Identifying code-mixed and code-switched hateful remarks on Social Media using NLP

by

Sumaiya Sinha
20101141
Naharin Siddiqui Nawar
24141298
Md. Abrar Faiaz Khan
19301106

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.
5. Readers' dicretion is advised.

Student's Full Name & Signature:

Naharin

Naharin Siddiqui Nawar
20101369

Sumaiya

Sumaiya Sinha
20101141

Md. Abrar Faiaz Khan
19301106

Approval

The thesis/project titled "Identifying code-mixed and code-switched hateful remarks on Social Media using NLP" submitted by

1. Sumaiya Sinha (20101141)

2. Naharin Siddiqui Nawar (20101369)

3. Md. Abrar Faiaz Khan (19301106)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 2024.

Examining Committee:

Supervisor:
(Member)

Farig Yousuf Sadeque

Farig Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Co-supervisor:
(Member)

Rahman

Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Thesis Coordinator)

____

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

____

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Online bullying has prevailed for years in the vast cesspool that is commonly known as the online social media. Increasing use of social media and online communication has led to a rise in cyberbullying– which is often facilitated by the abundant usage of code-mixing and code-switching. Research has been done to filter out these derogatory remarks. However, little research has been done on code-switched and code-mixed hateful remarks. English has blended into our Bangla language so effectively that people regularly use English letters to convey Bangla due to its convenience. English and Bangla are used interchangeably in regular conversations as well. Our main objective in this research is to detect these code-switched and code-mixed remarks– which we plan to do by taking advantage of the state-of-the-art natural language processing technologies.

Keywords: Online bullying, Cyberbullying, Social media, Code-mixing, Code-switching, Derogatory remarks, Hateful remarks, Natural language processing, Detection.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent times, the internet and social media provided individuals with unlimited opportunities to share and express their opinions, engage in discussions and connect with people globally. It has created an impact in both good and bad ways. The good thing is, we can now connect to people easily. We can share our experiences, knowledge, and thoughts with people effortlessly and many more. While the internet and social media have many good aspects, it also has many drawbacks, and hate speech is one of them. In general, hate speech indicates offensive speech or language that marks an individual or a class according to their basic characteristics like sexual orientation, religious belief, ethnicity, sexual preferences, and so on. It may harm the person or the group and can menace social peace. Hate speech is defined by the UN Strategy and Plan of Action on Hate Speech as any form of language that focuses or applies demeaning or biased phrases towards an individual or community on grounds of their beliefs, cultural background, the color of skin, gender, origin, or other characteristics. It provides a structure for the United Nations to point out the problem globally. In the era of the internet and online networking, hate speech is an increasingly prominent problem. People nowadays use different kinds of offensive language on different social media sites, social media posts, comment sections, messaging apps, and others. Hate speech can originate in many forms. Some common types of hate speech are threats, online trolling, harassment, derogatory language, etc. Sexism-related hate speech is another form of that. In recent years, the issue of sexism has extended its reach into the digital world as well. Sexism can be seen in online communication platforms and different social media sites. It has become really hard to detect sexism-related hate speech now because there is a lot of data on the internet that are either code-switched or code-mixed or both .

Data that is code-switched and code-mixed refers to mixing multiple languages within conversations or speech. This mixture of languages occurs in multilingual communities or individuals who are capable of speaking and knowing multiple languages. Various accents are employed in a single statement or speech when data is code-mixed. For instance, “I want to buy sabzi from the market”. Here, “sabzi” is a Hindi word that is mixed into an English sentence. Here, we can see a mixture of Hindi - English languages. This sentence is an example of code-mixed data. On the other hand, code-switched data means shifting from one language to another during a conversation. For example, if someone writes, “Voy a la tienda to buy some groceries”. Here, “Voy a la tienda” is a Spanish sentence that means I am going to the store. We can see the shift from Spanish to English language in the sentence. It is an example of code-switched data.

Nowadays the code-mixed and code-switched data has increased so much that it is really a complex task to detect and reduce sexism-related hate speech. These linguistic incidents are very common in today's era where people effortlessly mix languages to express their feelings, thoughts, and cultural identities. To give an example, let's assume there is a Twitter thread that is discussing women's rights which is a trending topic. There are many participants from various countries with different linguistic backgrounds. In such discussions, contributors may switch their languages like English, Hindi, and Bengali to express their thoughts, opinions and personal experiences. However, amidst this linguistic diversity, people may target women by passing sexist comments disguised within the code-mixed and code-switched statement. Identifying and addressing these instances becomes crucial for fostering an inclusive and respectful online environment.

Traditional approaches to hate speech detection predominantly focus on monolingual and single-language contexts. We can understand that with an example. Suppose, an English speaker who also knows Spanish. He wants to make a sexist comment while involved in an online conversation. Now, a monolingual model is trained on English data. It does not understand Spanish language. So, if he makes a sexist comment in Spanish, the model won't be able to detect it. We can see another example where a participant writes, "Women should stay at home. They cannot handle a leadership position. Netritto shudhu chelera dite pare, meyera noy". Which means "only men can lead, not women". This is a case of code-switched situation where the participant makes a sexist comment while switching his language from English to Bangla. This is very difficult for a monolingual model to detect and classify hateful remarks from code-mixed and code-switched data. Machines face significant challenges in adapting to the intricacies of code-switched and code-mixed data. The incorporation of multiple languages introduces linguistic variations, cultural references, and idiomatic expressions that may escape detection by existing systems. Analyzing instances that have code-mixed and code-switched data needs a deep understanding of language-specific cues, cultural context, and the ability to detect hateful intent within the interplay of languages. So, our research aims to identify sexism-related hate speech in Bengali code-mixed and code-switched data often found in social media posts and comments.

1.1 Research Statement

The primary objective of our thesis is to address the pressing issue of detecting and mitigating hate speech targeting sexism within Bengali code-switched and code-mixed data, considering the complex linguistic and cultural dynamics associated with these communication patterns. Code-switching and code-mixing which are very prevalent in diverse multilingual societies pose significant challenges in accurately identifying and effectively combating instances of hate speech. Hate speech, particularly when intertwined with sexism, perpetuates harmful stereotypes, reinforces discrimination, and creates an environment of hostility, exclusion, and potential harm, eroding the principles of equality, dignity, and respect within digital spaces.

Existing methodologies and tools for hate speech detection predominantly cater to monolingual and single-language contexts, lacking the adaptability, nuance, and contextual sensitivity required to navigate the intricate linguistic and cultural variations inherent in code-switched and code-mixed data. Moreover, the scarcity of adequately annotated datasets specifically curated for code-switched and code-mixed hate speech hampers the development and refinement of robust machine learning algorithms, natural language processing models, and computational linguistic approaches in this critical domain. These datasets are crucial for training and evaluating hate speech detection systems, as they provide the necessary annotations to guide the learning process and enhance the system's ability to recognize and classify instances of hate speech within code-switched and code-mixed data.

Thus, there is an urgent need to explore and develop innovative methodologies and approaches that effectively address the detection and mitigation of hate speech, specifically targeting sexism, within Bengali code-switched and code-mixed data. This research endeavor aims to bridge this critical gap by proposing novel techniques that leverage the complex linguistic and cultural characteristics of Bengali code-switched and code-mixed data. By incorporating linguistic features, cultural references, contextual clues, and language-specific nuances, the proposed approaches aim to enhance the accuracy, comprehensiveness, and contextual awareness of hate speech detection systems.

By advancing progressive machine learning algorithms, natural language processing models, and computational linguistic methods, this research seeks to add to the conception of more polished, nuanced, and adaptable systems that effectively detect sexist speech within Bengali code-switched and code-mixed digital communication platforms. Fundamentally, the discernments derived from this research aspire to nurture a more inclusive, respectful, and equitable digital landscape, advocating a culture of tolerance, diversity, and safety, thereby mitigating the detrimental impacts of hate speech and discrimination.

1.2 Research Objective

The main objectives of this research are to:

1. Analyze and identify the unique cultural and linguistic features of information that have been code-mixed and code-switched in the context of sexism, to gain insights into the manifestation and complexities of sexist language in multilingual communities.

2. Recognize and categorize the many forms of sexism included in code-mixed and code-switched data, such as openly insulting comments, hidden gender biases, stereotypical portrayals, and microaggressions, we may create a full taxonomy that encompasses the spectrum of sexist utterances.

3. Develop and curate a labeled dataset specifically tailored for code-mixed and code-switched sexist language, encompassing various language combinations, cul-

tural contexts, and domains, to provide a valuable resource for training and evaluating sexism detection models.

4. Explore and adapt existing machine learning algorithms, natural language processing techniques, and computational linguistic models to effectively detect instances of sexism in code-mixed and code-switched data, accounting for linguistic variations, cultural references, and the dynamic nature of language.

5. Enhance the accuracy and inclusivity of sexism detection systems by integrating contextual information, cultural sensitivity, and domain-specific knowledge into the detection algorithms, aiming to minimize false positives and false negatives while accounting for cultural nuances and the intended meaning within utterances that are either code-switched, or code mixed, or both.

The goal of this study is to improve sexism detection methods in code-switched and code-mixed information. By achieving these research goals, we can help create inclusive and equitable digital platforms and raise awareness of the negative effects of sexist language in online settings.

Chapter 2

Literature Review

We have divided our research topic into two parts. In the first phase, we will analyze the existing literature on hate speech. Cambridge dictionary defines hate speech as "public speech that expresses hate or encourages violence toward a person or group based on something such as race, religion, sex, or sexual orientation " [21]. Since hate speech covers multiple areas, we will concentrate only on a single type that is sexism. In the second part, we will analyze relevant code mixed and code switched papers.

2.1 Hate Speech Literature

Waseem, Z. & Hovy, D. (2016, June) recognize the importance of detecting hate speech in social media sites [1]. They believe that hate speech can incite physical violence. The authors used critical race theory to annotate more than 16k tweets. The tweets were divided into three categories, sexist, racist and neither of them. For this research, the authors compared the performance of character n-grams (sequence of characters) with word n-grams (sequence of words). They were able to find that the former outperforms the latter by using a bootstrap sampling test. So, they used character n-grams as their primary feature. This research came to the conclusion that character n-grams along with gender as a feature gives the best results. On the other hand, length of hate speech and its location brings insignificant improvement to results. The readers must keep in mind that the geographic distribution did not cover a wide range. So, there is still room for improvement.

In another paper, Waseem, Z. (2016, November) aims to detect hateful discourse on Twitter without bias by using various machine learning models and analyzing the impact from the perspective of human annotators [2]. The study collected a dataset of tweets containing racism, sexism and neither of them. We must be aware that the new dataset was created based on a sample from the original dataset released by Waseem and Hovy (2016) [1]. The new dataset was then labeled as hate speech or not by experts (feminist and anti-racism activists) and amateur annotators. The study found significant disagreement among the annotators due to the context-dependent nature of hate speech. Hate speech does not have a universal definition. The definition changes based on culture, geographical distribution and so on. The best-performing features for detection were token unigrams, cluster detection, length analyzer, and sarcasm detection. Machine learning models trained on annotations from multiple annotators performed better than those trained on a single annotator, highlighting the importance of considering human bias and variability when developing and evaluating hate speech detection models.

The next article acknowledges its predecessors and agrees with the fact that hate speech has a context dependent nature which makes it difficult for non-experts to annotate without having proper knowledge of the domain. The authors used a one-step and a two-step classification approach to detect abusive language containing racism and sexism [3]. The one-step approach trains a single classifier to identify tweets, while the two-step approach first labels potentially abusive tweets using a binary classifier and then labels them as either abusive or non-abusive using another classifier. The study uses three different types of CNN models, CharCNN, WordCNN, and HybridCNN, to classify a large dataset of 20 thousand tweets. The two-step approach outperforms the one-step approach in terms of F1 score and is better at avoiding false positives, while the one-step approach is better at identifying borderline cases.

Another research paper acknowledges that differentiating hate speech from other offensive language is a challenge. In order to build the dataset, the authors used a website called Hatebase.org to collect a list of terms and phrases that have been classified as hate speech online [4]. Then they use Twitter API to look for tweets that contain words from their original list. Overall, they were able to gather 85.4 million tweets. They then selected a random sample of 25k tweets from this corpus and had CrowdFlower (CF) employees manually code them. Each tweet was given a classification in one of three categories: hate speech, offensive but not hate speech, or neither offensive nor hate speech. In order to comprehend the tweets, the researchers employed a variety of techniques. They experimented with several data analysis techniques to see which one was most effective. They discovered that linear SVMs and logistic regression were the most efficient. These techniques were able to detect if a tweet contained hate speech, offensive language, or neither. They used all of their collected tweets to train their final model. Logistic regression with L2 regularization was used as a unique methodology to estimate the type of language. The model that performed the best overall has an F1 score of 0.90, a recall of 0.90, and a precision of 0.91. Just like the articles [2] and [3], the researchers concluded that hate speech is difficult to identify since each person has their own definition of what constitutes hate speech. For instance, people see homophobic and racist slurs as being hateful but might overlook sexist language. The future works thus need to address these biases.

The previous works have focused on detecting hostile sexism. However, according to Jha, A., Mamidi, R. (2017, August), sexism comes in two forms Hostile and Benevolent. Here, the primary aim of the researchers was to investigate the form of sexism that normally goes unnoticed [5]. An example of benevolent sexism could be ‘They’re probably surprised at how smart you are, for a girl’. Benevolent sexism can be subtle but dangerous. It can negatively impact women’s cognitive performance and create gender inequality.In order to build their extensive dataset, the researchers gathered data from various sources. They basically collected tweets which fell into three categories of sexism which are ‘hostile’, ‘benevolent’ and ‘others’. They made use of the Twitter Search API to collect tweets that fall under the ‘benevolent’ category. For the other two categories, they used the public corpus published by Waseem and Hovy (2016) [1]. Glick and Fisk (2019) proposed a groundbreaking ambivalent sexism theory. This theory was used by the authors to identify and annotate benevolent sexism [10]. The theory covers three important concepts. The concepts are paternalism, gender differentiation and heterosexuality. If we dive deeper into paternalism, we get to notice that it has two sides. One is the dominative paternalism which sees women with hostility. Consequently, viewing them as inept. The other kind is protective paternalism. This is a bit interesting. Even though this type is not exactly hostile, it normally concludes that women are to be cherished because they are weak. An example sentence could be “Women are like flowers who need to be cherished!”. In this way both gender differentiation and heterosexuality can have a hostile and a benevolent side. The authors utilized the FastText classifier, SVM, and sequence-to-sequence models to classify the tweets into their respective categories. They used the FastText classifier to get the best F1-score.

In a more recent paper than the preceding ones, authors Rodríguez-Sánchez, F., Carrillo-de-Albornoz, J., Plaza, L. (2020) talk about the need for effective detection technologies to identify hate speech online. In their work, they primarily treat hate speech as being any form of derogatory content [6]. Simply, hate speech is described as any form of communication that disparages people or groups based on attributes including gender, sexual orientation, race, nationality, or religion. They used two publicly accessible datasets that have been annotated for racist, sexist, hateful, or other offensive content on Twitter to assess their suggested methodology. In order to improve hate speech detection, the researchers used a pre-trained language model called BERT that learned from English Wikipedia and BookCorpus. Using publicly available datasets, they proposed a transfer learning approach on BERT. They also introduced new fine-tuning methodologies to BERT. Through this, they were able to get experiment results that outperformed previous works in terms of precision, recall, and F1-score. They also found out that their model was able to detect unfairness in the datasets they used. This is extremely useful because it will make sure that the hate speech datasets used to train models in the future are more impartial and balanced.

Until now we mainly focused on hate speech in the English Language. In the upcoming article, the researchers address the difficulties of detecting hate speech in Arabic language in the context of social media [7]. The hate speech can be based on gender, race or religion. It underlines the lack of study in Arabic hate speech detection compared to languages with Latin characters. It shows the difficulty in creating successful detection models due to the complexity and variety of the Arabic language due to its dialects, derivations and so on. The paper discusses the popularity of supervised machine learning approaches, unsupervised machine learning approaches, and semi-supervised machine learning approaches. It also emphasizes the growing usage of deep learning models, particularly recurrent neural networks (RNN) and convolutional neural networks (CNN), for the detection of hate speech. According to the article, future research should concentrate on adding the most recent deep learning architectures to create a model for Arabic hate speech identification that is more successful.

We can observe another implementation of BERT, a powerful deep learning model, to detect hate speech, especially sexism, in the next research paper. The authors created a dataset called MeTwo which contained more than 3000 Spanish Tweets [8]. The tweets were classified into three categories, “SEXIST'', “DOUBTFUL'' and “NON-SEXIST”. Just like research paper [5], the authors here have made sure to cover both types of sexism, ranging from explicit hate to subtle expressions. To detect sexism, the authors made a comparison between traditional methods and neural networks in Machine Learning. They implemented two approaches for feature extraction. For instance, traditional features like tf-idf and a quite newer one like word embeddings. BERT gave the best results compared to all the other algorithms. It has an accuracy of 74%. However, the authors encountered several difficulties while conducting the experiment. The difficulties were mainly due to the fact that language is complex. It can be harder to interpret sarcasm or ambiguity in words. Also, tweets are normally short and concise which makes it difficult to understand the context behind it. The size of the dataset can also be a factor. Obviously, a larger dataset would have led to more accurate results.

The last research paper in our hate speech category provided a twist to the earlier deep learning models. It used a deep generative model to detect hate speech [9]. Previously deep learning solutions have been suggested for modest sized datasets of a few thousands. However, these models only worked well for limited datasets. Due to this, the authors initially presented a dataset of 1 million hate and non-hate sequences generated by a deep generative model. Additionally, they used the generated data to train a well-researched DL detector, showing appreciable performance. The current labeled datasets were then updated with the generated data, thereby expanding the training sets with new information. When compared to models trained on the original, unaugmented datasets, the performance of the hate speech detection models performed better. Recall, precision, and F1 score were used to evaluate the efficacy of the augmentation strategy. The results of the studies showed that adding autonomously produced hate speech samples to the training sets significantly increased the recall, F1 score, and generalization ability of the hate speech detection models. The problem of false negatives (missing hate speech sequences) was addressed by the improved performance of the upgraded model.

2.2 Code-Mixed and Code-Switched Literature

The process of switching back and forth between two or more languages or dialects during a discussion is known as code switching. Usually, it happens when bilingual or multilingual people alternate between two or more languages depending on their social or cultural environment. On the other hand, code mixing happens when speakers use words, phrases, or grammar structures from one language while speaking or writing in another. Below we will analyze the code-mixed and code-switched papers.

In their paper, Barman, U., Das, A., Wagner, J. & Foster, J. (2014, October) describe a common phenomenon called code-mixing which is caused by multilingual users. According to the authors, linguistic analysis and computational modeling of code-mixed data is challenging due to linguistic intricacy brought on by irregular spelling and grammar differences, and transliteration [11]. The authors created a new dataset that contained both Facebook posts and comments that exhibit code mixing between Bengali, English and Hindi. Different techniques were employed, including a simple unsupervised dictionary-based approach, supervised word-level classification with and without contextual clues, and sequence labeling using Conditional Random Fields. For the dictionary-based approach, they integrated two dictionaries known as LexNormList and Training Data. The accuracy of this system was 93.12%. For the second technique, supervised word-level classification, they trained a Support Vector Machine (SVM) model using several features like character patterns, word length, capitalization, etc. They discovered that combining these features led to the best accuracy of 94.75%. But the CRF model with a combination of features gave an accuracy of 95.98%. So, the dictionary-based approach was surpassed by supervised classification and sequence labeling, and that it was important to take contextual clues into consideration. They drew the following conclusions as a result of the experimental findings: character n-gram features were helpful for this task; contextual information was also crucial; and information from dictionaries can be successfully incorporated as features.

Authors in the next research paper, analyze the different ways code-mixing can occur [12]. For instance, people can mix one word from another language or they can combine phrases from another language. The researchers also examined how frequently various words are used. They divided the words into three groups: clear Code-Mixing (CM) , clear Borrowings, and Ambiguous. Using a word from another language on purpose is known as clear CM. Words that are clearly translated from one language to another are called clear Borrowings. The distinction becomes blurry in the case of Ambiguous since it can be both clear CM and clear Borrowings. One issue the researchers had was with vulgar language and profanity. Without additional information, it can be challenging to discern the intended meaning of these terms because they are frequently confused with other languages. They also discovered that in order to comprehend code-mixing, one must take into account both the linguistic structure and the motivations behind language mixing. The researchers intend to keep working to enhance how computers comprehend multilingual communication and comprehend code-mixing.

In another article, researchers believed that despite its widespread use, little effort has been invested into creating computational methods or even fundamental linguistic resources to aid research into the processing of mixed-language data [12]. The authors encountered challenges in gathering code-switched tweets for their research. They found Nepali-English code-switching users through a collaborator and collected tweets from 42 frequent code-switchers. For Spanish-English, they used Twitter's search API and annotators to identify code-switching tweets. Demographic information was manually extracted. The corpora were split into training and test sets, with 11,400 (train) and 3,014 (test) tweets for Spanish-English and 9,993 (train) and 2,874 (test) tweets for Nepali-English. The researchers believe the techniques they incorporated for data collection and annotation could prove useful. More extensive and diverse datasets in other languages could be created through their techniques.

Moving on to the the next research article, the authors gathered tweets and annotated 345 samples of texts in two languages which are Hindi and English [13]. NLP (Natural Language Processing) techniques were used to carry out sentiment analysis on the data. The study concludes that we can accurately do sentiment analysis on code-switched data but there are several challenges we should be aware of. There is obviously a scarcity of systematic rules to handle such data. On the other hand, it is indeed very challenging to determine the language of a given word. The research highlights another very crucial point. One should consider the culture and the linguistic backgrounds of users when he is examining sentiment of code-switched data.

POS (Part-of-Speech) tagging is a very important technique in NLP. This technique allocate words in a text to their corresponding part-of-speech. The grammatical role and function of each word in a phrase are represented by a tag, such as noun, verb, adjective, adverb, pronoun, preposition etc. Sequiera, R., Choudhury, M., Bali, K. (2015, December) discusses Part-of-Speech (POS) tagging of Hindi-English Code-Mixed (CM) text from social media content [14]. The POS taggers were trained using Supervised Learning (SL). They initially used a model called VGSBC, but it didn't work properly. As a result, they shifted to SL. This model performed a lot better. In order to select the appropriate tags, it transmitted the entire text to a Hindi or English tagger. Although, they used language features like context, normalization etc. to see if it would improve the performance of their model but unfortunately it made little difference. The researchers discovered that the SL model with the best features provided them with the maximum accuracy. English and Hindi had 93% and 85.9% accuracy respectively. This indicates that the machine was able to identify POS correctly most of the time. A larger dataset was used to compare the models, and they discovered that the accuracy was consistently greater. This demonstrated that it is crucial to have additional data to learn from.

In another academic paper, researchers explore the English-Bengali code-mixing phenomenon. The researchers used different machine learning algorithms to build their system [15]. For instance, J48, IBk, Random Forest. They used n-grams and dictionaries as features. They formulated their dataset from facebook posts. They were able to reach an accuracy of 90%. But they discovered that the accuracy varied based on the particular data they examined. For instance, while accuracy declined for a different dataset based on Facebook discussions, it increased for a particular dataset dubbed FIRE 2013. They talked about their research's difficulties as well. The researchers faced difficulty to identify proper nouns (names of people, places etc.) in Bengali language due to variations in spellings, contextual ambiguity etc. To better understand the language and context of each word, sentiment analysis needs to be carried out. The authors mentioned that Bengali is still a relatively unexplored language in NLP. In future works, it will be beneficial to incorporate a Parts-of-Speech tagger.

The next paper mentions the ICON 2017 where a group of teams participated to identify sentiment analysis from a code-mixed dataset [16]. The dataset included Hindi-English text and Bengali-English. The teams were made to analyze the dataset and categorize the dataset into positive, negative or neutral sentiments.

The highest accuracy was achieved by the team that used features like word embeddings and n-grams. They made use of the SVM algorithm. The performance of deep learning models was not up to the mark due to the restricted dataset. Overall, research showed that even though it is difficult to understand code-mixed languages, the correct technique might give satisfactory results.

Till now we focused on sentiment analysis of code-mixed languages that typically involved English, Bengali or Hindi. However, the upcoming research paper addresses sentiment analysis of code-mixed for the transliterated Hindi and Marathi texts [17]. A dataset of 1,200 Hindi and 300 Marathi papers was gathered by the authors from chats, tweets, and YouTube comments. They first tested each of the individual languages separately. The authors employed multiple metrics, including precision, recall, and F-measure, to assess how well the system performed. They observed that the F1 score was highest for the Naive Bayes (NB) and Linear Support Vector Machine (SVM) algorithms, whereas the RBF-based SVM algorithm did not do that well. Various percentages of the dataset were used to train the models. It was found that, Hindi and Marathi datasets showed more or less similar outcomes. They faced difficulty while trying to manage the grammar of code-mixed scripts. It was challenging to accurately apply conventional grammar rules and part-of-speech identification to this hybrid script. They proposed that concentrating on these linguistic shifts might offer insightful information about sentiment analysis. The authors developed a Marathi Wordnet in Python to overcome these issues and made it available as an open-source project. They also discussed the necessity to establish a separate Marathi SentiWordNet (MSWN) and enhance the Hindi SentiWordNet (HSWN) by including more slang words for sentiment analysis.

The authors of our second-last research paper in mixed language category focus on building a corpus specifically intended for analyzing sentiment in code-mixed Tamil-English text [18]. The authors were aware of the importance of code-mixed data in sentiment analysis research. So, they offered valuable resources to study sentiment analysis in code-mixed data, especially involving the language pair Tamil-English. The dataset was collected from movie trailers and the sentiments in the text was annotated and several categories were created based on the sentiments. They were positive, negative, neutral, mixed, or other languages. The dataset was not balanced. There were more positive sentiments. The writers then experimented with several techniques to analyze the sentiments in the text. They employed techniques like support vector machines (SVM), deep learning, decision trees, logistic regression, and random forest classifiers. They calculated precision, recall, and F-score to assess the effectiveness of these techniques. These indicators show how well the techniques can foretell the sentiments in the text. Like the previous papers, the methods had difficulty in accessing sentiments in code-mixed or hybrid script. In comparison to SVM and deep learning techniques, the performance of logistic regression, random forest classifiers, and decision trees was somewhat better. This was due to the fact that the data was unique which made it difficult for the models to predict the sentiments.

As one of the latest research done on sentiment analysis of code-switched data, our last article uses the terms code-mixed and code-switched interchangeably [20]. The authors worked on four hybrid language pairs. They are Spanglish (Spanish-

English), Hinglish (Hindi-English), Tanglish (Tamil-English) and Malayalam-English. From our previous research papers, we are aware of the fact that traditional sentiment analysis techniques often call for a labeled dataset in which each text is assigned a sentiment (positive, negative, or neutral) that corresponds to it . We also saw the implementation of unsupervised learning in our prior papers. The authors in this particular paper also chose to carry out an unsupervised self-training approach in their research. The researchers gathered a dataset of tweets in various code-mixed languages to start their study. They assigned the labels HIN (Hindi), ENG (English), or 0 (symbol or special character) to each word in the dataset. In addition, they established a concept known as the Token Ratio. For instance, if we consider the Hinglish dataset, Token Ratio would indicate how much more Hindi is used in a statement than English. They trained the model using the code-mixed dataset they gathered using their unsupervised self-training approach. The algorithm used data from each training cycle to make predictions on fresh, unlabeled texts. The model supplied temporary labels using these predictions. It subsequently gained knowledge from the newly labeled data, which included both the authentically labeled sentences and the pseudo-labels. The model learned from its own predictions as the process was performed numerous times, improving its comprehension of code-mixed data. When compared to models that were trained using human annotations (supervised models), the researchers found that their unsupervised model performed competitively. This was a huge success because it eliminated the need for costly and time-consuming human labeling.

In the foregoing analysis, we have discussed the nature of hate speech, particularly sexism, and its impact in society. We analyzed various methods to detect it in the context of social media. Likewise, we have discussed the previous studies done on code-mixed and code-switched languages. We talked about how the authors of each paper collected their data and the models they used. Different models were compared in terms of accuracy, precision etc. We also discussed the difficulties the researchers faced in conducting their research and what improvements need to be made in the future. Overall, the subject is vast and we could not discuss all aspects of it. However, we believe that further research is needed on the subject in the near future.

Chapter 3

Dataset Description

3.1 Collection

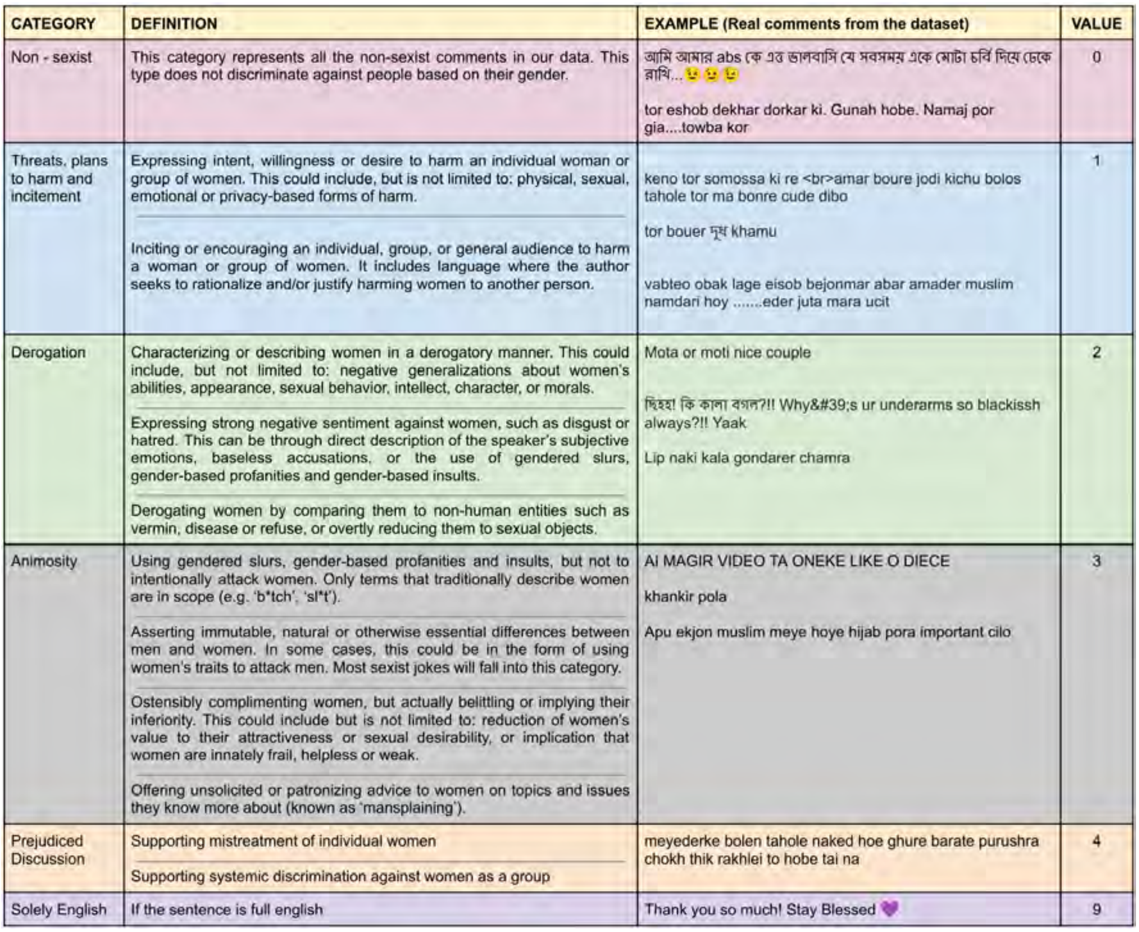
We observed the comment sections of a lot of different Youtube videos and Facebook public comment sections. For collection of Facebook comments, we used the help of CommentExp, which is a public API, to collect our public Facebook comments. Our main objective was to identify sexist content in code-mixed and code-switched data. So, we only chose the Youtube videos and public Facebook content that contained a significant amount of comments catering to our interest. We collected about 7k comments in total. Our dataset did not only just contain code-mixed and code-switched comments. It also had comments solely in English or Bangla.

We believe our dataset is special due to several reasons. First of all, it had linguistic complexity due to its code-mixed and code-switched nature. People who commented were particularly bilingual speakers, fluent in both Bangla and English. Secondly, our dataset was robust since people from diverse social and cultural backgrounds commented. So, the comments covered a wide range of topics and contexts. Lastly, the dataset covered different types of sexist expressions, ranging from subtle to overt sexist remarks. This ensured the analysis was done on a broad range of sexism.

3.2 Category Construction

Kirk et al. (2023) divides sexist content into four conceptually and analytically distinct categories [22]. We used the same categories and definitions to annotate our dataset. However, we added two extra rows to identify the non-sexist and solely English comments. The categories and definitions are highlighted in a table. We provided real code-mixed and code-switched comments from online communities as examples. Figure 3.2.1 illustrates the table.

Content warning: Please view the examples with caution. These are real comments from out dataset. These examples might cause discomfort to some.



CATEGORY	DEFINITION	EXAMPLE (Real comments from the dataset)	VALUE
Non - sexist	This category represents all the non-sexist comments in our data. This type does not discriminate against people based on their gender.	আমি আমার abs কে এত ভালবাসি যে সবসময় একে মোটা চর্বি দিয়ে ঢেকে রাখি...😉😉😉 tor eshob dekhar dorkar ki. Gunah hobe. Namaj por gia....towba kor	0
Threats, plans to harm and incitement	Expressing intent, willingness or desire to harm an individual woman or group of women. This could include, but is not limited to: physical, sexual, emotional or privacy-based forms of harm. Inciting or encouraging an individual, group, or general audience to harm a woman or group of women. It includes language where the author seeks to rationalize and/or justify harming women to another person.	keno tor somossa ki re amar boure jodi kichu bolos tahole tor ma bonre cude dibo tor bouer দুধ khamu vabteo obak lage eisob bejonmar abar amader muslim namdari hoy ....eder juta mara ucit	1
Derogation	Characterizing or describing women in a derogatory manner. This could include, but not limited to: negative generalizations about women's abilities, appearance, sexual behavior, intellect, character, or morals. Expressing strong negative sentiment against women, such as disgust or hatred. This can be through direct description of the speaker's subjective emotions, baseless accusations, or the use of gendered slurs, gender-based profanities and gender-based insults. Derogating women by comparing them to non-human entities such as vermin, disease or refuse, or overtly reducing them to sexual objects.	Mota or moti nice couple ছিহহ! কি কালা বগল?!! Why&#39;s ur underarms so blackissh always?!! Yaak Lip naki kala gondarer chamra	2
Animosity	Using gendered slurs, gender-based profanities and insults, but not to intentionally attack women. Only terms that traditionally describe women are in scope (e.g. 'b*tch', 'sl*t'). Asserting immutable, natural or otherwise essential differences between men and women. In some cases, this could be in the form of using women's traits to attack men. Most sexist jokes will fall into this category. Ostensibly complimenting women, but actually belittling or implying their inferiority. This could include but is not limited to: reduction of women's value to their attractiveness or sexual desirability, or implication that women are innately frail, helpless or weak. Offering unsolicited or patronizing advice to women on topics and issues they know more about (known as 'mansplaining').	AI MAGIR VIDEO TA ONEKE LIKE O DIECE khankir pola Apu ekjon muslim meye hoye hijab pora important cilo	3
Prejudiced Discussion	Supporting mistreatment of individual women Supporting systemic discrimination against women as a group	meyederke bolen tahole naked hoe ghure barate purushra chokh thik rakhlei to hobe tai na	4
Solely English	If the sentence is full english	Thank you so much! Stay Blessed 💜	9

Table 3.1: Different categories of sexism and their corresponding definitions, examples, and values.

3.3 Annotation

Our team of dedicated three members annotated the whole dataset with the help of an expert. Before beginning the annotation, the members received detailed guidelines and thorough training. Various case studies were used as examples to make the amateur annotators get familiar with the whole process. The team carried out weekly meetings with the expert annotator in order to discuss challenging cases and refine their understanding and approach to annotation. This rigorous and collaborative effort ensured that our dataset was of superior standard with high accuracy.

A three step process was carried out to annotate the whole dataset. Firstly, the amateur annotators were divided into pairs for different batches of the comments. Let's denote the members as N, F and S for simplicity. The first 500 comments were annotated by N and F separately, then by N and S for the next 500, and finally by F and S for another 500. The subsequent 500 was annotated by N and F again and so on. The pairs covered the whole dataset following this technique. The annotators assigned to a particular batch worked separately. For example, if N and F were assigned a batch of 500 comments, N made sure that she did not let F know which values she assigned for the comments. F did the same thing. This was done to ensure a variety of perspectives and to minimize biases on decision-making as much as possible.

Secondly, the members held regular meetings to check their annotations. If there was any disagreement between the members involved with a particular batch, the third member, who was not part of the pair for that batch, acted as a mediator. An example of this is, if N and F disagreed on a particular comment, S would decide which value is most appropriate. In this way, the third member would bring a new outlook to the situation and resolve the dispute between the rest of the members.

Thirdly, if the former process could not produce a consensus for ambiguous cases, the amateur annotators would seek help from the expert. The expert annotator played a crucial role in ensuring that our dataset was accurate and reliable. Overall, the whole meticulous process created a comprehensive dataset for the detection of sexist remarks in online platforms.

3.4 Statistics

Our dataset contains about 7k comments in total. Fig 3.2 shows our data distribution. We have almost 1.9k comments which are solely English. Our main objective was to remove these comments. These comments were not Bangla nor code-mixed or code-switched. Therefore we removed it when we cleaned our data for several reasons. For instance, our model BanglaBERT was designed specifically for the Bangla Language. In order for our model to work properly, we wanted to confuse it as little as possible with another completely different language. We hence felt purely English language would not add significant value to our dataset. Also, we believe that training large models requires computational resources and we wanted to allocate our resources as effectively as possible by focusing on our ultimate goal. Besides that, there are lots of models for purely English Language but little research has been done on Bangla language nuances. So we wanted to focus on Bangla and its variants. We have above 5k comments which are code-mixed and code-switched. Comments which are entirely Bangla have also been given the category of code-mixed and code-switched.

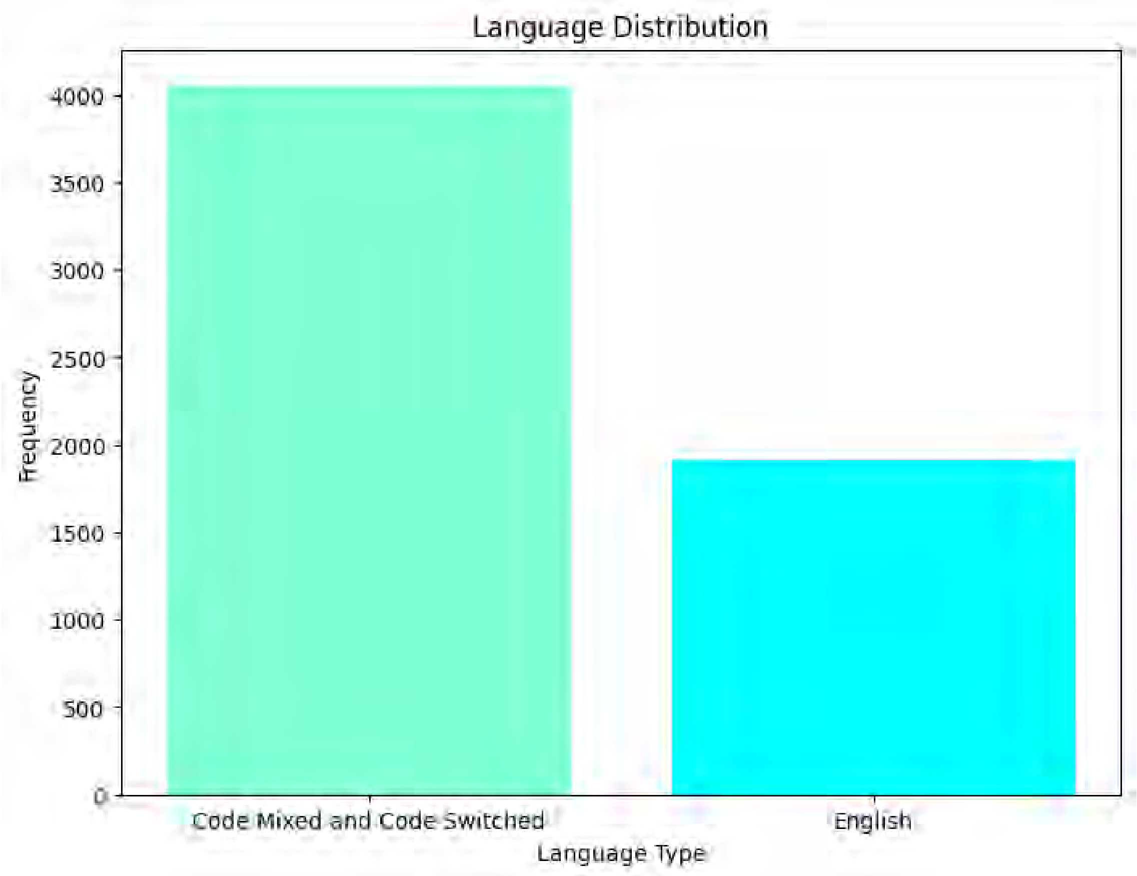
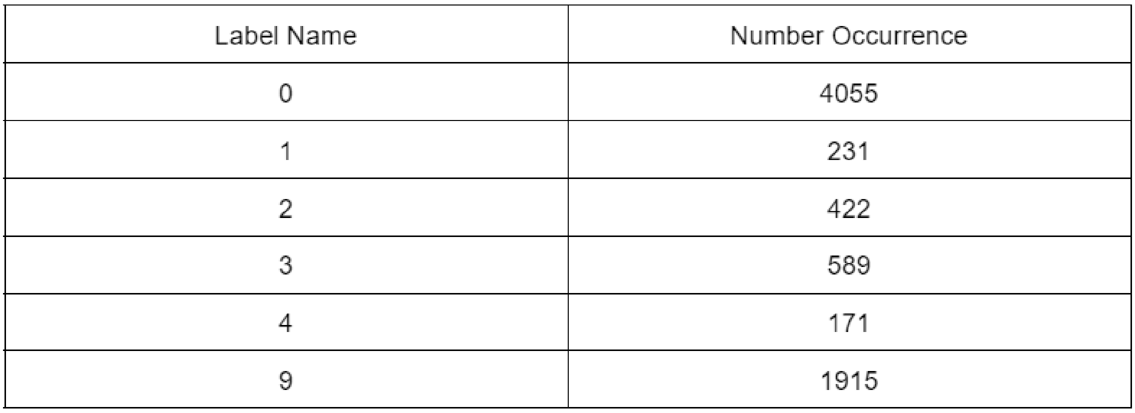


Figure 3.1: Comparison of code-mixed and code-switched vs. English data

Fig 3.3 illustrates the total count of our six categories. From the table it can be seen that our dataset contained about 4k comments from the code-mixed and code-switched category which were non-sexist. The four sexist types are also given subsequently which include 1500 comments in total. We also found 1915 solely English comments in our diverse dataset.



Label Name	Number Occurrence
0	4055
1	231
2	422
3	589
4	171
9	1915

Table 3.2: Distribution of showing number of occurrences of different label

Figure 3.4 illustrates only the sexist categories in a bar chart respectively. There are four sexist categories we used to annotate our dataset which are threats with the intention of harming, derogation and so on.

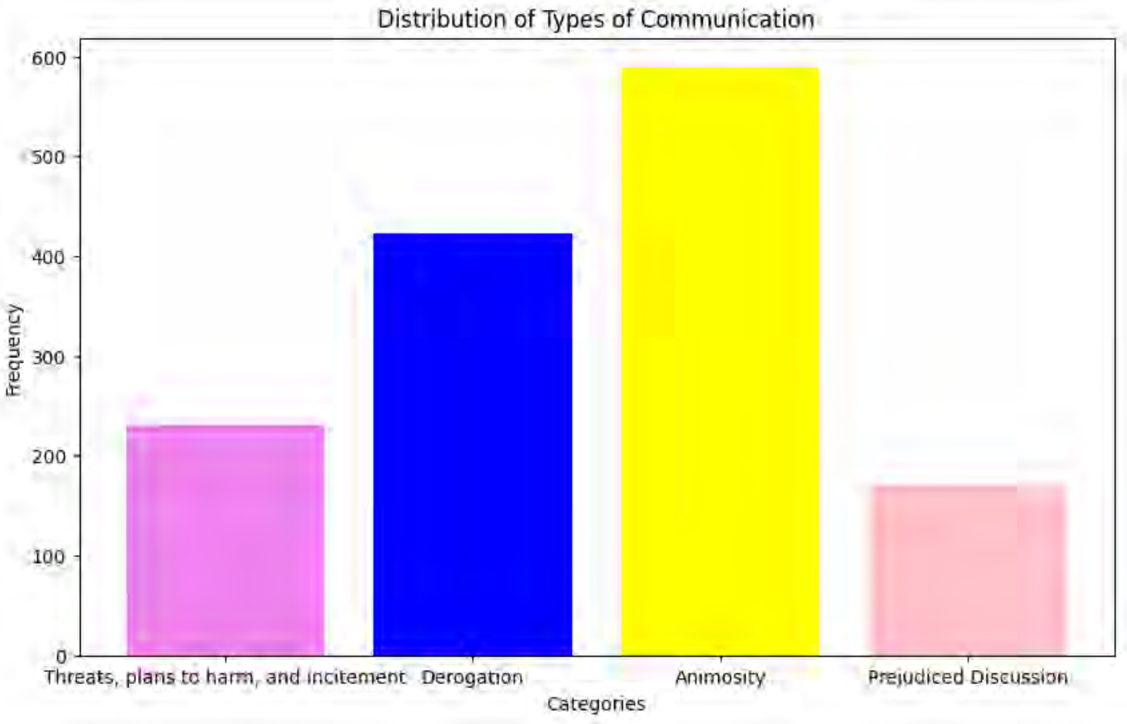


Figure 3.2: Comparison of different types of sexism from the dataset

Figure 3.4.4 gives us a visualization of our word cloud. It highlights the most used words in our sexist corpus. For instance, the words “khanki” and “magi” which are gendered slangs in the Bangla language have been been casually used numerous times in our dataset.

Content warning: Please be aware that some of the words might cause discomfort.

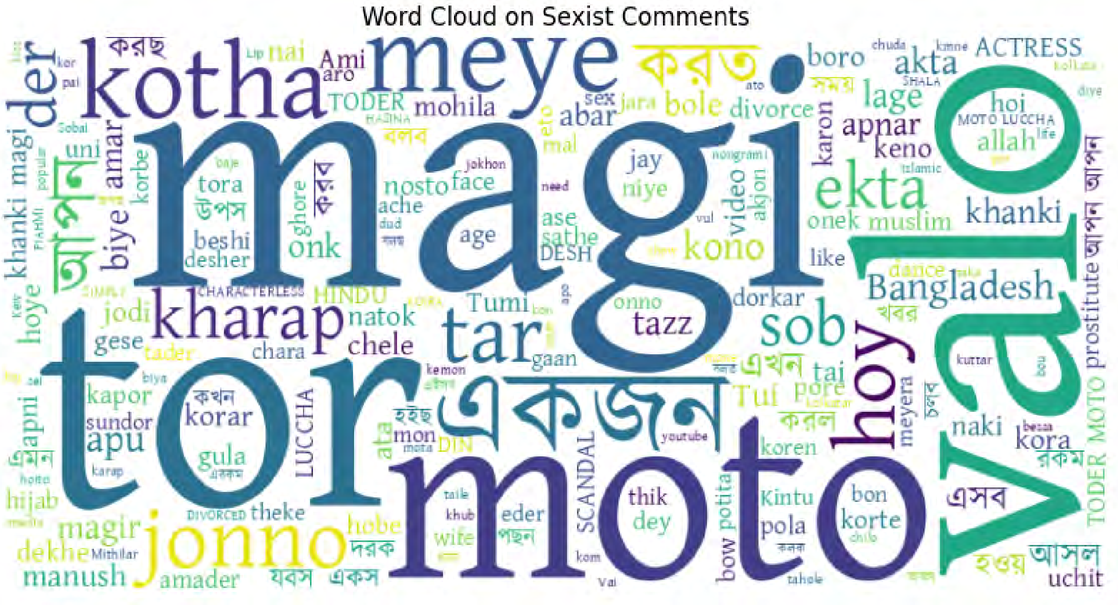


Figure 3.3: Word Cloud of sexist words

Chapter 4

Data Preprocessing

4.1 Data Cleaning

The custom annotated dataset containing nearly 7k comments is loaded using the pandas library. To avoid errors of chosen NLP model we cleaned our dataset which contains steps-

Handling Missing Values:
We search for columns that contain NaN value, and replace it with empty strings to avoid error and data bias.

Removing Duplicates:
We remove duplicate columns to avoid data bias in training and to increase accuracy of the model.

Removing unwanted sections:
To increase accuracy we remove characters that don't add value to the meaning of comments. So we remove URLs, html tags, hashtags, mentions and digits using regular expressions. And we remove emojis and replace it with empty strings by utilizing re modules' regex patterns.

4.2 Data Preprocessing

For ensuring quality full training of our chosen model we preprocess data to enhance the quality of our data and visualize and analyze following below steps-

Removing Punctuation: To remove punctuation from the English language we use nltk library. For Bangla language we take a list of punctuations and remove it from the text data.

Removing Stop Words: We use nltk library to access English and Bangla stop-words. Then we remove it from the text data column.

Translation: Some models are pre trained to handle monolingual data and some are trained to handle multiple languages. So, to increase accuracy of model we do translation on code-mixed and code-switched data to convert to Bangla language. We use Googletrans library version 4.0.0-rc1 to access Google Translate API. Then we identify the columns that are code switched or code mixed to convert them to Bangla text by translating.

Removing empty columns: Removing stop words, emoji, urls and mentions might create empty rows in the dataset. So to remove errors in training we remove any empty rows.

Handling Multi-label: Our dataset was annotated in a way that it is possible for each data row to have multiple labels. Rather than handling it as multi-class meaning creating a new category for multiple labels, we want to work with multi labels. So, we extract multiple labels from each row and count it towards the existing category.

4.3 Visualization

We want to visually analyze our data to get a better understanding of the custom annotated dataset. So we plot graphs to show distribution of labels and their implications.

Since we want to separately deal with binary classification for sexist versus non sexist data and multi label classification for our defined labels in dataset description. So we define and label the data where- label-0: Non - sexist, label-1: Threats, plans to harm, label-2: Derogation, label-3: Animosity, label-4: Prejudiced Discussion, label-9: English. Then we use scikit-learn to binarize our text data. For classification we create binary matrix representation of our original data. So we create a bar Chart for Labels 1 to 4 for multi-labels and for non-sexist data. And we visualize bar charts for English vs code-mixed and code switched data.

To get a better understanding of our dataset and analyze linguistic patterns we implement wordcloud on data before doing translation, to see frequently occurring Bangla and English words in sexist data. We utilize Bangla font ‘kalpurush.ttf’ to visualize Bangla words in the word cloud properly. We make use of wordcloud library to represent frequent words from both languages.

Chapter 5

Models, Experiment and Results

5.1 Models Introduction

Our dataset consists of code mixed and code switched sexist data. So to deeply analyze we first work with binary classification to identify if the text is sexist or not. And in multi label classification, we separately detect what category of sexism the text falls under. For this we used multiple NLP and deeplearning models. which are-

5.1.1 BanglaBERT

BanglaBERT, pre-trained on a vast Bangla data corpus, enhances its understanding of Bengali language nuances, syntax, semantics, and colloquial expressions. This specialization is crucial when dealing with Bangla code-mixed and code-switched data. The BanglaBERT model builds up on the foundational ELECTRA framework focusing on efficiency and effectiveness in order to better suit the intricacies of the Bangla language. It includes a 12-layer Transformer encoder structure. Each encoder layer is equipped with 768 embedding and hidden size units, 12 attention heads for complex pattern recognition, and a larger feed-forward network with a size of 3072. The unique pretraining objective of BanglaBERT is Replaced Token Detection(RTD) that sets it apart from BERT's Masked Language Modeling (MLM). Training parameters are carefully chosen to balance computational resources with performance. Using a v3-8 TPU instance, the model is trained over 2.5 million steps with a batch size of 256. Adam optimizer is used to handle the optimization with a learning rate of 2e-4 which includes a linear warmup over the first 10k steps. These parameters are selected to ensure that the model learns effectively while managing computational load. Basically, BanglaBERT represents a modified approach in developing language models for specific linguistic contexts, using an advanced architecture to process and understand the Bangla language with high efficiency.[25]

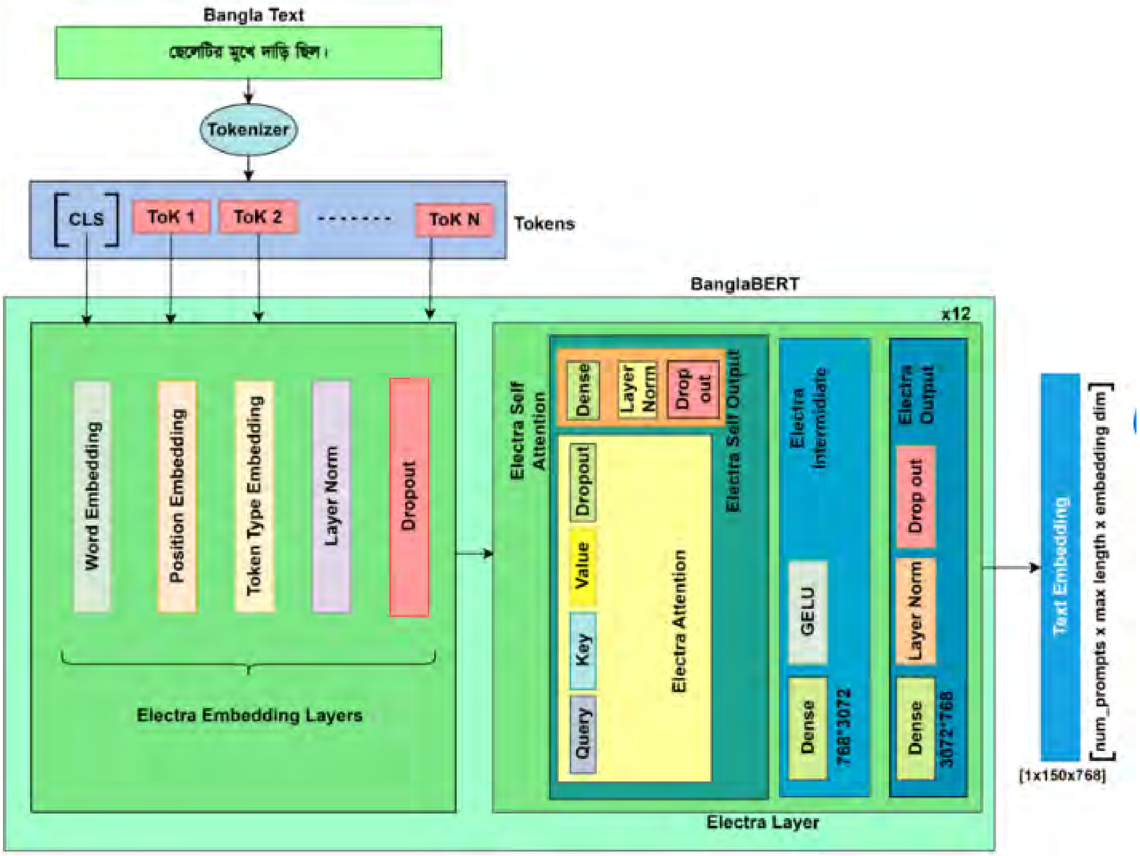


Figure 5.1: Architecture of BanglaBERT

5.1.2 BERT

The architecture of BanglaBERT is based on the ELECTRA model which is a variant of BERT. Basically, BanglaBERT is a version of the original BERT model which is modified for the Bangla language. It is built on the transformer architecture which is mentioned in the ”Attention is All You Need” paper [23]. BERT is a unique architecture that uses attention mechanisms in order to comprehend the meaning of a word in a sentence. It uses the encoder part of the Transformer and consists of a stack of identical layers, each with a multi-head self-attention mechanism and a fully connected feed-forward neural network.The Multi-Head Self-Attention component helps understand word context by focusing on different input sequence positions. BERT’s input representation uses WordPiece token embeddings, positional encodings, and segment embeddings to differentiate between different sentences. Unlike traditional language models, BERT considers all surrounding text, both left and right of the word. It is trained on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) to predict masked tokens and follow two sentences. BERT can be enhanced with additional output layers for various NLP tasks. The BERT architecture’s design effectively captures language nuances which enables advanced performance in different Natural Language Processing tasks.

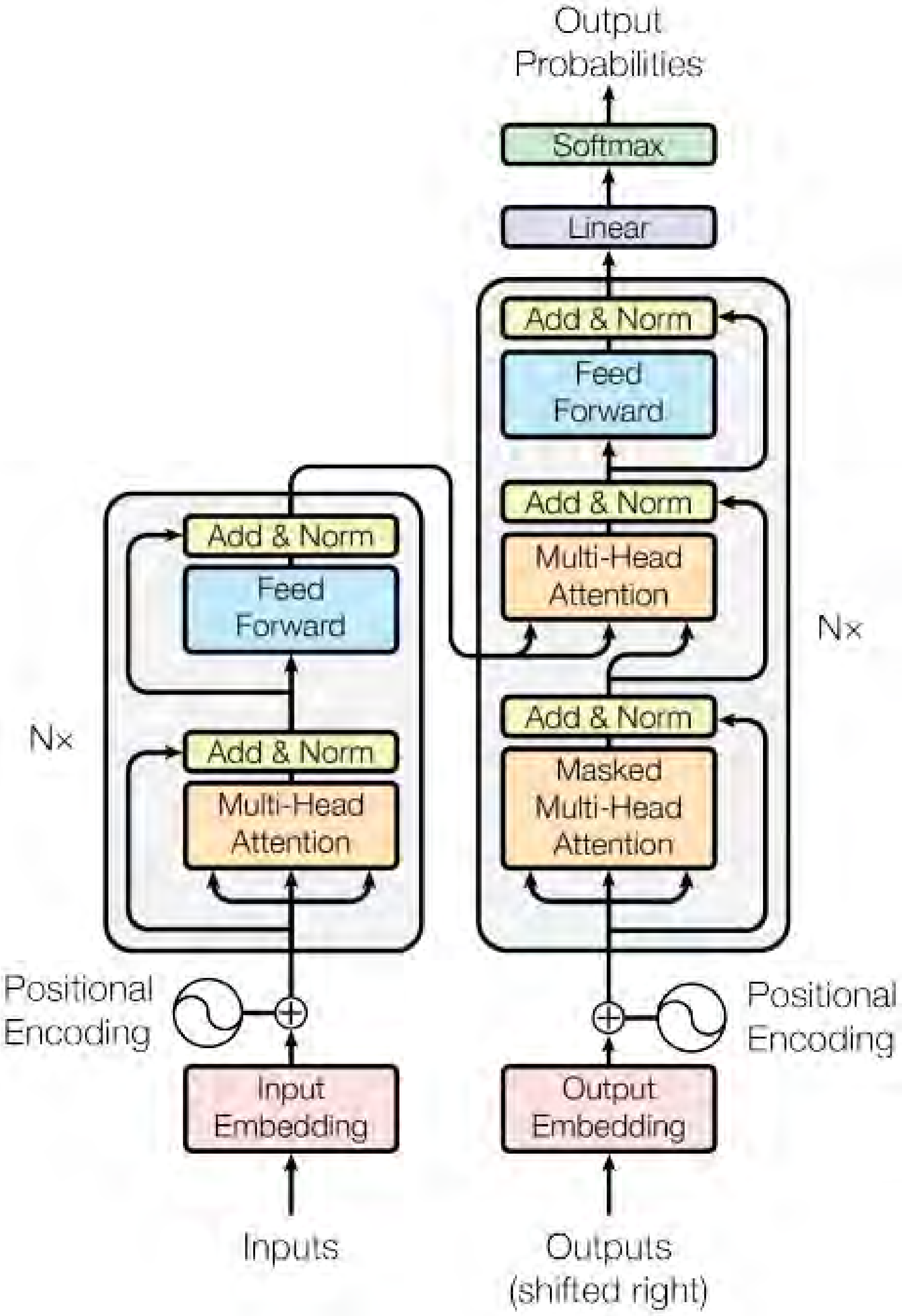


Figure 5.2: Architecture of BERT

5.1.3 Bi-LSTM

Bidirectional Long Short-Term Memory (BiLSTM) model is a kind of recurrent neural network (RNN) architecture. This is designed to process sequential data effectively by capturing dependencies in both forward and backward directions. Because of this bidirectional capability, this model is appropriate for tasks like text classification specially where understanding the context from the entire sequence of words is the most important part. This model in an extended version of the original

LSTM. It consists of two separate LSTM networks, one to process input sequence and one to perform backward LSTM. After the processing of the the entire sequence is done by both LSTMs, the hidden states are concatenated at each time step. This contributes in getting a comprehensive representation of each word that incorporates information from both directions by just combining the forward and backward context. This allows Bi-LSTM to understand the full context of each word[26].

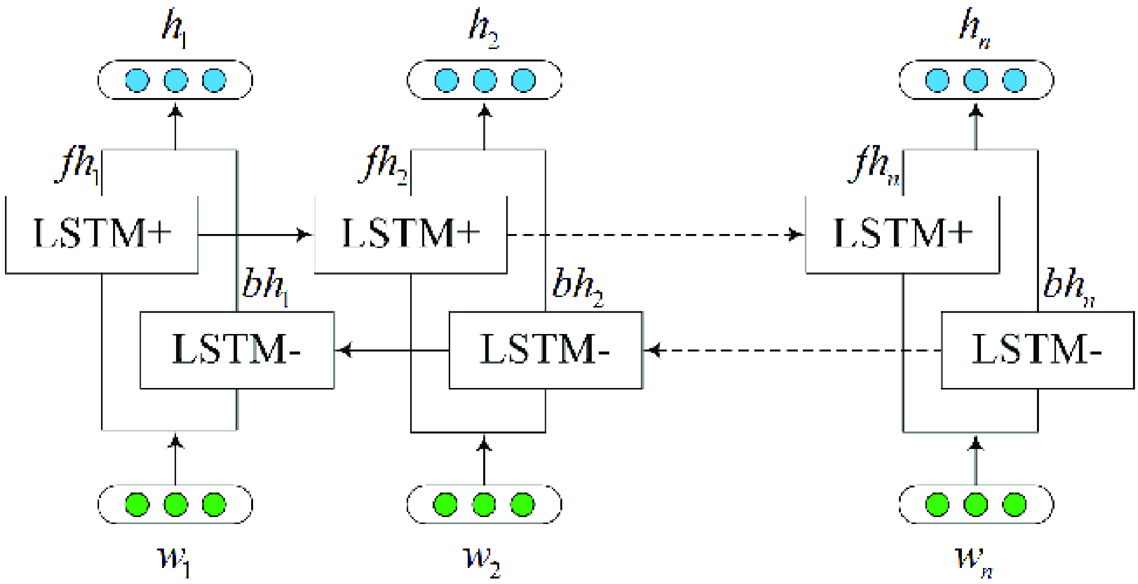


Figure 5.3: Architecture of Bi-LSTM

5.1.4 GRU

GRU model is a form of Recurrent Neural Network(RNN) which is very effective at handling sequence modeling tasks. It is designed to handle the vanishing gradient problem and has the ability to remember long-term dependencies in the data. GRU model is based on two gates - reset and update gates. The reset gate calculates and decides how much of the past it needs to forget. On the other hand, the update gate determines how much of the past knowledge gathered from earlier processes needs to be sent along.[24]

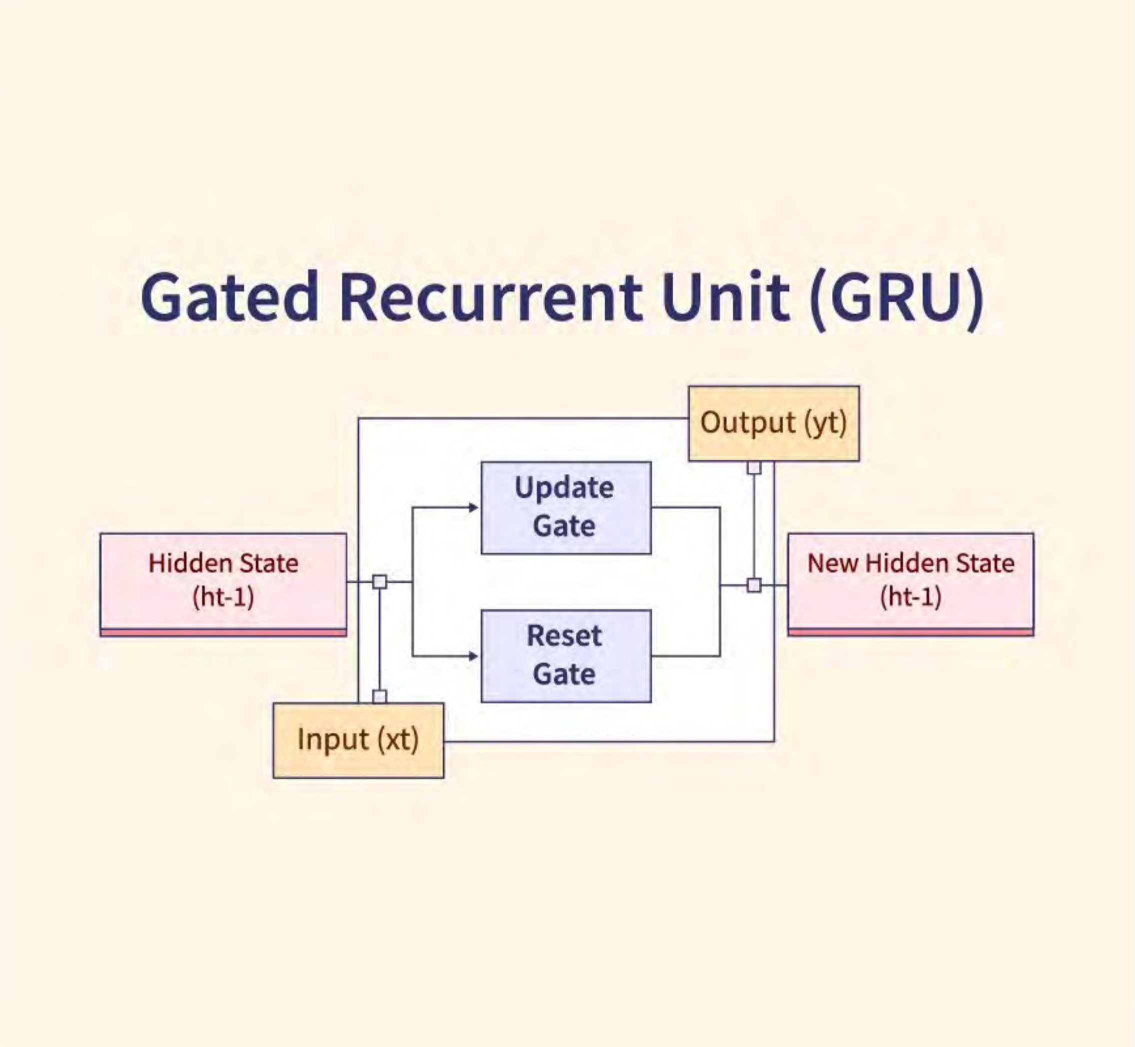


Figure 5.4: Architecture of GRU

5.1.5 MuRIL

MuRIL (Multilingual Representations for Indian Languages) is a pre-trained BERT model which has been trained on 17 Indian Languages including Bangla. The model has also been trained on the transliterated counterparts respectively. This means the model not only learns from the traditional scripts but versions of texts that have been converted into a common script. The model was pre-trained using data from Wikipedia, Common Crawl, PMINDIA and Dakshina respectively. We specifically used the MuRIL base because it's smaller in size needing less computational resources and therefore, it's faster to run. Even though MuRIL large might not meet the performance of its variant but it's suitable in our case because we wanted lower resource consumption and good performance speed. The MuRIL is trained on two types of data parallelly. One is the transliterated data and another is the translated data[27].

5.1.6 BanglishBERT

Banglishbert is a pretrained model that is developed by a NLP research group at BUET. This model was created for Bangla and “Banglish” text, which is very useful for handling the instances of code-mixed and code-switched content involving Bangla and English. Banglishbert has multiple layers of Transformer blocks where each block contains a multi-head self-attention mechanism and a fully connected feed forward network which are known as sublayers of the block. For Banglishbert, the tokenizer is trained on a mixed corpus of Bangla and English text, which helps it to handle the instances of Bangla mixed with the structure of English. The unique pretraining objective of BanglishBERT is Replaced Token Detection(RTD) that sets it apart from BERT’s Masked Language Modeling (MLM)[28].

5.2 Binary Classification

Avoid rows: For binary classification, we consider sexist and no-sexist data which includes rows labeled 0, 1, 2, 3 and 4. So we drop rows labeled 9 which hold data for english text and create objects for binary classification.

Split data: We use a cross validation process to split our dataset to avoid any kind of overfitting. For this we split the dataset so that the train set size is 80 percent and testset size is 20 percent of the main dataset.

Tokenize: To properly format our data to input in our BanglaBERT model, we tokenize our training and testing sets using pre trained ”csebuetnlp/banglabert” auto tokenizer. We load the banglaBERT tokenizer and sequence classification model using the Hugging Face Transformers library. We create a custom dataset class “BanglaDataset“ to fit our raw dataset with pytorch data loader to create dataset objects.

Training and Evaluating: We use the banglabert model to train on our dataset. We set custom training arguments and change these training arguments adapting with the models performance on the test dataset. We evaluate our model using accuracy, f1 score, precision and recall and loss. We visualize it by portraying it on training loss and validation loss graph over epochs and confusion matrix.

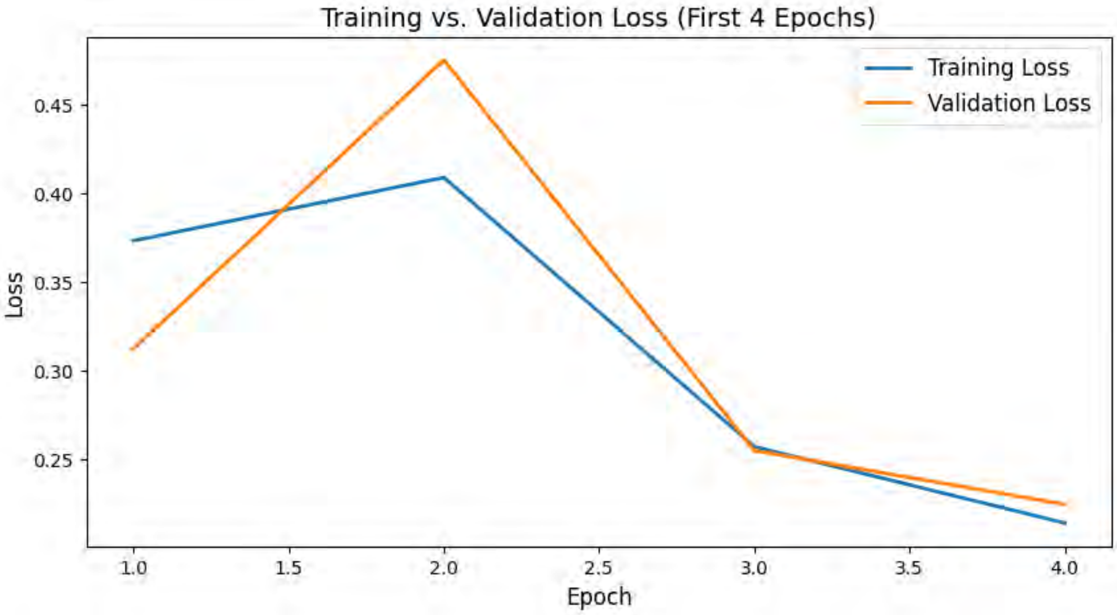


Figure 5.5: Training loss of Binary classification over epochs

5.3 Multi-label Classification

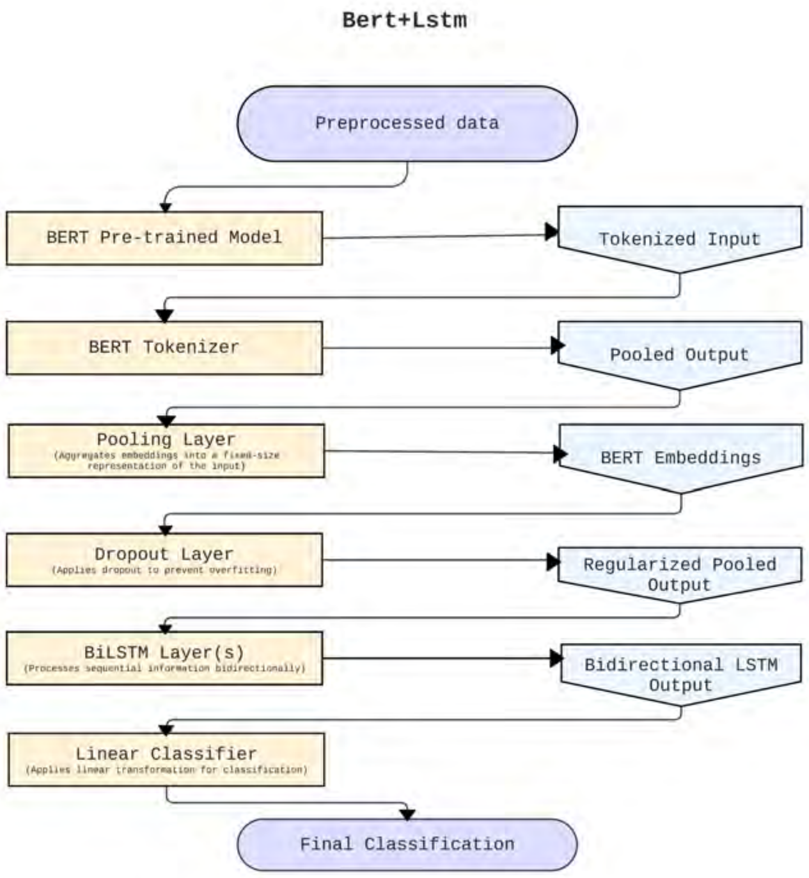
5.3.1 BERT+Bi-LSTM

Architecture

We start our experimentation with Bidirectional Encoder Representations from Transformers (BERT). As bert is known for its context capture ability, so to get the best result for multilabel text classification, we utilize contextual comprehension of BERT and the sequence modeling capabilities of BiLSTM.

For this algorithm, we initialized with Bert pre-trained model for better context and tokenized input, Then, used Bertotkenizer for our corpus, Bert's embedding layer maps input tokens to dense vectors which lays the foundation for future process. BERT's multi-layered in this case 12 layered bidirectional transformers are used to capture token relationships within a sequence. So the pooled output of bert corresponding to the [CLS] token, captures the entire sequence's contextual essence. The pooling layer aggregates bert embeddings into a fixed-size representation of the input to get proper BERT embeddings for this case. After getting the pooled output its passed through a dropout layer to mitigate over-fitting and to get the regularized pooled output.

We use BiLSTM to capture dependencies from both past and future contexts within the sequence, enhancing the model's ability to understand the full context. So, the dropout output is then fitted to the BiLSTM input requirements and passed through the BiLSTM layer. From this we get the output that comprises concentrated forward and backward hidden states, which means it contains the sequence's full context. Then its passed through a linear layer so that logits for each label can be computed, And we get the values of loss based on the logits and true labels for training and future improvement and evaluation. By doing so, we made our model well-equipped to handle the complexities of multi-label text classification.


Bert+Lstm
Preprocessed data
BERT Pre-trained Model
Tokenized Input
BERT Tokenizer
Pooled Output
Pooling Layer
(Aggregates embeddings into a fixed-size representation of the input)
BERT Embeddings
Dropout Layer
(Applies dropout to prevent overfitting)
Regularized Pooled Output
BiLSTM Layer(s)
(Processes sequential information bidirectionally)
Bidirectional LSTM Output
Linear Classifier
(Applies linear transformation for classification)
Final Classification

Note: Weights for each class in the linear classifier are updated using inverse weighting to address class imbalance during training.

Figure 5.6: BERT + LSTM Architecture

Performance Evaluation

Performance Evaluation: This model's accuracy over all epochs averaged at 0.822, with an F1-score of 0.817, precision of 0.826, and recall of 0.808. This gives an insight into how well this model if performing while classifying multi label classification.

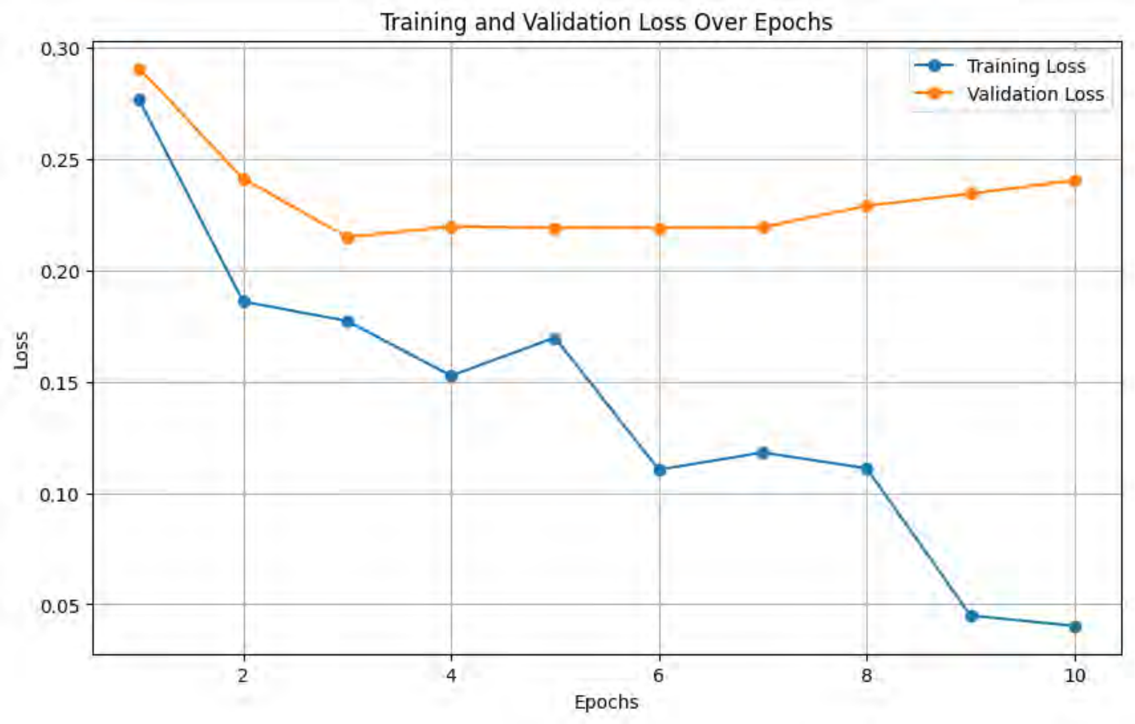


Figure 5.7: Training and validation Loss Graph of BERT + LSTM

This model undergoes an iterative optimization process and the parameters are adjusted accordingly to minimize loss function. The training loss here gradually decreases and drops as low as 0.011. Meanwhile validation loss shows some fluctuation and drops as low as 0.22. This proves that the model is successfully capturing underlying patterns and features in the codemixed and code switched text. In conclusion it is generalizing effectively, and making accurate predictions for multi-label text classification tasks.

5.3.2 BERT

Architecture

Experimenting with bert+lstm gave us impressive results, which inspired us to fine-tune BERT with custom parameters.

As established, BERT is a pre-trained model that captures context from little hints even on a small corpus.Basing on this, for this algorithm we recreated the same tokenization and process of getting the embeddings from BERT as our BERT+LSTM model. We apply dropout on this to avoid overfitting. Linear layer transforms these BETT outputs into classification format and then we calculate loss based on model prediction and true labels. And while training we keep updating model parameters based on it. Here we implement a function to calculate samples and inversely calculate weights for customizing loss computation for multi-label classification.

Performance Evaluation

Fine tuned bert with custom parameters also give noticeable results. Where accuracy is 0.833 and f1 score is 0.828 with precision 0.837 and recall 0.819. Observing the validation graph we can see that training loss gradually decreases a lot and drops to 0.017, Whereas, the validation loss is fluctuating but still drops to 0.19 at some point. These trends, coupled with high accuracy and f1 gives the idea that the model is reliable and sustainable real-world applications where accurate classification of text data across multiple labels is paramount.

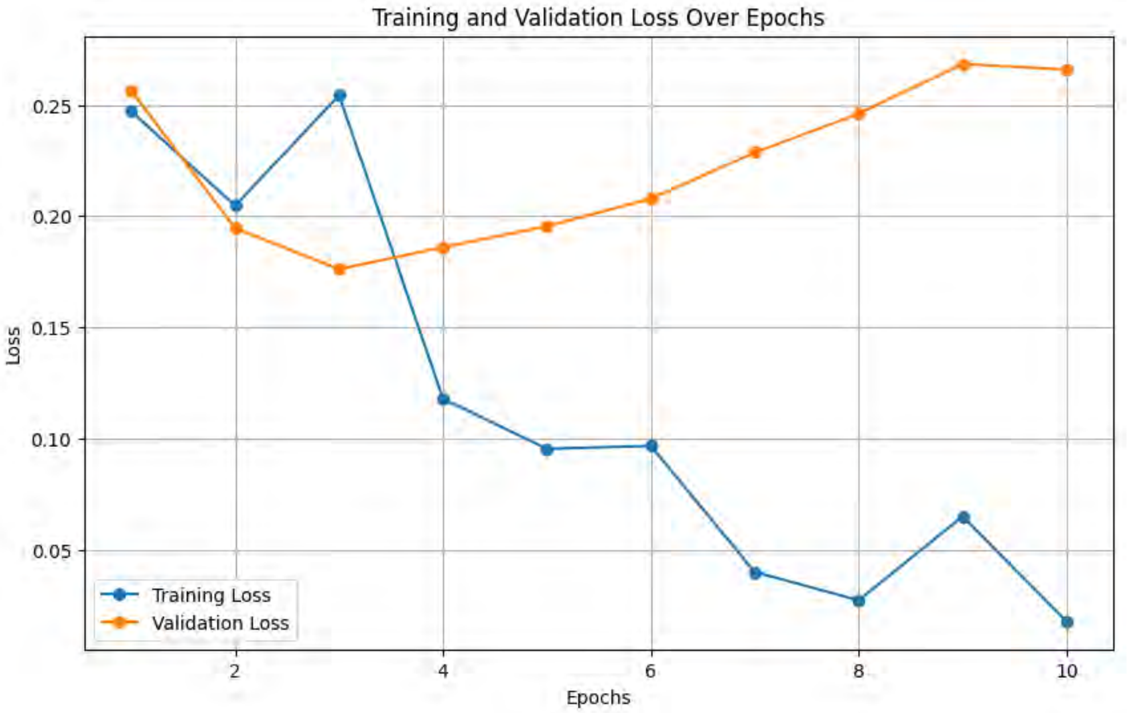


Figure 5.8: Training and validation Loss Graph of BERT

5.3.3 MuRIL+Bi-LSTM

Architecture

Our dataset contains text comments and their corresponding labels in a CSV format. The code-mixed and code-switched in our original dataset were converted to entirely Bangla to make it easier for the model to understand context better since MuRIL is based on Bert Model which was pre-trained on 17 Indian languages including Bengali. Hence, it's logical to convert the comments to Bengali. The pre-processed text data was converted to tokens using MuRIL's tokenizer. The tokens were further converted to embeddings using the MuRIL model. The main reason for doing this was due to the fact that the MuRIL model is able to capture the contextual relationships within text. The dataset was split into train, validation and test sets respectively. We tried to implement K-fold cross validation for splitting but the results were not satisfactory. This might be for several reasons. For instance, our dataset is imbalanced which means all classes might not be properly represented in each fold. Eventually, the splitted data was fed to an LSTM with an attention mechanism to capture the important parts of the text. Weights were calculated to handle class imbalance. A higher weight was given to the class that appeared less frequently in the dataset and vice versa. This was done by inverting the frequency of the class. Early stopping was implemented to ensure no overfitting took place. Also, different combinations of parameters were tested through random search and the parameters with the best results were selected eventually.

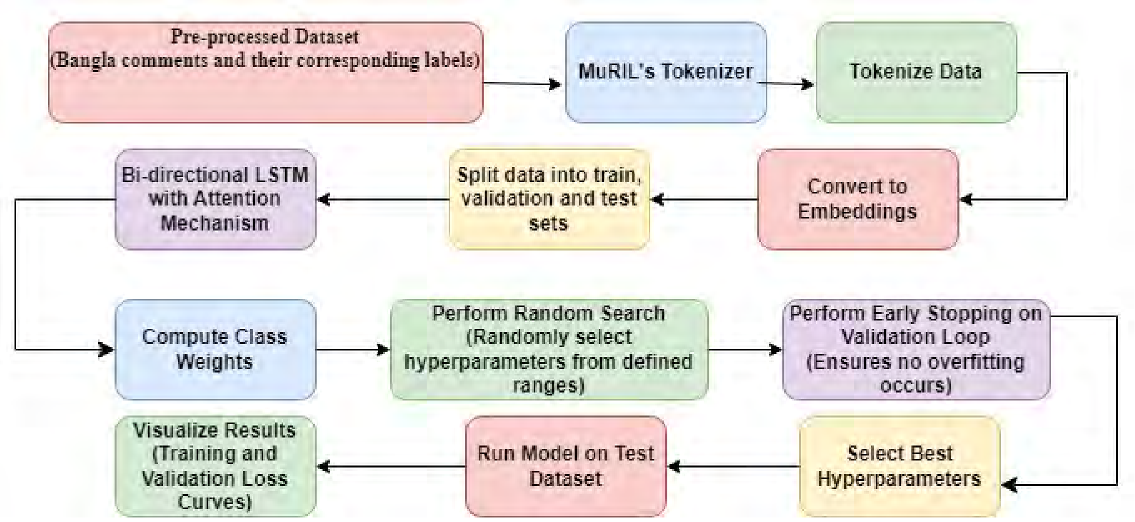

Pre-processed Dataset (Bangla comments and their corresponding labels)
MuRIL's Tokenizer
Tokenize Data
Convert to Embeddings
Split data into train, validation and test sets
Bi-directional LSTM with Attention Mechanism
Compute Class Weights
Perform Random Search (Randomly select hyperparameters from defined ranges)
Perform Early Stopping on Validation Loop (Ensures no overfitting occurs)
Select Best Hyperparameters
Run Model on Test Dataset
Visualize Results (Training and Validation Loss Curves)

Figure 5.9: Flowchart of MuRIL + Bi-LSTM methodology

Performance Evaluation

The MuRIL showed noticeable results with 0.713 accuracy, 0.788 precision, 0.757 recall and f1 score of f1 score of 0.772. The continuous decrease in training loss suggests that the model is effectively learning. The validation loss also decreases with the training loss which is a good sign. Both the graphs fluctuate a bit. The validation loss graph starts to increase after 35 epochs and it's stopped by early stopping ensuring no over-fitting occurs.

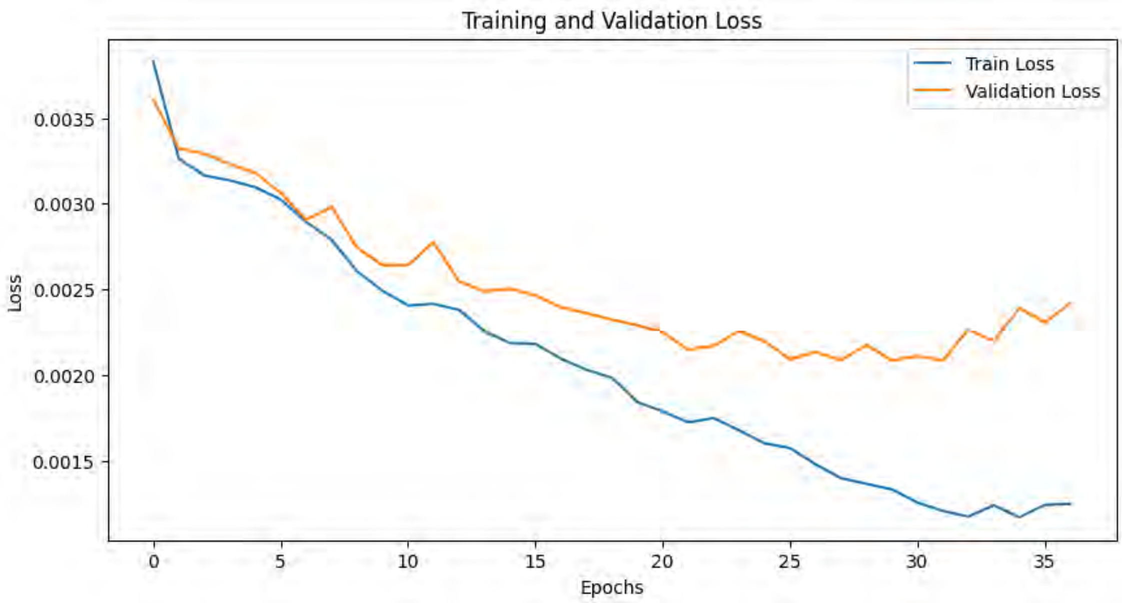


Figure 5.10: Train and Validation Loss Graph of MuRIL + Bi-LSTM

5.3.4 BanglishBERT+GRU

Architecture

For this integrated model, at first, we have done some preprocessing for the dataset. We did not translate the comments as we are using Banglishbert which is compatible with code-mixed and code switched data. Then the values were converted into a list of integers to capture the categorical nature of our labels which were one-hot encoded which converts categorical integers into binary matrices.

To prepare the data for model input, we utilized a tokenizer from the pretrained ”csebuetnlp/banglishbert”. This tokenizer is developed to handle Banglish text (a mixture of Bengali and English), making it ideal for our dataset. The tokenizer is used to convert the text data into tokens, which are essential for understanding and capturing the linguistic patterns within the text. We also defined a dataset class to manage the tokenized data and corresponding labels.

After that, we implemented a GRU based classifier layer on top of the Banglishbert model’s output. This setup was designed to leverage both the contextual embedding capabilities of BanglishBERT and the sequence modeling prowess of GRU, providing a robust framework for text classification. The GRU is supplemented by a dropout layer to prevent overfitting and a linear layer that maps the GRU’s output to our label space. Aditionally, to manage class imbalance, weights were dynamically adjusted during training. Common classes received lower weights and less frequent classes were highly weighted. Finally, the model was evaluated and tested with different matrices.

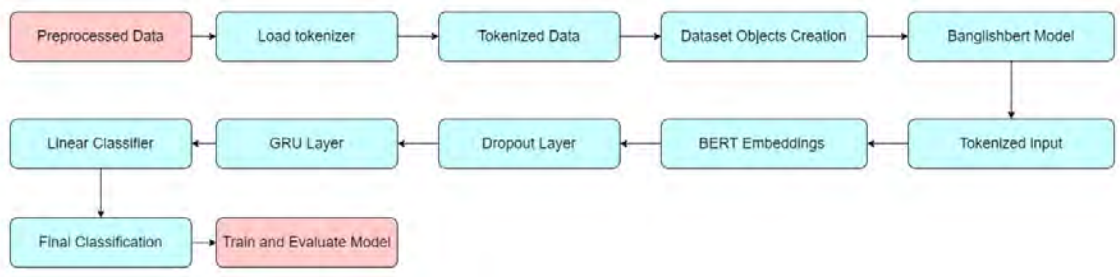


Figure 5.11: BanglishBERT + GRU Architecture

Performance Evaluation

After training the model, we have got 79.15% of the accuracy, 79.10% of F1 score, a precision of 80.58%, and 77.67% of recall.

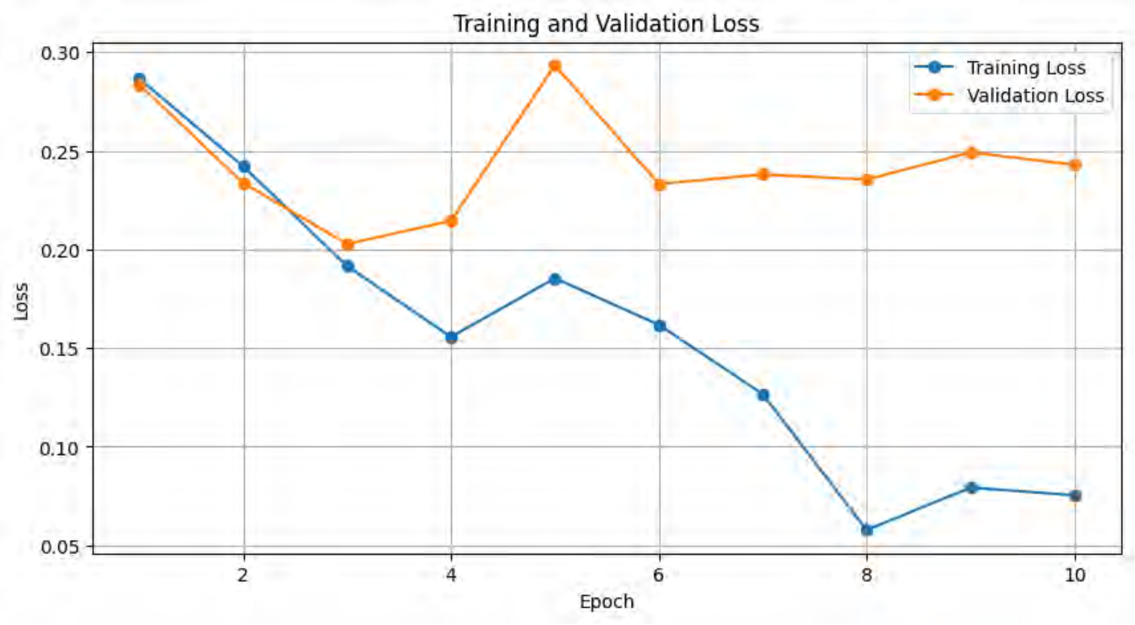


Figure 5.12: BanglishBERT + GRU Train and Validation Loss Graph

Placing the training and validation loss for the Banglishbert + GRU model in a graph, we can observe that both training and validation loss is decreasing over each epoch. As the number of epochs increases, the training loss decreases. 10 epochs are run on the model and after around 8 epochs, the training loss stabilizes.For validation loss, it shows an initial decrease but after epoch 4, it rises sharply. But after this spike, it steadily decreases through the remaining epochs. The graph indicates that the model is learning and generalizing well from the training data to the unseen validation data.

Chapter 6

Result Comparative Study

6.1 Binary classification

By evaluating Banglabert model, we see that our binary classification model provides a notable accuracy of 90.79%. And a loss score of 0.284 shows that the model is performing efficiently minimizing errors. Furthermore, the F1 score of 71.65% shows good balance of the model with Precision and recall scores of 77.18% and 66.86%.

6.2 Multi-label classification

Model Names	Accuracy (%)	F1 Score(%)	Precision(%)	Recall(%)
BERT	83.3795	82.8185	83.7073	81.9484
BERT + Bi-LSTM	82.2022	81.6981	82.6336	80.7836
MuRIL + Bi-LSTM	71.3147	77.2357	78.8381	75.6972
BanglishBERT + GRU	79.1545	79.0984	80.5846	77.6660

Table 6.2: Evaluation Metrics for Multi-label Classifications of different models

From the above table, we can analyse that BERT model achieves the highest accuracy, making it the most overall correct model in prediction across all instances. The MuRIL + Bi-LSTM model lags significantly in accuracy, suggesting it might be less effective for the task or dataset used.

While comparing the precision, BERT offers the best precision whereas, MuRIL + Bi-LSTM shows the lowest precision among the four models. For recall, the BERT model leads with the highest recall whereas, MuRIL + Bi-LSTM has recall percentage which is the lowest.

Now, the F1 score, balancing the precision and recall, is highest for the BERT model which indicates that it manages a balanced trade-off between precision and recall most effectively. The MuRIL + Bi-LSTM model that has the lowest F1 score which suggests a less optimal balance between precision and recall.

Seeing the overall result, the BERT model outperforms the other models in almost all matrics, suggesting it is the most balanced and effective model to detect sexism from code-mixed and code-switched comments. BERT + Bi-LSTM follows closely but still lags slightly behind the standalone BERT in every metric. The MuRIL + Bi-LSTM model shows notable weakness across all metrics, which might indicate that this particular architecture is not well suited to the dataset or task. The BanglishBERT + GRU model presents itself as a moderately effective model, offering decent performance across the board but it does not excel in any particular metric compared to the BERT models. We have also trained our dataset on other models but they do not show a satisfactory outcome.

Chapter 7

Limitations

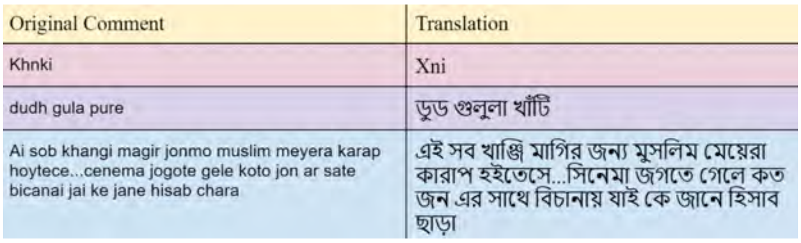
Imbalanced Dataset

Our dataset had fewer occurrences of class 1 and 4. This means our dataset wasn't fully balanced. We did not oversample our dataset because we wanted to have an authentic dataset that captured real situations on online platforms.

Quality of Translation

For the majority of our models, we translated our code-mixed and code-switched comments into bangla so that our models understand the data better. For translation, we used Googletrans which is a free library that implements Google Translate API. Issues arised with slang, idiomatic expressions, etc. Therefore, the translation had its deficiency which might have introduced noise and errors in our dataset which eventually affected our model training.

Content warning: Please view the examples with caution. These are real comments from out dataset. These examples might cause discomfort to some.



Original Comment	Translation
Khnki	Xni
dudh gula pure	ডুড গুলুলা খাঁটি
Ai sob khangi magir jonmo muslim meyera karap hoytece...cenema jogote gele koto jon ar sate bicanai jai ke jane hisab chara	এই সব খাঞ্জি মাগির জন্য মুসলিম মেয়েরা কারাপ হইতেসে...সিনেমা জগতে গেলে কত জন এর সাথে বিচানায় যাই কে জানে হিসাব ছাড়া

Table 7.1: Distinction between Original and Translated Comments

Availability of Resource

Most pre-trained models and NLP tools are developed for major languages and do not support occurrences with code-mixed and code-switched characteristics. This lack of resources hindered the development of an effective solution.

Limitation in Data Diversity

Even though we collected our dataset from Social Media like Youtube and Facebook which made our dataset somewhat robust, we still might have missed some nuances of code-mixed and code-switched data. Our dataset might not have captured every type of variability possible and this might have affected our results. Therefore, further improvement is obviously required.

Continual Data Update Necessity

Languages are constantly evolving and so is every social media platform. Our dataset is not dynamic sadly. In the near future, if new data arises, our dataset might lose its validity and can be misleading. Slangs, idioms, spellings and various other nuances in languages are changing day by day. New abbreviations might be added. Our dataset will not have these variabilities and might be obsolete in the future.

Difficulty in Capturing Language Subtlety

Our models might have suffered from language subtlety. For instance, it might have mixed sarcastic comments with serious ones. Distinguishing between these two types of comments is indeed very challenging. Although a difficult task, accurately distinguishing between these two types of nuances would indeed bring better results.

Chapter 8

Future Work

In the field of Natural language Processing and deep learning, there is significant potential to enhance the performance of detecting sexism in languages including codemixed and code switched cases. So we aim to build up on our existing work in the future. Such areas are-

Dataset Diversity: Our model is currently utilized on English, Bangla, codemixed and code switched languages. In the future, we want to expand our dataset in terms of language variety and will experiment with models to get the best possible performance on other languages as well.

Data collection method: Our existing dataset is collected from scratch, custom made and was annotated over a long period of time. With this baseline work we would like to utilize methods in the future so that we can collect data dynamically. So that current and reflective contemporary language trends are fed into the models. So that our models can keep up with the evolving languages.

Review System: In the future we would like to implement a real-world application. This will allow us to evaluate our model in social media and adapt the model. A review or feedback system can be implemented here so that users can report false positive and negatives.This will give opportunity for continuous improvement of the detection model thus enhancing its performance.

Finally, more fine tuning and experimentation will be done using various deep learning models to get better performance. Our goal is not only to detect overt instances of sexism but also to improve the model's capability to identify sarcastic or subtly expressed sexist remarks with higher accuracy

Chapter 9

Conclusion

The detection of sexist code-mixed and code-switched data is a crucial task that plays a significant role in combating online discrimination in present days. Our thesis has explored the challenges associated with identifying sexism in multilingual platforms like Youtube and Facebook. Since we are able to detect sexism effectively, in the near future, our research might help to mitigate the effects of these sexist comments from online platforms. Sexist comments targeted to a particular person or group are always detrimental for the mental health of the victims. Every person has different sensitivity levels. If sexist comments crosses a certain boundary, some victims might take drastic measures due to not being able to take the trauma. They might commit suicide. So, it's extremely crucial to remove these comments from online platforms because we want a peaceful world where people are cordial with each other in social media platforms. We believe our authentic dataset will help future researchers to further their research in this intriguing field of study. We explored several models and gave the results of the models that performed best. But there is always scope for improvement. Improving the performance of the models is our near future goal. We faced several limitations. We hope to mitigate these limitations in the future. We hope to further study and improve our research. Maybe we will be able to achieve results in the 90th percentile in the future. Even though our aims are novel, we have a strong desire to accomplish it. We will keep exploring the existing literature and research in this field to identify any nuances we might have missed. Overall, we believe our research will provide valuable insights into the various approaches and methodologies employed for sexism detection in code-mixed and code-switched data effectively for future researchers.

Chapter 10

Bibliography

[1] Waseem, Z., Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).

[2] Waseem, Z. (2016, November). Are you a racist or am I seeing things? annotator influence on hate speech detection on twitter. In Proceedings of the first workshop on NLP and computational social science (pp. 138-142).

[3] Park, J. H., Fung, P. (2017). One-step and two-step classification for abusive language detection on twitter. arXiv preprint arXiv:1706.01206.

[4] Davidson, T., Warmsley, D., Macy, M., Weber, I. (2017, May). 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, pp. 512-515).

[5] Jha, A., Mamidi, R. (2017, August). When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In Proceedings of the second workshop on NLP and computational social science (pp. 7-16).

[6] Rodríguez-Sánchez, F., Carrillo-de-Albornoz, J., Plaza, L. (2020). Automatic classification of sexism in social networks: An empirical study on twitter data. IEEE Access, 8, 219563-219576.

[7] Al-Hassan, A., Al-Dossari, H. (2019, February). Detection of hate speech in social networks: a survey on multilingual corpus. In 6th international conference on computer science and information technology (Vol. 10, pp. 10-5121).

[8] Rodríguez-Sánchez, F., Carrillo-de-Albornoz, J., Plaza, L. (2020). Automatic classification of sexism in social networks: An empirical study on twitter data. IEEE Access, 8, 219563-219576.

[9] Wullach, T., Adler, A., Minkov, E. (2020). Towards hate speech detection at large via deep generative modeling. IEEE Internet Computing, 25(2), 48-57.

[10] Glick, P., Fiske, S. T. (1997). Hostile and benevolent sexism: Measuring

ambivalent sexist attitudes toward women. Psychology of women quarterly, 21(1), 119-135.

[11] Barman, U., Das, A., Wagner, J., Foster, J. (2014, October). Code mixing: A challenge for language identification in the language of social media. In Proceedings of the first workshop on computational approaches to code switching (pp. 13-23).

[12] Barman, U., Das, A., Wagner, J., Foster, J. (2014, October). Code mixing: A challenge for language identification in the language of social media. In Proceedings of the first workshop on computational approaches to code switching (pp. 13-23).

[13] Sitaram, D., Murthy, S., Ray, D., Sharma, D., Dhar, K. (2015, July). Sentiment analysis of mixed language employing Hindi-English code switching. In 2015 International Conference on Machine Learning and Cybernetics (ICMLC) (Vol. 1, pp. 271-276). IEEE.

[14] Sequiera, R., Choudhury, M., Bali, K. (2015, December). Pos tagging of hindi-english code mixed text from social media: Some machine learning experiments. In Proceedings of the 12th international conference on natural language processing (pp. 237-246).

[15] Chanda, A., Das, D., Mazumdar, C. (2016, November). Unraveling the English-Bengali code-mixing phenomenon. In Proceedings of the second workshop on computational approaches to code switching (pp. 80-89).

[16] Patra, B. G., Das, D., Das, A. (2018). Sentiment analysis of code-mixed indian languages: An overview of sail$_c ode-mixedsharedtask@icon-2017.arXivpreprint-arXiv : 1803.06745$.

[17] Ansari, M. A., Govilkar, S. (2018). Sentiment analysis of mixed code for the transliterated hindi and marathi texts. International Journal on Natural Language Computing (IJNLC) Vol, 7.

[18] Chakravarthi, B. R., Muralidaran, V., Priyadharshini, R., McCrae, J. P. (2020). Corpus creation for sentiment analysis in code-mixed Tamil-English text. arXiv preprint arXiv:2006.00206.

[19] Maharjan, S., Blair, E., Bethard, S., Solorio, T. (2015, June). Developing language-tagged corpora for code-switching tweets. In Proceedings of The 9th Linguistic Annotation Workshop (pp. 72-84).

[20] Gupta, A., Menghani, S., Rallabandi, S. K., Black, A. W. (2021). Unsupervised self-training for sentiment analysis of code-switched data. arXiv preprint arXiv:2103.14797.

[21] Hate speech. Cambridge Dictionary. (n.d.).https://dictionary.cambridge.org/dictionary/english/hate-speech

[22] Kirk, H. R., Yin, W., Vidgen, B., Röttger, P. (2023). SemEval-2023 Task 10: Explainable Detection of Online Sexism. arXiv preprint arXiv:2303.04222.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[24] Malingan, N. (2023) Gated Recurrent Unit (GRU), Scaler Topics. Available at: https://www.scaler.com/topics/deep-learning/gru-network/ (Accessed: 20 May 2024).

[25]Saha, A. K., Arnob, N. M. K., Rahman, N. N., Haque, M., Al Masud, S. M. R., Rahman, R. (2023). Mukh-Oboyob: Stable Diffusion and BanglaBERT Enhanced Bangla Text-to-Face Synthesis. International Journal of Advanced Computer Science Applications, 14(11).

[26] Xie, J., Chen, B., Gu, X., Liang, F., Xu, X. (2019). Self-attention-based BiLSTM model for short text fine-grained sentiment classification. IEEE Access, 7, 180558-180570.

[27] Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., ... Talukdar, P. (2021). Muril: Multilingual representations for indian languages. arXiv preprint arXiv:2103.10730.

[28]Bhattacharjee, A., Hasan, T., Ahmad, W. U., Samin, K., Islam, M. S., Iqbal, A., ... Shahriyar, R. (2021). BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. arXiv preprint arXiv:2101.00204.