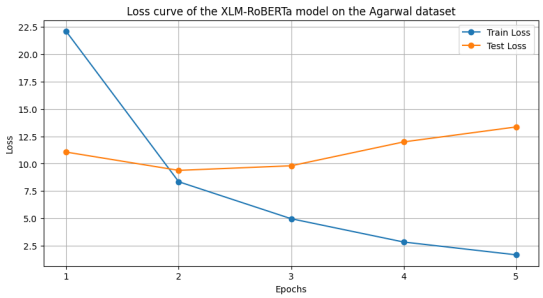
Model: XLM-RoBERTa, Dataset: Agarwal



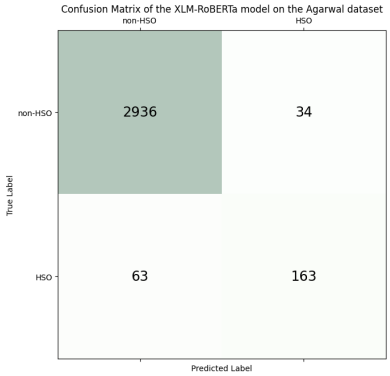
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

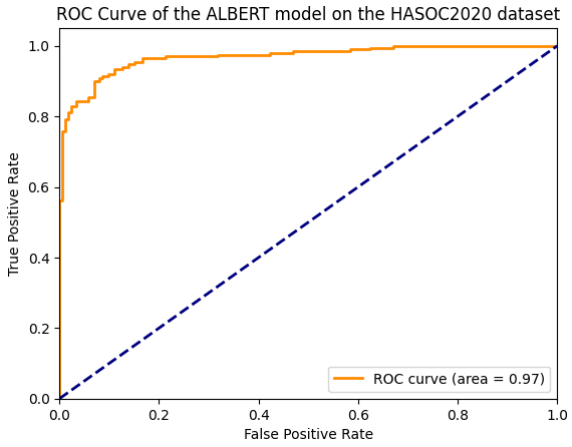


(d) Confusion matrix

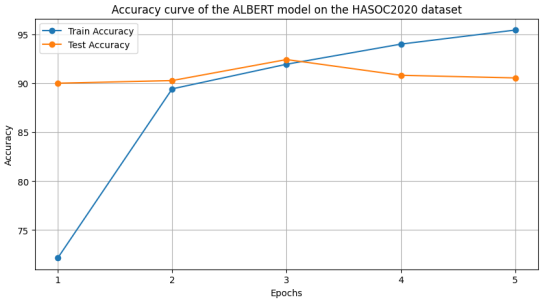
Figure 5.3: Performance of the XLM-RoBERTa model on the Agarwal dataset

Fig 5.3 displays output characteristics curves and confusion matrix of the XLM-RoBERTa model on the Agarwal dataset. This model was trained for five epochs. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

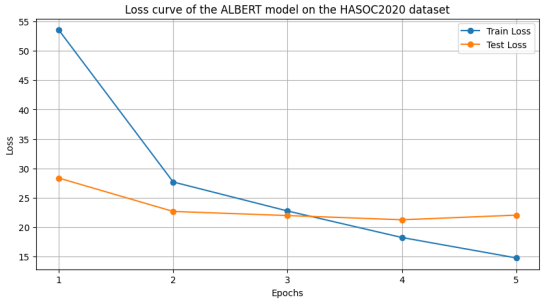
Model: ALBERT, Dataset: HASOC2020



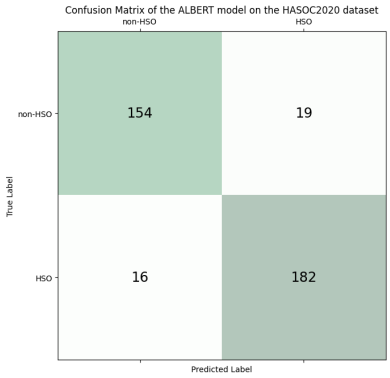
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

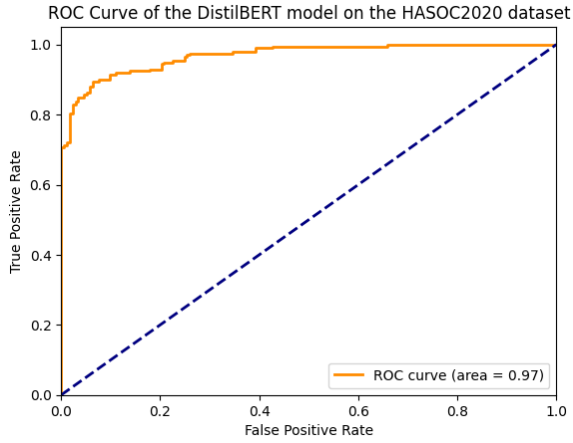


(d) Confusion matrix

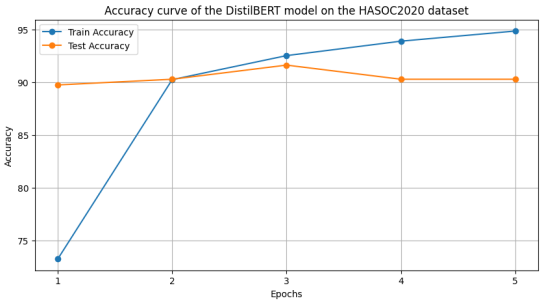
Figure 5.4: Performance of the ALBERT model on the HASOC2020 dataset

Fig 5.4 displays output characteristics curves and confusion matrix of the ALBERT model on the HASOC2020 dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

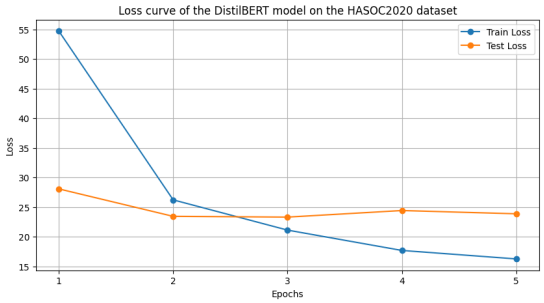
Model: DistilBERT, Dataset: HASOC2020



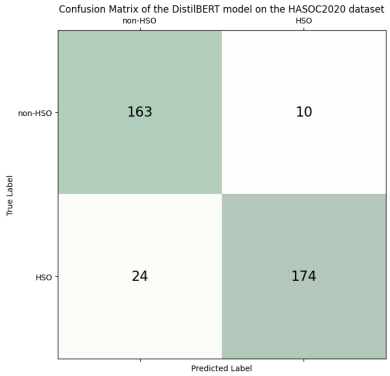
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

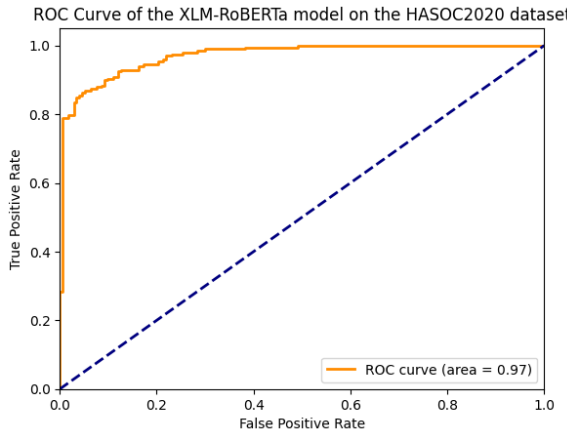


(d) Confusion matrix

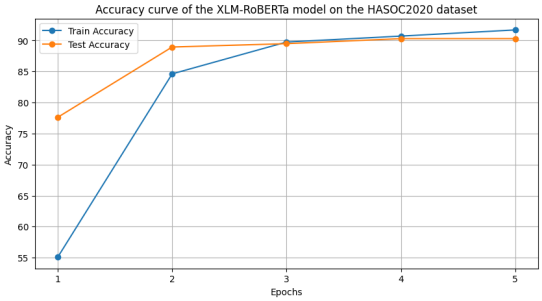
Figure 5.5: Performance of the DistilBERT model on the HASOC2020 dataset

Fig 5.5 displays output characteristics curves and confusion matrix of the DistilBERT model on the HASOC2020 dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

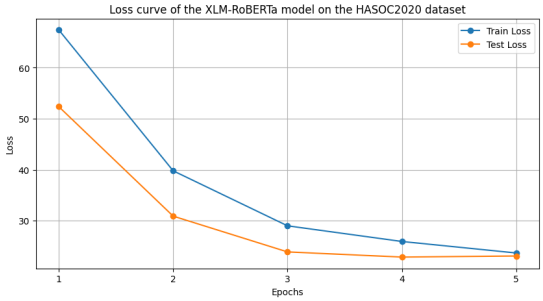
Model: XLM-RoBERTa, Dataset: HASOC2020



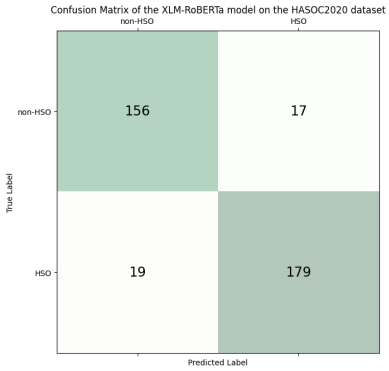
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

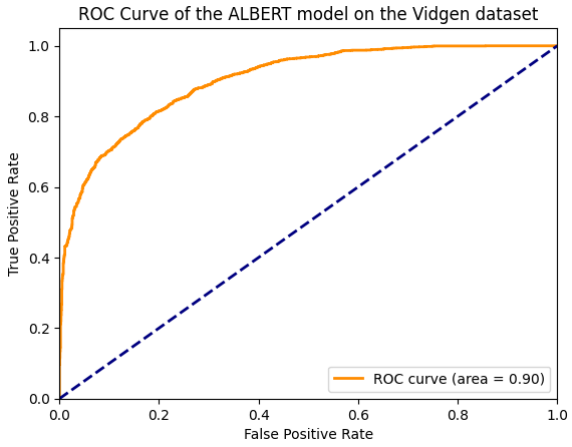


(d) Confusion matrix

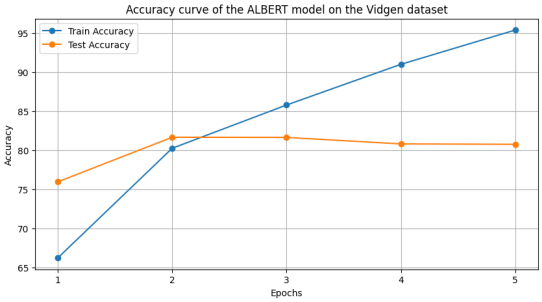
Figure 5.6: Performance of the XLM-RoBERTa model on the HASOC2020 dataset

Fig 5.6 displays output characteristics curves and confusion matrix of the XLM-RoBERTa model on the HASOC2020 dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

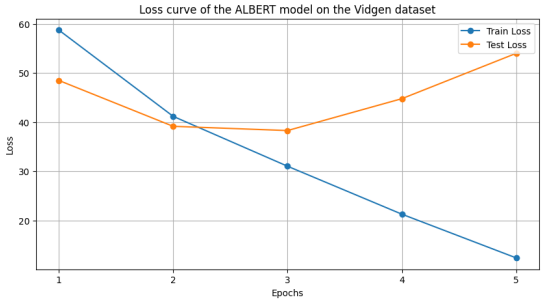
Model: ALBERT, Dataset: Vidgen



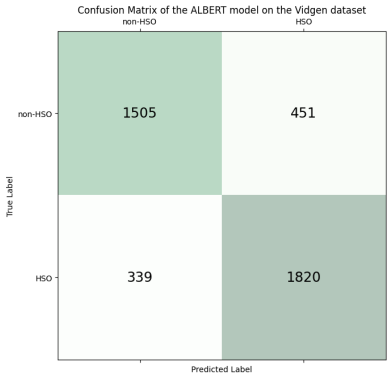
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

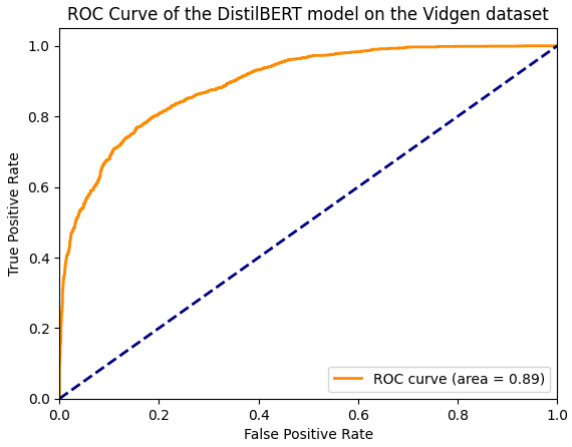


(d) Confusion matrix

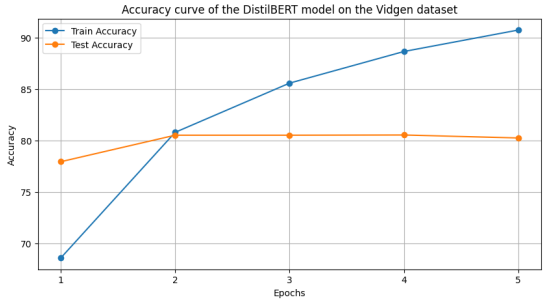
Figure 5.7: Performance of the ALBERT model on the Vidgen dataset

Fig 5.1 displays output characteristics curves and confusion matrix of the ALBERT model on the Vidgen dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

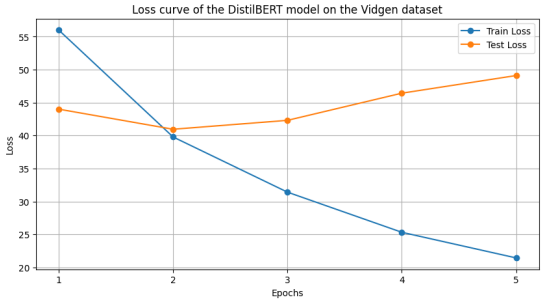
Model: DistilBERT, Dataset: Vidgen



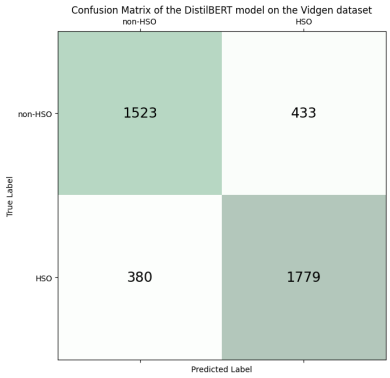
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

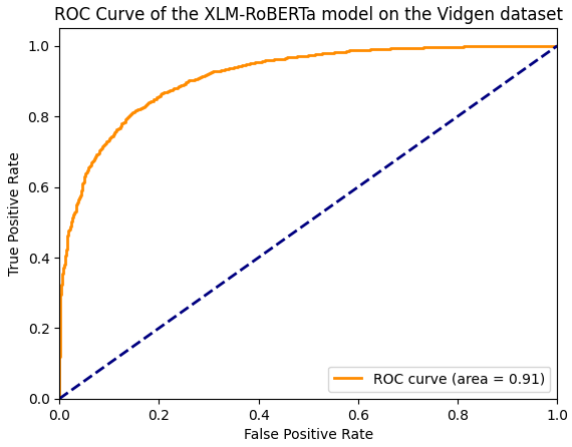


(d) Confusion matrix

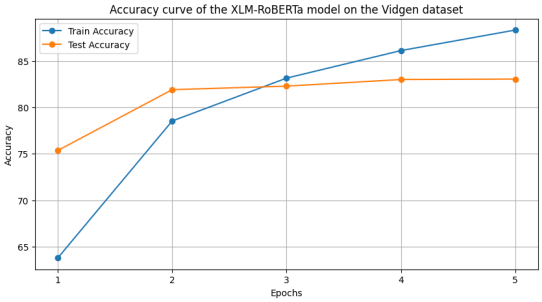
Figure 5.8: Performance of the DistilBERT model on the Vidgen dataset

Fig 5.2 displays output characteristics curves and confusion matrix of the DistilBERT model on the Vidgen dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

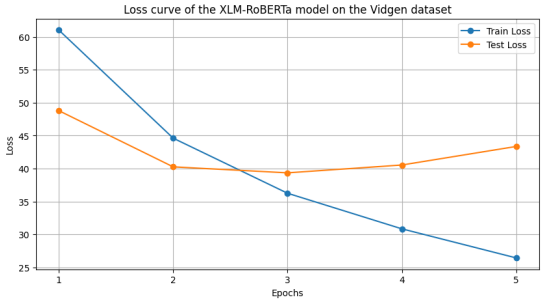
Model: XLM-RoBERTa, Dataset: Vidgen



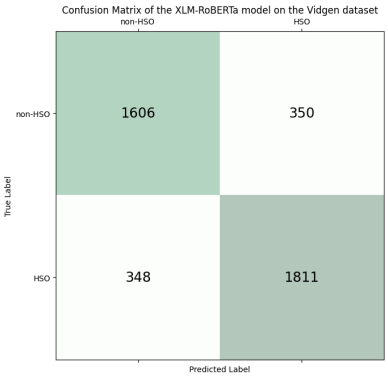
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

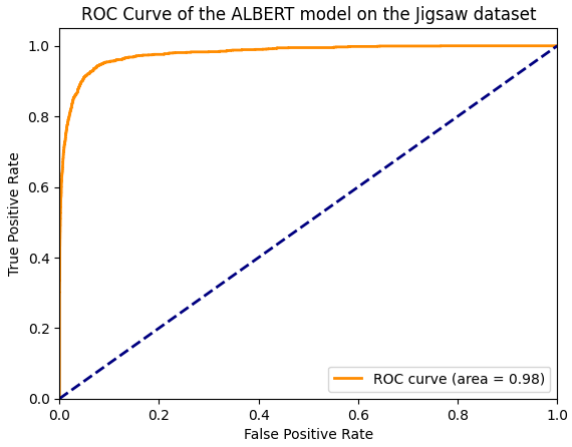


(d) Confusion matrix

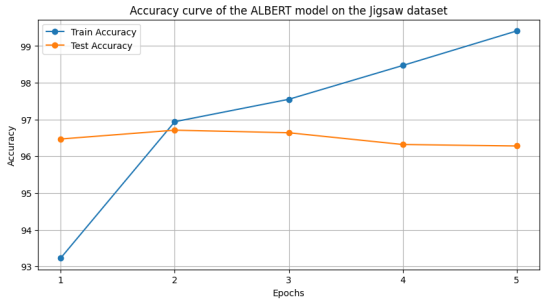
Figure 5.9: Performance of the XLM-RoBERTa model on the Vidgen dataset

Fig 5.9 displays output characteristics curves and confusion matrix of the XLM-RoBERTa model on the Vidgen dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

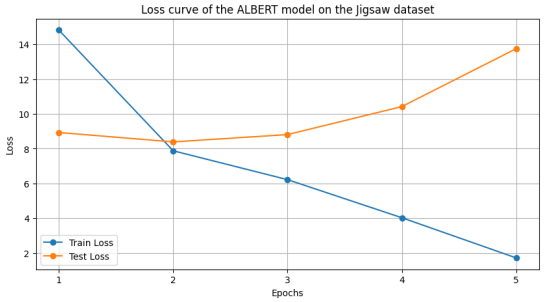
Model: ALBERT, Dataset: Jigsaw



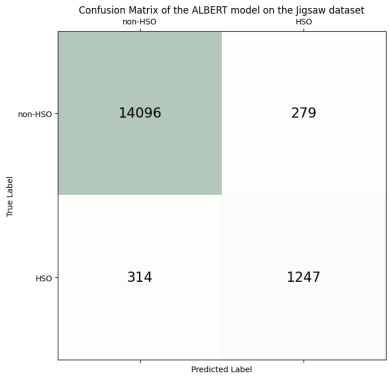
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

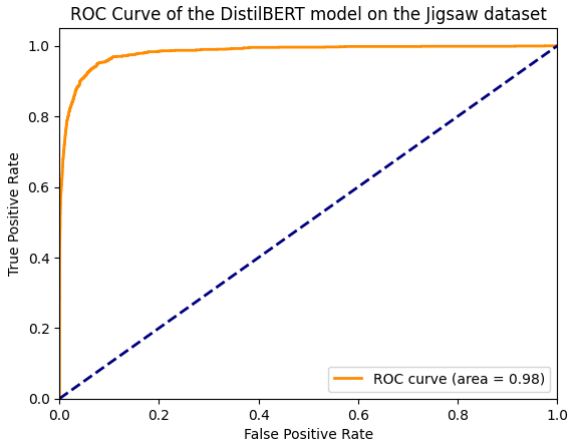


(d) Confusion matrix

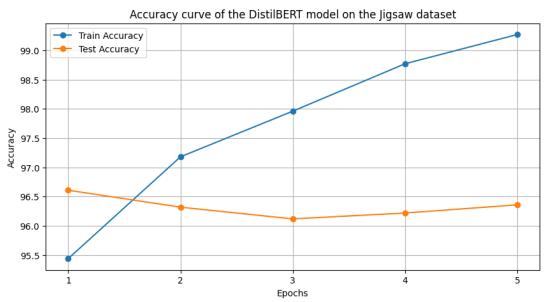
Figure 5.10: Performance of the ALBERT model on the Jigsaw dataset

Fig 5.10 displays output characteristics curves and confusion matrix of the ALBERT model on the Jigsaw dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

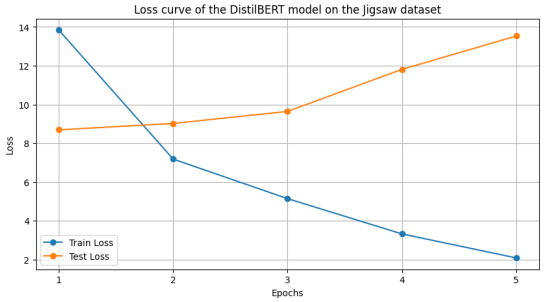
Model: DistilBERT, Dataset: Jigsaw



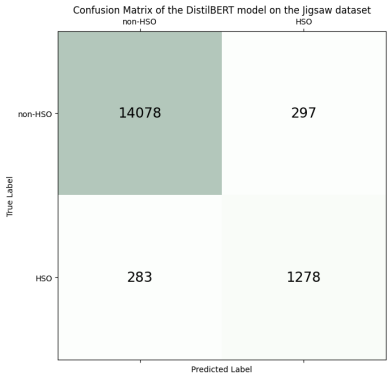
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

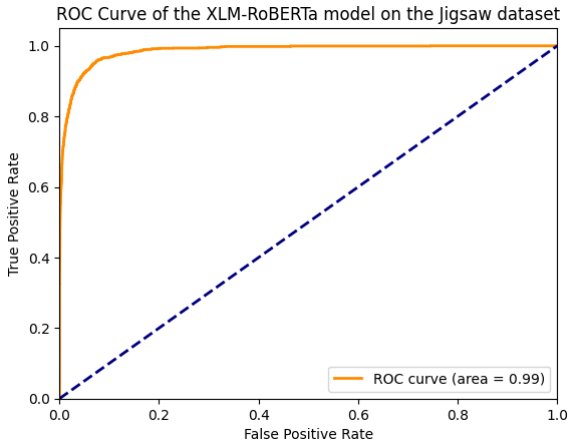


(d) Confusion matrix

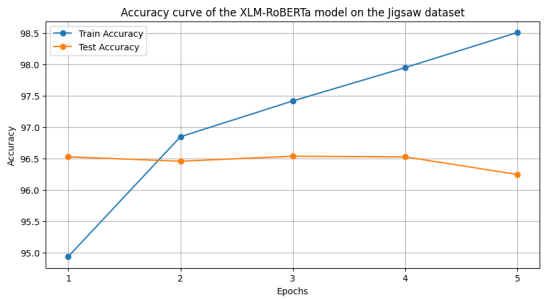
Figure 5.11: Performance of the DistilBERT model on the Jigsaw dataset

Fig 5.11 displays output characteristics curves and confusion matrix of the DistilBERT model on the Jigsaw dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

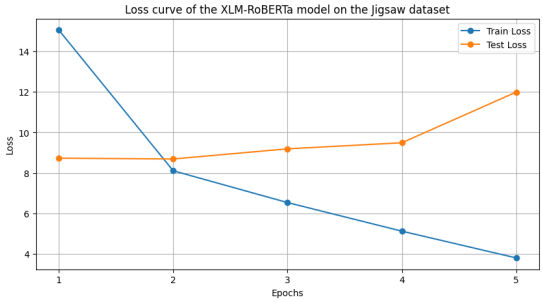
Model: XLM-RoBERTa, Dataset: Jigsaw



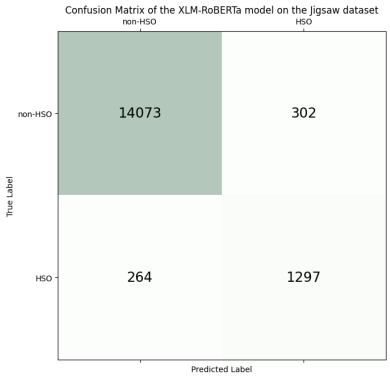
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

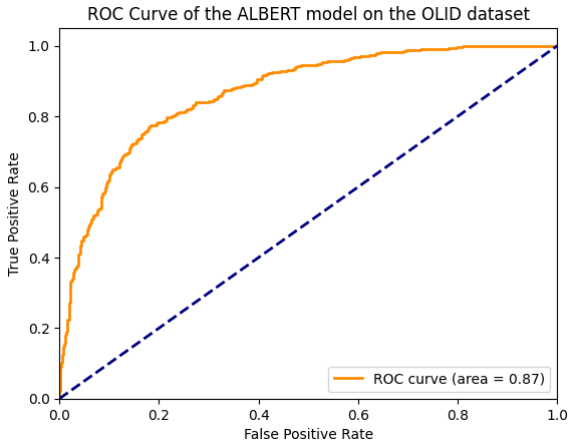


(d) Confusion matrix

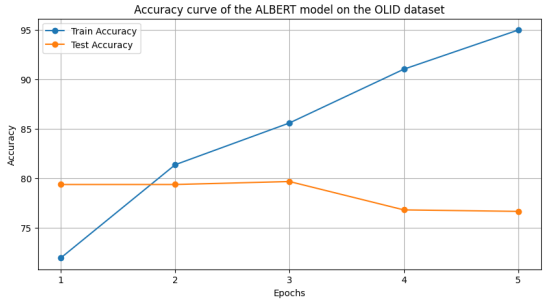
Figure 5.12: Performance of the XLM-RoBERTa model on the Jigsaw dataset

Fig 5.12 displays output characteristics curves and confusion matrix of the XLM-RoBERTa model on the Jigsaw dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

Model: ALBERT, Dataset: OLID

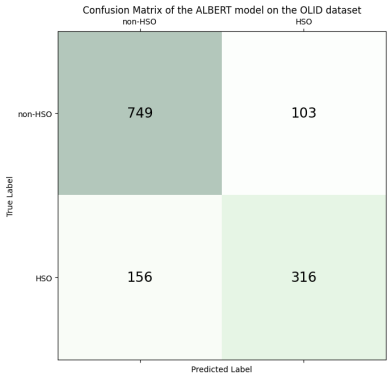


(a) ROC curve



(b) Accuracy curve

(c) Loss curve

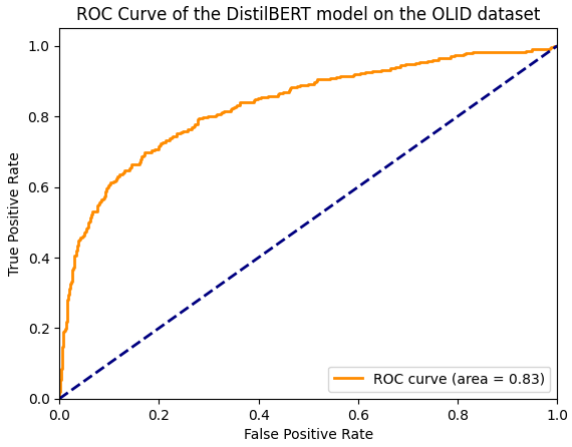


(d) Confusion matrix

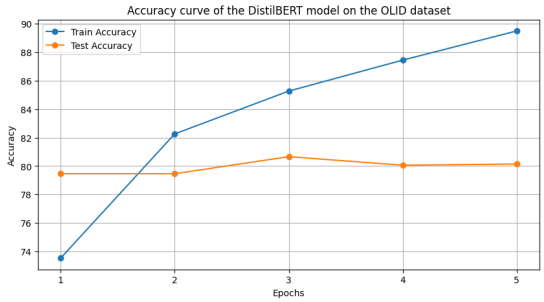
Figure 5.13: Performance of the ALBERT model on the OLID dataset

Fig 5.13 displays output characteristics curves and confusion matrix of the ALBERT model on the OLID dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

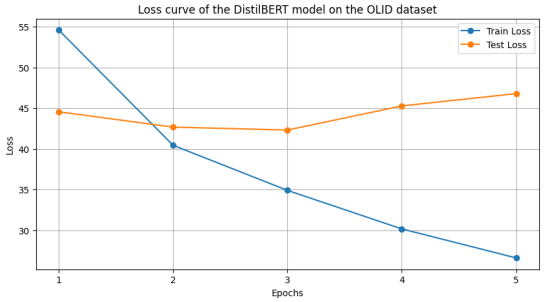
Model: DistilBERT, Dataset: OLID



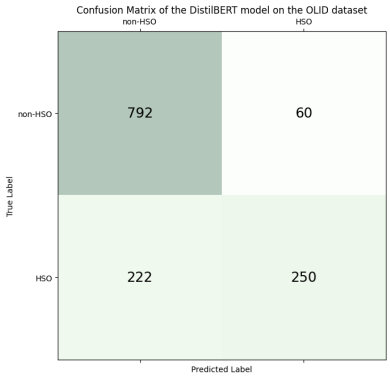
(a) ROC curve



(b) Accuracy curve



(c) Loss curve

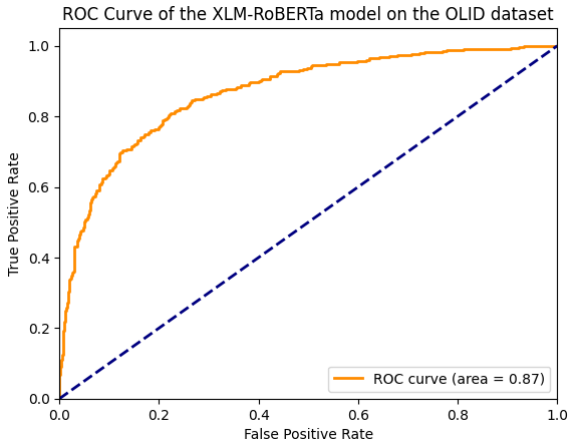


(d) Confusion matrix

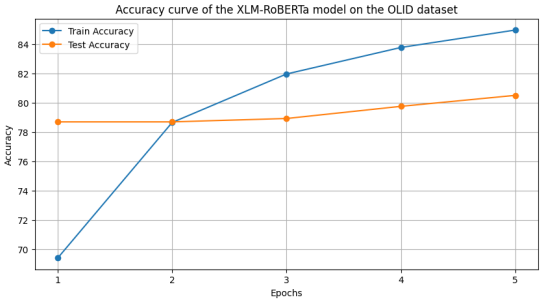
Figure 5.14: Performance of the DistilBERT model on the OLID dataset

Fig 5.14 displays output characteristics curves and confusion matrix of the DistilBERT model on the OLID dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

Model: XLM-RoBERTa, Dataset: OLID

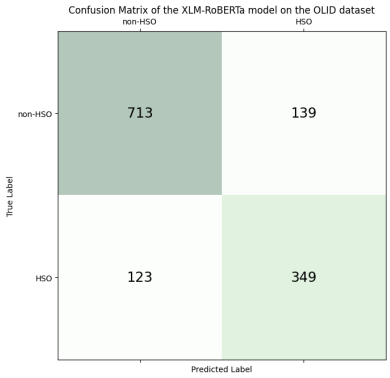


(a) ROC curve



(b) Accuracy curve

(c) Loss curve



(d) Confusion matrix

Figure 5.15: Performance of the XLM-RoBERTa model on the OLID dataset

Fig 5.15 displays output characteristics curves and confusion matrix of the XLM-RoBERTa model on the OLID dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.9 and table 5.10.

We can see the model architectures did not affect the cross-dataset analysis result to a noteworthy degree. This is on par with the argument of [5] - the nature of the dataset is more important than the model architecture. Also, any significant impact of class balance and dataset size on generalization ability was not found. Due to the innate randomness of hate speech samples, the large datasets used here might have a very sparse distribution of different domains and demographics. So, even though the models trained on them have seen more samples to learn from, due to the sparsity, they failed in the cross-dataset analysis. This indicates that to make a model generalizable, it needs to be trained on a dataset that has a huge amount of samples from all domains, demographics, and types. Certainly, all the datasets used here were not adequately large enough to hold that characteristic.

5.5.2 Ensemble Modeling

After the cross-dataset analysis, a total of 18 models were constructed using three different model architectures - ALBERT, DistilBERT, and XLM-RoBERTa. In the ensemble modeling experiment, for each dataset, three models, one of each type, were taken together and used to perform two types of ensemble operation - hard-voting ensemble and weighted ensemble. The purpose was to improve the results obtained by the models alone. Table 5.12 and 5.13 display the achieved result through the ensemble operation in comparison with each model.

Table 5.12: Comparison of f1 scores and accuracies of the models

Datasets	**ALBERT**		**DistilBERT**		**XLM-RoBERTa**		**Hard-voting Ensemble**		**Weighted Ensemble**	
	F1-score	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Accuracy**
Agarwal	74.35	96.31	72.77	96.56	**77.07**	96.81	75.79	96.90	76.09	**97.00**
MHS	76.86	81.26	77.21	81.63	77.27	**81.91**	76.12	81.43	**77.35**	81.72
HASOC2020	**92.78**	**92.45**	92.03	91.64	90.86	90.30	91.28	90.84	92.02	91.33
Vidgen	83.36	81.70	81.54	80.53	83.96	82.99	84.40	**83.38**	**84.64**	83.20
Jigsaw	82.31	**96.71**	82.12	96.32	82.37	96.46	**82.92**	96.66	82.55	96.62
OLID	72.52	79.68	72.05	80.66	73.04	78.93	73.15	81.65	**73.67**	**82.08**

We can see that the models' performance improved a lot using ensemble modeling. In four of the six datasets, the ensemble modeling achieved greater performances. The weighted ensemble model outperformed all others in the MHS dataset, the Vidgen dataset, and the OLID dataset. The hard-voting ensemble model came out as best in the Jigsaw dataset. However, in the case of Agarwal and Hasoc2020 datasets, ensemble modeling could not improve the result. It performed better than some models but not better than the best performers in those cases. Afterward, the output characteristics of weighted ensemble models on different datasets are shown.

Table 5.13: Comparison of precisions and recalls of the models

Datasets	ALBERT		DistilBERT		XLM-RoBERTa		Hard-voting Ensemble		Weighted Ensemble	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Agarwal	78.72	65.49	82.58	65.04	81.31	71.24	84.70	68.58	85.47	69.55
MHS	76.08	77.66	76.79	77.64	77.27	77.85	78.56	73.83	76.91	77.79
HASOC2020	94.74	90.91	93.72	90.40	91.33	90.40	92.71	89.90	90.91	93.17
Vidgen	79.71	87.36	81.15	81.94	83.08	84.85	83.15	85.69	81.35	88.20
Jigsaw	86.84	78.22	78.39	86.23	80.41	84.43	83.13	82.70	83.50	81.62
OLID	70.02	75.21	74.32	69.92	67.14	80.08	76.44	70.13	77.43	70.26

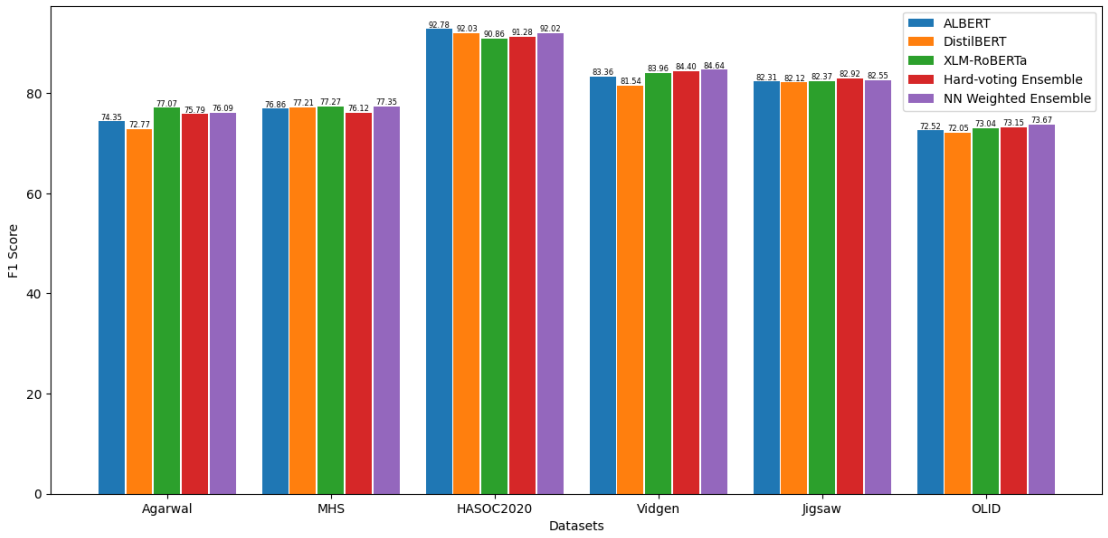
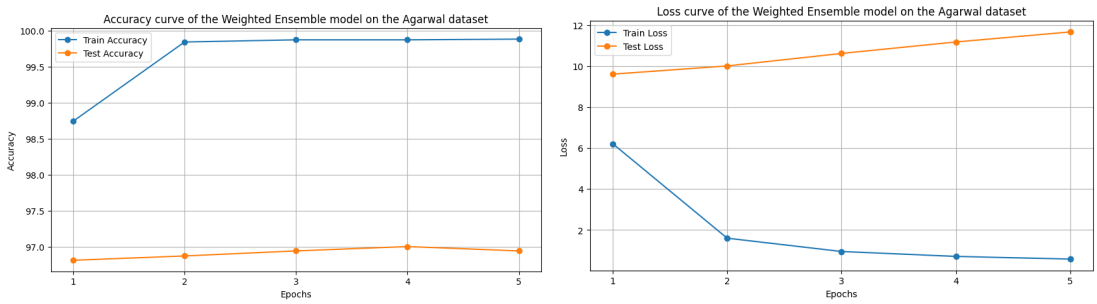


Figure 5.16: Comparison of different models on each dataset

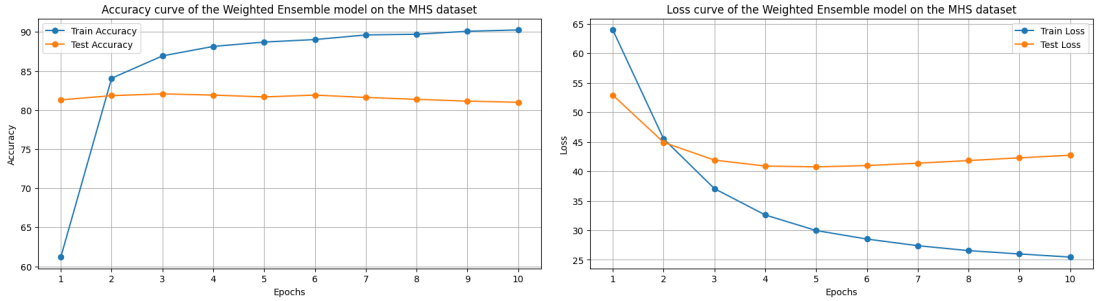
Fig 5.16 shows a figurative comparison of the f1 scores of all models in each dataset. An interesting observation here is that the models trained on the same dataset displayed nearly the same type of behavior even though they have different types of architectures. They achieved f1 scores in the same range. This proves that the dataset has more control over model performance rather than the model architecture.

Fig 5.17 displays output characteristics curves of the Weighted ensemble model on the Agarwal dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.



(a) Accuracy curve (b) Loss curve

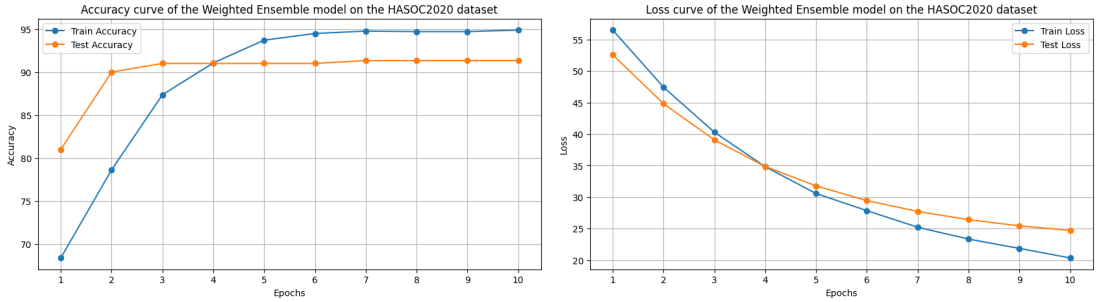
Figure 5.17: Characteristics of the ensemble network on the Agarwal dataset



(a) Accuracy curve (b) Loss curve

Figure 5.18: Characteristics of the ensemble network on the MHS dataset

Fig 5.18 displays output characteristics curves of the Weighted ensemble model on the MHS dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.



(a) Accuracy curve (b) Loss curve

Figure 5.19: Characteristics of the ensemble network on the HASOC2020 dataset

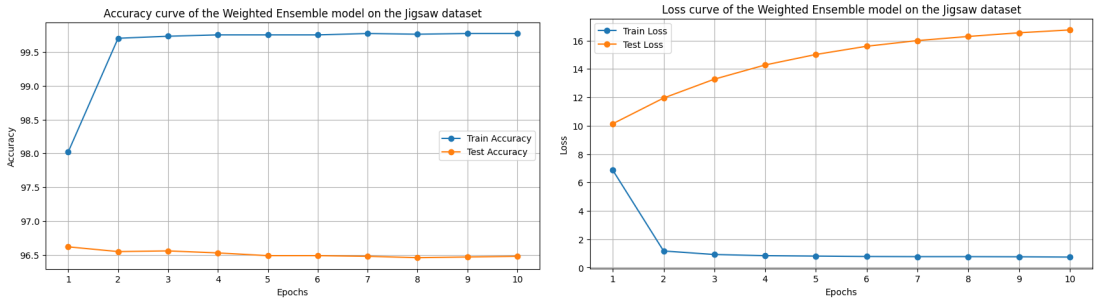
Fig 5.19 displays output characteristics curves of the Weighted ensemble model on the

HASOC2020 dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.

(a) Accuracy curve (b) Loss curve

Figure 5.20: Characteristics of the ensemble network on the Vidgen dataset

Fig 5.20 displays output characteristics curves of the Weighted ensemble model on the Vidgen dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.

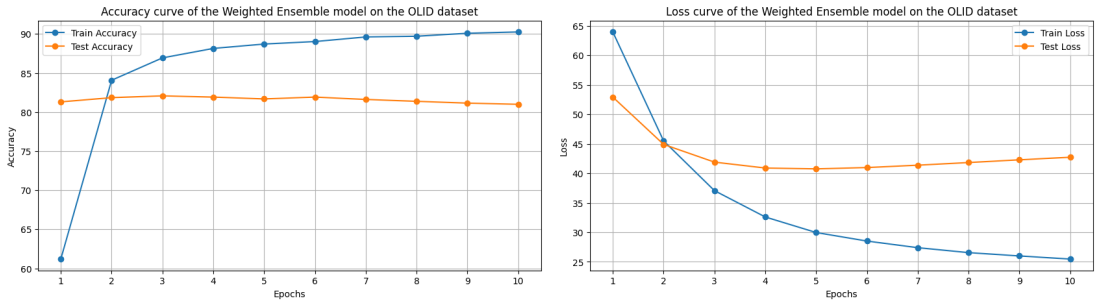


(a) Accuracy curve (b) Loss curve

Figure 5.21: Characteristics of the ensemble network on the Jigsaw dataset

Fig 5.21 displays output characteristics curves and confusion matrix of the Weighted ensemble model on the Jigsaw dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.

Fig 5.22 displays output characteristics curves of the Weighted ensemble model on the OLID dataset. The value of the acquired f1 score, accuracy, precision, and recall can be found in table 5.12 and table 5.13.

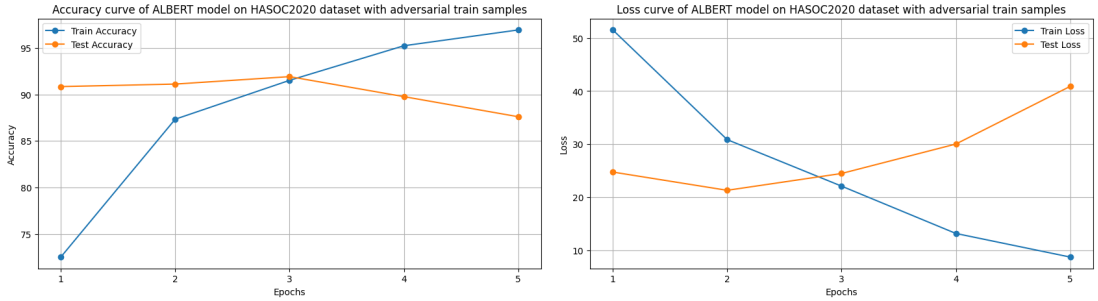


(a) Accuracy curve (b) Loss curve

Figure 5.22: Characteristics of the ensemble network on the OLID dataset

5.5.3 Adversarial Attack

In the adversarial attack experiment, the HASOC2020 dataset and the OLID dataset were put through an augmentation process and two larger datasets with adversarial examples was produced. Then, these augmented datasets were used on three model architectures - ALBERT, DistilBERT, and XLM-RoBERTa. For the HASOC2020 dataset, this approach improved the f1 score of the XLM-RoBERTa model by more than 1%. The effect on the ALBERT model was very miniscule. However, DistilBERT was worsened by this approach. Its f1 score dropped about 1.5%. On the other hand, the performances of models trained on the OLID dataset were greatly improved. It showed more than 2% improvement in the DistilBERT model. The other two were also improved. Table 5.14 shows the corresponding f1 score of different models used in different settings.

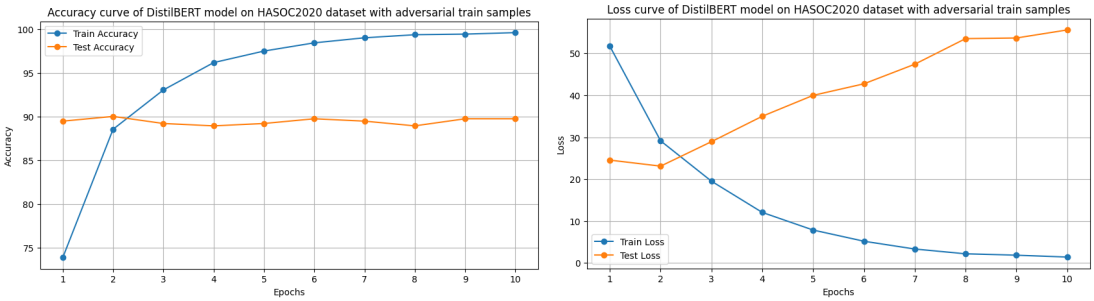


(a) Accuracy curve (b) Loss curve

Figure 5.23: ALBERT model on the augmented HASOC2020 dataset

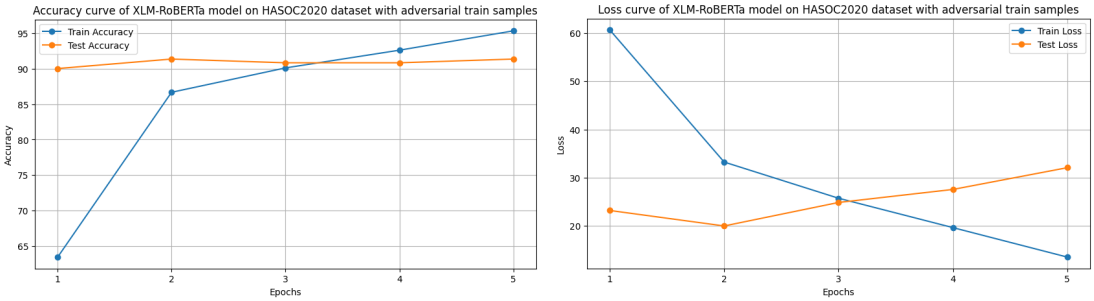
5.5.4 Contrastive Learning

In this experiment, contrastive loss was introduced along with adversarial samples where the augmented samples were considered positive samples to the original and others



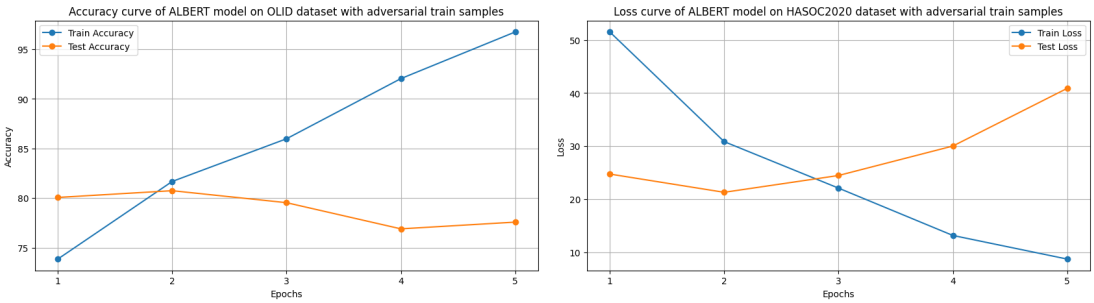
(a) Accuracy curve (b) Loss curve

Figure 5.24: DistilBERT model on the augmented HASOC2020 dataset



(a) Accuracy curve (b) Loss curve

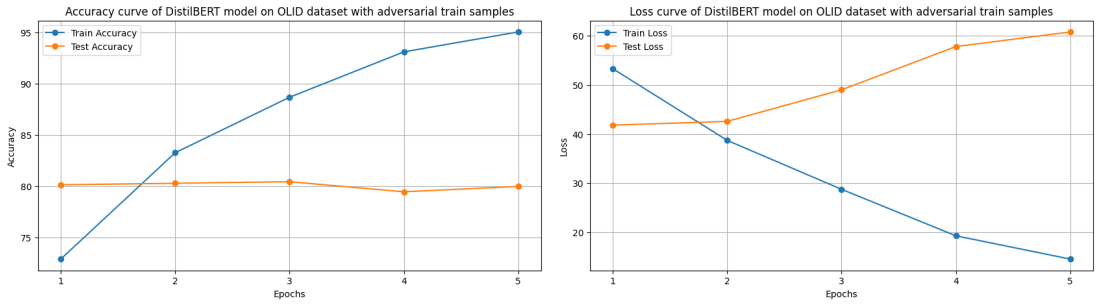
Figure 5.25: XLM-RoBERTa model on the augmented HASOC2020 dataset



(a) Accuracy curve (b) Loss curve

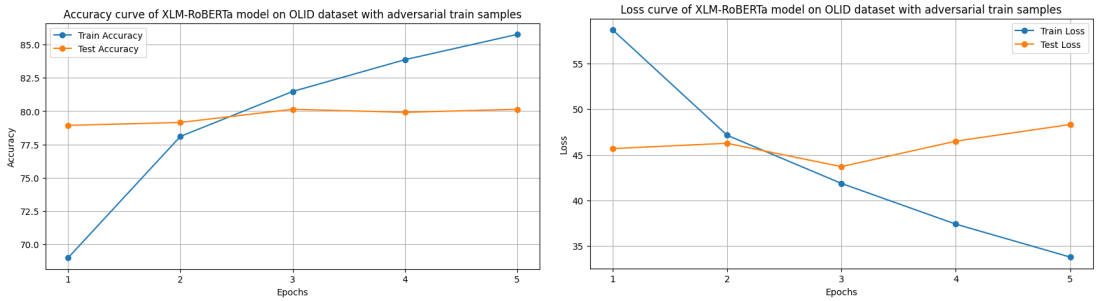
Figure 5.26: ALBERT model on the augmented OLID dataset

with the same origin. And, samples from different origins were considered negative to each other. A pull-push mechanism was in effect. This effect was implemented by using contrastive loss. Table 5.14 shows a comparison of the f1 scores of three types of models in three types of settings. After experimenting with multiple settings, improvements could be seen.



(a) Accuracy curve (b) Loss curve

Figure 5.27: DistilBERT model on the augmented OLID dataset



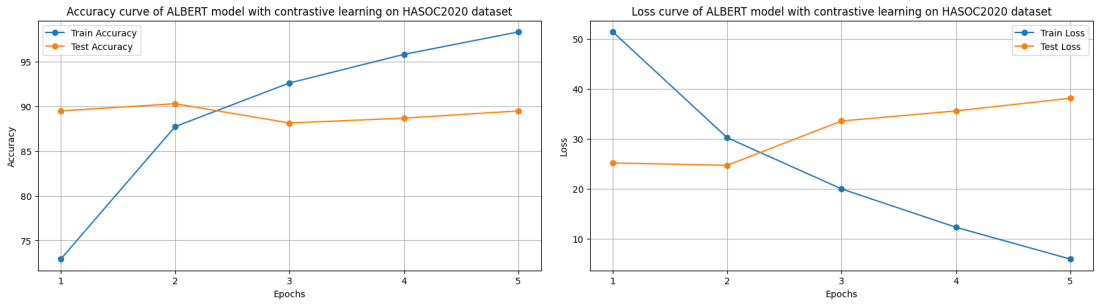
(a) Accuracy curve (b) Loss curve

Figure 5.28: XLM-RoBERTa model on the augmented OLID dataset

Table 5.14: Comparison of f1 scores of models in different settings

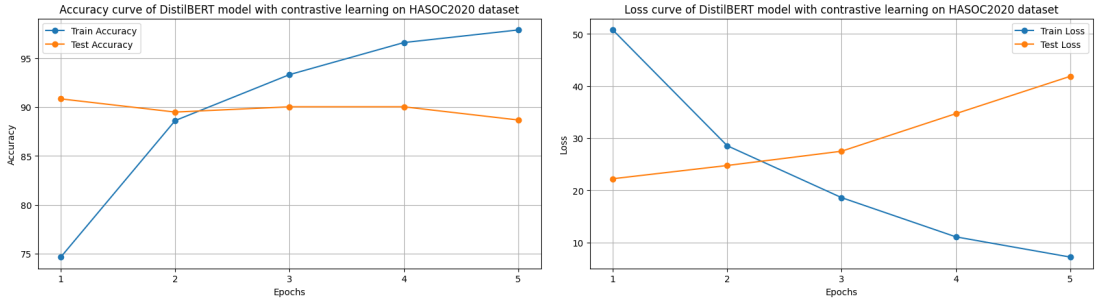
Dataset	**Model**	**Original Dataset**	**Adversarial Attack**	**Contrastive Learning**
HASOC2020	ALBERT	92.78	92.35	93.09
	DistilBERT	92.03	90.40	91.65
	XLM-RoBERTa	90.86	91.92	91.00
OLID	ALBERT	72.52	73.02	73.22
	DistilBERT	72.05	74.13	71.06
	XLM-RoBERTa	73.04	73.15	73.47

It can be seen from table 5.14 that the ALBERT model on the HASOC2020 dataset gained a drastic and noticeable improvement while trained with adversarial samples and contrastive loss. It achieved a f1 score of 93.09%, which is the highest any model has performed on the HASOC2020 dataset so far. It also improved the result of the



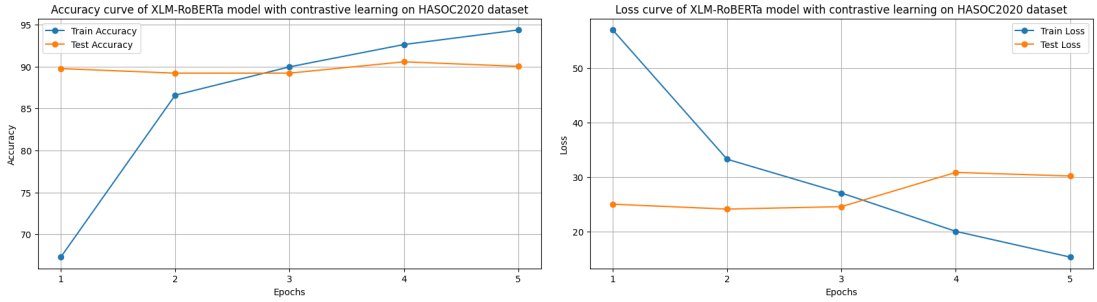
(a) Accuracy curve (b) Loss curve

Figure 5.29: ALBERT model with contrastive loss on HASOC2020 dataset



(a) Accuracy curve (b) Loss curve

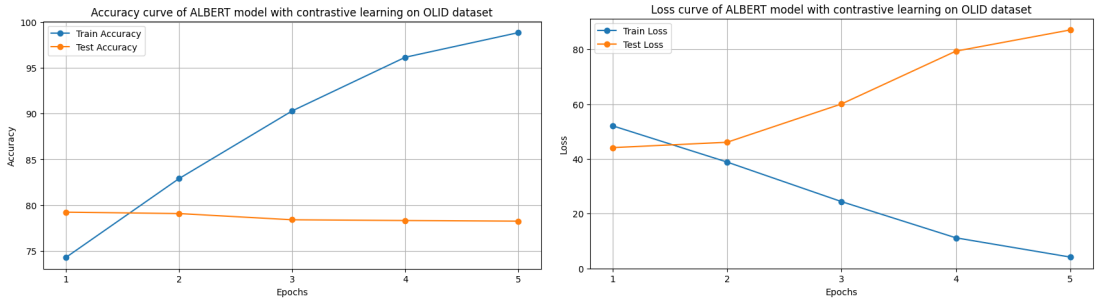
Figure 5.30: DistilBERT model with contrastive loss on HASOC2020 dataset



(a) Accuracy curve (b) Loss curve

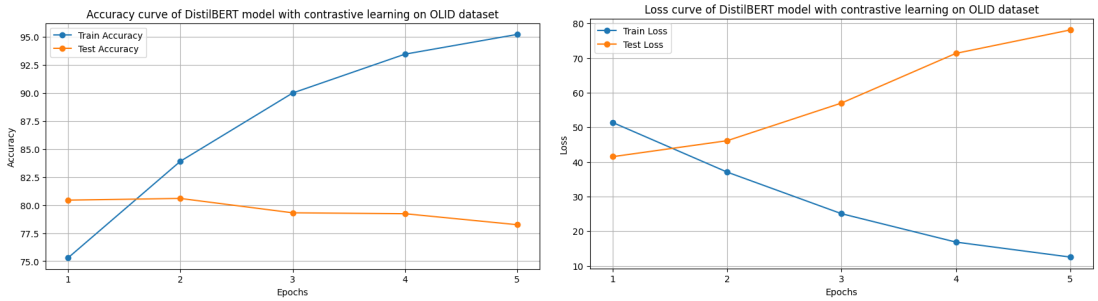
Figure 5.31: XLM-RoBERTa model with contrastive loss on HASOC2020 dataset

XLM-RoBERTa model and reached a f1 score of 91%. For DistilBERT architecture, it improved the result for the adversarial samples by about 1.25%. However, it could not outperform the model trained with the original dataset with no augmented samples in the case of DistilBERT. Table 5.14 also shows that the ALBERT and XLM-RoBERTa models on the OLID dataset were also improved over their corresponding previous



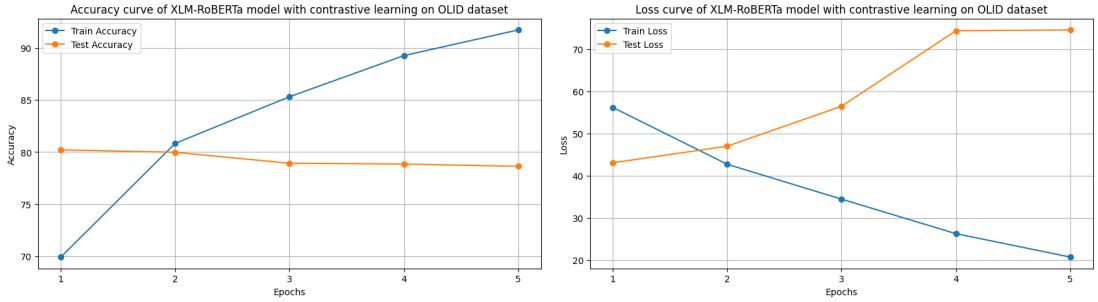
(a) Accuracy curve (b) Loss curve

Figure 5.32: ALBERT model with contrastive loss on OLID dataset



(a) Accuracy curve (b) Loss curve

Figure 5.33: DistilBERT model with contrastive loss on OLID dataset



(a) Accuracy curve (b) Loss curve

Figure 5.34: XLM-RoBERTa model with contrastive loss on OLID dataset

versions. XLM-RoBERTa achieved a f1 score of 73.47, which is the highest any model trained on this dataset gained on this test set. But the performance of DistilBERT dropped. In both cases, DistilBERT models were hampered by the contrastive loss, but ALBERT and XLM-RoBERTa improved. This is an interesting observation.

5.6 Objective Achieved

The first objective of this thesis was to conduct a thorough study of different transformer-based architectures. This thesis studied BERT, ALBERT, DistilBERT, RoBERTa, and XLM-RoBERTa and implemented three of them in the later parts of the thesis. The second objective was to perform a cross-dataset analysis on multiple datasets using multiple model architectures. An extensive cross-dataset analysis was performed on six different datasets using three model architectures in this thesis so far. XLM-RoBERTa achieved the best scores most of the time. The third objective was to apply several types of ensemble modeling techniques. Two different types of ensemble modeling have been performed using the models produced by the cross-dataset analysis. These are - hard-voting and weighted ensembles. This experiment showed notable performance upgradation. Then adversarial samples were produced from the HASOC2020 dataset and the OLID dataset and multiple models were trained on those augmented datasets. The effect of that operation was observed. This was the fourth objective. Finally, contrastive learning with the pull-push mechanism was investigated. A loss function was designed and implemented. The performance of the ALBERT model and the XLM-Roberta model on both datasets were well improved by this process. This was the final objective.

5.7 Financial Analyses and Budget

Transformer-based architectures are very resource-hungry and expensive to implement. Especially, the 'large' versions of the large language models are so massive that they can not be implemented in a normal computer. In this experiment, the base versions of these models were used. Several online and free-to-use GPU-providing platforms such as Kaggle Notebook and Google Colaboratory were used to implement the models. However, even these platforms are not adequate to implement the large versions effectively. To completely fulfill the objectives of this study, advanced transformer-based researches need to be performed with the latest technology available. A computer with a powerful CPU, RAM, SSD, and GPU is needed. As the model size grows bigger and more complex, the capacity and capability of the GPU and CPU will need to be scaled too. A fast and continuous internet connection will need to be maintained.

However, this study used freely available online resources and platforms to conduct the research. So, no additional financial support was required.

5.8 Conclusion

Throughout the experiments, multiple different approaches of hate classification models were implemented and analyzed. Six different datasets were scrutinized using three different model architectures. The OLID dataset showed an outstanding demonstration of its high generalizability. The XLM-RoBERTa model outperformed ALBERT and DitilBERT in many cases. Two types of ensemble modeling approaches were investigated. These ensemble networks improved the results of the individual models in a notable fashion. Then, adversarial samples were generated from two datasets and different models were developed to test the effect on the result. Finally, the contrastive loss was introduced in the adversarially augmented dataset analysis process. The ALBERT and the XLM-RoBERTa models trained with contrastive loss on the augmented datasets displayed considerable improvement.

Chapter 6

Societal, Health, Environment, Safety, Ethical, Legal and Cultural Issues

6.1 Intellectual Property Considerations

If any organization wants to deploy this transformer-based architecture to detect and classify hate speech in text, that organization will have to discuss with the thesis writer. The commercial use of this model in any form or shape must be carried out with the permission of the thesis writer. Also, for obvious reasons, the publicly available parts not originating from this study are not under such consideration.

6.2 Ethical Considerations

This thesis used publicly available hate speech datasets from online. They are used here with proper references and acknowledgments. All the resources regarding the thesis were collected from free-to-use sources. No data or other resources were taken without the consent of the respective owners. All the papers this study mentions are also properly referenced.

6.3 Safety Considerations

In the context of this hate speech detection system, safety considerations were crucial to ensure that the system operated ethically and responsibly while minimizing potential harm to individuals or communities. One safety consideration involved implementing measures to prevent the misidentification or false labeling of non-hateful content as hate speech, which could have led to censorship or suppression of legitimate expression.

Rigorous testing and validation procedures were in place to minimize the risk of false positives and ensure the system's accuracy and fairness across diverse demographics and linguistic contexts. Additionally, safeguards were implemented to protect user privacy and confidentiality, particularly when processing sensitive or personal information to train or deploy the detection model. Moreover, the system incorporated mechanisms for transparent and accountable decision-making, enabling users to understand how hate speech classifications were made and providing avenues for recourse in case of errors or disputes. By addressing these safety considerations, the hate speech detection system contributed to fostering a safer and more inclusive online environment while upholding fundamental principles of freedom of speech and expression.

6.4 Legal Considerations

In the development of a hate speech classification system, legal considerations were paramount to ensure compliance with relevant laws and regulations governing speech and content moderation. One key legal consideration involved adherence to laws protecting freedom of speech, ensuring that the system did not inadvertently censor or suppress lawful expression. Furthermore, it was important to consider liability issues related to the classification of hate speech, including potential legal consequences for false positives or misidentifications. By addressing these legal considerations, the hate speech classification system aimed to navigate legal challenges effectively while upholding ethical principles and contributing to a safer and more inclusive online environment. Integrating legal considerations into project planning and execution from the outset helped mitigate risks and ensured the responsible conduct of the research endeavor.

6.5 Impact of the Project on Societal, Health, and Cultural Issues

6.5.1 Societal Impact

In the context of a hate speech classification system, the societal issue it directly addressed was the proliferation of hate speech and its detrimental effects on individuals and communities. Hate speech contributed to the perpetuation of discrimination, intolerance, and violence against marginalized groups, exacerbating societal divisions and hindering efforts toward inclusivity and equality. By developing a hate speech classification system, the project aimed to mitigate this societal issue by providing a tool for identifying and addressing hate speech online. This endeavor sought to promote a safer and more respectful online environment, fostering constructive dialogue and promoting social cohesion. By combatting hate speech, the classification system contributed

to broader societal goals of promoting tolerance, respect, and equal treatment for all individuals, regardless of their background or identity.

6.5.2 Health Impact

The hate speech detection system will have a significant impact on mental health outcomes by providing a safer online environment for individuals and communities. Hate speech can contribute to feelings of fear, anxiety, and distress, particularly among targeted groups such as minorities, and marginalized communities. By accurately identifying and mitigating hate speech online, the detection system will help reduce exposure to harmful content, thereby mitigating the negative psychological effects associated with online harassment and discrimination. Additionally, by fostering a more inclusive and respectful online environment, the system will promote mental well-being and contribute to the overall mental health of internet users. Overall, the hate speech detection system will play a crucial role in safeguarding mental health and promoting a more positive online experience for individuals and communities in the future.

6.5.3 Cultural Impact

In the context of cultural impact, a hate speech detection system can play a vital role in preserving cultural diversity and promoting inclusivity within digital spaces. Hate speech targeted at specific cultural or ethnic groups can erode cultural identity, perpetuate stereotypes, and undermine the sense of belonging among affected communities. By accurately identifying and mitigating hate speech online, the detection system can help protect the cultural heritage of diverse communities and foster a more respectful and inclusive online environment. This contributes to the preservation of cultural identity and promotes a sense of belonging among individuals from marginalized or targeted cultural groups. Furthermore, by combating hate speech, the system supports efforts to promote cultural appreciation, understanding, and tolerance, thus fostering greater cultural diversity and harmony within society. Overall, the cultural impact of a hate speech detection system lies in its ability to safeguard cultural heritage, promote inclusivity, and foster a more respectful and harmonious digital space for individuals of all cultural backgrounds.

6.6 Impact of Project on the Environment and Sustainability

6.6.1 Environmental Impact

In the context of environmental impact, a hate speech detection system may indirectly contribute to environmental conservation efforts by fostering a more positive online environment. While the direct environmental impact of such a system may be minimal, its role in promoting respectful and constructive online discourse can have broader societal implications that indirectly benefit the environment. Hate speech and online toxicity can contribute to social tensions, conflict, and division, which may divert attention and resources away from environmental conservation efforts. By mitigating hate speech and promoting a more harmonious online community, the detection system can help redirect focus towards environmental issues and encourage collective action towards sustainability. Additionally, a more inclusive and respectful online environment may foster greater collaboration and cooperation among individuals and organizations working towards environmental conservation goals. Overall, while the environmental impact of a hate speech detection system may be indirect, its role in promoting social cohesion and collaboration can contribute to broader efforts towards environmental sustainability and conservation.

6.6.2 Sustainability Impact

In the context of sustainability impact, a hate speech detection system may contribute to fostering a more inclusive and equitable society, which aligns with broader sustainability goals. Hate speech and discriminatory behavior can create social divisions and hinder efforts toward building sustainable communities. By identifying and mitigating hate speech online, the detection system helps promote a culture of respect, tolerance, and inclusion, thereby contributing to social cohesion and resilience. Sustainable development encompasses not only environmental considerations but also social and economic aspects, with a focus on creating thriving and equitable communities. A hate speech detection system supports this by fostering a digital environment where individuals from diverse backgrounds feel safe and valued, promoting social well-being and community resilience.

6.6.3 Waste Reduction

A hate speech detection system may indirectly contribute to promoting a more sustainable and environmentally friendly society. While the direct impact of such a system on waste reduction may not be evident, its role in fostering a culture of respect, empathy,

and collaboration can lead to broader societal changes that support waste reduction efforts. Hate speech and discriminatory behavior can create social divisions and barriers to cooperation, hindering collective action on environmental issues, including waste management and recycling initiatives. By identifying and mitigating hate speech online, the detection system helps promote social cohesion and community resilience, fostering an environment where individuals are more willing to collaborate and participate in waste reduction and recycling programs.

Chapter 7

Addressing Complex Engineering Problems and Activities

7.1 Complex engineering problems associated with the current thesis

Addressing complex engineering problems associated with transformer-based hate speech detection, adversarial sample analysis, and contrastive learning requires a combination of technical expertise, algorithmic innovation, and ethical considerations to develop effective and responsible solutions for mitigating online hate speech. The crucial issues of complex engineering problems such as depth of knowledge required, depth of analysis required, familiarity of issues, extent of applicable codes, and interdependence are applied to this study. Table 7.1 discusses these issues with a detailed explanation.

Table 7.1: Complex Engineering problems associated with the current thesis

Attribute	Addressing the Attributes of Complex Engineering Problems	
Depth of knowledge required	P1	This study needed a deep understanding of transformer-based architectures, large language models and their application. Many core concepts of NLP were explored and experimented with. The internal mechanism of self-attention, deep neural networks, embedding space, etc are required to be thoroughly understood to perform this thesis.
Range of conflicting requirements	P2	Not applicable
Depth of analysis required	P3	The analysis needs an in-depth understanding of machine learning algorithms and functionalities. The internal process of text analysis must be extensively studied to perform this experiment. The model output characteristic curves need to be properly observed and the result must be thoroughly investigated to assess the model performance correctly. Reaching right conclusions by analyzing evaluation metrics is a must for this thesis.
Familiarity of issues	P4	Combining adversarial sampling analysis and contrastive learning mechanism to deal with the randomity and variations of hate speech is a relatively newer approach in the hate speech classification domain. These techniques are not that familiar in the research fields in this regard.
Extent of applicable codes	P5	The application of the codes behind this experimentation can be extended to a variety of use cases such as flagging offensive comments in social media networks, news websites, online gaming communities, educational platforms, content moderation systems, etc. Suppose there is a need for an automatic hate speech classification system in any domain. In that case, the code behind this study can be briefly modified and applied.
Extent of stakeholder involvement and conflicting requirements	P6	Not applicable
Interdependence	P7	This thesis contained 4 steps of analysis. Each of the states needed another part to properly work. If the transformer-based models can not produce good outputs, ensembling them won't bring any success. Again, without producing well-thought adversarial examples, contrastive learning can not be applied effectively either. They are inherently dependent on each other.

7.2 Complex engineering activities associated with the current thesis

Addressing complex engineering activities associated with transformer-based hate speech, adversarial sample analysis, and contrastive learning detection is a significant thing to discuss. The crucial issues of complex engineering activities such as the range of resources, consequences for society and the environment, and familiarity are applied to perform this study. Table 7.2 discusses these issues with a detailed explanation.

Table 7.2: Complex Engineering activities associated with the current thesis

Attribute	Addressing the Attributes of Complex Engineering Activities	
Range of resources	A1	To conduct this study, a total of six datasets were used, Three different types of model architectures were applied. More than fifty research papers were extensively analyzed. Also, to develop the backbone code for this thesis, a wide range of programming resources were explored. Multiple online GPU systems were needed to efficiently perform the experiments. So, in a nutshell, the range of resources behind this study is considerably wide.
Level of interaction	A2	Not applicable
Innovation	A3	Applying contrastive loss mechanism on transformer-based models trained with adversarially augmented dataset to improve the classifiers' performance is a new and innovative approach in hate speech classification domain.
Consequences for society and the environment	A4	If the result of this thesis is applied in the real-time online environment, it can bring a very positive change in online interactions. If the hate speech classification models are trained in the experimented mechanism, the applied domain will see a considerable decrease in the use of hateful remarks. This will positively affect the society and environment.
Familiarity	A5	Combining adversarial sampling analysis and contrastive learning mechanism to deal with the randomity and variations of hate speech is a relatively newer approach in the hate speech classification domain. These techniques are not that familiar in the research fields in this regard.

Chapter 8

Conclusions

8.1 Summary

Throughout the study, multiple different approaches of hate classification models were implemented and analyzed. Six different datasets were extensively scrutinized. These were the Agarwal dataset, the MHS dataset, the HASOC2020 dataset, the Vidgen dataset, the Jigsaw dataset, and the OLID dataset. A cross-dataset analysis was performed using three model architectures - ALBERT, DistilBERT, and XLM-RoBERTa. The OLID dataset showed an outstanding demonstration of its high generalizability. The XLM-RoBERTa model often outperformed ALBERT and DitilBERT, displaying its superior computational ability. Two types of ensemble modeling approaches were investigated - hard-voting and weighted ensembles. These ensemble networks improved the results of the individual models in a notable fashion. Then, adversarial samples were generated from the HASOC2020 and the OLID datasets and multiple models were developed using that to test the effect on the result. Finally, the contrastive loss was introduced in the adversarially augmented datasets analysis process. The ALBERT and the XLM-RoBERTa models trained with the contrastive loss on the augmented datasets improved in the contrastive learning experiment.

8.2 Limitations

Due to resource constraints, only the base versions of different large language models were used. The larger versions of these large language models have more parameters, more layers, and more computational power. Naturally, they would produce better results than the base versions. Most of the state-of-the-art models were produced using these large versions of them. So, the result this thesis achieved might not have surpassed the

state-of-the-art models due to this limitation.

Even the base versions of these large language models are so large that they occupy most of the GPU memory. As a result, during implementation, many problems were faced. The batch size had to be kept within 100. Maximum sequence length needed to be kept within 128, coinciding with the batch size. During the ensemble, multiple models couldn't be trained at the same time, instead, they were trained independently. The shared learning approach couldn't be explored. They were time-consuming too. While training the models on the MHS dataset, containing over 135 thousand samples, each epoch took 45 minutes to complete.

The datasets used here mostly contain explicit hate speeches. Explicit hate speeches are comparatively easier to detect and understand. But, nowadays, implicit hate speeches are seen in many cases. These implicit hate speeches can not be captured without training the model with proper examples. The experimented models may fail to capture such content in real-life scenarios. Also, there are many more publicly available datasets on different online platforms. But, due to time constraints, this thesis only worked with six of them. If it was possible, some more datasets should have been included in this study.

8.3 Recommendations and Future Works

A good hate speech classification model should have the ability to tackle any variations of hate speech within its domain and demographic. It should also be able to perform at least reasonably on other domains instead of getting completely broken down. How prone a model is to adversarial attacks is also important. Without properly training the model to understand and recognize the variations of the data, it can not be made to perform well in real-life scenarios. So, the adversarial training approach and contrastive learning are promising research areas in this regard. Since they work on teaching the model these variations and randomness in a controlled environment, they offer a lot of creative opportunities to research. To improve the dataset characteristics, machine-generated samples can be considered too. Implicit hate speech detection is another promising research area. As the dataset nature is more significant than model architectures, the focus should be made more on improving the formulation process of the datasets than other parts of the process.

References

[1] F. M. Plaza-Del-Arco, M. D. Molina-González, L. A. Ureña-López, and M. T. Martín-Valdivia, "A multi-task learning approach to hate speech detection leveraging sentiment analysis," *IEEE Access*, vol. 9, pp. 112 478–112 489, 2021.

[2] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PloS one*, vol. 14, no. 8, p. e0221152, 2019.

[3] "Meta policy on hate speech." [Online]. Available: https://transparency.fb.com/en-gb/policies/community-standards/hate-speech/

[4] "Twitter's policy on hateful content." [Online]. Available: https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

[5] T. Gröndahl, L. Pajola, M. Juuti, M. Conti, and N. Asokan, "All you need is" love" evading hate speech detection," in *Proceedings of the 11th ACM workshop on artificial intelligence and security*, 2018, pp. 2–12.

[6] A. Rodríguez, C. Argueta, and Y.-L. Chen, "Automatic detection of hate speech on facebook using sentiment and emotion analysis," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019, pp. 169–174.

[7] A. K. Das, A. A. Asif, A. Paul, and M. N. Hossain, "Bangla hate speech detection on social media using attention-based recurrent neural network," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 578–591, 2021. [Online]. Available: https://doi.org/10.1515/jisys-2020-0060

[8] F. Del Vigna12, A. Cimino23, F. Dell'Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook," in *Proceedings of the first Italian conference on cybersecurity (ITASEC17)*, 2017, pp. 86–95.

[9] S. Zimmerman, C. Fox, and U. Kruschwitz, "Improving hate speech detection with deep learning ensembles," 05 2018.

[10] N. Alswaidan and M. E. B. Menai, "A survey of state-of-the-art approaches for emotion recognition in text," *Knowledge and Information Systems*, vol. 62, pp. 2937–2987, 2020.

[11] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–30, 2018.

[12] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the fifth international workshop on natural language processing for social media*, 2017, pp. 1–10.

[13] Z.-J. Chuang and C.-H. Wu, "Multi-modal emotion recognition from speech and text," in *International Journal of Computational Linguistics & Chinese Language Processing, Volume 9, Number 2, August 2004: Special Issue on New Trends of Speech and Language Processing*, 2004, pp. 45–62.

[14] N. A. Setyadi, M. Nasrun, and C. Setianingsih, "Text analysis for hate speech detection using backpropagation neural network," in *2018 international conference on control, electronics, renewable energy and communications (ICCEREC)*. IEEE, 2018, pp. 159–165.

[15] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proceedings of the 26th international conference on World Wide Web companion*, 2017, pp. 759–760.

[16] Y. Zhou, Y. Yang, H. Liu, X. Liu, and N. Savage, "Deep learning based fusion approach for hate speech detection," *IEEE Access*, vol. 8, pp. 128 923–128 929, 2020.

[17] N. Romim, M. Ahmed, H. Talukder, and M. Saiful Islam, "Hate speech detection in the bengali language: A dataset and its baseline evaluation," in *Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2020*. Springer, 2021, pp. 457–468.

[18] P. K. Roy, A. K. Tripathy, T. K. Das, and X.-Z. Gao, "A framework for hate speech detection using deep convolutional neural network," *IEEE Access*, vol. 8, pp. 204 951–204 962, 2020.

[19] M. M. H. Manik, F. Haque, M. Hashem, M. A. Habib, M. Z. Islam, and T. Ahmed, "A hybrid framework for sentiment analysis from bangla texts," in *2022 25th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2022, pp. 517–522.

[20] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: a comprehensive review," *ACM computing surveys (CSUR)*, vol. 54, no. 3, pp. 1–40, 2021.

[21] P. Kapil and A. Ekbal, "A deep neural network based multi-task learning approach to hate speech detection," *Knowledge-Based Systems*, vol. 210, p. 106458, 2020.

[22] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proceedings of the first workshop on abusive language online*, 2017, pp. 85–90.

[23] S. Dowlagar and R. Mamidi, "Hasocone@ fire-hasoc2020: Using bert and multilingual bert models for hate speech detection," *arXiv preprint arXiv:2101.09007*, 2021.

[24] U. Naseem, I. Razzak, and I. A. Hameed, "Deep context-aware embedding for abusive and hate speech detection on twitter." *Aust. J. Intell. Inf. Process. Syst.*, vol. 15, no. 3, pp. 69–76, 2019.

[25] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the international AAAI conference on web and social media*, vol. 11, no. 1, 2017, pp. 512–515.

[26] J. Golbeck, Z. Ashktorab, R. O. Banjo, A. Berlinger, S. Bhagwan, C. Buntain, P. Cheakalos, A. A. Geller, R. K. Gnanasekaran, R. R. Gunasekaran *et al.*, "A large labeled corpus for online harassment research," in *Proceedings of the 2017 ACM on web science conference*, 2017, pp. 229–233.

[27] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[28] L. Yuan, T. Wang, G. Ferraro, H. Suominen, and M.-A. Rizoiu, "Transfer learning for hate speech detection in social media," *arXiv preprint arXiv:1906.03829*, 2019.

[29] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A bert-based transfer learning approach for hate speech detection in online social media," in *Complex Networks and Their Applications VIII: Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019 8*. Springer, 2020, pp. 928–940.

[30] R. T. Mutanga, N. Naicker, and O. O. Olugbara, "Hate speech detection in twitter using transformer methods," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020.

[31] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[32] S. S. Sabry, T. Adewumi, N. Abid, G. Kovács, F. Liwicki, and M. Liwicki, "Hat5: hate language identification using text-to-text transfer transformer," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[33] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," 2019.

[34] M. R. Karim, S. K. Dey, T. Islam, S. Sarker, M. H. Menon, K. Hossain, M. A. Hossain, and S. Decker, "Deephateexplainer: Explainable hate speech detection in under-resourced bengali language," in *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2021, pp. 1–10.

[35] T. Alam, A. Khan, and F. Alam, "Bangla text classification using transformers," *arXiv preprint arXiv:2011.04446*, 2020.

[36] J. A. García-Díaz, S. M. Jiménez-Zafra, M. A. García-Cumbreras, and R. Valencia-García, "Evaluating feature combination strategies for hate-speech detection in spanish using linguistic features and transformers," *Complex & Intelligent Systems*, vol. 9, no. 3, pp. 2893–2914, 2023.

[37] S. G. Roy, U. Narayan, T. Raha, Z. Abid, and V. Varma, "Leveraging multilingual transformers for hate speech detection," *arXiv preprint arXiv:2101.03207*, 2021.

[38] P. Alonso, R. Saini, and G. Kovács, "Hate speech detection using transformer ensembles on the hasoc dataset," in *International conference on speech and computer*. Springer, 2020, pp. 13–21.

[39] K. Mnassri, P. Rajapaksha, R. Farahbakhsh, and N. Crespi, "Hate speech and offensive language detection using an emotion-aware shared encoder," *arXiv preprint arXiv:2302.08777*, 2023.

[40] A. C. Mazari, N. Boudoukhani, and A. Djeffal, "Bert-based ensemble learning for multi-aspect hate speech detection," *Cluster Computing*, pp. 1–15, 2023.

[41] Y. Khan, W. Ma, and S. Vosoughi, "Lone pine at semeval-2021 task 5: fine-grained detection of hate speech using bertoxic," *arXiv preprint arXiv:2104.03506*, 2021.

[42] D. Antypas and J. Camacho-Collados, "Robust hate speech detection in social media: A cross-dataset empirical evaluation," 2023.

[43] T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar, "Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection," *arXiv preprint arXiv:2203.09509*, 2022.

[44] K. Madukwe, X. Gao, and B. Xue, "In data we trust: A critical analysis of hate speech detection datasets," in *Proceedings of the fourth workshop on online abuse and harms*, 2020, pp. 150–161.

[45] Y. Kim, S. Park, and Y.-S. Han, "Generalizable implicit hate speech detection using contrastive learning," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 6667–6679.

[46] Q. Chen, R. Zhang, Y. Zheng, and Y. Mao, "Dual contrastive learning: Text classification via label-aware data augmentation," *arXiv preprint arXiv:2201.08702*, 2022.

[47] J. Lu, H. Lin, X. Zhang, Z. Li, T. Zhang, L. Zong, F. Ma, and B. Xu, "Hate speech detection via dual contrastive learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[48] L. Pan, C.-W. Hang, A. Sil, and S. Potdar, "Improved text classification via contrastive adversarial training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 11 130–11 138.

[49] S. Choi, M. Jeong, H. Han, and S.-w. Hwang, "C2l: Causally contrastive learning for robust text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 526–10 534.

[50] A. F. Adoma, N.-M. Henry, and W. Chen, "Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2020, pp. 117–121.

[51] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018. [Online]. Available: http://arxiv.org/abs/1802.05365

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[53] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[54] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[55] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[56] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[57] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[58] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[59] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[60] P. Fortuna, J. Soler, and L. Wanner, "Toxic, hateful, offensive or abusive? what are we really classifying? an empirical analysis of hate speech datasets," in *Proceedings of the 12th language resources and evaluation conference*, 2020, pp. 6786–6794.

[61] C. J. Kennedy, G. Bacon, A. Sahn, and C. von Vacano, "Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application," *arXiv preprint arXiv:2009.10277*, 2020.

[62] P. Sachdeva, R. Barreto, G. Bacon, A. Sahn, C. Von Vacano, and C. Kennedy, "The measuring hate speech corpus: Leveraging rasch measurement theory for data perspectivism," in *Proceedings of the 1st Workshop on Perspectivist Approaches to NLP@ LREC2022*, 2022, pp. 83–94.

[63] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, "Learning from the worst: Dynamically generated datasets to improve online hate detection," 2021.