National College of Ireland

Sentiment Analysis in Tamil-English Code-Mixed Data Using Hybrid Deep Learning Techniques

MSc Research Project
Data Analytics

Diwakar Muthuraj
Student ID: X23106824

School of Computing
National College of Ireland

Supervisor: Aaloka Anant

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Diwakar Muthuraj

Student ID: X23106824

Programme: MSc - Data Analytics **Year:** 2024

Module: Research Project

Supervisor: Aaloka Anant

Submission Due Date: 12/12/2024

Project Title: Sentiment Analysis in Tamil-English Code-Mixed Data Using Hybrid Deep Learning Techniques

Word Count: 6581 **Page Count:** 21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Diwakar Muthuraj

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Sentiment Analysis in Tamil-English Code-Mixed Data Using Hybrid Deep Learning Techniques

Diwakar Muthuraj
X23106824

Abstract

Sentiment Analysis (SA) is the process of classifying the sentiments found in data as positive, negative, or neutral. Some of the important real-world applications of sentiment analysis are social media trends, customer feedback, political discourse, and market insights. With this vast application, it has a major role in Natural Language Processing. Looking at most social media posts, comments, ecommerce product reviews, and online forums, the bilingual communities largely use code-mixed text, which is a frequent interchange of words from different languages to express their opinions. This code-mixed text has non-standard grammar, transliterations, slang words; thus, these complexities introduce challenges in sentiment analysis. Tamil-English being one of the most used code-mixes is chosen for this research project to examine fine-tuning hyperparameters of hybrid models to efficiently classify sentiment in Tamil-English code-mixed data. In this project, various experiments are done with a base model mBERT+TextGCN, with different tools and techniques to prepare the data for the model. These steps include preprocessing, handling class imbalance, feature engineering, feature extraction etc. Then to improve the efficiency of proposed IndicBART+TextGCN further, fine-tuning of hyperparameters are performed and evaluated using accuracy, precision, recall, F1 Score and confusion matrix. By following these effective techniques, the IndicBART+TextGCN model achieved a weighted average of precision 0.71, recall 0.68, f1-score 0.67. This result shows that the preprocessing, handling class imbalance, feature engineering and efficient fine-tuning of IndicBART+TextGCN has improved this hybrid model's ability to classify sentiments from the Tamil-English code-mixed data.

1 Introduction

The merging of two languages in a single sentence is very prevalent in multilingual communities which are known as code-mixed. Most of the code-mixes are bilingual pairs such as Chinese-English, French-English, Hindi-English, Tamil-English, etc (Winata, 2023). These code-mixes spread rapidly in the digital realm as the increase in online platforms usage among the multilingual communities such as social media, e-commerce, and digital forum.

Sentiment analysis (SA) is a method of classifying the emotion from the text data, into positive, negative, neutral sentiments. SA in monolingual data is very efficient as it has more tools and models which can accurately classify SA. However, when it comes to the code-mixed data, the complexity arises because there are fewer resources which leads to insufficient data clarity and lack of appropriate tools (Chutia, 2024).

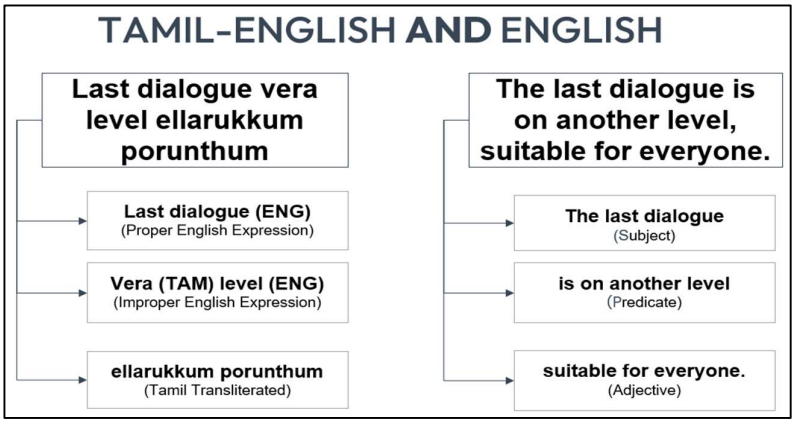


Figure 1: Tamil-English codemix data.

Majority of code-mixed data has multiple out-of-vocabulary words, slang words, non-standard grammar and transliterated words. This research project focuses on the Tamil-English code-mixed which also has similar complexities (Mahadzir, 2021). Figure. 1 shows the complexity of the code-mixed data that makes the sentiment analysis more challenging. However, this has many real-world applications and plays an important role in natural language processing (NLP).

Almost all the tech giants strive to understand their customers or users reviews shared online as these feedbacks may impact market value of their product or technology (Majumder, 2022). For example, a small startup or new YouTube channels can look at the comments to understand what type of their content are well received by everyone. In doing so, they can not only streamline their resources to create similar context in the future, but also be able to generate a higher revenue in return. Furthermore, the same is true for e-commerce sites such as Amazon. Sellers would benefit from using sentiment analysis to improve their products and services based on customers sentiment and their experiences. Finally, sentiment analysis is also equally helpful in politics. By analysing Tamil-English discussions on social media, we can be able to get real-time feedback on how people feel about political leaders, policies, or events. Thus, understanding the complexities of the codemix data and extracting meaningful insight such as sentiment from the code-mixed data is a valuable field of research.

1.1 Background and motivation

In our day-to-day activities, such as social media posts, comments, messaging and all those informal activities online involve code-mixed text. As most of us are from the bilingual communities of different parts of the world and English being the primary digital language, these code mix data will be a mix of the native language and the English (Androutsopoulos, 2007). By observing these activities and exploring studies conducted on these code-mixed data, it was identified that most of the studies were recent and used techniques and tools which are efficient but still in need of evolving. This showed a significant research gap in the sentiment analysis of code-mixed data in NLP.

This research aims to provide a model that is efficient in classifying sentiment on Tamil English code-mixed. The first step where to find any related research has been done in this

field. Chakravarthi B. M. (2020)'s initial work was to focus on the corpus creation of this analysis. Primarily the study only looked at code-mixes of Tamil-English. However, a year later on the basis of Chakravarthi B. M. (2020) primary study, Chakravarthi B. R. (2022) published a newer study, which not only has the addition of more Dravidian language code-mixed SA into it, but also has offensive language detection.

It was based on this corpus that the traditional machine learning (ML) models were trained and evaluated as baseline models. Despite better results, the models lacked efficiency due to the complex vocabulary, nonstandard grammar and sentiment annotations of lexicons. Both deep learning (DL) models and pre-trained models performed well comparatively as they benefited from contextual understanding, word embedding and transfer learning ability.

Although these models did considerably well in general aspects, they still needed to be fine-tuned for specific tasks or needed a classification layer to be added or an ensembled technique to be adapted (Sampath, 2024; Winata, 2023). Dowlagar (2021) study used the mBERT+TextGCN hybrid model for their research, the graph convolutional network with an undirected graph on the word embedding from a pre-trained model data worked great. Hence, this research project aims to identify a hybrid model, and then fine-tuning it to examine how well the model can perform SA on Tamil-English code-mixed data. By adapting (Dowlagar, 2021) suggestion of the similar hybrid architecture for better classification, mBERT+TextGCN is chosen as the initial base model. Then the tools and techniques such as preprocessing, handling class imbalance, feature engineering are examined and pre-trained models such as IndicBART is adapted to efficiently classify Tamil-English SA.

1.2 Research Question

The aim of this research project is to evaluate and improve the performance of a hybrid model by fine-tuning it to classify sentiments in Tamil-English code-mixed data.

Research Question: "How well can a hybrid model namely IndicBART+TextGCN be fine-tuned to efficiently classify sentiments on Tamil-English code-mixed data?"

1.3 Research Objective

The objectives of this research project are as follows.

Objective 1: Exploring and analyzing the tools, libraries, and techniques used for sentiment analysis on Tamil-English code-mixed data by a detailed review of literature.

Objective 2: Examining various preprocessing, feature engineering and class imbalance handling techniques with a hybrid base model namely mBERT+TextGCN.

Objective 3: Adapting more efficient pre-trained model namely, IndicBART in the hybrid architecture and then fine-tuning it.

1.4 Contribution

NLP on monolingual data such as English, French, German has enormous research and developments done, whereas the bilingual code-mixed data such as Tamil-English are still evolving at the initial stage of research. To contribute to this evolution, this study aims to identify efficient DL techniques for preprocessing, feature engineering, feature extraction. This is done by examining them with a hybrid base model for better classification of sentiment in Tamil-English code-mixed. The performance evaluation of the base model is then evaluated where the techniques which yield the better outcome are selected to be adapted into more effective models such as IndicBART of hybrid model architecture. The IndicBART + TextGCN model is chosen for this research as it showed improved performance in the sentiment classification and further fine-tuned. Based on this, this research project contributes to exploring the suitable data cleaning, preprocessing, feature transformation, feature extraction, model building, fine-tuning and evaluation methods for efficient sentiment analysis of Tamil-English code-mixed data. This gives valuable understanding of the existing tools, libraries, models and techniques and also suggests any improvement needed.

The proposed methodology is properly documented in this report in the following structure where section 2 contains related work, followed by methodology, design specification, implementation, and finally evaluation.

2 Related Work

NLP is a branch of Artificial intelligence that involves improving how machines understand human languages. It has a widespread task from translation, summary, text generation, sentiment analysis, offensive language identification. Sentiment Analysis (SA) is an area of study in NLP, which aims to analyze and identify the sentiment such as positive, negative, neutral from textual data (Khurana, 2022). The research on SA initially started with statistical methods, which then later evolved into computational linguistics. The SA is based on ML and DL methods to improve its ability to handle the complications of languages in the real world (Hermawan, 2020).

In recent years, numerous models and preprocessing tools have emerged making SA with high accuracy on monolingual text like English, French, German. Notable models like mBERT, RoBERTa have shown high accuracy around 94% in most monolingual SA around the globe (Hermawan, 2020). Vaswani (2023) introduced a transformer architecture with the self-attention mechanism that allows parallel processing of the inputs. These transformer-based models produce better results when compared with traditional ML models and RNNs. Though, despite all these improvements in the monolingual SA, there is still a significant need for research to address challenges in SA of code-mixed data. Code-mixed data is a multilingual text that consists of words from different languages in the form of transliterated and slang words. This makes ML models and DL models struggle when it comes to SA of code-mixed data. As the researchers turned focus on the code-mixed SA.

Although various tools and models were experimented and developed (Majumder, 2022), the complex structure of the code-mixed text are challenges which are yet to be handled effectively (Chutia, 2024). As most of the online textual data are multilingual and don't follow

a proper formal grammar or pattern, this complex structure has non-standard grammar, cultural and slang words, transliterate words, out of vocabulary words and spelling errors. Hence this creates a pressing need for research on this code-mixed data SA in order to better understand the code-mixed data from the digital media and communication platforms (Winata, 2023).

2.1 Sentiment Analysis on Code-Mixed Data

The rise in digital gadgets and internet among the multilingual communities has led to an increase in code-mixed data over all digital communication platforms, creating a research need in this area. Initially the study of code-mixed data focuses only on language identification and lexicon creation, but it was later improvised to include sentiment and offensive language detection (Winata, 2023).

It is very common practice for multilingual communities to use code-mixed in their day-to-day activities. All codemixed languages have their own complexities such as tonal phonetics, semantics, out of vocabulary words, slang words and resource scarcity (Winata, 2023). The early stages of all these language-based researches started with three major tasks such as language identification, corpus creation, and lexicon-based approaches. Lexicon based methods such as 'SentiWord' and 'SenticNet' were effective when paired with the linguistic resources. At later stages with involvement of ML and DL methods, it has led to the development of the preprocessing techniques as well (Mahadzir, 2021).

Preprocessing techniques do support and improve the performance of models such as text normalization, stop word removal, spelling corrections. Babu (2021) shows some techniques such as part-of-speech tagging and sequence labelling increases the model performance by improving the contextual. Traditional ML models such as decision tree, logistic regression, naïve bayes, support vector machine and maximum entropy classifiers are models which initially showed limited accuracy but with the right adaptation of the pre-processing techniques, it has shown an improved performance (Perera, 2024). Logistic regression performed well with 65% precision on the Tagalog-English code-mixed data. However, the lack of understanding the contextual relationships and nonlinear relationships has led to reduction in precision significantly (Ehsan, 2024). Similarly, the decision tree also performed well with almost 60% recall, but the model was overfitting and had noisier data (Winata, 2023). Yusuf (2023) studied Hausa-English but the less resources and complex semantics of the data has limited their SVM model accuracy to around 61%.

Most ML models achieved around 70% of F1 score in classifying sentiments but these models struggled when the data is imbalanced, has noise, a non-linear relationship among words, limited resources and complex semantics. These global researches on the international languages of code-mixed SA emphasizes the need to have proper preprocessing, feature engineering, efficient model building and fine-tuning for capturing the linguistic complexity (Khurana, 2022). Tamil – English is one among the language pairs which has unique syntactic structure and very low resources in terms of supporting SA on code-mixed data. Hence, this project chose to focus on Tamil-English code-mixed SA research gap.

2.2 Sentiment Analysis on Tamil-English Code-Mixed Data

Tamil is one of the official languages in India, Sri Lanka and Singapore, widely spoken by a large diaspora around the globe. Hence, why the usage of Tamil-English code-mixed is so predominant among the multilingual community and directly reflects their engagement on digital platforms. This growing Tamil-English code-mixed data culture has introduced challenges in processing and understanding the context of its textual data due to its complex structure and vocabulary (Shanmugavadivel, 2022). In the earlier stage, researchers were solely working on the corpus creation and language identification using 'TamilMixSentiment' dataset using the YouTube comments as it was fairly annotated data with labels of sentiment. It also has multiple baseline models experimented on it, such as Logical regression and SVM, that showed better performance around 60% (Chakravarthi B. M., 2020).

Later Chakravarthi B. R. (2022) extended the dataset to include other south Indian languages and datasets for offensive language detection as well. This corpus creation provided a strong foundation for the SA and offensive language detection of code-mixed data for most south Indian languages. Yet the limited linguistic resources and data imbalance issues would need to be addressed in Tamil-English even though ML models like Decision Tree and the Support Vector Machine (SVM) do achieve a better performance. Most ML models lacked contextual understanding when it comes to a non-linear relationship between words, but the addition of preprocessing techniques like language identification, part-of-speech tagging, normalization and handling class imbalance effectively improved accuracy (Mahadzir, 2021).

Developments of the DL models helps to address some limitations faced by ML methods by the adapting of word embeddings and the contextual representations. For Tamil-English SA, models such as LSTM and Bi-LSTM demonstrated an accuracy range of 60-75%. These models mostly outperformed ML approaches because of their ability to capture long-term dependencies and handle low resource setting (Ahmad, 2019). However, DL models usually need higher computational resources than ML and as their capability of analysing sentiments in low-resourced data like Tamil-English code-mixed only works with a proper annotated dataset. Thus, this highlights that there is potential for room for further enhancement (Shanmugavadivel, 2022).

Furthermore, tokenizers like 'fastText' and 'word2vec' have also shown improved performance by providing a meaningful embedded representation for the code-mixed data (Dowlagar, 2021). But it was the introduction of transformer-based architecture pre-trained models, like BERT and mBERT that truly revolutionised SA for code-mixed languages. These models have a self-attention mechanism that is able to process any input in parallel, allowing an in-depth understanding of linguistic tones. This is reflected in the improved performance of pre-trained models like RoBERTa, which have achieved accuracy rates around 68 -70% for Tamil-English sentiment tasks (Babu, 2021). Specific Indian language models, such as IndicBART has shown an extraordinary contextual understanding while Adaptive-BERT achieved an accuracy of 79% for Tamil-English code-mixed SA (Shanmugavadivel, 2022). These language models had language integration and have learned to adapt to the linguistic complexities after being trained on most of the regional Indian languages.

Hence, fine-tuning these models with additional classifier layers and hyperparameter optimisation may significantly improve their effectiveness. In SA for code-mixed languages,

preprocessing steps is very important. Studies by Dowlagar (2021) show that key preprocessing steps, such as lowercasing, punctuation, emoji, and stopword removal, as well as lemmatisation have significantly enhanced model accuracy. Next, addressing the class imbalance with suitable techniques such as oversampling or under sampling may also improve the performance significantly in hybrid models when combined with pre-trained models. Furthermore Dowlagar (2021) has demonstrated Graph Convolutional Networks (GCN) in their study. Models based on these advanced methods have shown higher precision in classifying positive and negative sentiments than other classes. Thus, by adapting these existing methods from the above works as a base methodology, this project hopes to enhance the further the experiment steps as preprocessing, feature engineering, tokenisation, word embeddings, and hybrid model techniques like GCN for capturing nonlinear relationships.

2.3 Limitations and takeaway from the literature review

A limitation that was taken note of is that even though many class imbalance handling methods were implemented, the model's misclassification still persists (Banerjee, 2020; Singhal, 2024). The mBERT model has a contextual understanding of Tamil and English language but many other models which are specifically trained on the regional Indian languages such as IndicBART models, ensuring most effective word embedding extraction from the code-mixed contexts and also IndicBART has a language tagging feature with the word embeddings which increases the quality of word embeddings, requires the text to be in proper Tamil and English script (Dabre, 2022). As this suggests the IndicBART pretrained model with the TextGCN as a hybrid architecture may perform more efficiently to identify sentiments in Tamil English code-mixed data.

The takeaway from the literature review is that hybrid model with a pretrained model and TextGCN layers are a better choice for code-mixed SA (CMSA) as it does take advantage of the pre-trained model and GCN.

This hybrid model shows a high performance with their ability to understand the non-linear relationship among the code-mixed data by graph-based feature representation. The proper preprocessing, feature engineering and handling data imbalance has to be taken into account. The preprocessing techniques such as removal of emojis, special characters, extra whitespace, converting text to lowercase for normalization, stop word removal were also adapted for this project (Dowlagar, 2021). Additionally, handling the class imbalance has to be examined with different methods as resampling techniques such as under sampling or oversampling which doesn’t show improvement in most of the references (Banerjee, 2020; Singhal, 2024). Hence, the class imbalance handling method is adapted based on experimenting with various methods. Next, feature engineering which is done by the back transliteration process where Tamil and English words are being converted into their respective scripts where then each word is embedded based on the language tag being performed (Dowlagar, 2021).

As discussed, these key points from the literature reviews are the foundation for this research project. Based on these findings, this research will focus on data cleaning, preprocessing, feature engineering, model building (IndicBART+TextGCN), fine-tuning and evaluation methods for efficient SA of Tamil-English code-mixed data.

3 Research Methodology

This section explains about the data mining methodology used in this research project. Knowledge discovery in database (KDD) methodology is followed for the implementation of this research project.

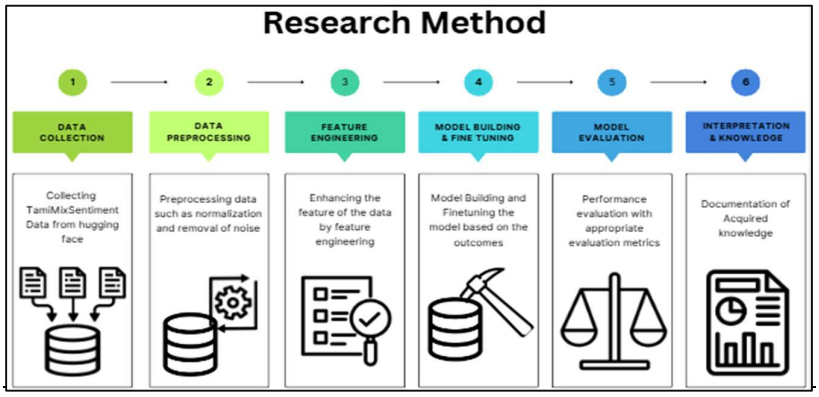


Figure 2: KDD Methodology.

It starts with data collection, followed by pre-processing based on the understanding of the data characteristics by a detailed exploratory data analysis. Based on the literature references, the feature engineering method is adapted for Tamil-English code-mixed SA. With a base model, the proposed hybrid model is adapted and its hyperparameters are fine-tuned. Evaluation metrics of the model performance is then evaluated and its parameters, pre-processing and feature engineering processes are documented for the future references. By this methodology, this research project aims to ensure that each step of the research methodology is well structured as shown in the Figure. 2.

3.1 Data collection

Using the datasets library, 'Tamilmixsentiment' data from Hugging Face is directly loaded into the Python environment using Google Colab as a dictionary structure. Once the data is loaded, it is then converted into a panda data frame for easier manipulation and analysis. This approach eliminates the need for manual downloading or uploading of datasets, ensuring streamlined data access for further processing.

3.2 Data preprocessing

After storing the data as the panda dataframe, a detailed EDA is performed to understand the data properties, features, class distributions and visualization of frequently occurring words using WordCloud. This helps to pre-process the data better. Firstly, special characters, numbers, and emojis were removed using RegEx and emoji library. The text is then normalised by converting it to lowercase, eliminating stopwords using the NLTK stopwords corpus. Next, misspelled words were corrected with the SpellChecker library while lemmatisation was

applied through NLTK's WordNetLemmatizer to reduce words to their root form (Dowlagar, 2021).

3.3 Feature Engineering

Feature engineering of code-mixed text data is first tokenised using whitespace into a new column named 'Wordset'. These tokenized words are back transliterated using the XlitEngine from 'ai4bharat-transliteration' which then converts the Romanised Tamil words into proper Tamil script. Next, a new column named 'combined_words' is created by combining the English words in English script and Tamil words in Tamil script respectively. This ensures proper representation of the text for embedding, allowing the model to perform better performance SA (Dowlagar, 2021).

3.4 Handling Class Imbalance

Figure. 3 shows highly imbalanced data. Hence, methods such as oversampling, under sampling, adding class weights are performed on the models using various tools and techniques that were examined with the base model. A method that works best and more suitable is adapted to the proposed model.

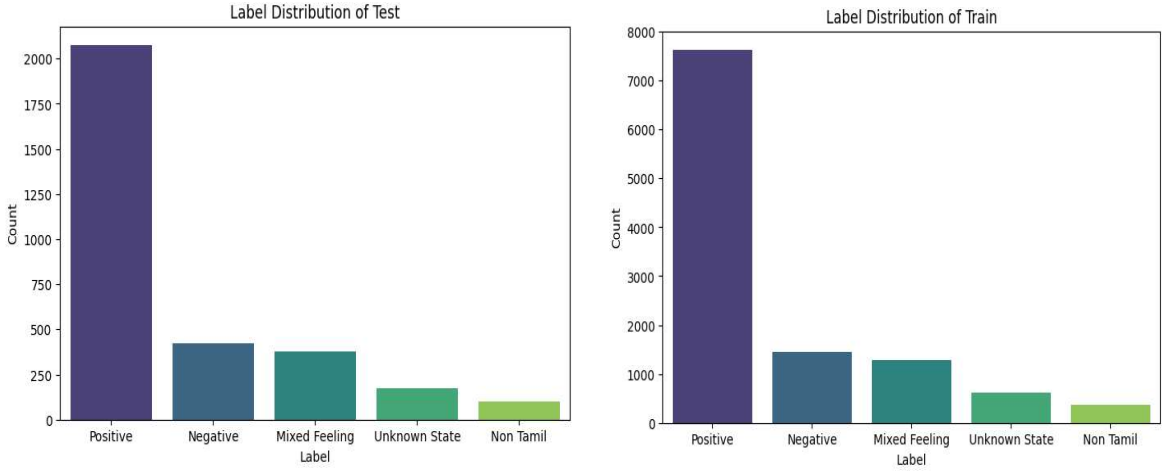


Figure 3: Class distribution of data.

3.5 Model Building and Fine-tuning

The mBERT+TextGCN is the base model and it is experimented with different tools and techniques. This includes pre-processing, feature engineering and class imbalance handling techniques including class weights, oversampling like ADASYN, and under sampling. The combination of IndicBART's contextual understanding on Indian languages and TextGCN's graph-based learning for better context representation creates an effective approach for identifying the non-linear understanding of data, making code-mixed SA effective (Dowlagar, 2021; Dabre, 2022).

Based on the model's performance across these experiments, the most effective method for handling class imbalance, and feature engineering is used with the proposed IndicBART+TextGCN model. IndicBART+TextGCN's hyperparameters are then fine-tuned to improve its overall performance in SA of the code-mixed data (Yusuf, 2023).

3.6 Interpretation and Evaluation

This is a multiclass classification problem, therefore the evaluation metrics as the following have been used. The confusion matrix shown in Figure. 4 is visualised in order to understand the model performance in classifying each class of the data correctly.

TP (Class 0)	FP (Class 0)	FN (Class 0)
FN (Class 1)	TN (Class 1)	FP (Class 1)
FP (Class 2)	FN (Class 2)	TN (Class 2)

Figure 4: Confusion matrix.

The True Positive (TP) is the number of positive samples that the model correctly classifies as positive. The false positive (FP) is the number of negative samples that are classified as positive incorrectly and the True Negative (TN) is the number of negative samples that the model predicts as negative correctly. Finally, the False negative (FN) is the number of the positive sample that is wrongly predicted as negative.

The Accuracy, precision, Recall, F1-score are calculated as follows:

1. Accuracy $= \frac{\text{True Positives+True Negatives}}{\text{Total Samples}}$ is used to measure the number of correct predictions made among all the samples.

2. Precision $= \frac{\text{True Positives}}{\text{True Positives + False Positives}}$ is used to indicate how accurate the model is in correctly predicting positive sentiments.

3. Recall $= \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$ are the measures of the proportion of true positive sentiments predicted by the model to the total positive sentiments in the data.

4. F1-Score $= \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}$ seeks to merge both precision and recall to provide a useful way of considering two measures of performance.

These metrics are calculated for each sentiment class and give an average overall performance score, where the weighted and micro average are considered as data is highly imbalanced.

4 Design Specification

This section explains in detail about the design specification of the proposed work as shown in Figure. 5.

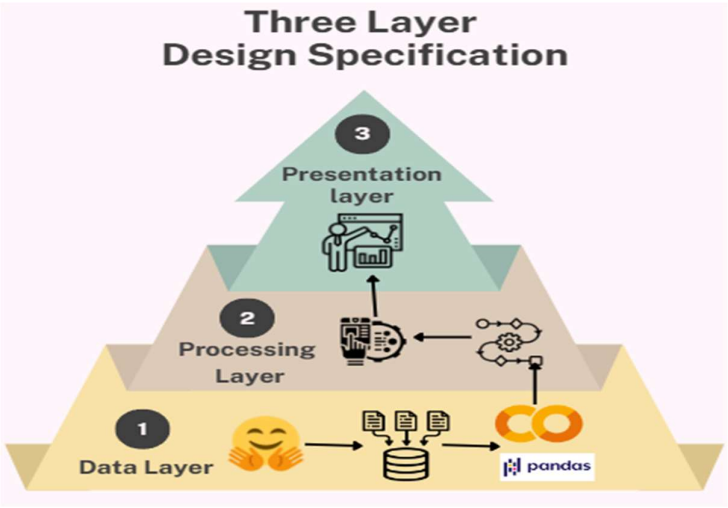


Figure 5: Design specification.

Data Layer : The data is collected from Hugging Face. It is directly loaded into the Google Collab using the dataset library of Hugging Face for further operations.

Processing Layer: The data collected in the data layer is then cleaned and pre-processed, feature engineering, class imbalance, and model trained and evaluated in this layer.

Presentation Layer: Based on the outcome of the processing layer, the most efficient method used is documented and its interpretation findings of the whole process is then presented for the future reference.

5 Implementation

5.1 Data

The data 'tamilmixsentiment' by Chakravarthi B. M. (2020) is collected from the Hugging Face for the SA of Tamil-English code-mixed data. This data has around 15744 Youtube comments with their sentiment annotations labelled. This contains labelled text data with sentiment class from 0 to 4, where '0' is positive, followed by '1' being negative, '2' is unknown, '3' is mixed, and '4' is non-tamil. This 'tamilmixsentiment' dataset from the library of Hugging Face is then loaded into the Google Colab environment. The class distribution is

visualised using the barchart with the help of Matplotlib and Seaborn, highlighting a class imbalance in this dataset. Figure 6 shows WordCloud generated using its library to visually represent the frequently occurring words in data.

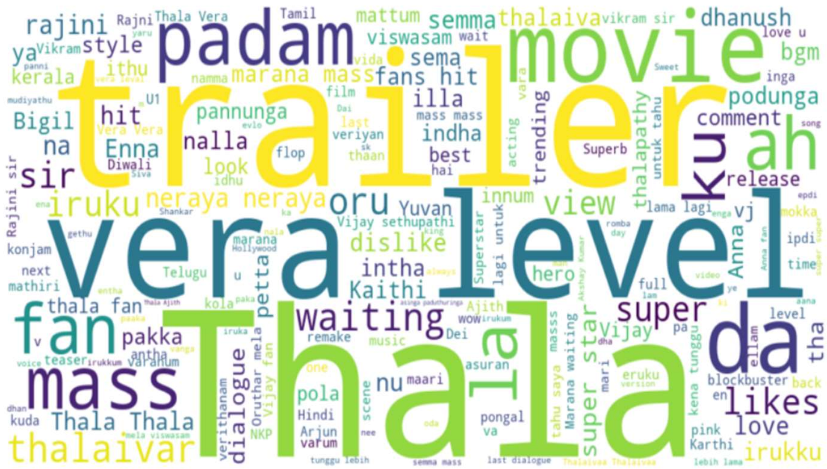


Figure 6: Word cloud.

5.2 Data Pre-Processing

The first step in pre-processing starts off with the conversion of texts into lowercase using the ‘str.lower’ function to maintain consistent data. The punctuation, special characters, numbers and the emojis were also removed using the ‘re.sub’, a regex function. Next, the more common stop words, words without semantic meaning are then filtered out with the help of NLTK libraries. These texts are then lemmatised in order to reduce words to their root form such as changing the words from ‘running’ to ‘run’ using ‘WordNetLemmatizer’ from NLTK (Dowlagar, 2021). These steps are applied uniformly to both train and test datasets to maintain consistency. Finally, the resulting dataset is ensured to be free from noise, consistent, ready for feature engineering and modeling.

5.3 Feature Engineering

Firstly, tokenisation process takes place. It is the process of splitting the data into tokens or individual words using whitespace as the delimiter. These tokenised words are stored in a new column as ‘Wordset’. Next very important step in terms of preserving the meaning of certain words, is where code-mixed texts are back-transliterated into Tamil script using ‘XlitEngine’ from the ‘ai4bharat-transliteration’ library due to the translation is often distorted when left in its romanised form (Dowlagar, 2021).

Then transliterated words are verified and combined with the English words to form a new column named ‘combined’. Figure. 7 shows the ‘combined’ column that ensures the text is in suitable form to represent Tamil and English words respectively in their proper script. For example, the text *“Trailer like and view paaka vanthavanga like panunga”* from ‘original_text’ column will be transformed into *“trailer like and view பாகா வந்தவங்க like*

பணைங்கள" and named as 'combined' column. This column is then used to train the model. This helps to improve the pre-trained embeddings to represent the words more accurately (Dowlagar, 2021). That will in return improves the model's accuracy to capture both languages in the code-mixed data.

Figure 7: Feature Engineering outcome.

5.4 Handling Class imbalance

Class imbalance is a main concern in SA of Tamil-English code-mixed datasets. The observed distribution of the data has 67% positive class, 13% negative while the neutral and mixed feeling classes represent only 5% and 3%, respectively. The annotations of mixed feelings classes are particularly challenging due to their low distribution and annotation difficulty. This imbalance does affect the performance of sentiment classifiers (Srinivasan, 2021). They resulted poorly for neutral and mixed feelings and with higher precision, recall, and F1-Scores for the positive class due to the improper annotation of these labels (Chakravarthi B. M., 2020).

To address this imbalance and annotation issue, a most effective approach will be by combining random under sampling with less meaningful labels, specifically mixed feelings and other language classes. With this intervention, the noise was minimised, and the model can better generalise across the remaining sentiments. This way of handling class imbalance significantly improved performance, particularly for neutral and negative classes, with higher F1-Scores and better generalisability (Gokhale, 2022).

5.5 IndicBART + TextGCN Model Architecture

The proposed IndicBART + TextGCN model architecture has a pre-trained model for word embeddings with 3 layers of TextGCN. IndicBART, is made specifically for Indian regional languages like Tamil, Hindi, Bengali (Dabre, 2022). This model is pre-trained on a large corpus of Indian regional languages, enabling it to effectively handle the details of Tamil and other similar languages in a code-mixed data. The IndicBART model has encoders and decoders. The encoder, MBartEncoder, consists of 12 layers of transformer blocks, where each block contains multi-head self-attention mechanisms, feed-forward networks, and layer normalisation. IndicBART captures the context and relationships of words across multiple languages, both including Tamil and English. It will also include a language identification tag

for each word embedding to properly differentiate the generated embeddings for each Tamil and English words (Dabre, 2022).

Then, a graph data is visualised using this embedding with its language tag with words as nodes and relationship as edges. This is then passed as the input vector for 'batch_size' and 'seq_len, 1024' to the TextGCN module. The graph convolution layers 'gcn1' and 'gcn2' refines these embeddings by considering the graph structure of the data, where each word is connected to other words in the sentence based on syntactic or semantic similarities (Dowlagar, 2021). These two layers find the first order and second order relationship between the words. Once the graph convolutions, the embeddings are then passed through the fully connected layers 'fc1' and 'fc2', where the final layer produces a sentiment classification output with dimensions of 'batch_size, 3', corresponding to the sentiment classes whether positive, neutral, or negative. The mixed feelings are taken in consideration as is not properly annotated and is a non-tamil class (Chakravarthi B. M., 2020).

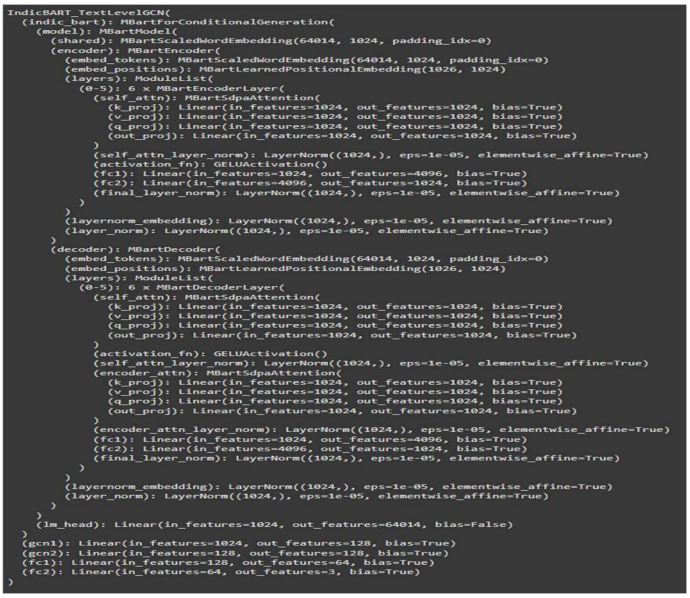


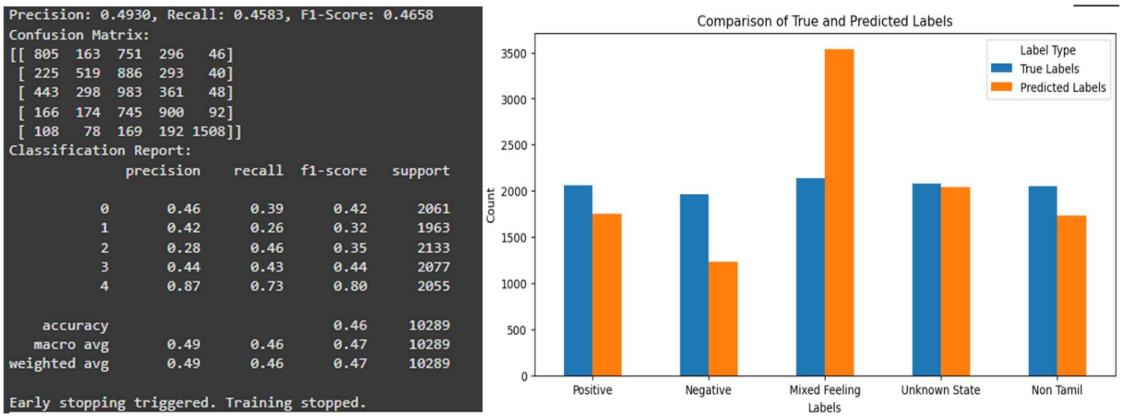
Figure 9: IndicBART+TextGCN Architecture.

6 Evaluation

6.1 Experiment 1 / Handling Class imbalance

Various methods for handling class imbalance were explored using the highly imbalanced class distribution dataset that were discussed above as it is important to achieve a reasonable performance across all sentiment classes (Chakravarthi B. M., 2020). Experiment 1 initially looks at the class weights adjustment method by penalizing the majority class and prioritizing the minority classes during training (Sahoo, 2023), thus the results after 7th epochs of training data did show some improvements. However, its performance metrics only showed overall weighted average precision of '0.49', with '0.46' of recall, and a F1-Score of '0.47', including train loss of '0.78' and test loss of '1.42'. Hence, this model did not properly

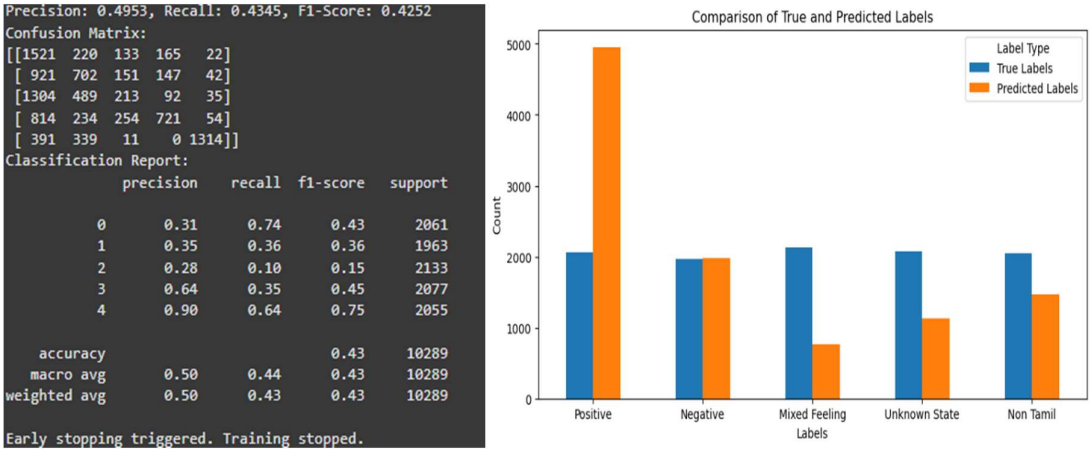
generalize the data and had a significant overfitting despite the improvements in the model's ability to handle class imbalance and being able to classify most of the classes correctly. Figure. 10 is a bar chart with mixed feelings and negative class showing misclassification due to the challenges in annotating the mixed feeling labels (Chakravarthi B. M., 2020). Furthermore, the early stopping triggered in a lesser epoch also suggested that the model was not generalizing well, particularly in mixed and negative classes.



```
Precision: 0.4930, Recall: 0.4583, F1-Score: 0.4658
Confusion Matrix:
[[ 805  163  751  296   46]
 [ 225  519  886  293   40]
 [ 443  298  983  361   48]
 [ 166  174  745  900   92]
 [ 108   78  169  192 1508]]
Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.39      0.42      2061
           1       0.42      0.26      0.32      1963
           2       0.28      0.46      0.35      2133
           3       0.44      0.43      0.44      2077
           4       0.87      0.73      0.80      2055

    accuracy                           0.46     10289
   macro avg       0.49      0.46      0.47     10289
weighted avg       0.49      0.46      0.47     10289

Early stopping triggered. Training stopped.
```


Figure 10: Outcomes of Class weight adjusted model.

The next approach examined was oversampling of the minority classes using ADASYN, since the class weights showed a slightly better performance. This approach is also able to generate synthetic samples for the underrepresented classes (Banerjee, 2020). However, this method has led to overfitting of the positive class and excessive oversampling of the other classes, causing more noise and discrepancies in the distribution. Hence, this noise has made it difficult for the model to generalise. The results of this approach are shown in Figure. 11 with a train loss of '0.23', test loss of '2.95', precision of '0.50', recall of '0.44' and F1-Score of '0.43'. Overfitting issues were highlighted in the Classification Report where the positive class are overrepresented while the model struggled with the rest of the classes. The synthetic samples introduced through oversampling did not however capture the inherent complexities of the data (Sahoo, 2023).



```
Precision: 0.4953, Recall: 0.4345, F1-Score: 0.4252
Confusion Matrix:
[[1521  220  133  165   22]
 [ 921  702  151  147   42]
 [1304  489  213   92   35]
 [ 814  234  254  721   54]
 [ 391  339   11    0 1314]]
Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.74      0.43      2061
           1       0.35      0.36      0.36      1963
           2       0.28      0.10      0.15      2133
           3       0.64      0.35      0.45      2077
           4       0.90      0.64      0.75      2055

    accuracy                           0.43     10289
   macro avg       0.50      0.44      0.43     10289
weighted avg       0.50      0.43      0.43     10289

Early stopping triggered. Training stopped.
```


Figure 11: Outcomes of ADASYN oversampled model.

Random under sampling combined with label exclusion was used in this approach to balance and reduce the size of the positive class of the dataset. The results from this experiment demonstrated better performance, as the model was able to generalize better across the most classes except negative and mixed feelings. The results performance metrics shown in Figure. 12 has a precision score of '0.47', recall at '0.45' and F1-Score of '0.43'. The bar chart showed a balanced distribution of predictions across the classes excluding negative and mixed feelings, but with particularly high performance in the negative class instead of mixed. Despite this improvement, the mixed feelings label still posed challenges due to its ambiguity (Chakravarthi B. M., 2020). By excluding these ambiguous labels, such as the mixed feeling and the non-Tamil, it is evident that higher performance can be achieved.

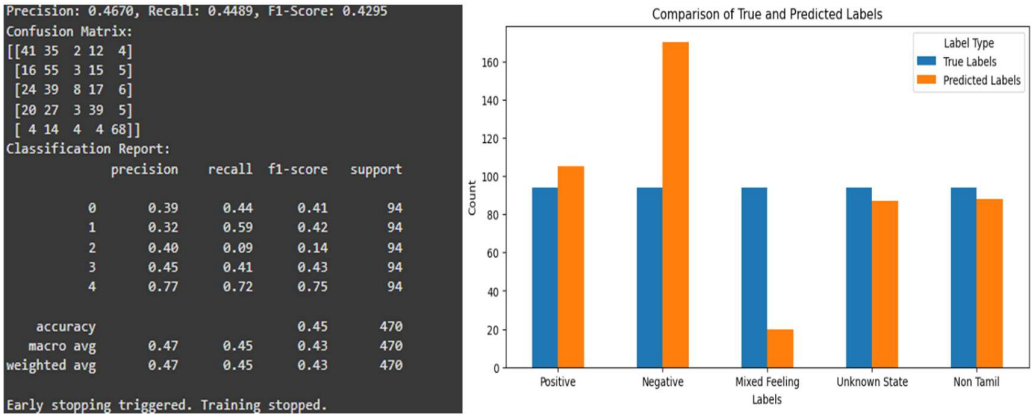


Figure 12: Outcomes of Random under sampled model.

The final approach has addressed the challenges of a class imbalance effectively more than the first two methods. The results showed that the exclusion of noisy, low-frequency labels such as mixed feelings and non-Tamil has allowed the model to focus on more clearly defined sentiment classes. Hence, the result produced better performance overall, especially in terms of generalizability and a well-rounded learning (Chakravarthi B. M., 2020).

6.2 Experiment 2 / Transliteration or Translation

This section discusses how two methods namely transliteration and translation are compared on how they preserved the meaning and structure of the text for SA (Puranika, 2021). The process began with tokenising the code-mixed text into tokens or individual words into a column called 'Wordset' to separate and process Tamil and English words more effectively (Madhani, 2023). Transliteration was performed using 'AI4Bharat-XlitEngine', which accurately converted romanised Tamil words into proper Tamil script with its meaning intact and kept the English words unchanged (Dowlagar, 2021; Dabre, 2022). For instance, the sentence 'Last dialogue vera level ellarukkum porunthum' was converted to 'Last dialogue வேற level எல்லாருக்கும் பொருந்தும்', while maintaining the bilingual nature of each respective script. On the other hand, Google translation API method was also attempted to see if the translator will translate the entire code-mixed sentence into a single language.

Unfortunately, it struggled to maintain the intended meaning and flow. Hence, it is proven that any translation process would be unsuitable for accurately handling bilingual text as it often caused errors, including incorrect mappings and disruptions in the sentence structure (Puranika, 2021).

This shows that transliteration would be the best choice for these types of datasets. Transliteration not only works well but it's able to create a new column with 'combined_words' and also merge the transliterated Tamil script with the unchanged English script (Puranika, 2021). 'Combined_words' column has allowed the models to utilise pre-trained multilingual embeddings and improves SA performance. Overall, transliteration outperformed translation by retaining the bilingual structure and linguistic context of the text. While Google Translation API translation introduced inaccuracies and disrupted the sentence flow,'AI4Bharat-XlitEngine' successfully captured the nuances of both languages, making it a more effective approach for analyzing Tamil-English code-mixed text. This highlights the importance of transliteration in tasks like SA, where maintaining the original linguistic meaning is crucial (Dowlagar, 2021).

6.3 Experiment 3 / Adapting IndicBART+TextGCN

Before the adaptation of the proposed model IndicBART+TextGCN, the initial mBERT-based model was replaced with IndicBART to improve the word embeddings and language-specific understanding. IndicBART was best suited for Tamil-English text as it was pre-trained on Indian languages (Dabre, 2022; Dowlagar, 2021). It also uses a 12-layer transformer encoder and 1024-sized embeddings like mBERT, but is specialised in pre-training superior contextual representations for Indian languages with a tagging feature to identify specific language using tags.

The TextGCN module is integrated to improve word relationship understanding. Once the encoder of IndicBART generates embeddings, a graph data is then created, representing nodes are words and edges are syntactic relationships (Dowlagar, 2021). Through graph convolutional layers, these relationships are refined, enabling the model to learn contextual and semantic dependencies in the code-mixed text. The two layers of the TextGCN helps to identify the first order and second order relationship. These refined embeddings then pass through fully connected layers to provide output of sentiment classification as positive, neutral, and negative classes. Using this hybrid model, the data is trained and its hyper parameters fine-tuned. Learning rate of '0.00001', graph window of '3', Adam optimizer and cross entropy function are considered to avoid overfitting. Finally, the model is trained and evaluated using performance metrics.

IndicBART has demonstrated a much better performance than the base model due to its Indian language-focused training. The IndicBART+TextGCN model, however has achieved a test accuracy of 68.6% with the positive class scoring a precision of '0.79', recall of '0.69', and F1-Score of '0.73', the negative class has a precision of '0.60', recall of '0.90', and F1-Score of '0.72'; meanwhile the neutral class has a precision of '0.76', recall of '0.46', and F1-Score of '0.57'.

Figure 13: Outcomes of proposed model.

The IndicBART+TextGCN model does effectively handle positive and negative sentiments, however the neutral sentiments have been misclassified. An early stopping at epoch 22 has helped to optimise the performance by preventing overfitting with the model achieving a final loss of '0.69', overall weighted precision of '0.71', recall of '0.68' and F1-Score of '0.67'. These results highlight the effectiveness of combining IndicBART's well-rounded language-specific embeddings with TextGCN's graph-based learning for complex code-mixed SA.

6.4 Discussion

This research project has performed three experiments on SA of Tamil-English code-mixed data to improve the classification efficiency of the hybrid model IndicBART+textGCN. The first experiment showed that random under sampling combined with label exclusion was the most effective approach among the first other two approaches in handling class imbalance. It has significantly improved the performance for the neutral and negative classes once the positive class size was reduced and noisy labels were excluded. The second experiment was to determine whether transliteration or translation is best suited for models' understanding. Using the 'AI4Bharat-XlitEngine' for transliteration has enabled the romanised Tamil words to be converted into proper Tamil script while retaining English words intact. The final experiment was the adaptation of IndicBART on tokenise embeds words. The integration of IndicBART and TextGCN has improved the model to achieve more accurate sentiment classification.

Furthermore, fine-tuning the IndicBART+TextGCN model using the Adam optimizer with weight decay and a learning rate of '0.00001' has prevented overfitting. The training ran for 22 epochs with a batch size of '1', although with an early stop training, there was no further improvement in validation loss. These fine-tuned hyperparameters, when combined with the proper data pre-processing techniques, class imbalance handling approaches and feature engineering, has helped the model to achieve an accuracy of 68%. Even though positive and negative classes showed higher precision, recall, F1-Scores and the model's final loss of '0.69' has demonstrated good generalization, the neutral class has remained challenging by achieving lower recall and F1-Scores. These results have highlighted the importance of addressing class imbalance, using transliteration over translation, and fine-tuning specialized models like IndicBART with TextGCN to handle the complexities of Tamil-English code-mixed SA.

7 Conclusion and Future Work

This project has successfully demonstrated that a hybrid model namely IndicBART+TextGCN can be efficiently fine-tuned to classify sentiments on Tamil-English code-mixed when several methods and approaches are combined.

As a recommendation for future work, future researchers could create a customised word dictionaries that include slang, abbreviations, and interjection words will help in handling informal language better, making the text cleaner for better analysis and accuracy.

Moreover, we can also experiment with other effective variants of GNNs, such as GCN with self-attention node features or Graph Attention Networks (GAT) to improve the model's parallelism and batch processing abilities can help in handling large-scale datasets faster and more efficiently. In conclusion, by implementing the above improvements, we can ensure that the model is not only well able to work effectively with large datasets, but also efficiently giving better sentiment classification results.

References

Ahmad, G. S. (2019). 'Review on Sentiment Analysis of Indian Languages with a Special Focus on Code Mixed . *International Conference on Automation, Computational and Technology Management (ICACTM).*

Androutsopoulos, J. (2007). Bilingualism in the mass media and on the internet. *Bilingualism: A Social Approach.* URL:https://www.researchgate.net/publication/304731364_Bilingualism_in_the_Mass_Media_and_on_the_Internet

Babu, Y. (2021). Sentiment Analysis on Dravidian Code-Mixed YouTube Comments using Paraphrase XLM-RoBERTa Model. *Proceedings of FIRE 2021: Forum for Information Retrieval Evaluation.* URL:https://ceur-ws.org/Vol-3159/T6-22.pdf

Banerjee, S. J. (2020). Sentiment Analysis of Code-Mixed Dravidian text using XLNet. *Proceedings of the FIRE 2020 Working Notes Track on Dravidian-CodeMix.* CEUR Workshop Proceedings. URL:https://doi.org/10.48550/arXiv.2010.07773

Chakravarthi, B. M. (2020). Corpus creation for sentiment analysis in code-mixed Tamil-English text. *Proceedings of the 1st Joint SLTU and CCURL Workshop (SLTU-CCURL 2020), at the Language Resources and Evaluation Conference (LREC 2020), Marseille, France.* European Language Resources Association (ELRA). URL: https://aclanthology.org/2020.sltu-1.28.pdf

Chakravarthi, B. R. (2022). DravidianCodeMix: Sentiment analysis and offensive language identification dataset for Dravidian languages in code-mixed text. *Language Resources and Evaluation.* URL: https://doi.org/10.1007/s10579-022-09583-7

Chutia, T. (2024). A review on emotion detection by using deep learning techniques. *Artificial Intelligence Review.* URL:https://link.springer.com/article/10.1007/s10462-024-10831-1

Dabre, R. S. (2022). IndicBART: A pre-trained model for Indic natural language generation. *Findings of the Association for Computational Linguistics: ACL 2022.* Association for Computational Linguistics. URL: https://doi.org/10.18653/v1/2022.findings-acl.145

Dowlagar, S. (2021). Graph Convolutional Networks with Multi-headed Attention for Code-Mixed Sentiment Analysis. *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages (DravidianLangTech 2021)* (pp. 59–67). Association for Computational Linguistics. URL: https://aclanthology.org/2021.dravidianlangtech-1.8.pdf

Ehsan, T. T. (2024). Sentiment Analysis of Code-Mixed Tamil and Tulu by Training Contextualized ELMo Representations. . University of Gujrat, Pakistan; University of Jaffna, Sri Lanka; Hamad : DravidianLangTech-EACL2024.

Gokhale, O. P. (2022). Optimize_Prime@DravidianLangTech-ACL2022: Emotion Analysis in Tamil. *Association for Computational Linguistics.* ACL 2022 DravidianLangTech Workshop. URL: https://aclanthology.org/2022.dravidianlangtech-1.35

Hermawan, M. (2020). Sentiment analysis using deep learning techniques: A review. *International Journal of Technology.* URL: https://link.springer.com/article/10.1007/s13735-023-00308-2

Khurana, D. K. (2022). Natural language processing: State of the art, current trends and challenges. *Springer Nature.* URL:https://link.springer.com/article/10.1007/s11042-022-13428-4

Madhani, Y. P. (2023). Aksharantar: Open Indic-language transliteration datasets and models for the next billion users. URL: https://doi.org/10.48550/arXiv.2205.03018

Mahadzir, N. O. (2021). Sentiment analysis of code-mixed text: A review. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE.* URL: https://www.researchgate.net/publication/378548409_Sentiment_Analysis_of_Code-Mixed_Text_A_Comprehensive_Review

Majumder, M. G. (2022). Perceived usefulness of online customer reviews: A review mining approach using machine learning & exploratory data analysis. *Journal of Business Research.* URL:https://www.sciencedirect.com/science/article/abs/pii/S0148296322005446

Perera, A. (2024). Sentiment Analysis of Code-Mixed Text: A Comprehensive Review. *JUCS - Journal of Universal Computer Science*, (pp. 30, 242-261.). URL 10.3897/jucs.98708.

Puranika, K. B. (2021). IITT@Dravidian-CodeMix-FIRE2021: Transliterate or translate? Sentiment analysis of code-mixed text in Dravidian languages. *CEUR Workshop Proceedings.*

Sahoo, C. W. (2023). Sentiment analysis using deep learning techniques: a comprehensive review.. *Int J Multimed Info Retr 12, 41.* URL: https://doi.org/10.1007/s13735-023-00308-2

Sampath, K. (2024). Transformer Based Sentiment Analysis on Code Mixed Data. *Procedia Computer Science, 5th ICIDCA 2024.* URL: https://www.sciencedirect.com/science/article/pii/S1877050924006173

Shanmugavadivel, K. S. (2022). Deep learning-based sentiment analysis and offensive language identification on multilingual code-mixed data. *Computer Speech & Language,.* URL https://doi.org/10.1016/j.csl.2022.101407

Singhal, K. (2024). Sentiment Analysis of Code-Mixed Tamil Using RoBERTa. . Thapar Institute of Engineering and Technology.: Transformers@DravidianLangTech-EACL2024.

Srinivasan, R. (2021). Sentimental analysis from imbalanced code-mixed data using machine learning approaches. *Distributed and Parallel Databases.* URL:https://link.springer.com/article/10.1007/s10619-021-07331-4

Vaswani, A. e. (2023). Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.* URL:https://arxiv.org/abs/1706.03762

Winata, G. A. (2023). The decades progress on code-switching research in NLP: A systematic survey on trends and challenges. *Findings of the Association for Computational Linguistics: ACL 2023.* URL: https://aclanthology.org/2023.findings-acl.185.pdf

Yusuf, A. S. (2023). Fine-tuning Multilingual Transformers for Hausa-English Sentiment Analysis. *2023 13th International Conference on Information Technology in Asia (CITA).* URL: https://ieeexplore.ieee.org/document/10262742