Offensive Text Detection in Code-mixed Dravidian Languages Towards Marginalized Groups and Women

A Thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech.

in

Computer Science and Engineering

By

Joshua Mahadevan (106120047)

Lokkamithran M (106120061)

Mubeena (106120071)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI – 620015

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **Offensive Text Detection in Code-mixed Dravidian Languages Towards Marginalized Groups and Women** is a Bonafide record of the work done by

Joshua Mahadevan (106120047)

Lokkamithran M (106120061)

Mubeena (106120071)

in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2023-24.

[illegible]

Dr. C. Oswald	**Dr. S. Mary Saira Bhanu**
Guide	Head of the Department

Project Viva-voce held on: ___________________________

Internal Examiner	**External Examiner**

ABSTRACT

In recent times, the proliferation of sharing content and expressing opinions across various online platforms has become very common. Individuals actively engage in articulating their perspectives on diverse subjects, often resorting to commenting on shared content. This textual discourse has evolved considerably, incorporating elements such as text emoticons and multimedia content. Moreover, the trend of code-mixing, blending English with regional languages, has emerged as a preferred mode of personal expression. This phenomenon poses a significant challenge for current language processing Models, necessitating Novel approaches for understanding, analysing, and generating Code-mixed data.

While existing research has addressed this challenge to some extent, there remains a notable gap, particularly in the exploration of code-mixed data in Dravidian languages. Additionally, the prevalence of offensive content targeting Marginalized groups and Women on Social Media platforms increases the urgency for effective detection mechanisms.

This research project aims to develop a robust system capable of identifying potentially offensive text towards marginalised groups and women. This is achieved by using a Code-Mixed Data Corpus available online [REF] and manually Annotating it to suit our objective. Furthermore, we conduct a comparative analysis of various Deep Learning Models and Large Language Models to evaluate their efficiency in processing code-mixed data. Performance metrics such as Accuracy, F1 Score, Recall, and Precision are employed to assess the capabilities of these models. Through this project, we seek to contribute to the advancement of Natural Language Processing research in Code-Mixed Data and foster the development of more inclusive and socially responsible digital environments.

Keywords: Code-mixed Data, Dravidian Languages, Online Platforms, Offensive Content Detection, Annotation, Deep Learning models, Large Language Models, Evaluation Metrics.

ACKNOWEDGEMENTS

We would like to express our sincere gratitude to our project guide, Dr. C. Oswald, for his valuable guidance and support throughout this final year project. He has been a constant source of inspiration and encouragement for us and has helped us overcome many challenges and difficulties. We are grateful for his expertise, patience and constructive feedback that enabled us to improve our work and achieve our goals. He has provided us with useful resources, insights and suggestions that enhanced our understanding and skills in the field of study. He has also been very supportive and flexible in accommodating our needs and requests. We would further like to thank the Project Committee for their time and effort in reviewing our project and giving us constructive criticism and feedback. We want to express our gratitude to them for providing us with useful recommendations which have shaped this project.

Finally, we would like to thank Dr. G. Aghila, Director, National Institute of Technology, Tiruchirappalli, for giving us this opportunity to work on this project.

This project has been a rewarding and memorable experience for us, and we have learned a lot from it. We are proud of what we have accomplished, and we hope that our project will contribute to the advancement of knowledge.

Joshua Mahadevan (106120047)

Lokkamithran M (106120061)

Mubeena (106120071)

TABLE OF CONTENTS

Title		Page No.
ABSTRACT		ii
ACKNOWLEDGEMENT		iii
TABLE OF CONTENTS		iv
LIST OF FIGURES AND TABLES		vi
CHAPTER 1	**INTRODUCTION**	1
1.1	Introduction to the Problem Statement	1
1.2	Motivation	1
1.3	Existing Solutions and their Gaps	2
1.4	Problem Statement	4
1.5	Objectives	5
CHAPTER 2	**LITERATURE SURVEY**	6
2.1	Related Work	6
2.2	Research Gaps and Challenges	8
CHAPTER 3	**METHODOLOGY**	13
3.1	Overview	13
3.2	Architecture Diagram of the Proposed Work	14
3.2.1	Annotating the Code-Mixed Tamil Corpus	15
3.2.2	Data Pre-processing	16
3.2.3	Deep Learning Model	16
3.2.4	Machine Learning Model	19
3.2.5	Classification Task	20
3.3	Classification of an Input Sequence	20

CHAPTER 4 RESULTS AND DISCUSSION 22

4.1 Experimental Setup 22

4.2 Results 23

4.2.1 Annotation Results 23

4.3 Evaluation Metrics 25

4.4 Analysis 28

CHAPTER 5 SUMMARY AND CONCLUSION 29

5.1 Summary 29

5.2 Conclusions 29

5.3 Scope for Future Work 30

REFERENCES 31

APPENDIX 35

LIST OF FIGURES AND TABLES

FIGURES:		**Page no**
3.1	Architecture Diagram	14
3.2	Input Sentence Illustration	21
4.1	Visualisation of Performance Metrics for Binary Classification (Deep Learning Models)	27
4.2	Visualisation of Performance Metrics for Multi-Label Classification (Machine Learning Models)	27

TABLES:

2.1	Literature Survey	9
4.1	Dataset before Annotation	23
4.2	Dataset after Annotation	23
4.3	Classification Labels for Example Sentences	24
4.4	Evaluation Metrics of Machine Learning Models	26
4.5	Evaluation Metrics of Deep Learning Models	26
4.6	Evaluation Metrics of Combination Deep Learning Models and Machine Learning Models	26

CHAPTER 1

INTRODUCTION

The following chapter gives an idea about the problem statement and states the Motivation and Objectives that we have set out to achieve through this project.

1.1 Introduction to the Problem statement

Code-mixing is broadly defined as the mixing of two or more languages or language variants within a single utterance or text. This has become a common feature of online communication and is widely found in many online platforms and social media. It is often used by people to express their emotions, opinions, thoughts and beliefs using references from other languages. What makes it widespread across global platforms is the flexibility it gives to the users to use their own language to be mixed with a widely used language which gives a sense of relatability.

The rise of online platforms has increased the ease of communication and has given everyone freedom of expression and thought. This same freedom makes these online platforms very vulnerable to the spread of hate and offensive content which might be targeted towards marginalized groups and women, which makes them feel unsafe in such platforms. Our problem statement helps with mitigating such offensive content by identifying it, and the particular group to which it might be offensive.

1.2 Motivation

In recent years, the digital landscape has become increasingly intertwined with societal dynamics. Even in times of trouble, people resort to Social Media platforms for staying up to date with the latest news. However, alongside the proliferation of technology and online platforms, a troubling trend has emerged: the widespread dissemination of offensive content targeting various individuals and groups within these linguistic spaces. Marginalized groups and women, in particular, are disproportionately affected by online harassment and discrimination, facing threats to their safety and well-being. This reality underscores the critical need for effective mechanisms to identify and mitigate such harmful content, creating a digital environment where all individuals can participate freely without fear of being targeted or marginalized based on their identity.

At the heart of this endeavour lies a commitment to protecting vulnerable communities and upholding principles of inclusivity and diversity. Code-mixing reflects the nuanced tapestry of languages and cultures present in online communities, but it also presents opportunities for the spread of hate speech and discriminatory attitudes. By actively working to combat offensive content within code-mixed communication, we seek to ensure that linguistic diversity is celebrated and respected, rather than exploited for harmful purposes. This fosters a more welcoming and inclusive digital environment where users feel valued and accepted for who they are, contributing to a sense of belonging and mutual respect among online participants.

Furthermore, social media is constant source of hatred and it is often misused. Social media comments can harm by contributing to cyberbullying, spreading misinformation, hate speech, trolling, privacy violations, coercion, and online harassment. Despite platforms' efforts to moderate, harmful comments can still evade detection, underlining the challenge in addressing misuse.

By actively working to mitigate offensive content, we not only promote healthier online interactions but also contribute to the creation of a more informed and engaged citizenry. When users feel safe and respected in digital spaces, they are more likely to participate in meaningful discussions on important social issues, leading to greater awareness and collective action. This, in turn, fosters a culture of mutual understanding and collaboration, where diverse perspectives are valued and leveraged to address complex challenges facing society.

1.3 Existing Solutions and their Gaps

Various strategies are proposed to detect hate speech in multilingual contexts. The MoH pipeline [1] combines language identification, transliteration, and fine-tuned BERT and MuRIL models to address hate speech in Hindi-English code-switched text. Another approach combines SentBERT [5] with LSTM and KNN-based pipelines for offensive speech detection in code-mixed dialogue, emphasizing contextual understanding.

Additionally, a novel technique called minority positive sampling [6] enhances statistical language modeling by strategically sampling minority language instances, improving model pe `1rformance in code-mixed scenarios. In Dravidian languages, offensive language identification [8] employs zero-shot and few-shot learning techniques with models like XLM-RoBERTa and mBERT, alongside selective translation and transliteration methods to effectively identify offensive language in low-resource settings.

These strategies leverage advanced models and techniques to address hate speech detection challenges in multilingual and code-switched contexts. The MoH pipeline integrates language identification, transliteration, and fine-tuned BERT and MuRIL models, while another approach combines SentBERT with LSTM and KNN-based pipelines to emphasize contextual understanding in offensive speech detection.

Additionally, minority positive sampling enhances statistical language modeling performance in code-mixed scenarios, and in Dravidian languages, zero-shot and few-shot learning techniques with models like XLM-RoBERTa and mBERT, alongside selective translation and transliteration methods, effectively identify offensive language in low-resource settings.

Below are the gaps in the existing solutions which we aimed to address through our project:

1) Limited Focus on Targeted Audience: While offensive text detection in code-mixed language is addressed, there's a lack of emphasis on categorizing hate text based on the target audience. Understanding whom the offensive comments target could enhance hate text detection and identification, contributing to a safer and more inclusive online environment.

2) Insufficient Representation of Marginalized Groups: The existing solutions may not adequately represent the diverse experiences and vulnerabilities of marginalized groups. There's a need for more targeted research and data collection efforts to capture the nuances of offensive content targeting specific demographic groups.

3) Scalability and Generalizability: Some approaches may lack scalability and generalizability, particularly in low-resource settings or across different Dravidian languages. Enhancements in model robustness and adaptation to diverse linguistic contexts are essential to ensure broad applicability and effectiveness.

1.4 Problem statement

Our Problem statement deals with collecting Offensive comments and annotating them and training and testing various Deep Learning Models and Machine Learning Models to get the best accuracy and efficiency.

As part of the problem statement, we have recognized and made a list of targeted groups and the different forms in which such offense can exist.

The following categories will be focused on in this study.

- Offense towards Women
- Offense towards Religion
- Offense towards Caste
- Offense towards Race
- Offense towards Handicapped individuals
- Offense towards LGBTQ+ community

The model should also be able to detect general offense which doesn't fall under these categories and detect if it's not offensive too.

Now we re-define our problem statement for better understanding:

Problem statement: Annotating Dataset according to the defined classes, detecting Offensive text present in Code-Mixed Tamil Language by initially performing Binary Classification which predicts if the comment is offensive or not, followed by Multi-Label Classification which classifies into one of the 7 categories if detected as offensive and analysing the performance.

1.5 Objectives of the proposed work

Objectives outlined for the above problem statement are:

1) **Develop an Annotated Dataset**: This objective involves creating a comprehensive dataset of code-mixed text in Dravidian languages, annotated with labels indicating offensive content targeted towards marginalized groups and women. This dataset serves as the foundation for training and evaluating the offensive speech detection model, providing a diverse and representative sample of online communication.

2) **Create an Offensive Speech Detection Model:** Building upon the annotated dataset, the next objective is to develop a model capable of accurately detecting offensive speech in code-mixed Dravidian language text. This involves implementing state-of-the-art Natural Language Processing (NLP) techniques, such as Deep Learning and Transformer models, to effectively capture the nuanced characteristics of offensive language in a multilingual context.

3) **Evaluate and Benchmark Model Performance**: Once the offensive speech detection model is trained, the third objective is to evaluate its performance using standard evaluation metrics. This includes measuring the model's accuracy, precision, recall, and F1-score on a separate test dataset to assess its effectiveness in identifying offensive content. Benchmarking the model against existing approaches and baselines helps gauge its comparative performance and identify areas for improvement.

By achieving these objectives, the proposed offensive text detection tool aims to enhance diversity and inclusion in online communities by addressing cultural sensitivity and fostering positive interactions. Moreover, by focusing on code-mixed Dravidian languages, the tool contributes to preserving linguistic diversity and empowering users from underrepresented linguistic backgrounds to participate safely in online discourse.

CHAPTER 2

LITERATURE SURVEY

This chapter focuses on providing essential background knowledge. Hence to learn about pre-existing issues where there can be an improvement, a detailed discussion of each paper we have reviewed.

2.1 Related Work

Current measures to combat online hate speech are insufficient, particularly in languages with scarce hate speech detectors. Arushi Sharma et.al (2022) [1] devised a new approach focusing on analysing hate speech in Hindi-English code-switched language. This method, called 'MoH' (Map Only Hindi), employs transformation techniques to represent text accurately. It includes language identification, transliteration, and utilizes fine-tuned Multilingual Bert and MuRIL language models. Quantitative experiments on three datasets showed 'MoH' improved F1 scores by 13% with classical ML models, by 6% compared to baseline models, and by 15% over existing transliteration libraries.

Hiren Madhu et.al (2023) [5] recognize that context is crucial in understanding communication. Previous approaches treated hate speech recognition solely as a text classification problem, ignoring contextual cues. Their work extends prior efforts by considering both the current and preceding tweets in a conversation. They introduce a method for extracting context, focusing on code-mixed Hindi. They demonstrate that including context improves accuracy with their best pipeline producing a macro F1 score of 0.892.

Arindam Chatterjee et.al (2020) [6] experimented with both statistical and neural language modelling techniques, highlighting switching points in code-mixed text as the main challenge leading to performance drops compared to monolingual models. To address this, they introduce minority positive sampling to improve performance, yielding a perplexity of 139 for Hinglish code-mixed Language Modelling.

Siva Sai et.al (2021) [8] concluded detecting offensive language in Dravidian languages is difficult due to limited resources. They used zero-shot and few-shot learning approaches to detect offensive speech in these languages. They introduce a novel method of selective translation and transliteration to improve results from models like XLM-RoBERTa and mBERT. Additionally, the study experiments with transfer learning techniques across languages and tasks to enhance model performance using resources from the English language and related tasks.

To address the lack of real code-mixing data, L3Cube-HingCorpus [12] offers large-scale Hindi-English code-mixed data from Twitter. Models like HingBERT are pre-trained on this corpus for tasks like sentiment analysis and language identification. HingGPT, a GPT2-based model can generate full tweets. Additionally, L3Cube-HingLID Corpus and HingBERT-LID model are released for language identification.

Pratapa et.al (2018) [13] compare existing bilingual word embedding methods with a new one trained on synthetic code-mixed text. Results show the new approach consistently outperforms existing methods in sentiment analysis and part-of-speech tagging for code-mixed text. This suggests a need for tailored multilingual word embeddings for code-mixed text processing.

Sentiment analysis tools for European languages are well-developed, but scarce for Indian languages due to a lack of pre-processing tools like POS taggers and shallow parsers. Additionally, the rise of social media has led to code-mixed content, where English and regional languages are mixed, posing a challenge for sentiment analysis. Mahata et.al [16] aim to address this challenge by developing a sentiment analysis model for English-Tamil code-mixed data using bi-directional LSTMs and language tagging.

Kusampudi et.al (2021) [18] discuss the challenges of sentiment analysis on Code-Mixed Telugu-English Text (CMTET) due to its unstructured nature stemming from informal language, transliterations, and spelling errors. The authors introduce an annotated dataset for sentiment analysis in CMTET and achieve an accuracy of 80.22% using a novel unsupervised data normalization technique with a Multilayer Perceptron (MLP) model. They propose that this technique can be applied to various NLP tasks involving CMTET and report a 2.53% increase in accuracy due to this data normalization approach.

The aim of code-mixed language identification (LID) is to determine the language used in a given piece of text, including cases where multiple languages are mixed. Yigezu et.al (2022) [19] employ the CoLI-Kenglish dataset, comprising English, Kannada, and mixed-language words, for model training. Through a series of experiments, they found that the Bi-LSTM model yielded the best results, achieving an F1-score of 0.61%.

Thara et.al (2018) [21] examine CM's role in various Natural Language Processing (NLP) tasks such as language identification, Part-of-Speech (POS) tagging, Named Entity Recognition (NER), and others. Additionally, the paper touches on CM's applications in Machine Translation, Dialect identification, and Speech technologies, and discusses trends, techniques, and evaluation measures for accuracy.

2.2 Research Gaps and Challenges

The several key issues prevalent in the current landscape of offensive speech detection in code-mixed Dravidian languages are:

Focus on Binary Classification: Most existing research in offensive speech detection simplifies the problem by treating it as a binary classification task: offensive or not offensive [5][8]. This approach fails to capture the nuances and complexities of offensive language, particularly when it comes to targeting specific marginalized groups within a community.

Neglect of Marginalized Groups: By predominantly focusing on hate speech that targets a broad audience or generic hate, the current research overlooks the distinct forms of discrimination and abuse faced by marginalized groups such as different races, religions, sexual orientations, and disabilities [1][8][12].

Lack of Attention to Gender-Based Hate: Another critical blind spot in existing research is the limited attention given to hate speech specifically targeted at women [5][8][16][18]. Gender-based hate speech is a pervasive issue online and offline, yet it often receives inadequate consideration in offensive speech detection studies.

The Literature Survey of the reviewed papers is in Table 2.1.

Table 2.1: Literature Survey

Paper	Publication	Concepts/Techniques Used	Merits	Demerits	Datasets	Performance
Ceasing hate with MoH: Hate Speech Detection in Hindi-English Code-Switched Language [1]	Elsevier, January 2022	Employs a 'MoH' or (Map Only Hindi) pipeline which consists of language identification, Roman to Devanagari script transliteration, and employs the fine-tuned Multilingual BERT and MuRIL language models.	MoH is a novel architecture which takes advantage of the MBERT and MURIL models which have been pre-trained for this purpose.	This approach only deals with the specific case of Hindi-English code-mixed hate speech.	HOT (Mathur et al., 2018) [2] HS (Bohra et al., 2018) [3] TRAC-I (Kumar et al., 2018b) [4]	They achieved a peak F1 score of 0.71, 0.90 and 0.85 with the three datasets, respectively.
Detecting offensive speech in conversational code-mixed dialogue on social media: A contextual dataset and benchmark experiments [5]	Elsevier, April 2023	They have scraped dialogues from social media and annotated them to include context. This dataset was then experimented on with various LLMs (SentBERT) and Classifiers (LSTM) to find the impact of context in determining offensive text.	Introduces a clearly defined way of extracting context from conversational code-mixed dialogues. Presents the first dataset for conversation-based Hate Speech classification.	Their ICHCL dataset only contains contextual tags for Hindi-English code-mixed data.	ICHCL (Identification of Conversational Hate-Speech in Code-Mixed Languages) dataset [5]	Their best performing pipeline achieves a F1 score of 0.89 in classifying Hindi-English code-mixed text.

Minority Positive Sampling for Switching Points - an Anecdote for the Code-Mixing Language Modelling [6]	ACL Anthology, May 2020	Analysis shows code-switching points are the main challenge for performance drops. Introduces minority positive sampling to selectively induce samples for better performance.	Selectively adding more switching points in the training data for the model without adding noise to improve the perplexity (or bias) of the model.	Minority Positive Sampling is not tested against Dravidian code-mixed data for better performances.	ICON 2017 Hinglish sentiment analysis dataset (Patra et al., 2018) [7]	SPS applied to Eng-Hin switching points reduces model perplexity to 650 from 730 for 100k samples.
Towards Offensive Language Identification for Dravidian Languages [8]	ACL Anthology, April 2021	Implements zero-shot and few-shot learning paradigms for low resource languages with pre-trained, fine-tuned and ensembled variants of the XLM-RoBERTa model.	This paper focuses on Dravidian languages and provides a helpful way to deal with their low resource availability.	It only focuses on binary classification of Hate Speech.	HASOC-Dravidian CodeMix-FIRE 2020 [9] Sentiment Analysis for Dravidian Languages in Code-Mixed Text [10] Offensive Language Identification in Dravidian Languages [11]	They achieved F1 scores of 0.90, 0.79 and 0.82 for Tanglish, Manglish and Kanglish

L3Cube-HingCorpus and HingBERT: A Code Mixed Hindi-English Dataset and BERT Language Models [12]	ACL Anthology, June 2022	They developed a large scale code-mixed dataset and trained various LLMs on the corpus. They also analyse the effectiveness of these models in various tasks like sentiment analysis.	Developed HingCorpus, the first large-scale Hindi-English code-mixed data with over 52M sentences.	This approach only deals with Hindi-English code-mixed data.	L3Cube-HingCorpus [12]	The models achieve a F1 score of 0.86 in language identification and 0.66 in sentiment analysis.
Word Embeddings for Code-Mixed Language Processing [13]	ACL Anthology, October 2018	They compare popular bilingual embedding techniques on two code-mixed related tasks - sentiment analysis and POS tagging.	Developed a novel word embedding technique which outperforms existing embedding methods.	They do not focus on hate speech classification.	English-Spanish parallel corpora [14] Synthetic CM data [15]	Achieved a F1 score of 0.64 in sentiment analysis and accuracy of 84.9% in POS tagging.
Sentiment Classification of Code-Mixed Tweets using Bi-Directional RNN and Language Tags [16]	ACL Anthology, April 2021	They used FastText [17] and language tagging with Bi-LSTMs to accurately classify the sentiment of code-mixed Tamil-English data.	Proposes using Bi-LSTMs to mitigate the low resources available for Tamil-English code-mixed data.	Focuses on sentiment analysis rather than on offensive classification.	Dravidian-CodeMix - FIRE 2020 [10]	Garners precision, recall, and F1 scores of 0.59, 0.66, and 0.58 respectively.

Sentiment Analysis in Code-Mixed Telugu-English Text with Unsupervised Data Normalization [18]	ACL Anthology, September 2021	They collect their data from online sources, transliterate it and perform data normalization followed by feature extraction and sentiment classification.	Developed an annotated dataset for code-mixed Telugu-English text. Improved accuracy with novel data normalization technique.	Only focuses on Telugu-English code-mixed text.	Introduced a new dataset for CMTET (Code-Mixed Telugu-English Text) [18]	Accuracy of 80.22% with a improvement of 2.5% due to their data normalization technique.
Word Level Language Identification in Code-mixed Kannada-English Texts using Deep Learning Approach [19]	ACL Anthology, December 2022	Experimented and decided on the best performing model in language identification tasks in Kannada-English code-mixed text by training on a dataset containing English, Kannada and mixed-language words.	They experimented with the dataset and performed hyper parameter tests to decide on the parameters that maximized the model's performance.	Word level identification is not performed on Tamil-English code-mixed data.	CoLI-Kenglish dataset (Hosahalli Lakshmaiah et al., 2022) [20]	Obtained a macro F1 score of 0.61 with Bi-LSTM.
Code - Mixing: A Brief Survey [21]	IEEE, September 2018	Highlights a study of CM in the fields of Natural Language Processing (NLP) including language identification, Part-of-Speech (POS) tagging, Named Entity Recognition (NER), Polarity Identification and Question Answering.	Develop unique and novel evaluation measures to derive a model's accuracy. Offers analysis and comparisons of various related articles on code-mixed text.	Less focus on Dravidian Code-Mixed data.	Social Media Code-Mixed Corpora [22] Bengali-English Code-Mixed Corpus [23]	Many approaches that are compared yield F1 scores greater than .90

CHAPTER 3

METHODOLOGY

This chapter talks about the methodology employed to build the model. It contains the proposed methodology diagram with an analysis on every part of the model and its significance.

3.1 Overview

The model has been divided into 3 phases:

- **Annotation Phase:** The first phase involves annotating the Code-Mix Tamil Corpus with the relevant categories for classification, such as offensive language targeting women and marginalized groups. Additionally, preprocessing steps are applied to the data to ensure consistency and quality, including tokenization, normalization, and removal of noise or irrelevant information. This stage lays the foundation for subsequent model training by providing labelled data that accurately represents the categories of offensive content present in the corpus.
- **Training and Testing Phase**: In the second phase, Deep Learning and Machine Learning models are trained, tested, and evaluated using the annotated and pre-processed data. This involves selecting appropriate architectures and algorithms, such as recurrent neural networks (RNNs), Large language models (LLM's), or support vector machines (SVMs), and fine-tuning them to achieve optimal performance on the task of offensive content classification. The models are trained on a portion of the annotated data, and finally tested on a holdout set to assess their generalization ability and effectiveness in accurately identifying offensive content.
- **Classification and Performance analysis:** The final phase entails concatenating the results of the models to predict the category of offense for each instance of code-mixed text. By integrating the outputs of multiple models,

we aim to enhance the overall accuracy and reliability of the offense classification process and decreasing the computational time as well.

3.2 Architecture Diagram of the proposed work

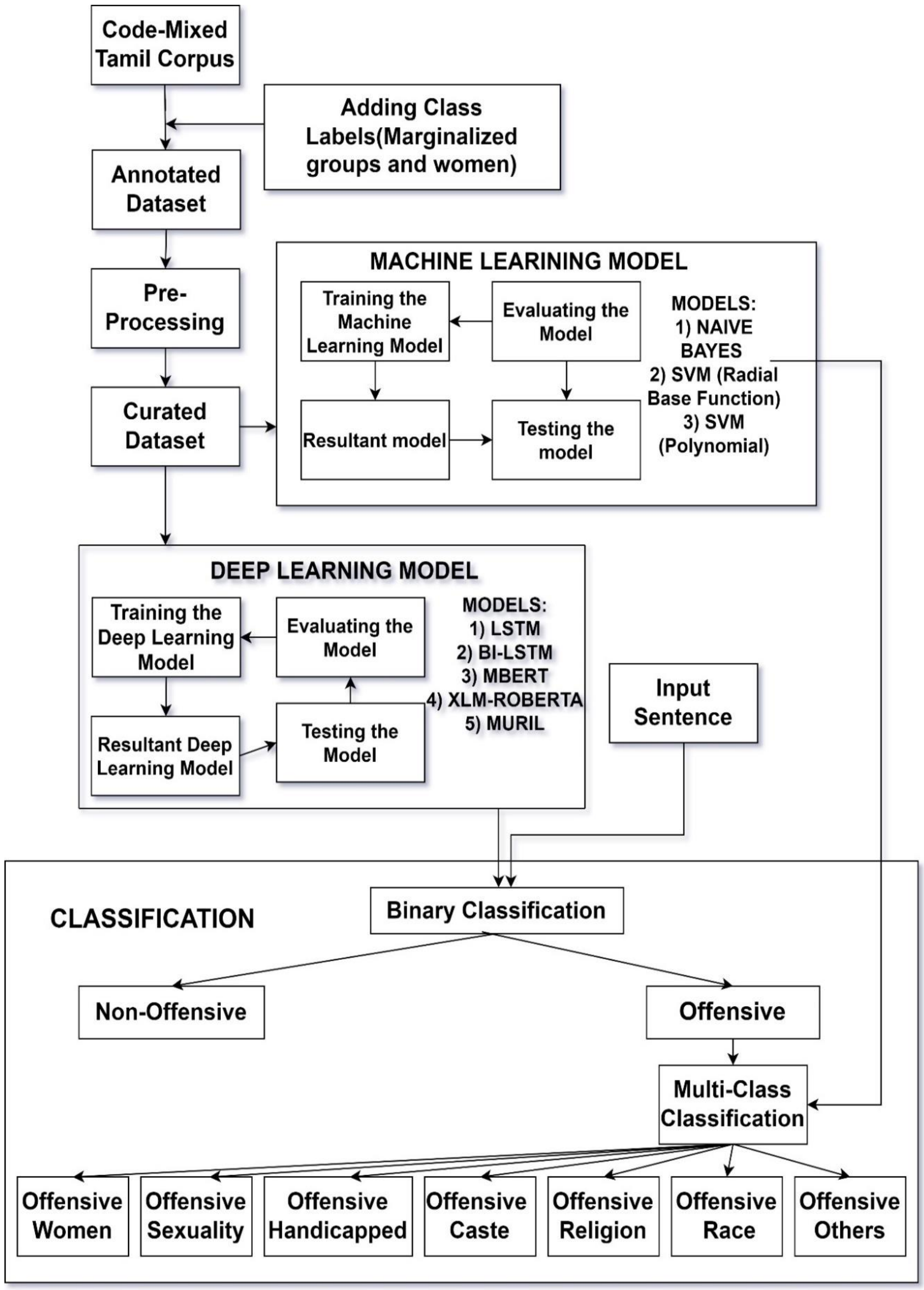


Figure 3.1 Architecture diagram

The Above diagram gives a complete picture of the architecture of the model. Let’s break down individual components to understand it.

3.2.1 Annotating the Code-Mixed Tamil Corpus

The Corpus used is a dataset of a collection of already existing code-mixed Tamil sentences made by scrapping you-tube movie trailer and review comments. It is part of the shared task Overview of the HASOC-Dravidian Code-Mix Shared Task on Offensive Language Detection in Tamil and Malayalam [24][31]. The dataset is annotated is annotated for binary classification and each comment is classified as offensive or non-offensive.

The Annotation part includes assigning tags according to the required problem statement. As stated above there are eight categories into which the comment can fall into:

1) Offensive Women

2) Offensive Religion

3) Offensive Caste

4) Offensive Race

5) Offensive handicapped

6) Offensive sexuality (LGBTQ+ community)

7) Offensive Others

8) Not offensive

A fine tag is assigned to the comment manually based on the above categories using an automated code written in python to speed up the process. [Appendix]

3.2.2 Data Preprocessing

The annotated dataset is further pre-processed to remove unnecessary information and denoise the data. After Annotation comments in Pure Tamil script were removed so that standard pre-processing techniques can be applied.

The following tasks were performed were applied:

1) cleantext library [25] was used to each comment in the dataset and the following thigs were eliminated.

a) Unicode errors were fixed in the text.

b) Non-ASCII characters were converted to their closest ASCII equivalents.

c) Multiple consecutive whitespaces were normalized into a single whitespace.

d) All text was converted to lowercase.

e) Line breaks from the text were removed.

f) Emojis from the text were removed

g) It was made sure all the text is in English.

After the preprocessing now the Curated Dataset is obtained which can be used for phase two which is training and testing the model.

3.2.3 Deep Learning Model

This part of the model is responsible for classifying the given text into Offensive or Non-offensive. There were totally five Deep Learning Models which were explored and pre-trained models were fine-tuned according to the problem statement along with variation of input and output parameters.

Let’s understand each of the Deep Learning model used to get a better understanding of the models.

1) **LSTM** [26]: Long Short-Term Memory (LSTM) is a specialized type of recurrent neural network (RNN) designed for sequential data processing, such as text or time series. It addresses the vanishing gradient problem of traditional

RNNs by incorporating a memory cell and three gates: input, forget, and output. These gates regulate information flow, allowing LSTMs to capture long-term dependencies and retain relevant information. In natural language processing, LSTMs excel in tasks like sentiment analysis, machine translation, and text classification due to their ability to understand context and maintain memory over extended sequences. For the given problem, LSTM was employed to comprehend text, with additional layers like dropout and dense layers used to map LSTM outputs to classification results. This architecture enables effective understanding and processing of textual data for various applications.

2) **Bi-LSTM** [26]: Bidirectional Long Short-Term Memory (Bi-LSTM) enhances traditional LSTM architecture by processing input sequences in both forward and backward directions. Unlike standard LSTM, which only captures dependencies from past to future, Bi-LSTM employs two separate LSTM layers to process input in both directions. By combining information from both directions, Bi-LSTM captures contextual cues from past and future contexts at each time step. This bidirectional processing enables Bi-LSTM to better capture long-range dependencies and context in sequential data. It finds applications in tasks like sequence labelling, named entity recognition, and machine translation where understanding the entire sequence is crucial. The model was chosen to address limitations of LSTM and improve efficiency by leveraging bidirectional processing.

3) **mBERT (Transformers)** [27]: mBERT, a variant of the BERT model, facilitates multilingual natural language processing by encoding language-agnostic representations through diverse corpus training. It enables parallel processing, enhancing text classification and other NLP tasks across multiple languages. Its key advantage lies in transferring knowledge across languages, requiring minimal language-specific tuning. mBERT's shared representations learned from multilingual data make it versatile and efficient for various NLP applications. By eliminating the need for separate models or extensive fine-tuning, it lowers barriers to multilingual NLP, aiding researchers and practitioners. With training on 104 languages, mBERT excels at understanding

code-mixed text, offering a global-scale solution for accessible and effective language.

4) **XLM-Roberta** [28]: XLM-RoBERTa, an extension of RoBERTa, revolutionizes multilingual NLP by simultaneously training on diverse languages. Its cross-lingual pre-training captures shared linguistic features, enhancing understanding across languages. With knowledge transfer capabilities, it excels in tasks without language-specific fine-tuning. This makes it ideal for machine translation, document classification, and information retrieval across languages. XLM-RoBERTa's robust performance stems from its ability to discern linguistic nuances and hidden contexts. Its vast model comprehensively grasps diverse linguistic data, offering state-of-the-art results. Researchers and practitioners favor it for its versatility and effectiveness on multilingual datasets. The model's adeptness at differentiating languages and contexts underscores its significance in bridging linguistic gaps. Overall, XLM-RoBERTa stands as a cornerstone in multilingual NLP, elevating capabilities in cross-lingual understanding and application.

5) **MURIL** [29]: MuRIL, built on BERT, tackles multilingual NLP challenges in Indian languages by training on a diverse corpus of 17 Indian languages and English. Its training encompasses various linguistic nuances, enabling effective text processing across languages. Notably, MuRIL excels in handling code-mixed text, prevalent in social media, enabling tasks like sentiment analysis and language identification. Its robust performance across NLP tasks signifies a significant advancement in multilingual NLP, especially in linguistically diverse regions like India. During training, hyperparameters like Epochs, Batch size, and test-train split size were optimized for better performance. MuRIL's versatility and effectiveness mark it as a valuable tool for users and developers in multilingual environments. With its focus on Indian languages and adeptness in diverse linguistic tasks, MuRIL stands as a pivotal advancement in bridging linguistic gaps in NLP applications.

3.2.4 Machine Learning Model

This part of the model is used for multi-level classification. Once the model binary classifies the comment if the comment is offensive it tries to further classify the comment into the above-mentioned categories. The Machine Learning models used in this part are:

1) **Naïve Bayes:** Naive Bayes, based on Bayes' theorem, is favored for its simplicity and efficiency, particularly in text classification and sentiment analysis. Despite assuming feature independence, it performs well in practice and is resilient to data noise. In multi-level classification, Naive Bayes efficiently handles hierarchical structures by treating each level as a separate classification problem. Its ability to compute conditional probabilities independently simplifies computation and enables effective classification across multiple levels. This makes Naive Bayes a practical choice for hierarchical classification tasks, offering ease of implementation and robust performance.

2) **Support Vector Machine (Radial Basis Function)** [30]**:** SVM with RBF kernel is renowned for its ability to handle nonlinear relationships in classification tasks. The RBF kernel projects data into a higher-dimensional space, facilitating the discovery of optimal separating hyperplanes. It captures intricate decision boundaries through Gaussian function-based similarity computations, ideal for nonlinear and non-separable data. This flexibility enables SVM to model complex patterns accurately across different classes. However, achieving optimal performance requires meticulous tuning of hyperparameters like C and gamma. Despite potential computational expenses, SVM with RBF kernel remains a potent tool in image recognition, text classification, and various ML applications.

3) **Support Vector Machine (Polynomial)** [30]: Support Vector Machine (SVM) with a polynomial kernel is a variant commonly utilized for classification tasks, operating by computing dot products between data points raised to a specified degree. This facilitates mapping input data into a higher-dimensional space to discern nonlinear decision boundaries. The degree parameter dictates the complexity of the polynomial used for mapping, with higher degrees capturing

more intricate feature relationships. While effective for nonlinear datasets, the performance heavily relies on hyperparameter selection, including degree, coefficient, and regularization parameter (C). Overfitting can occur, especially with high-degree polynomials, and it may struggle with highly imbalanced classes. Despite limitations, SVM with polynomial kernel remains valuable for multi-level classification tasks, often augmented with SMOTE to handle imbalanced datasets, making it a widely employed tool in machine learning across various domains.

3.2.5 Classification Task

This segment of the model represents the core prediction stage, where the input sentence undergoes classification to determine its offensive or non-offensive nature. Initially, the sentence is fed into the binary classifier, which serves as the first filter in the pipeline. If the output from this classifier indicates that the sentence is non-offensive, the process pauses, as no further classification is required. However, if the output suggests that the sentence is potentially offensive, it proceeds to the next stage for further evaluation.

Subsequently, the comment undergoes classification using a Machine Learning model specifically designed for multi-level classification. This stage involves categorizing the comment into different levels or categories based on its offensive content. The Machine Learning model analyses various aspects of the comment, such as its language, sentiment, and context, to determine the appropriate classification. By leveraging techniques from natural language processing and Machine Learning, this model can effectively identify and categorize offensive content, enabling more nuanced and accurate moderation of online discourse. Overall, this multi-level classification process enhances the model's capability to accurately detect and address offensive content in diverse online environments.

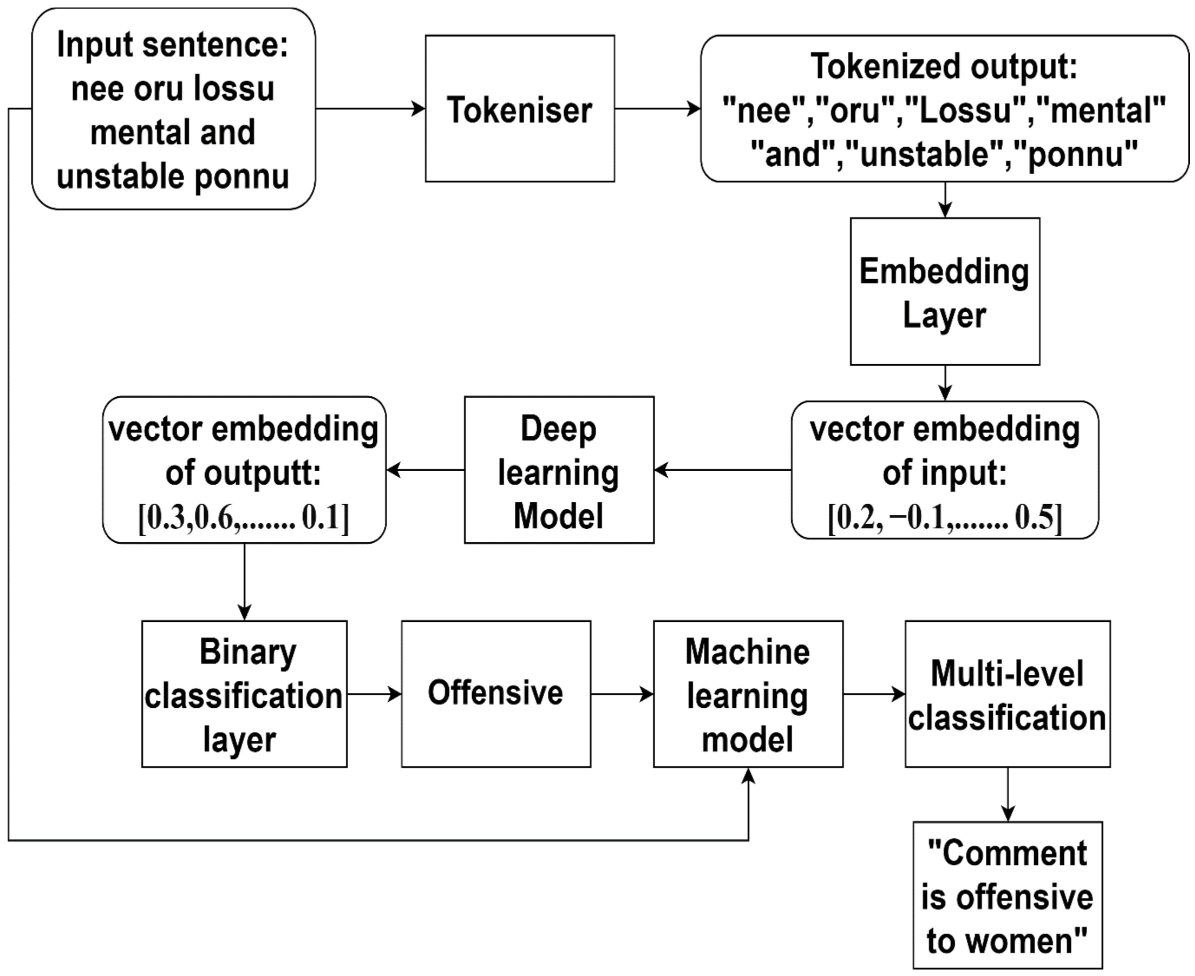
3.3 Classification of an Input Sequence

Let’s assume there is a sentence “nee oru lossu mental and unstable ponnu”. Let’s look at how it transcends into through the model.

- The sentence is initially passed into the Deep Learning Model and the sentence is converted is tokenized and the respective vector embedding is generated.
- This embedding is fed into the model and the model understands the context of the sentence. The model outputs vector embedding consisting valuable information about the input sequence's temporal dependencies, which can be further utilized for various tasks.
- We can furthermore classify it as offensive using the binary classifier.
- Once we have decided the comment is offensive, we will have to classify it's into one of the categories.
- The comment is passed through the Machine Learning Model in its raw form, which perform Multi Level Classification and declares it as "Offensive to Women"

The illustration of the same is below.

Figure 3.2 Input Sentence Illustration


Input sentence: nee oru lossu mental and unstable ponnu
Tokeniser
Tokenized output: "nee","oru","Lossu","mental" "and","unstable","ponnu"
Embedding Layer
vector embedding of input: [0.2, −0.1,....... 0.5]
Deep learning Model
vector embedding of outputt: [0.3,0.6,....... 0.1]
Binary classification layer
Offensive
Machine learning model
Multi-level classification
"Comment is offensive to women"

CHAPTER 4

RESULTS AND DISCUSSION

This chapter talks about the results obtained after executing the methodology proposed in the previous chapter followed by graphs and analysis.

4.1 Experimental Setup

- The Dataset was generated using a pre-existing Tamil Code-mixed dataset. This corpus was taken as base for obtaining offensive content and was annotated based on the requirements of the problem statement. The Dataset consists of 43919 comments out of which 10,325 Offensive comments were used to build the dataset.
- Comments that were composed of entirely Tamil script were eliminated resulting in a dataset of 9073 comments, which has been used for training, testing and experimentation. Pre-trained Deep Learning models were imported from Hugging Face which were custom fine-tuned according to the requirement. Machine Learning models were implemented using pre-existing python libraries and were trained and tested and a few techniques were applied to improve the results.
- Simulation Environment: Jupyter Notebook
- H/W resources:
- S/W resources:
- Deep Learning models implemented:
    1. LSTM
    2. Bi-LSTM
    3. MuRIL
    4. XLM-RoBERTa
    5. mBERT
- Machine Learning models implemented:
    1. SVM – RBF
    2. SVM – Polynomial

3. Naïve Bayes

- Input:
  - o Any Tamil-English code-mixed sentence or a list of such sentences.
- Output:
  - o Predicted class of the input from one of the pre-determined eight classes.

4.2 Results

This section comprises of different results obtained throughout the project.

4.2.1 Annotation Results

Dataset Before annotation is present in Table 4.1:

Table 4.1 Dataset Before Annotation

TAG	*COMMENT*
Offensive_Untargetede	*Lighta adanga maaru bgm maathiri theriyuthu*
Offensive_Untargetede	*Sema...last dialogue awesome....sanda pottu edukku saaganum...neenga daa diwali hit tu..*
Offensive_Targeted_Insult_Other	*Nice But pesura dialogue onnum puriyala romba unnipa ketka vendieruku... Music over loaded*
Not_offensive	*BGM asusual therikka vitaru Sam cs*

Dataset after annotation is present in Table 4.2:

Table 4.2 Dataset After Annotation

FINE TAG	*COARSE TAG*	*COMMENT*
Not_Offensive	*Offensive_Untargetede*	*Lighta adanga maaru bgm maathiri theriyuthu*
Offensive_others`	*` Offensive_Untargetede*	*Sema...last dialogue awesome....sanda pottu*

		edukku saaganum...neenga daa diwali hit tu..
Offensive_others	*Offensive_Targeted_Insult_Group*	*Nice But pesura dialogue onnum puriyala romba unnipa ketka vendieruku... Music over loaded.*
Not_offensive	*Not_offensive*	*BGM asusual therikka vitaru Sam cs*

Some example predictions made by the various combinations of the Deep Learning and Machine Learning models in our pipeline were as follows:

Table 4.3 Classification labels for example sentences

Model/Sentence	nee laam uyir vaala ve kudathu	indha ponnu waste da	Matha naatu kaarangala India kulla vidave kudathu
LSTM + Naïve Bayes	Offensive sexuality	Offensive sexuality	Offensive sexuality
LSTM+SVM(RBF)	Offensive others	Offensive others	Offensive others
LSTM+SVM(Poly)	Offensive caste	Offensive caste	Offensive caste
Bi-LSTM + Naïve Bayes	Offensive sexuality	Offensive sexuality	Offensive sexuality
Bi-LSTM + SVM(RBF)	Offensive others	Offensive others	Offensive others
Bi-LSTM + SVM(Poly)	Offensive caste	Offensive castes	Offensive caste
mBERT + Naïve Bayes	Offensive sexuality	Offensive Sexuality	Offensive sexuality
mBERT + SVM(RBF)	Offensive others	Offensive women	Offensive others
mBERT + SVM(Poly)	Offensive caste	Offensive caste	Offensive caste

XLM-RoBERTa + Naïve Bayes	Offensive sexuality	Offensive women	Offensive sexuality
XLM-RoBERTa + SVM(RBF)	Offensive others	Offensive others	Offensive others
XLM-RoBERTa + SVM(Poly)	Offensive caste	Offensive caste	Offensive caste
MURIL + Naïve Bayes	Offensive sexuality	Offensive sexuality	Offensive sexuality
MURIL + SVM(RBF)	Offensive others	Offensive others	Offensive others
MURIL +SVM(Poly)	Offensive caste	Offensive caste	Offensive caste

4.3 Evaluation Metrics

The evaluation metrics used are Accuracy, Recall, Precision and F1 Score. Each of the metrics are explained below:

- **Accuracy:** This metric is the percent of the test data that the model gives a correct prediction for.
- **Recall:** Recall is a metric that measures how often a model correctly identifies positive instances (true positives) from all the actual positive samples in the test dataset.
- **Precision:** Precision is the indicator of the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions (i.e., the number of true positives plus the number of false positives).
- **F1 Score:** The F1 score of a model is a metric calculated by taking the harmonic mean of the Recall and Precision scores of the model. It is calculated by dividing 2 by the sum of the inverses of Recall and Precision.

The performance metrics for Binary Classification (Deep Learning Models) has been listed in Table 4.4. The same values have been visualised in Figure 4.1. Similarly, the performance metrics for Multi-Label Classification (Machine Learning Models) has been listed in Table 4.5. The same values have been visualised in Figure 4.2. Different combinations of Deep Learning and Machine Learning Models were tried on a example test dataset. The accuracy values of this experiment are listed in Table 4.6

Table 4.4 Evaluation Metrics of Deep Learning Models

Model Name/Metrics	Accuracy	F1-score	Precision	Recall
LSTM	0.608	0.689	0.669	0.71
BI-LSTM	0.604	0.66	0.627	0.697
MURIL	0.587*	0.7398*	0.615*	0.932*
mBERT	0.586*	0.7398*	0.605*	0.954*
XLM-RoBERTa	0.588*	0.7398*	0.606*	0.955*

Note: Values denoted with (*) have been obtained through k-fold cross validation with k=5, others have been tested on 20% of the dataset

Table 4.5 Evaluation Metrics of Machine Learning Models

Model Name/Metrics	Accuracy	F1-score	Precision	Recall
Naïve-Bayes	0.89	0.895	0.895	0.912
Support Vector Machine (RBF)	0.861*	0.814*	0.808*	0.86*
Support Vector Machine (Polynomial)	0.85	0.81	0.92	0.85

Note: Values denoted with (*) have been obtained through k-fold cross validation with k=5, others have been tested on 20% of the dataset

Table 4.6 Accuracy of Proposed Architecture (Tested on 20% of the Dataset)

Model Name/Metrics	Naive Bayes	SVM(RBF)	SVM(Poly)
LSTM	0.541	0.523	0.516
Bi-LSTM	0.537	0.520	0.513
XLM-RoBERTa	0.522	0.505	0.499

mBERT	0.522	0.505	0.498
MURIL	0.522	0.505	0.498

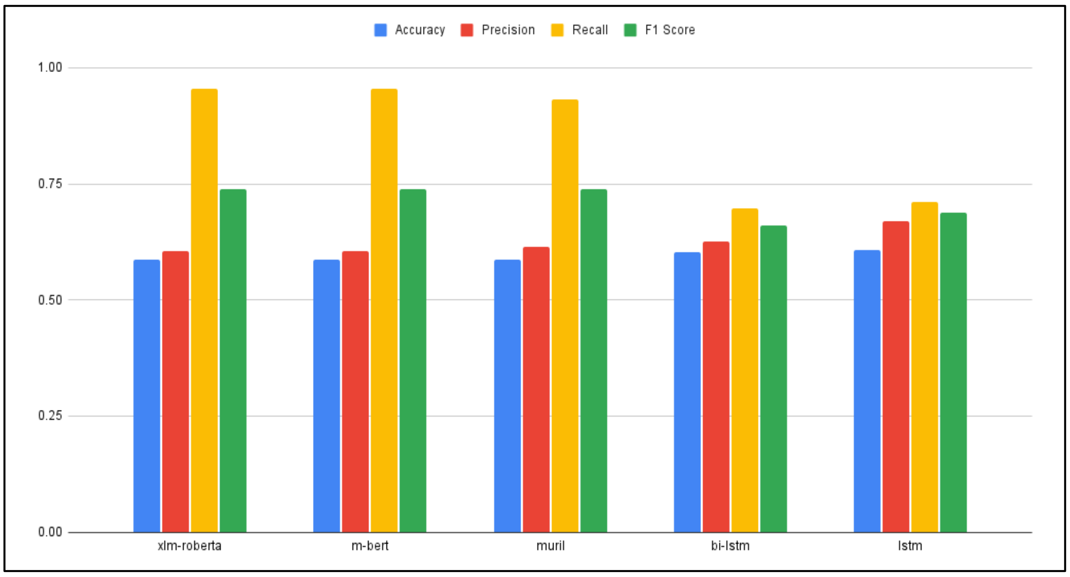


Figure 4.1 Visualisation of Performance Metrics for Binary Classification

(Deep Learning Models)

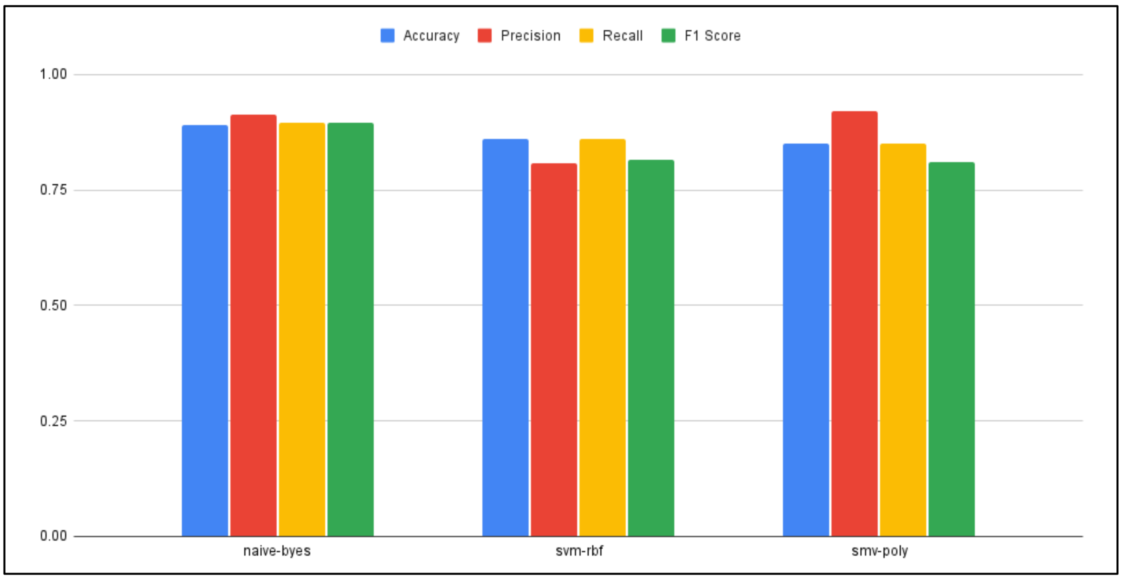


Figure 4.2 Visualisation of Performance Metrics for Multi-Label Classification

(Machine Learning Models)

4.4 Analysis

From the above performance metrics, it can be derived that the best performing combination of Deep Learning and Machine Learning models is that of mBERT and SVM. The lower than usual accuracies of the models can be attributed to the non-availability of high-quality and high-volume code-mixed Tamil-English data to suit our specific needs.

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 Summary

Our work focuses on adapting various pre-existing models like LSTM, Bi-LSTM, mBERT, XLM-RoBERTa, and MuRIL for multiclass classification of offensive text toward marginalized groups. The pipeline comprises three phases: text pre-processing, binary classification by a Deep Learning model, and multiclass classification by a Machine Learning model. In the pre-processing phase, text is converted to lowercase, and emojis, punctuations, numbers, and stop words are removed. The processed text is then vectorized and passed to the Deep Learning model for binary classification. If the sentence is not offensive, the pipeline ends; otherwise, it proceeds to the Machine Learning model. This model, also pre-trained, classifies the offensive text into one of six predetermined categories: "Race," "Religion," "Caste," "Women," "Sexuality," "Handicaps," or "Others" if the offense is generic. Predictions are printed to the console accordingly. Overall, the pipeline aims to identify and classify offensive text toward marginalized groups efficiently.

5.2 Conclusions

The project accurately classifies offensive and hate speech in code-mixed Tamil-English data based on the speech's intended target by leveraging both Deep Learning and Machine Learning models. Since the models are pre-trained and fine-tuned, a single sentence or a database of sentences can be readily passed into the pipeline and be predicted immediately. This does not require any time for training which makes it very resource and time efficient. Moreover, our pipeline is modular, meaning any of the Deep Learning models can be switched for another Deep Learning model. This follows for the Machine Learning models too, thereby enabling us to choose the combination of models which work the best.

5.3 Scope for Future Work

- **Balancing the Dataset:** The overall accuracy can be increased by Balancing the Dataset which will make more efficient prediction. It can be achieved by augmenting more relevant data in each category or using sampling techniques.
- **Model Adaptation and Fine-tuning:** Continuously adapt and fine-tune the existing Deep Learning and Machine Learning models to better capture the nuances of offensive language targeting marginalized groups in code-mixed text. This involves retraining models on updated or expanded datasets, exploring techniques like domain adaptation to generalize models to new data distributions, and investigating transfer learning approaches to leverage pre-trained models from related tasks or domains.
- **Removal of offensive comment:** The proposed work majorly focuses on identification of offensive content but doesn’t really focus on how it will be removed from such online platforms. So, a further study can be conducted exploring these areas.

REFERENCES

[1] Sharma, Arushi, Anubha Kabra, and Minni Jain. "Ceasing hate with moh: Hate speech detection in Hindi–English code-switched language." Information Processing & Management 59, no. 1 (2022): 102760.

[2] Mathur, P., Sawhney, R., Ayyar, M., & Shah, R. (2018). Did you offend me? classification of offensive tweets in hinglish language. In Proceedings of the 2nd Workshop on Abusive Language Online (ALW2) (pp. 138–148).

[3] Bohra, A., Vijay, D., Singh, V., Akhtar, S. S., & Shrivastava, M. (2018). A dataset of hindi-english code-mixed social media text for hate speech detection. In Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media (pp. 36–41).

[4] Kumar, R., Reganti, A. N., Bhatia, A., & Maheshwari, T. (2018b). Aggression-annotated corpus of hindi-english code-mixed data. arXiv preprint arXiv:1803.09402.

[5] Madhu, Hiren, Shrey Satapara, Sandip Modha, Thomas Mandl, and Prasenjit Majumder. "Detecting offensive speech in conversational code-mixed dialogue on social media: A contextual dataset and benchmark experiments." Expert Systems with Applications 215 (2023): 119342.

[6] Chatterjere, Arindam, Vineeth Guptha, Parul Chopra, and Amitava Das. "Minority positive sampling for switching points-an anecdote for the code-mixing language modeling." In Proceedings of the Twelfth Language Resources and Evaluation Conference, pp. 6228-6236. 2020

[7] Patra, B. G., Das, D., and Das, A. (2018). Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @icon-2017. CoRR, abs/1803.06745.

[8] Sai, Siva, and Yashvardhan Sharma. "Towards offensive language identification for Dravidian languages." In Proceedings of the first workshop on speech and language technologies for Dravidian languages, pp. 18-27. 2021.

[9] Bharathi Raja Chakravarthi, M Anand Kumar, John Philip McCrae, Premjith B, Soman KP, and Thomas Mandl. 2020c. Overview of the track on ”HASOC-Offensive Language Identification- DravidianCodeMix”. In Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE ’20.

[10] Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020d. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pages 202–210, Marseille, France. European Language Resources association.

[11] Adeep Hande, Ruba Priyadharshini, and Bharathi Raja Chakravarthi. 2020. KanCMD: Kannada CodeMixed dataset for sentiment analysis and offensive language detection. In Proceedings of the Third Workshop on Computational Modeling of People’s Opinions, Personality, and Emotion’s in Social Media, pages 54–63, Barcelona, Spain (Online). Association for Computational Linguistics.

[12] Nayak, Ravindra, and Raviraj Joshi. "L3Cube-HingCorpus and HingBERT: A Code-Mixed Hindi-English Dataset and BERT Language Models." arXiv preprint arXiv:2204.08398 (2022).

[13] Pratapa, Adithya, Monojit Choudhury, and Sunayana Sitaram. "Word embeddings for code-mixed language processing." In Proceedings of the 2018 conference on empirical methods in natural language processing, pp. 3067-3072. 2018.

[14] David Vilares and Miguel A Alonso. 2016. En-es-cs: An English-Spanish code-switching twitter corpus for multilingual sentiment analysis. In LREC.

[15] Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1543–1553.

[16] Mahata, Sainik, Dipankar Das, and Sivaji Bandyopadhyay. "Sentiment classification of code-mixed tweets using bi-directional rnn and language tags." In Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 28-35. 2021.

[17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606.

[18] Kusampudi, Siva Subrahamanyam Varma, Preetham Sathineni, and Radhika Mamidi. "Sentiment analysis in code-mixed telugu-english text with unsupervised data normalization." In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), pp. 753-760. 2021.

[19] Yigezu, Mesay Gemeda, Atnafu Lambebo Tonja, Olga Kolesnikova, Moein Shahiki Tash, Grigori Sidorov, and Alexander Gelbukh. "Word Level Language Identification in Code-mixed Kannada-English Texts using Deep Learning Approach." In Proceedings of the 19th International Conference on Natural Language Processing (ICON): Shared Task on Word Level Language Identification in Code-mixed Kannada-English Texts, pp. 29-33. 2022.

[20] Shashirekha Hosahalli Lakshmaiah, Fazlourrahman Balouchzahi, Anusha Mudoor Devadas, and Grigori Sidorov. 2022. CoLI-Machine Learning Approaches for Code-mixed Language Identification at the Word Level in Kannada-English Texts. acta polytechnica hungarica.

[21] Thara, S., and Prabaharan Poornachandran. "Code-mixing: A brief survey." In 2018 International conference on advances in computing, communications and informatics (ICACCI), pp. 2382-2388. IEEE, 2018.

[22] Jamatia, Anupam, Bjrn Gambck, and Amitava Das. ”Collecting and Annotating Indian Social Media Code-Mixed Corpora.” In the 17th International Conference on Intelligent Text Processing and Computational Linguistics, pp. 3-9. 2016.

[23] Mandal, Soumil, Sainik Kumar Mahata, and Dipankar Das. ”Preparing Bengali-English Code-Mixed Corpus for Sentiment Analysis of Indian Languages.” arXiv preprint arXiv:1803.04000 (2018).

[24] https://ceur-ws.org/Vol-3159/T3-1.pdf

[25] https://pypi.org/project/clean-text/

[26] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. Cambridge, MA: MIT Press.

[27] Pires, Telmo, Eva Schlinger, and Dan Garrette. "How multilingual is multilingual BERT?." *arXiv preprint arXiv:1906.01502* (2019).

[28] Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. "Unsupervised cross-lingual representation learning at scale." arXiv preprint arXiv:1911.02116 (2019).

[29] Khanuja, Simran, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam et al. "Muril: Multilingual representations for indian languages." arXiv preprint arXiv:2103.10730 (2021).

[30] Steinwart, Ingo, and Andreas Christmann. Support vector machines. Springer Science & Business Media, 2008.

[31] Chakravarthi, Bharathi Raja, Ruba Priyadharshini, Vigneshwaran Muralidaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. "Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text." Language Resources and Evaluation 56, no. 3 (2022): 765-806.

Appendix

1) Annotation steps:

```python
import pandas as pd

def mapping(arg):
    switcher = {
        0: "Offensive_women",
        1: "Offensive_race",
        2: "Offensive_handicapped",
        3: "Offensive_religion",
        4: "Offensive_caste",
        5: "Offensive_sexuality",
        6: "Offensive_others",
        7: "Not_offensive",
        8: "Pure Tamil",
        9: "Save and exit:)",
    }

    return switcher.get(arg, "invalid")

if __name__ == "__main__":

    df = pd.read_csv("Mubys_part.csv", sep="`", on_bad_lines='warn')

    print("\nInteger to Tag mapping:\n")
    for i in range(1, 10):
        print(f"{i} - {mapping(i)}")

    for index in df.index:

        if df.iloc[index,0] != "placeholder":
            continue

        print(f"\n\nThe comment is ({index+1} of {len(df.index)}; {len(df.index)-index} remaining):")
        print(df.iloc[index,2])

        choice = 0
        while choice < 1 or choice > 9:
            try:
                choice = int(input("\nEnter integer: "))
                fine_tag = mapping(choice)
                if fine_tag == "invalid":
                    print("Invalid. Enter an integer from 1 to 8.")
            except ValueError:
                print("Invalid. Enter a integer.")

        if fine_tag == "Save and exit:)":
            break
        df.iloc[index,0] = fine_tag
        df.to_csv("Mubys_part.csv", index=False, sep="`")
```

2) Pre-Processing Steps:

```
import pandas as pd
from cleantext import clean

df = pd.read_csv("dataset.csv", sep="`", on_bad_lines="warn")

for index in df.index:
    df.loc[index, "Comment"] = clean(df["Comment"][index], fix_unicode=True, to_ascii=True,
                normalize_whitespace=True, lower=True, no_line_breaks=True, no_emoji=True,lang="en")

```

```
df.to_csv('final_dataset_preprocessed.csv', index=False, sep="`")
```

3) LSTM:

```
f = df.dropna()
df = df.sample(frac=1.0).reset_index(drop=True)
x = df["Comment"]
y = df["Fine Tag"]
encoding = [one_hot(words,vocab_size) for words in x]
emb_doc=pad_sequences(encoding,padding='pre',maxlen=100)
final_x = np.array(emb_doc)
final_y = np.array(y)
model = Sequential()
model.add(Embedding(vocab_size,50,input_length=100))
model.add((LSTM(50)))
model.add(Dense(1,activation='sigmoid'))

model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['binary_accuracy']
)
```

4) Bi-LSTM:

```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=100))
model.add(Bidirectional(LSTM(100)))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['binary_accuracy']
```

5) MURIL:

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-uncased")
```

6) mBERT

```python
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(“multilingual-bert-uncased")
```

7) XLM-RoBERTa

```python
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-uncased")
```

8) Naïve Bayes

```python
Tfidf_vect = TfidfVectorizer(max_features=17293)
fit = Tfidf_vect.fit(x)
x = fit.transform(x)


encoder = LabelEncoder()
y = encoder.fit_transform(y)
print(encoder.classes_)
print(encoder.transform(encoder.classes_))
oversample = SMOTE(sampling_strategy = 'not majority')
x, y = oversample.fit_resample(x, y)

unique, frequency = np.unique(y, return_counts = True)
print(unique)
print(frequency)


x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3, random_state=42)
```

9)SVM (RBF)

```python
def fit_rbf(x_train, x_test, y_train, y_test):

    start = time.process_time()
    rbf = svm.SVC(kernel = 'rbf', gamma = 1, C = 1).fit(x_train, y_train)
    end = time.process_time()
    pred = rbf.predict(x_test)

    return [calMetric(y_test, pred)]
```

10)SVM (Poly)

```
def fit_poly(x_train, x_test, y_train, y_test):
    poly = svm.SVC(kernel = 'poly', degree = 4, C = 10).fit(x_train, y_train)
    pred = poly.predict(x_test)

    return calMetric(y_test, pred)
```

11) Classification Task

```
DL_MODEL = "xlm-roberta"
# DL_MODEL = "bert-multilingual"
# DL_MODEL = "muril"

ML_MODEL = "rbf_only_offensive"
# ML_MODEL = "poly_only_offensive"
from transformers import AutoTokenizer

model_checkpoint = f"/content/drive/MyDrive/fyp/models/{DL_MODEL}-finetuned"

    tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
    from transformers import AutoTokenizer
    from transformers import AutoModelForSequenceClassification

    def dl_inference( test_sentence ):
      model_checkpoint = f"/content/drive/MyDrive/fyp/models/{DL_MODEL}-finetuned"
      tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

```
  model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint)

  inputs = tokenizer( clean_text, return_tensors="pt" )

  logits = model(**inputs).logits
  print( logits )
  prediction = logits.argmax().item()
  print( "Classification : ", prediction )

  return prediction
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint)
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer( max_features = 1e5 )
vec = pickle.load( open("/content/drive/MyDrive/fyp/models/ml-models/tfidf.pickle", "rb") )

ml_model = pickle.load( open(f"/content/drive/MyDrive/fyp/models/ml-models/{ML_MODEL}.pkl", "rb") )

def ml_inference( test_sentence):
  test_sentence = vec.transform( [test_sentence] )
  prediction = ml_model.predict( test_sentence )[0]
```

```

        print( "offensive class label: ", prediction )

        return mapping( prediction )
```

B33_thesis

ORIGINALITY REPORT

[illegible]

18%	14%	9%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	aclanthology.org Internet Source	2%
2	www.coursehero.com Internet Source	1%
3	dokumen.pub Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	preview.aclanthology.org Internet Source	1%
6	www.fortunejournals.com Internet Source	1%
7	Rahul, Vasu Gupta, Vibhu Sehra, Yashaswi Raj Vardhan. "Ensemble Based Hinglish Hate Speech Detection", 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021 Publication	<1%