Chapter 5

Sentiment Analysis on Dravidian Code-Mixed Data

Following the exploration of word-level language identification on code-mixed data, we transition to another research objective outlined in Section 1.8 of this thesis. This chapter undertakes an investigation into sentiment analysis (SA) on Dravidian code-mixed data. The structure of this chapter is organized as follows. Section 5.1 offers an elaborate presentation of the problem statement and research questions to be addressed here. The dataset is comprehensively described in Section 5.2, accompanied by an exposition of the preprocessing techniques employed. Section 5.3 provides a detailed account of the proposed methodology and experimental setup. Subsequently, in Section 5.4, we present the results and conduct a comparative analysis of the performance against state-of-the-art models, in terms of standard evaluation metrics, alongside error analysis and ensuing discussion. Ultimately, our findings are encapsulated in Section 5.5, drawing conclusions from the conducted study.

5.1 Problem Statement

The problem we are going to explore in this chapter is motivated from the Dravidian CodeMix Sentiment Analysis task conducted at FIRE 2020 and 2021. The goal of the shared task is to determine the sentiment polarity of code-mixed data from Dravidian language pairs (Malayalam-English, Tamil-English, and Kannada-English) from the comment section of YouTube videos. The text is grouped into five categories: Positive, Negative, Mixed_feelings, unknown_state and not-<*language*>[1]. The shared task is run for two years in a row. The task encountered a number of research issues and the dataset was offered to investigate and explore them. In this work, we focus on the following research questions (RQs).

- *RQ-1:* If a given dataset contains a mixture of pure monolingual, transliterated monolingual, single-script multilingual and mixed script multilingual text, what is the best strategy for conducting sentiment analysis?
- *RQ-2:* How important is the language identification (LID) task for code-mixed data processing? Can a language identification model in parallel with a pre-trained model improve the overall system performance any further? We seek to find the answer particularly, with respect to sentiment analysis task.
- *RQ-3:* Is it possible to see SA as a multi-level problem rather than a multi-class problem? If yes, can a multi-level hierarchical model enhance the performance?

5.2 Datasets

The Dravidian-CodeMix shared task[2] organizers provide a dataset comprising a training, a development and a test set. The organizers of the study disseminated the training and development datasets in the format of tab-separated values (.tsv) files. These files are structured with two distinct columns: “text” and “label”. Each individual row

[1]The language might be Tamil, Kannada or Malayalam
[2]https://dravidian-codemix.github.io/2021/index.html

within these files represents a single data sample containing a youtube comment. Test dataset has an additionally "id" field for each data-item. Entire dataset maintained this format consistently across all three language pairs: Tamil-English (TA-EN) [120], Kannada-English (KA-EN) [121] and Malayalam-English(MA-EN) [122]. The statistics of training, development, and test data corpus collection and their class distribution are shown in Table 5.1. The details of the dataset and benchmark results are given in overview [123] and findings [124] of the sentiment analysis of Dravidian languages. Even though the entire dataset is segregated as language pairs, there are examples of

Table 5.1: Data Distribution for sentiment detection of code-mixed text in Dravidian languages

Tamil - English				
Class	**Training**	**Development**	**Test**	**Total**
Positive	20070	2257	2546	24873
Negative	4271	480	477	5228
not-Tamil	1667	176	244	2087
Mixed_feelings	4020	438	470	4928
unknown_state	5628	611	665	6904
Total	35656	3962	4402	44020
Kannada - English				
Class	**Training**	**Development**	**Test**	**Total**
Positive	2823	321	374	3518
Negative	1188	139	157	1484
not-Kannada	916	110	110	1136
Mixed_feelings	574	52	65	691
unknown_state	711	69	62	842
Total	6212	691	768	7671
Malayalam - English				
Class	**Training**	**Development**	**Test**	**Total**
Positive	6421	706	780	7907
Negative	2105	237	258	2600
not-Malayalam	1157	141	147	1445
Mixed_feelings	926	102	134	1162
unknown_state	5279	580	643	6502
Total	15880	1766	1962	19616

impurity: i.e., a TA-EN data may contain comments in Hindi (HI) or other languages. The dataset also suffers from other general problems of social media data, particularly code-mixed data like the sentences are short with a lack of well-defined grammatical structures and many spelling mistakes. The example text from three language pairs are shown in Table 5.2.

Table 5.2: Example of code-mixed text in Dravidian languages from three language pairs for all classes

Text	Label
Tamil - English	
வன்னியர் சாதி சார்பாக படம் வெற்றி பெற வாழ்த்துக்கள்....	Positive
Il FAUT que WTC voit ca	not-Tamil
வெந்து வேகதா அரை வேகடு ஹீரோது	Negative
தல போல வராது.... Waiting viswasam trailer	Mixed_feelings
Rajini fans comment. Neutral fans like.	unknown_state
Kannada - English	
Tiktokers present situation... nನೋಡುವವರು ಯಾರು ನಮ್ಮ ವೀಡಿಯೋನೂ	Negative
I do like Guru Deshpande Movies. Im gonna rush	not-Kannada
ಮನೆಯಿಂದಲೇ ಸಂಪಾದಿಸಬೇಕೇ? With income proof. WhatsApp 6362******	unknown_state
Super ಸಾಂಗ್ ವೆರಿ ನೈಸ್....	Positive
@Wild Rex ಕಟ್ಟಬೇಕು bronಖಂಡಿತಾ ಕಟ್ಟುತ್ತೆ bro	Mixed_feelings
Malayalam - English	
ഫഹദിന്റെ ഭാര്യായിട്ട് അഭിനയിച്ച ചേച്ചി ഔട്സ്റ്റാന്ഡിങ് ആക്ടിങ്	Positive
ആമയെ വെള്ളത്തിൽ മുക്കി കൊല്ലുന്ന ഇനമാ.. ..	unknown_state
yenthu chali annu bro ethu	Negative
nyc but best actor poley aayaaal seriyaavilla	Mixed_feelings
The face of indiaaikkkkkkkaaaaaa luv uuuuuu	not-Malayalam

Each dataset is labeled into five classes. The class labels are as follows.

- **Positive:** A comment comprises a textual cue, either explicit or implicit, that suggests the speaker is in a good frame of mind.
- **Negative:** Comments contain an explicit or implicit clue in the text suggesting that the speaker is in a negative state.
- **Mixed_feelings:** Comments contain an explicit or implicit clue in both positive and negative feelings.
- **unknown_state:** The comment does not contain an explicit or implicit indicator of the speaker's emotional state.

- **Not in intended language:** The text does not contain the desired Dravidian language. For example, in TA-EN dataset, if the sentence does not contain any Tamil word written in Tamil script or Roman script then it is not-Tamil.

5.3 Methodology and Experiment Setup

This section provides a detailed description of the methodology and experimental setup employed in this study for the different RQs presented in Section 5.1. Here we first set necessary technical background and briefly discuss some of the theoretical concepts, which have been used in further sections.

5.3.1 Multilingual BERT or mBERT

Word embedding play a crucial role in natural language processing (NLP) tasks by representing words as dense numerical vectors in a continuous vector space. Here we use mBERT for word embeddings. mBERT is an variant of BERT already discussed (Section 2.9) trained with multilingual dataset. Multilingual BERT or mBERT (`bert-base-multilingual-cased`[3]) is pre-trained on cased text in the top 104 languages with the largest Wikipedia and has a total of 179M parameters with 12 transformers blocks, 768 hidden layers and 12 attention head. This model takes a special `[CLS]` token as input first, followed by a sequence of words as input. It then passes the input to the next layer. `[CLS]` here stands for Classification. Each layer applies self-attention and passes the result through a feedforward network to the next encoder.

5.3.2 Language Identification tool by Googletrans

We use Googletrans[4], a free and unlimited Python library that implements Google Translate API. It uses the Google Translate Ajax API to make calls to such methods

[3]https://github.com/google-research/bert/blob/master/multilingual.md
[4]https://pypi.org/project/googletrans/

as detect (for language detection) and translate (to a target language).

5.3.3 Data Pre-processing

The dataset provided was not very clean and required pre-processing. The pre-processing pipeline contains several steps to ensure the cleanliness and standardization of the text data. Following are some of the techniques adopted.

- In order to mitigate the effect of repeated characters, a restriction is imposed wherein only two consecutive repetitions are allowed. For instance, the word "goooooooood" is simplified to "good".
- Hashtags and URLs are eliminated from the text.
- Exclamations and other punctuation marks are removed.
- Non-ASCII characters, symbols, numbers, and special characters are all excluded.
- Emojis are replaced with their corresponding semantic text.
- Any trailing or excessive white spaces are stripped off, ensuring a uniform text format.

The cleaned data is then subject to different steps as the research questions demanded. We detail the methodology adopted for each such RQ below.

5.3.4 Methodology for RQ-1

In order to check if the dataset is code-mixed or not, we apply word-level language identification to determine which languages are involved in an utterance. We deploy a language identification (LID) module (implemented using Googletrans), which provides word-level language identification for each word. We see the number of constituent languages used in a sentence and the frequency with which a user changes languages in a given sentence and CMI is calculated for each sentence. All the sentences in the dataset are then divided into two categories. The sentences having a CMI value of zero are monolingual. The rest are classified as code-mixed sentences (See Figure

5.1). Subsequently, we train the model on training and validation data separately on monolingual and code-mixed data.

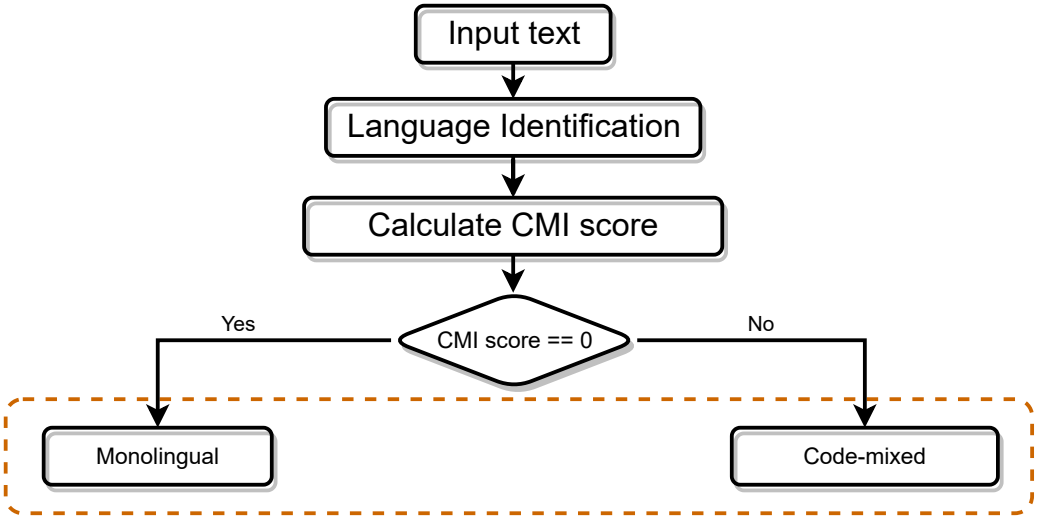

Figure 5.1: Model Architecture for identifying monolingual and code-mixed data

Our initial objective is to first investigate the impact of training using different components of data from the given datasets: using

1. all the original data (code-mixed and monolingual together)
2. only code-mixed data
3. employing only monolingual data

Separation of (2) and (3) is done using CMI as discussed above. Subsequently, we conduct experiments using test datasets on each of the cases above.

The given task here is seen as single-stage classification with five classes.

The sentences are converted to vectors using word embedding according to (mBERT) pre-trained model [125] to get a vector as an embedding for the sentence to be used for classification. On top of that, we also use a multi-layer perceptron (MLP) where we freeze the parameters of the pre-trained model and generate the prediction.

For the implementation, HuggingFace's transformers library [5] is utilized. HuggingFace is a Python package that offers pre-trained and adaptable transformer models

[5]https://huggingface.co/transformers/pretrained_models.html

to be used for various NLP tasks. The implementation environment is the PyTorch library, which supports GPU computation. The mBERT models are run using Google Colab. We train our classifier 10 epochs with a batch size of 32. The `AdamW` optimizer is used, and the dropout value is set at 0.1. The learning rate is 2e-5. For tokenization, we utilize the HuggingFace transformers' pre-trained BERT tokenizer. We utilize the HuggingFace library's `BertForSequenceClassification` module for tinkering and sequence classification.

5.3.5 Methodology for RQ-2

In RQ-1, we considered sentiment analysis task as a multi-class classification problem. However, for the given dataset, one of the classes is "Not in intended language", that, according to the original description of the dataset, is any statement which does not contain a word from the specified language. This class needs language identification while the other four are based on sentiment identification. In RQ-2, we hypothesize that this language identification should be used directly in classification. While a classifier can be used for finding the sentiments, another can be used for finding the language / not language part. Consequently, we approach it as a multi-class classification problem like in RQ-1. But additionally we use another language identification (LID) module to identify not in language to augment the classification task.

For the multi-class classification, we use the mBERT model which we already discuss in the previous section. We use the combined (monolingual and code-mixed) dataset for training, development and testing for this experiment.

For the language identification (LID), like before, we use Googletrans API to decide whether a sentence is in <language> or 'not-<language>'. The mBERT predictions as sentimental tags (positive, negative, mixed_feelings or unknown_state) are kept only if the output is <language>, otherwise, are marked as 'not-<language>' and added to the class of 'not-<language> by mBERT as well (see Figure 5.2).

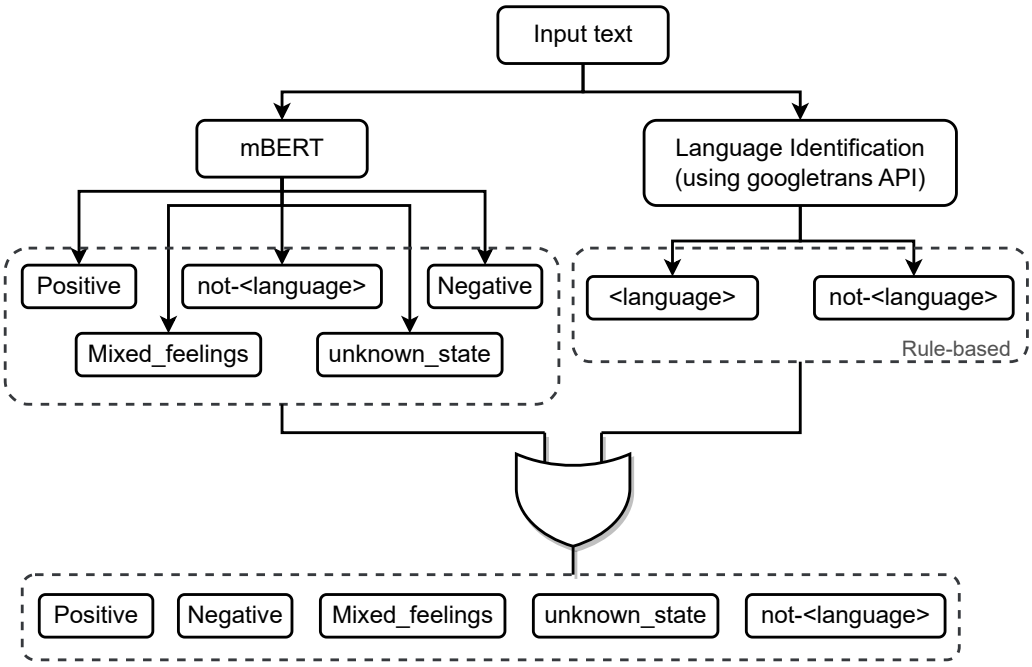

Figure 5.2: Model architecture for multi-class classification with ruled-based language tag

5.3.6 Methodology for RQ-3

In this experiment, we develop our system in a hierarchical fashion. First, we use the LID module to determine <language> and one 'not-<language>'. All the sentences labeled as <language> only by the LID-module are fed to the mBERT module for sentiment classification. The 'not-<language>' are solely decided by the LID module, unlike in RQ-2 (see Figure 5.3).

For this experiment, we utilise the combined dataset (consisting of monolingual and code-mixed samples) for training, development, and testing purposes.

5.4 Results and Discussion

This section reports experimental results conducted to address the RQs on the three language pairs followed by discussion on the findings. We evaluate and compare every model's performance in terms of precision, recall, F_1-score, weighted average F_1-score and accuracy. Furthermore, in Section 5.4.4, an in-depth exploration into error analysis

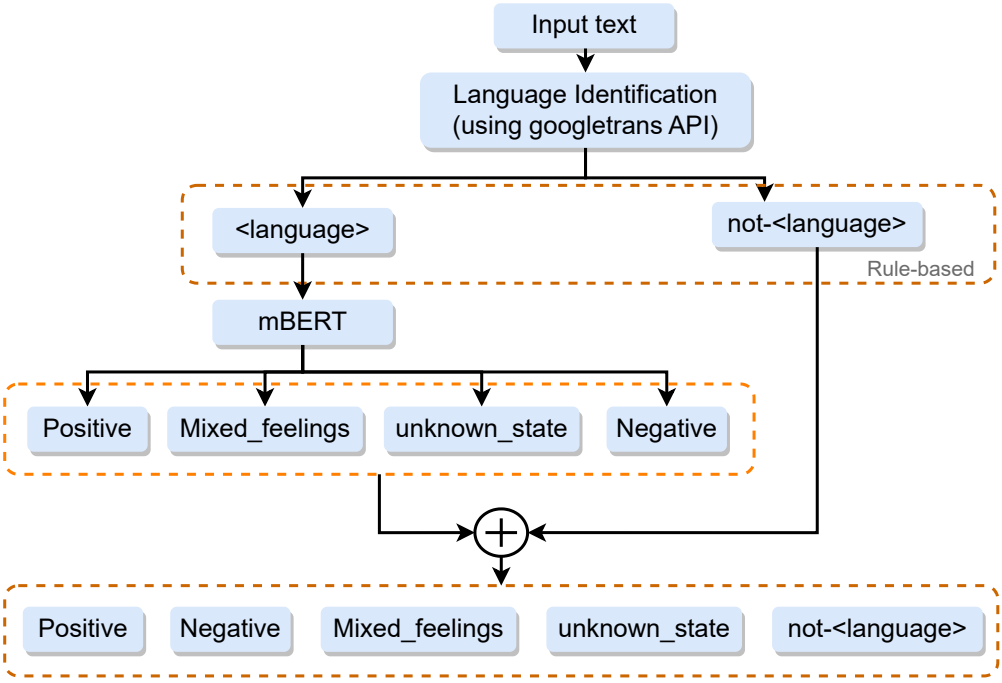

Figure 5.3: Model architecture for hierarchical approach with mBERT

is also undertaken.

5.4.1 Results for RQ-1

The primary objective of RQ-1 is to check whether the data is code-mixed or not. It is clear from Table 5.3 that the dataset contains a large percentage of monolingual data. In the Tamil-English dataset, more than 40% of the data is monolingual in the three distributions: training, development, and test. In contrast, the Malayalam-English dataset has lower monolingual percentages, with 35.8%, 37.5%, and 34.5% of the data being monolingual in the training, development, and test distributions, respectively. However, share of monolingual data is the highest in Kannada-English: 57.4%, 57.4%, and 56.6% in training, development, and test data respectively.

Table 5.4 depicts the code-mixing level involved in each distribution of three language pairs. We report the average CMI values in two ways: first, considering the original dataset including code-mixed and monolingual (represented as CMI-All) and

Table 5.3: The statistics of monolingual and code-mixed data involved in training, development, and test datasets of all three language pairs.

Tamil - English					
Training		Development		Test	
Monolingual	Code-mixed	Monolingual	Code-mixed	Monolingual	Code-mixed
14385	21271	1588	2374	1783	2619
Kannada - English					
Training		Development		Test	
Monolingual	Code-mixed	Monolingual	Code-mixed	Monolingual	Code-mixed
3568	2644	397	294	435	333
Malayalam - English					
Training		Development		Test	
Monolingual	Code-mixed	Monolingual	Code-mixed	Monolingual	Code-mixed
5702	10186	664	1102	676	1286

then, considering only the pure code-mixed ones (discarding the monolingual ones, represented as CMI-Mixed). Since, the given datasets contain a substantial share of monolingual sentences having CMI values zero, the scores increase from CMI-All to CMI-Mixed as these values are discarded from calculation in the second.

Table 5.4: Level of code-mixing (CMI values) involved in training, development, and test datasets of all three language pairs.

	Tamil-English		Kannada-English		Malayalam-English	
	CMI-All	CMI-Mixed	CMI-All	CMI-Mixed	CMI-All	CMI-Mixed
Train	18.39	30.83	13.67	32.13	19.16	29.89
Dev	18.63	31.09	14.45	33.97	18.82	30.16
Test	18.38	30.9	13.5	31.13	20.09	30.65

We see the impact of different types of training data: all original data with monolingual (mono) and code-mixed (CM) combined; only code-mixed data; and only monolingual data for 3 language-pairs on testing with all data (monlingual + code-mixed). The performance of our proposed models is shown in terms of precision, recall, macro-averaged F_1-score, and weighted average F_1-score in Table 5.5, 5.6 and Table 5.7, respectively.

The findings reveal a performance decline for all three language pairs when we exclusively employ either code-mixed or monolingual data only for training. For instance,

in the case of Malayalam-English, the weighted average F_1 score was 0.69. However, when we train the model solely on code-mixed data and conduct test on the combine dataset, the weighted average F_1 score decreases to 0.66, indicating a performance drop of approximately 4% (Table 5.7). Furthermore, when we employ monolingual data for training and test the model on the combine dataset, the weighted average F_1 score decrease by 14%, from 0.69 to 0.59.

These outcomes suggest that utilizing solely code-mixed or monolingual data for training adversely affects the model's performance on the combined test dataset, which consists of both code-mixed and monolingual samples.

Table 5.5: Precision, recall, F_1-scores, and support for all experiments on Tamil-English test data

	Train on Mono and CM data			**Train on CM data only**			**Train on Mono data only**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.26	0.26	0.26	0.22	0.23	0.22	0.22	0.21	0.21	470
Negative	0.40	0.34	0.36	0.38	0.31	0.34	0.36	0.25	0.29	477
Positive	0.75	0.79	0.77	0.75	0.75	0.75	0.72	0.81	0.76	2546
not-Tamil	0.65	0.53	0.58	0.61	0.46	0.52	0.59	0.45	0.51	244
unknown_state	0.40	0.38	0.39	0.37	0.43	0.40	0.38	0.33	0.35	665
macro avg	0.49	0.46	**0.47**	0.47	0.44	0.45	0.45	0.41	0.43	4402
weighted avg	0.60	0.61	**0.60**	0.59	0.58	0.58	0.57	0.59	0.58	4402
Accuracy		0.61			0.58			0.59		

Table 5.6: Precision, recall, F_1-scores, and support for all experiments on Kannada-English test data

	Train on Mono and CM data			**Train on CM data only**			**Train on Mono data only**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.23	0.28	0.25	0.12	0.14	0.13	0.11	0.12	0.12	65
Negative	0.66	0.54	0.59	0.59	0.50	0.54	0.65	0.52	0.58	157
Positive	0.70	0.74	0.72	0.70	0.72	0.71	0.70	0.76	0.73	374
not-Kannada	0.59	0.59	0.59	0.53	0.55	0.54	0.62	0.59	0.61	110
unknown_state	0.31	0.29	0.30	0.24	0.24	0.24	0.33	0.29	0.31	62
macro avg	0.50	0.49	**0.49**	0.44	0.43	0.43	0.48	0.46	0.47	768
weighted avg	0.61	0.60	**0.60**	0.57	0.56	0.56	0.60	0.60	0.60	768
Accuracy		0.60			0.56			0.60		

We also look at the performance when the system is exclusively trained on pure code-mixed data and separate tests are conducted on monolingual, code-mixed, and combined datasets. Table 5.8 presents the weighted average F_1 scores for three language pairs (We chose weighted average F_1 to summarize the performances without going to the details). Notably, the performance of Kannada-English and Malyalam-English language

Table 5.7: Precision, recall, F_1-scores, and support for all experiments on Malayalam-English test data

	Train on Mono and CM data			**Train on CM data only**			**Train on Mono data only**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.40	0.16	0.23	0.34	0.31	0.32	0.22	0.14	0.17	134
Negative	0.55	0.53	0.54	0.56	0.44	0.49	0.40	0.37	0.38	258
Positive	0.74	0.81	0.77	0.72	0.78	0.75	0.64	0.75	0.69	780
not-Malayalam	0.83	0.71	0.77	0.78	0.63	0.70	0.61	0.72	0.66	147
unknown_state	0.71	0.74	0.72	0.67	0.71	0.69	0.67	0.58	0.62	643
macro avg	0.64	0.59	**0.61**	0.61	0.57	0.59	0.51	0.51	0.51	1962
weighted avg	0.69	0.70	**0.69**	0.66	0.67	0.66	0.59	0.60	0.59	1962
Accuracy		0.70			0.67			0.60		

pairs exhibit a decline when utilizing both the combined and monolingual test data. However, the overall performance for TN-EN pair is not hampered.

It is essential to consider both the weighted average F_1 score and the support value as crucial metrics. A higher weighted average F_1 score signifies superior overall performance, while a larger support value indicates higher number of data points. This implies that a larger dataset was used, which could be the result of collecting data from a larger population, a longer time span, or a more extensive sampling process. The utilization of a larger dataset provides a broader range of information, potentially capturing more diverse scenarios, variations. The observations depicted in Table 5.8 indicate that, relative to monolingual data, the models perform marginally better when subjected to tests using code-mixed data. Regarding the Tamil-English language pair, the performances of the models exhibit a comparable trend on both code-mixed and combined datasets, with monolingual data showing promising results. Nevertheless, when considering the support value, the model achieves optimal performance on combined datasets, followed by code-mixed data and then monolingual data. In the case of the Kannada-English and Malayalam-English language pairs, conducting tests on code-mixed data yields favorable outcomes in comparison to monolingual and combined datasets. In the specific scenario of Malayalam-English language pair, training and testing the model on code-mixed data lead to a weighted average F_1 score of 0.69. However, training on code-mixed data and testing on the complete dataset result in a

weighted average F_1 score of 0.66, indicating a decline in performance of approximately 4%. Remarkably, utilizing monolingual data for testing purposes further decreases the weighted average F_1 score by 11.5%, from 0.69 to 0.61.

Table 5.8: Weighted average F_1-scores and support for three language pairs where model is trained only on CM data but tested on all, Monolingual and CM data

Language pair	Test on all data		Test on monolingual data only		Test on code-mixed data only	
	Weighted avg. F_1-score	support	Weighted avg. F_1-score	support	Weighted avg. F_1-score	support
Tamil-English	0.58	4402	**0.60**	1783	0.58	2619
Kannada-English	0.56	768	0.53	435	**0.60**	333
Malayalam-English	0.66	1962	0.61	676	**0.69**	1286

5.4.2 Results for RQ-2

Here we use the mBERT model to forecast the sentiment polarity for the Tamil-English, Kannada-English and Malayalam-English language pairs. However, we also use a LID module followed by a rule-based approach to divide sentences into two categories: language and not-<language>. Finally we combine these predictions with the mBERT prediction and then provide prediction.

Table 5.9, 5.10 and 5.11 display the performance of the test outcomes for the mBERT model and mBERT + Ruled-based systems. For all three language pairs, we can observe a performance increase in terms of weighted average F_1 scores. Our model witnessed an increase of 6.6%, 10% and 4.3% on Tamil-English, Kannada-English and Malayalam-English respectively. In this study we wanted particularly to improve the not-language tag performance with the addition of LID module which we can clearly observe for all language pairs. For Tamil-English language F_1 score of not-Tamil class improve from 0.58 to 0.87 (almost 50%). For Kannada-English language F_1 score of not-Kannada class improve from 0.59 to 0.83 (almost 40%). For Malayalam-English language F_1 score of not-Malayalam class improve from 0.77 to 0.89 (almost 15%).

Not only the not-<language> class, we also see slight performance improvement in other classes as well. The reason behind is that mBERT sometimes wrongly classifies not-<language> as other classes which create a large set of false positives. When these

are removed as detected by the (LID + rule-based) system, finally the precision value of all classes across the language pairs improve. Overall, mBERT + Rule-based model outperforms the mBERT model for all three language pairs.

Table 5.9: Comparison of Precision, recall, F_1-scores, and support with our proposed model for RQ-2 on Tamil-English test data

	mBERT			**mBERT + Ruled-based systems**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.26	0.26	0.26	0.31	0.19	0.23	470
Negative	0.40	0.34	0.36	0.44	0.39	0.41	477
Positive	0.75	0.79	0.77	0.76	0.83	0.80	2546
not-Tamil	0.65	0.53	0.58	0.79	0.98	0.87	244
unknown_state	0.40	0.38	0.39	0.45	0.43	0.44	665
macro avg	0.49	0.46	0.47	0.55	0.56	**0.55**	4402
weighted avg	0.60	0.61	0.60	0.63	0.66	**0.64**	4402
Accuracy		0.61			0.66		

Table 5.10: Comparison of Precision, recall, F_1-scores, and support with our proposed model for RQ-2 on Kannada-English test data

	mBERT			**mBERT + Ruled-based systems**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.23	0.28	0.25	0.21	0.28	0.24	65
Negative	0.66	0.54	0.59	0.71	0.54	0.61	157
Positive	0.70	0.74	0.72	0.77	0.76	0.76	374
not-Kannada	0.59	0.59	0.59	0.71	1.00	0.83	110
unknown_state	0.31	0.29	0.30	0.39	0.24	0.30	62
macro avg	0.50	0.49	0.49	0.56	0.56	**0.55**	768
weighted avg	0.61	0.60	0.60	0.67	0.67	**0.66**	768
Accuracy		0.60			0.67		

Table 5.11: Precision, recall, F_1-scores, and support with our proposed model for RQ-2 on Malayalam-English test data

	mBERT			**mBERT + Ruled-based systems**			
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	support
Mixed_feelings	0.40	0.16	0.23	0.43	0.25	0.32	134
Negative	0.55	0.53	0.54	0.63	0.56	0.59	258
Positive	0.74	0.81	0.77	0.77	0.82	0.79	780
not-malayalam	0.83	0.71	0.77	0.81	0.97	0.89	147
unknown_state	0.71	0.74	0.72	0.72	0.74	0.73	643
macro avg	0.64	0.59	0.61	0.67	0.67	**0.66**	1962
weighted avg	0.69	0.70	0.69	0.72	0.73	**0.72**	1962
Accuracy		0.70			0.73		

5.4.3 Results for RQ-3

Here we propose a hierarchical model. In the first level of hierarchical model, we use a LID module followed by a rule-based approach to divide sentences into two categories: <language> and not-<language>. In the second level, all the sentences labeled as <language> are sent to mBERT module for sentiment classification among four classes: positive, negative, unknown_state and mixed_feelings.

Table 5.12, 5.13, and 5.14 display the result of the test data for the mBERT model and mBERT + Hierarchical model. For all the three language pairs, we observe a performance gain in terms of weighted average F_1 scores. Our model witness an increase of 3.33%, 11.6% and 6% on Tamil-English, Kannada-English and Malayalam-English respectively.

We observe a significant number of misclassifications after the first-level of language identifications: <language> and not-<language>. A substantial number of sentences classified into not-<language> classes are wrongly shown belonging to other classes in the gold standard data (for details, see below). Our method identifies 430 sentences as not-Tamil for the Tamil-English test dataset, whereas the gold data only indicates 244 instances.

This causes a decrease in the support count across all classes. In Table 5.12, we specifically observe that 23 sentences are classified as Mixed_feelings, which should be classified as not-Tamil sentences. The same thing is observed in other classes also, like 30 from the negative class, 259 from the positive class, and 75 from the unknown_state class.

Despite such mis-classifications, our model's weighted average F_1 score improves compared to the mBERT model. This improvement indicates that our model's performance is becoming more accurate and reliable when applied to the test data.

Similar patterns are observed in Table 5.13 for the Kannada-English dataset, where our propose model consistently exhibited superior performance compare to mBERT.

For the Malayalam-English dataset, as depicted in Table 5.14, the performance trends are similar.

Table 5.12: Comparison of Precision, recall, F_1-scores, and support with our proposed model for RQ-3 on Tamil-English test data

	mBERT				**mBERT + Hierarchical Model**			
	Precision	Recall	F_1-score	support	Precision	Recall	F_1-score	support
Mixed_feelings	0.26	0.26	0.26	470	0.36	0.20	0.25	447
Negative	0.40	0.34	0.36	477	0.39	0.36	0.37	447
Positive	0.75	0.79	0.77	2564	0.74	0.85	0.80	2305
unknown_state	0.40	0.38	0.39	665	0.41	0.35	0.38	590
macro avg	0.45	0.44	0.44	4176	0.48	0.45	**0.47**	3789
weighted avg	0.59	0.61	0.60	4176	0.60	0.64	**0.62**	3789

Table 5.13: Comparison of Precision, recall, F_1-scores, and support with our proposed model for RQ-3 on Kannada-English test data

	mBERT				**mBERT + Hierarchical Model**			
	Precision	Recall	F_1-score	support	Precision	Recall	F_1-score	support
Mixed_feelings	0.23	0.28	0.25	65	0.39	0.12	0.19	56
Negative	0.66	0.54	0.59	157	0.70	0.54	0.61	154
Positive	0.70	0.74	0.72	374	0.77	0.76	0.76	312
unknown_state	0.31	0.29	0.30	62	0.37	0.38	0.37	37
macro avg	0.46	0.44	0.46	658	0.54	0.48	**0.52**	559
weighted avg	0.60	0.61	0.60	658	0.66	0.68	**0.67**	559

Table 5.14: Comparison of Precision, recall, F_1-scores, and support with our proposed model for RQ-3 on Malayalam-English test data

	mBERT				**mBERT + Hierarchical Model**			
	Precision	Recall	F_1-score	support	Precision	Recall	F_1-score	support
Mixed_feelings	0.40	0.16	0.23	134	0.38	0.23	0.29	129
Negative	0.55	0.53	0.54	258	0.56	0.50	0.53	252
Positive	0.74	0.81	0.77	780	0.74	0.82	0.78	745
unknown_state	0.71	0.74	0.72	643	0.71	0.71	0.71	613
macro avg	0.60	0.56	0.56	1815	0.60	0.57	**0.58**	1739
weighted avg	0.67	0.66	0.66	1815	0.69	0.70	**0.70**	1739

On closer analysis, we discover that our LID-module divides the sentences into <language> and not-<language> correctly for a majority of the cases. The deviation from the actual tags is primarily due to the error in the annotation. However, there are some errors as well made by the LID module as discussed below.

5.4.4 Error Analysis

We categorize the errors into two types: errors in annotation of gold standard and errors made by our model.

Table 5.15 presents few instances where our model made correct predictions but annotations in the gold standard are wrong. Column Gold shows the annotations of gold standard while column Predicted shows results of our model. As we can see that the labels predicted by our model align more accurately with the reality rather than in the gold standard. From Example 1 to 5, it is clear that all of the sentences do not contain any Tamil words, so this should come under the not-Tamil class. For Example 6, despite the presence of many Tamil words according to the LID module, the gold standard labels it as not-Tamil, contradicting the reality.

Table 5.15: Errors in gold standard

No.	Sample text from dataset	Gold	Predicted
1	Govt should take action	neutral	not-Tamil
2	Police pls take action	neutral	not-Tamil
3	Government take this type of peoples civiliar actions	neutral	not-Tamil
4	JJ mam we miss u	Positive	not-Tamil
5	Kangana Ranaut ke fans like here	unknown_state	not-Tamil
6	Funny to hear tamil language for urdu/punjabi speaking people. at puta nagada duta mala keda under duka etc. lol	not-Tamil	Tamil
7	Super song bro	Positive	not-Kannada
8	@Ranganath S yes	unknown state	not-Kannada
9	and rocking star fans also	Positive	not-Kannada
10	YouTube logic 66k likes + 6.3k dislikes = 16k views..	Negative	not-Malayalam

LID module plays a significant role in cases where our model failed to predict the correct classes. Table 5.16 presents instances where our model made incorrect predictions. For instance, in Example 1, the LID module labels “shetty” as a Kannada word possibly because the word is an out-of-vocabulary (OOV) word, leading our model to predict it as Kannada. However, this is a part of the name of a person or a named entity (NE), and other words are English. Example 2 is actually a Tamil expression, but the LID module identifies it as Kannada. In Example 3, “mammookka” was labeled as a

Malayalam word by the LID module, while it is a name (NE). In Example 5, the word "alle," is a Malayalam word, but labeled as Italian by the LID module.

In general, we observe that LID module struggled to accurately identify the proper language tag for named entities (NEs). Consequently, the performance of the word-level language identification module was inadequate, as it failed to detect the appropriate tags. In the Tamil-English dataset alone, we encountered a total of 28 language tags, prompting us to develop a rule-based model where we designated these tags as unknown tags.

These findings emphasize the limitations of the LID module and the challenges it poses in accurately determining the language for certain words and sentences.

Table 5.16: Errors made by the LID Model

No.	Sample text from dataset	Gold	Predicted
1	Lv u rrr shetty super	not-Kannada	Kannada
2	Pora rai koodi tulla	not-Kannada	Kannada
3	One an only megastar mammookka	not-Malayalam	Malayalam
4	aa musicinu vendi wait cheyyuva	not-Malayalam	Malayalam
5	300 crores next. Ok alle	unknown_state	not-Malayalam
6	MEGA STAR FANS undo like aadiki	unknown_state	not-Malayalam
7	4M viewsmammookka we are waiting	not-Malayalam	Malayalam

5.4.5 Discussion

The findings from our research provide valuable insights into sentiment analysis in code-mixed languages and offer practical implications for handling such datasets effectively. The three research questions are interconnected and contribute collectively to a comprehensive understanding of SA in code-mixed datasets. RQ-1 underscores the significance of dataset composition, with the combination of code-mixed and monolingual data influencing model performance. In RQ-1, we apply a state-of-the-art model and report the best scores that are used later as the baselines to compare with the other RQs. RQ-2 highlights an important role of LID in code-mixed data processing, to further enhance the ability of the sentiment analysis model to handle multiple languages

effectively. Tables 5.9, 5.10 and 5.11 display the results of the test data for the mBERT model (RQ-1) and mBERT + Rule-based model (RQ-2). On comparison between RQ-1 and RQ-2, we observe a noticeable performance gain in RQ-2 in terms of weighted average F_1 scores. RQ-3 explores a hierarchical approach that incorporates both LID and SA, showcasing improved performance and the complementary nature of these tasks in code-mixed datasets. Tables 5.12, 5.13, and 5.14 display the result of the test data for the mBERT model (RQ-1) and mBERT + Hierarchical model (RQ-3). For all three language pairs, we observe a performance gain in RQ-3 in terms of weighted average F_1 scores. Between RQ-2 and RQ-3, the main difference is whether LID should be used in parallel to classfier (RQ-2) or should be used first followed by the classifier (RQ-3) to augment the classification task. For the given datasets, experiments for RQ-2 provide comparable performances to RQ-3. However, real comparison between RQ-2 and RQ-3 is only possible when the datasets are free of annotation errors and performance of LID module is reasonably high. At the moment, with the amount of annotation errors, it is premature to comment on the comparative performances of experiments on RQ-2 and RQ-3.

The organizers [126] provided language-wise rank lists for sentiment analysis (Task A). Table 5.17 shows the rank list for Task A in the Tamil-English track. Similarly, Table 5.18 for Malayalam-English and Table 5.19 for Kannada-English. Our submission, IRLAB, outperformed others in both the Malayalam and Kannada tracks. However, due to an error in the submission file for the Tamil track, we missed appearing on the rank list. Nonetheless, based on the scores, we did better than other official submissions in the Tamil-English track ($F_1 score$ 0.44).

5.5 Summary of this Chapter

The exponential growth of data generated by social media users in the era of Web 2.0 has necessitated advanced techniques for sentiment analysis. While numerous stud-

Table 5.17: Rank list for Task A: Tamil track published by organiser

Team Name	**Precision**	**Recall**	F_1**-score**	**Rank**
SRMNLP	0.340	0.330	0.270	1
BharathNLP	0.190	0.220	0.190	2
bilstm	0.220	0.190	0.190	3
SSN-CSE	0.220	0.260	0.170	4
Sentiment	0.240	0.220	0.170	5
MUCS	0.240	0.190	0.160	6
Fnet	0.150	0.130	0.130	7
JPMCAI	0.020	0.160	0.020	8

Table 5.18: Rank list for Task A: Malayalam track published by organiser

Team Name	**Precision**	**Recall**	F_1**-score**	**Rank**
IRLAB	0.670	0.670	0.660	1
Fnet	0.660	0.620	0.640	2
Sentiment	0.620	0.630	0.630	3
MUCS	0.610	0.610	0.610	4
NITK	0.600	0.600	0.600	5
SRMNLP	0.610	0.550	0.570	6
lone_warrior	0.520	0.590	0.520	7
bilstm	0.490	0.580	0.500	8
BharathNLP	0.160	0.270	0.200	9
JPMCAI	0.340	0.200	0.140	10
SSN-CSE	0.090	0.140	0.110	11

ies have focused on sentiment analysis in monolingual datasets, there is a scarcity of research in code-mixed datasets. In this chapter, we present a methodology for identifying sentiment polarities in code-mixed languages using a dataset of Tamil-English, Kannada-English and Malayalam-English YouTube social media comments. Our approach goes beyond traditional models by incorporating novel characteristics, such as the language tag module, which proves to yield improved results in the majority of cases.

Specific to our dataset, we draw the following conclusions for the RQs.

RQ-1: If a given dataset contains a mixture of pure monolingual, transliterated monolingual, single-script multilingual and mixed script multilingual text, what is the

Table 5.19: Rank list for Task A: Kannada track published by organiser

Team Name	**Precision**	**Recall**	F_1**-score**	**Rank**
IRLAB	0.560	0.560	0.550	1
Sentiment	0.520	0.500	0.510	2
lone_warrior	0.470	0.510	0.480	3
NITK	0.480	0.500	0.480	4
Fnet	0.500	0.490	0.480	5
AI Defenders	0.490	0.480	0.480	6
SRMNLP	0.540	0.440	0.460	7
JPMCAI	0.550	0.430	0.450	8
MUCS	0.470	0.460	0.440	9
bilstm	0.480	0.500	0.430	10
QWERTY	0.460	0.350	0.350	11
BharataNLP	0.290	0.330	0.300	12
SSN-CSE	0.120	0.170	0.110	13

best strategy for conducting sentiment analysis?

The dataset comprises a combination of code-mixed and monolingual sentences. We find that if a dataset contains a mix of code-mixed and monolingual data, it is preferable to train the model using the same type of data. Training the model solely on code-mixed data resulted in a performance drop. However, when the text data consists of exclusively of code-mixed content, training the model using code-mixed data leads to better results. Therefore, the combination of code-mixed and monolingual data is a crucial factor to consider.

RQ-2: How important is the language identification (LID) task for code-mixed data processing? Can a separate language identification model with a pre-trained model improve the overall system performance? We seek to find the answer particularly, with respect to sentiment analysis task.

Language identification is a very important step. Across all three language pairs, our models demonstrate improved performance in terms of weighted average F_1 scores. Specifically, we observe performance increases of 6.6%, 10%, and 4.3% for Tamil-English, Kannada-English, and Malayalam-English, respectively, after incorporating

a separate LID module that augments a simple, single-level training-based approach.

RQ-3: Is it possible to see SA as a multi-level problem rather than a multi-class problem? If yes, can a multi-level hierarchical model enhance the performance?

We explored a hierarchical model with LID and mBERT module. Across all three language pairs, our model showcased enhanced performance in terms of weighted average F_1 scores. Notably, we achieved performance increases of 3.33%, 11.6%, and 6% for Tamil-English, Kannada-English, and Malayalam-English, respectively.

Language identification and sentiment analysis are two independent tasks. But in case of code-mixed data, these two are intertwined. However, to improve the performance of SA, LID is an important step - that should be taken separately. Our experiments could not show clearly whether LID should be done in parallel to augment the classification task (RQ-2) or it should be done first followed by SA on in-language data (RQ-3) because of annotation errors in the gold standard data. We hypothesize that the second should yield better performance if the performance of LID is reasonably good and the gold standard data is error-free. However, this needs to be experimentally validated and it requires an error-free data.

Wrong predictions made by our model can be attributed to our reliance on "Google-trans," a word-level language identification module, which is not entirely accurate and thus affects our evaluation scores. We acknowledge the need for further research and development of our own word-level identification model to improve the performance scores in the future. Furthermore, it is important to acknowledge that the system's performance can be significantly influenced by the extent of code-mixing. The observed substantial disparity between CMI-Mixed and CMI-All values indicates the presence of numerous monolingual sentences, with varying degrees of code-mixing across different sentences. This aspect opens up a promising area of future research, wherein investigations could be conducted on the impact of distinct levels of code-mixing in sentences on downstream tasks. Such analyses would contribute valuable insights to the field.