Offensive language -malayalam

ABSTRACT

Offensive language identification in code-mixed Dravidian languages is a critical task in Natural Language Processing (NLP), aiming to mitigate offensive content on social media platforms. This system addresses the complexities of code-mixed text and utilizes a combination of pre-processing techniques and advanced machine learning models. The text is pre-processed to address linguistic code-switching, tokenized with multilingual support, and fed into a fine-tuned XLM-Roberta (XLM-R) model to classify offensive content. Our dataset comprises code-mixed comments and posts in Dravidian languages (Malayalam-English) annotated at the post/comment level. The pre-processing step includes handling class imbalance, text normalization, and label encoding. With a focus on balancing training and validation data, the model achieves a 97% accuracy rate. This system description outlines the methodology, pre-processing steps, and model architecture to provide a comprehensive view of the approach used in this task.

INTRODUCTION

Offensive language identification on social media is a growing area of interest in Natural Language Processing (NLP) due to its implications for online safety and community standards. With the proliferation of code-mixed languages on social media platforms, traditional approaches to offensive language detection are increasingly inadequate. This study addresses the challenges posed by code-mixed Dravidian languages, focusing on Malayalam-English, which exhibit complex linguistic patterns due to code-switching and language blending. Our approach utilizes advanced pre-processing techniques and transformer-based models to effectively classify offensive language in social media comments and posts. We collected a dataset from real-world social media sources, annotated for offensive content, with inherent class imbalance to reflect practical scenarios. The system architecture involves robust pre-processing steps to clean and normalize code-mixed text, along with label encoding and tokenization using multilingual support. We fine-tuned a pre-trained transformer model, specifically XLM-Roberta, to classify offensive content. The training and validation processes account for class imbalance, with strategies to ensure balanced learning and accurate classification. Our results demonstrate that the system achieves high accuracy in offensive language identification, with a focus on code-mixed text, indicating its potential for practical application in moderating harmful content on social media platforms. This research contributes to the field by providing a scalable and effective solution for offensive language detection in multilingual contexts, with implications for broader applications in online content moderation and community management.

TABLE 1: An example of code-mixed text

Malayalam-English code-mixed text	
Text	entha full glass vechond irkunath enthlum problem undo.
Tags	mal eng eng mal mal mal eng mal.

System Description

Introduction

Offensive language identification involves classifying text as offensive or non-offensive. The code-mixed nature of Dravidian languages presents unique challenges, as traditional models trained on monolingual data tend to underperform due to the complexity of code-switching at different linguistic levels. This system uses a fine-tuned version of the XLM-Roberta (XLM-R) model to classify offensive language in code-mixed Malayalam-English text.

Data Collection

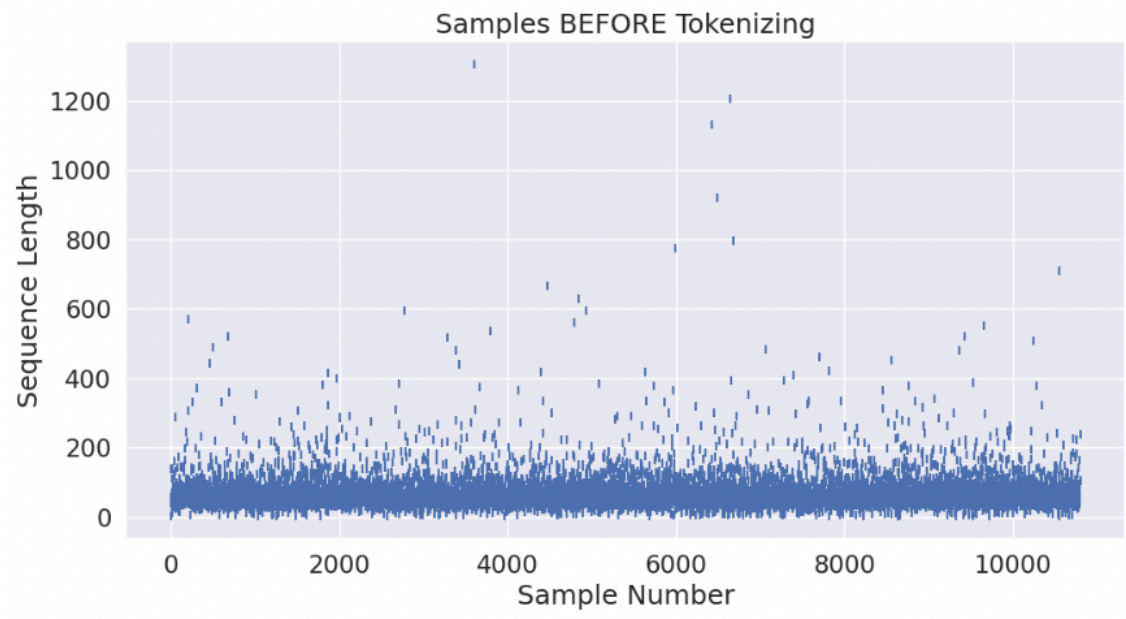
The dataset used in this research contains social media comments and posts in code-mixed Dravidian languages, annotated at the post/comment level. The data is split into training, validation, and test sets to facilitate model development and evaluation. Given the real-world nature of social media data, the dataset exhibits class imbalance, with more non-offensive samples than offensive ones.

text	category
പലദേശം. പല ഭാഷ ഒരേ ഒരു രാജാവ് അല്ലാതെ സ്വന്തം രാജവയത് അല്ല	Not_offensive
ഈ ഓണം ഏട്ടനും പിള്ളേർക്ക് ഉള്ളതാണ് എന്ന് ഉള്ളവർ ലൈക് അടി	Not_offensive
ആരണ്ട ആരണ്ട തലുണ്ടാകാണാ ആരണ്ട ഞാൻ ആണ്ട ഞാൻ ആണ്ട ഞാൻ Royal Mech ആടാ ആരണ്ട ആരണ്ട മീശ പിരിക്കുന്ന ആരണ്ട ഞാൻ ആണ്ട ഞാൻ ആണ്ട ഞാൻ royal Mech ആടാ	Not_offensive
Sushin syam Shaiju khalid Midhun manual	Not_offensive
J A K E S. B E J O Y !!!	Not_offensive
Pwoli item padam kananda ennu karuthiyatha pakshe ini kaanum	Not_offensive
Oru ratchasan feel kittitu ullalo. Bgm athra adipoliyano	Not_offensive
Super casting	Not_offensive
ഷൈലോക്ക് ന്റെ നല്ല ടീസർ ആയിട്ട് പോലും ട്രോളി നടന്ന ലാലേട്ടൻ ഫാൻസിന് കിട്ടിയൊരു നല്ലൊരു തിരിച്ചടി തന്നെ ആയിരിന്നു ബിഗ് ബ്രദർ ന്റെ ട്രെയ്‌ലർ	Not_offensive
ന്റമ്മോ ഇജ്ജാതി trailor.. Uffftrailor ethegilum.. Padam. Pwoliyo pwoli	Not_offensive
ഇത് ബോംബ് ആകും എന്ന് തോന്നുന്നവർ Like അടിക്ക്....	Not_offensive
Ee asurante varavin katta waiting.....MAMMOOKKA.. uyir	Not_offensive
Rajuvettan chunkalla chunkidipaan adi makalle like Neela adi chunk കളെ	Not_offensive
Ettan kurch koode perfect akum enn vicharikunu by H	Not_offensive
മാമാങ്കത്തിൽ മമ്മൂക്കയെ കണ്ടു . ഇതിപ്പോ ആരാ .	Not_offensive
Vada.. ithu kettu romancham vannavarku likenulla coment	Not_offensive
Haneef adeni pwolikkum Maass trailer Mommookkayude Face kanum yenn pradeekshichu Sambavam marana maass aann	Not_offensive
padam vere leval kandu ada oru figth	Not_offensive
ഓണത്തിനു ജെട്ടിമാണി നൽകിയ ക്ഷീണം ക്രിസ്മസിനു ഇതു മാറ്റുമെന്ന് പ്രതീക്ഷിക്കുന്നു '	Not_offensive
ആ ജോഷി തന്നെയാണോ??? joshiy എന്നു കണ്ടതുകൊണ്ടു ചോതിച്ചതാണ്???ആരാണെങ്കിലും trailer അടിപൊളി!	Not_offensive
തെറ്റ് കൾ തിരുത്തിയാൽ നന്നായിരിക്കും. അല്ലങ്കിൽ തിരുത്താൻ പറ്റാത്ത തെറ്റായിരിക്കും.	Not_offensive
Ikka polikumen urappullavar like and comment please	Not_offensive
December 12 inn katta waitingullar like poad	Not_offensive
Jobi jorge padam ano. Potti mone	Offensive_Targeted_Insult_Individual

Pre-processing Techniques

Pre-processing is a critical step in ensuring the data is clean, consistent, and ready for model training. Here's an overview of the pre-processing techniques used in this study:

1. Data Cleaning:
    - Comments and posts may contain noise, such as special characters, extra spaces, or irrelevant text. The pre-processing step involves cleaning the text to remove unnecessary characters, reduce excessive spacing, and convert text to lowercase for consistency.
2. Label Encoding:
    - The class labels for offensive language are encoded into numerical values using a LabelEncoder. This step simplifies the data structure, making it suitable for feeding into the machine learning model.
3. Handling Class Imbalance:
    - Given the class imbalance in the dataset, strategies such as resampling, weighted loss functions, or class weights are considered. The compute_class_weight function is used to assign appropriate weights to different classes to reduce the impact of class imbalance during training.
4. Tokenization:
    - The text is tokenized using the AutoTokenizer from Hugging Face Transformers, with specific attention to code-mixed language support. Tokenization converts text into a sequence of tokens that can be processed by the model. We set a maximum token length to ensure consistency and avoid excessive computational costs.

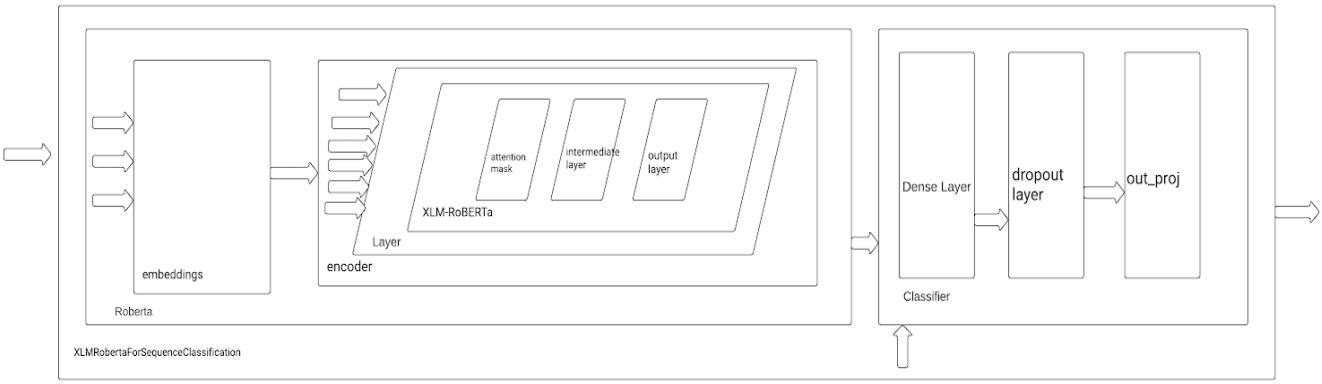


5. Padding :

- In this step it pads the sequences (Each batch of tokenized sentences and labels) to match the length of the longest sequence in the batch using the tokenizer's pad token ID. For each sentence, it calculates the required padding tokens, pads the sentence, and creates an attention mask. Finally, it converts the padded inputs, attention masks, and labels into PyTorch tensors and appends them to their respective lists. This ensures uniform sequence length for efficient neural network batch processing.

6. Data Batching:
    - `make_smart_batches`, is designed to prepare batches of text samples and their corresponding labels for processing in the model. The function ensures that the tokenized data is split into batches and are randomly selected from the sorted samples to prevent any bias. After selecting the batches, it pads the sequences within each batch to match the length of the longest sequence in that batch and creates attention masks accordingly. Finally, it converts the batches into PyTorch tensors and returns them as a tuple containing inputs, attention masks, and labels, ready for use in training or evaluation.

Architecture of the Model

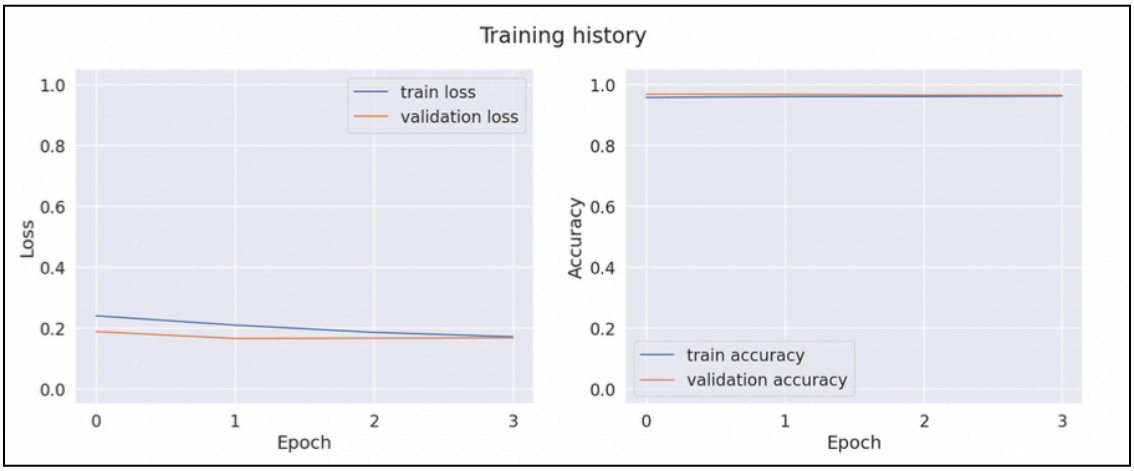


The architecture of the model follows a hierarchical structure aimed at sequence classification tasks. At its core is the XLMRobertaForSequenceClassification model, serving as the primary entity for classification. This model utilizes the roberta backbone, which consists of the XLM-RoBERTa model's components. These include the embeddings layer, responsible for handling token, position, and token type embeddings, and the encoder layer, which incorporates multiple XLM-RoBERTa layers. Each XLM-RoBERTa layer contains an attention mechanism for capturing contextual dependencies, an intermediate layer for feature transformation, and an output layer for producing final representations. Additionally, the model includes a classifier component for classification tasks, comprising a linear transformation (dense), a dropout layer for regularization, and a final linear transformation (out_proj) for generating classification outputs. This architecture enables effective sequence classification by leveraging pre-trained embeddings and capturing intricate relationships within input sequences through attention mechanisms.

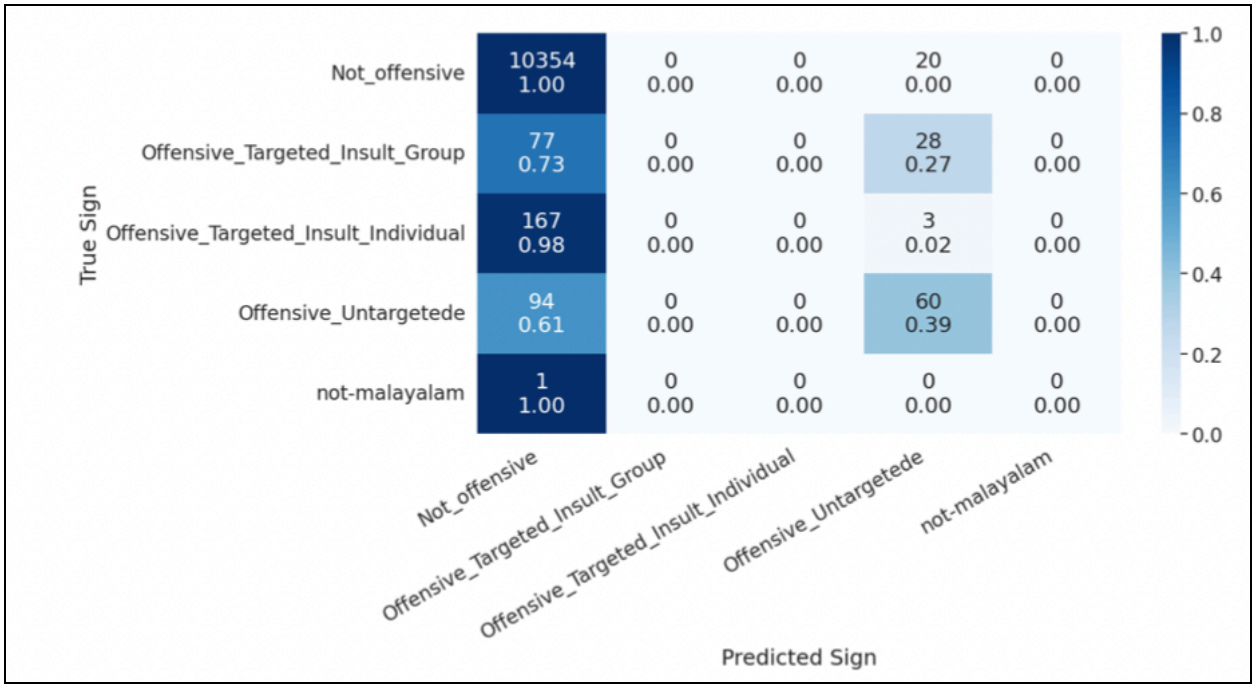
Model Training

The model used in this system is XLM-Roberta (XLM-R), a pre-trained multilingual transformer model. The model is fine-tuned to classify offensive language in code-mixed text. The following steps outline the training process:

1. Model Initialization:
    - The pre-trained XLM-R model is initialized with a specified number of output classes (corresponding to the unique labels in the dataset). An appropriate tokenizer is also loaded to ensure compatibility with the model.
2. Optimizer and Scheduler:
    - The AdamW optimizer is used, with a learning rate scheduler (get_linear_schedule_with_warmup) to control the learning rate during training. This setup helps maintain stability and avoids overfitting.
3. Training Loop:
    - The training loop involves multiple epochs of training, with each epoch comprising multiple batches. The model's loss is calculated and backpropagated to update the model's weights. The scheduler is updated to adjust the learning rate, ensuring smooth training.

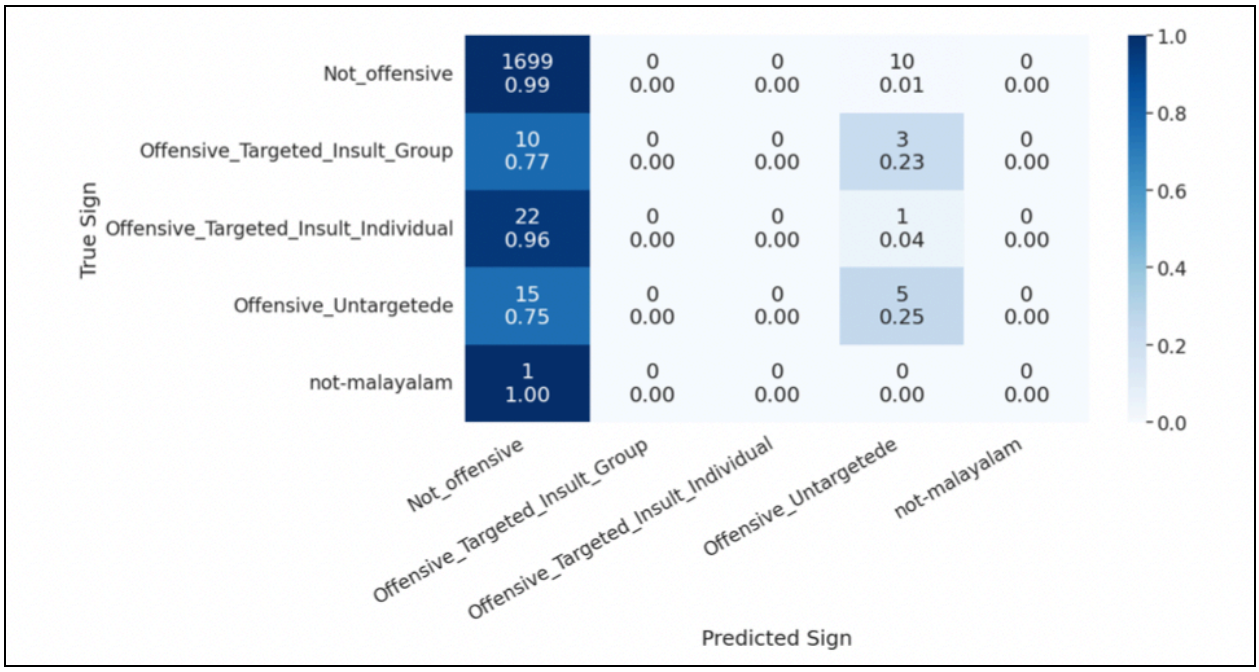


	precision	recall	f1-score	support
Not_offensive	0.97	1.00	0.98	10374
Offensive_Targeted_Insult_Group	0.00	0.00	0.00	105
Offensive_Targeted_Insult_Individual	0.00	0.00	0.00	170
Offensive_Untargetede	0.54	0.39	0.45	154
not-malayalam	0.00	0.00	0.00	1
accuracy			0.96	10804
macro avg	0.30	0.28	0.29	10804
weighted avg	0.94	0.96	0.95	10804



4. Validation:
    - After training, the model is evaluated on the validation set to monitor its performance. Metrics such as accuracy, F1-score, and confusion matrix are calculated to measure the model's effectiveness and identify areas for improvement.

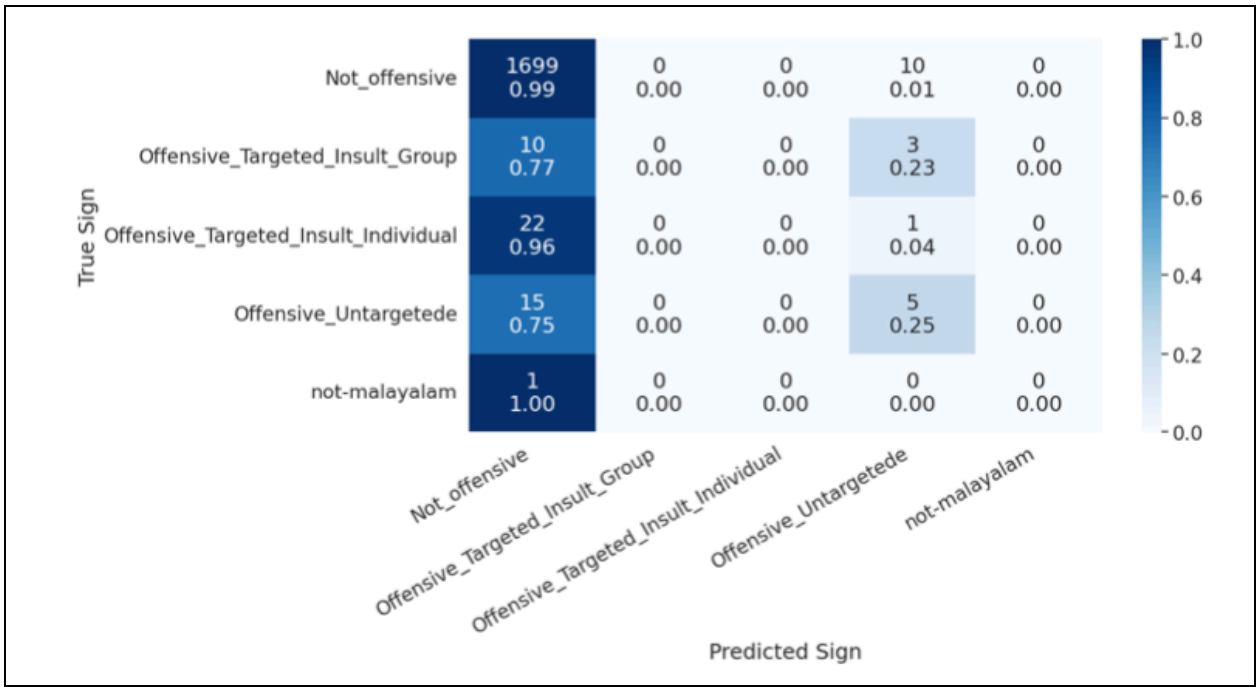
	precision	recall	f1-score	support
Not_offensive	0.97	0.99	0.98	1709
Offensive_Targeted_Insult_Group	0.00	0.00	0.00	13
Offensive_Targeted_Insult_Individual	0.00	0.00	0.00	23
Offensive_Untargetede	0.26	0.25	0.26	20
not-malayalam	0.00	0.00	0.00	1
accuracy			0.96	1766
macro avg	0.25	0.25	0.25	1766
weighted avg	0.94	0.96	0.95	1766



Results and Evaluation

After validation, the model's performance is evaluated using the testing dataset. The evaluation includes metrics like accuracy, F1-score, and confusion matrix. The classification results indicate strong performance in identifying instances of "Not_offensive" content, with a precision of 97% and a recall of 99%, yielding an F1-score of 98% with a substantial support of 1709 instances. The overall accuracy of the model is 96%, with a macro average F1-score of 25% and a weighted average F1-score of 95%, considering all classes equally and by their support, respectively, in the evaluation, suggesting that the pre-processing techniques and model architecture effectively address the complexities of code-mixed text.

	precision	recall	f1-score	support
Not_offensive	0.97	0.99	0.98	1709
Offensive_Targeted_Insult_Group	0.00	0.00	0.00	13
Offensive_Targeted_Insult_Individual	0.00	0.00	0.00	23
Offensive_Untargetede	0.26	0.25	0.26	20
not-malayalam	0.00	0.00	0.00	1
accuracy			0.96	1766
macro avg	0.25	0.25	0.25	1766
weighted avg	0.94	0.96	0.95	1766



Conclusion

This system description outlines the approach used for offensive language identification in code-mixed Dravidian languages, focusing on Malayalam-English text. The pre-processing techniques ensure the data is clean and consistent, while the use of the XLM-R model provides robust support for multilingual text. The training and evaluation methods contribute to achieving high accuracy, demonstrating the effectiveness of the approach. Further research could explore additional pre-processing techniques, different model architectures, and other strategies for handling class imbalance to improve performance.