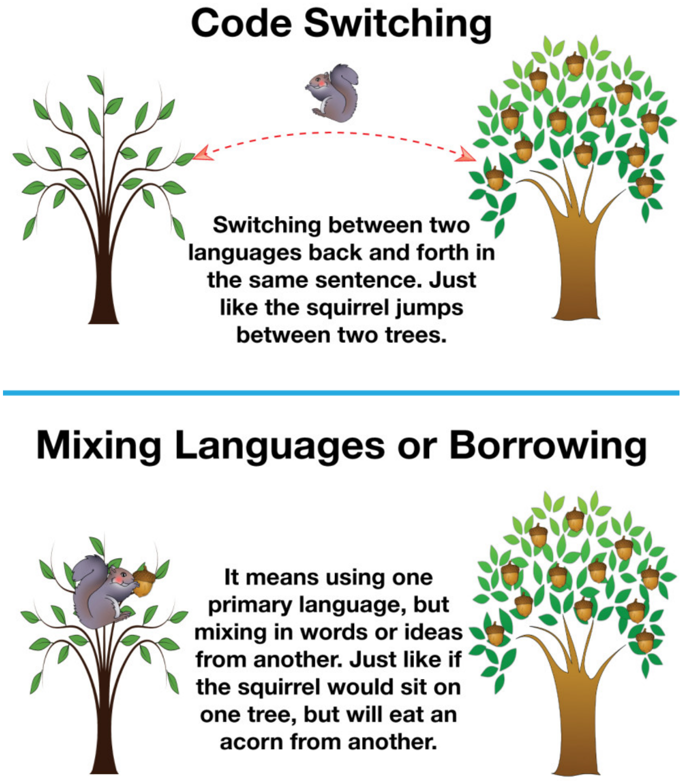
DISSERTATION REPORT

Submitted in partial fulfilment of the degree Master of Technology in Computer Science

SENTIMENT ANALYSIS ON HINDI-ENGLISH CODE-MIX DATA



Code Switching

Switching between two languages back and forth in the same sentence. Just like the squirrel jumps between two trees.

Mixing Languages or Borrowing

It means using one primary language, but mixing in words or ideas from another. Just like if the squirrel would sit on one tree, but will eat an acorn from another.

Prepared By:

Balwant Singh Bisht

MTech Computer Science

Roll No. CS2121 (2021-23)

Supervisor:

Dr.Ujjwal Bhattacharya

Associate Prof. (ISI Kolkata)

CVPR Unit

Indian Statistical Institute

Declaration

I, Balwant Singh Bisht, hereby affirm that I have completed all the necessary requirements as specified by the Indian Statistical Institute for the submission of my M.Tech dissertation in Computer Science.

I attest that the project presented is an original piece of work, reflecting the culmination of my independent investigations and research. It is devoid of any form of plagiarism. I affirm that this work has not been previously submitted to any other educational institution or university for the purpose of obtaining a degree, diploma, or any other form of academic recognition.

Furthermore, I declare that all text, diagrams, and materials acquired from external sources, including but not limited to books, journals, and the internet, have been duly acknowledged, referenced, and cited in accordance with the best of my knowledge and understanding.

Date: 21/06/2023

Balwant

Balwant Singh Bisht
Roll No. CS2121
MTech Computer Science (2021-23)
Indian Statistical Institute

Certificate by the Supervisor

This is to certify that the work presented in this dissertation titled “Sentiment Analysis on HIndi-English code-mix data”, submitted by Balwant Singh BIsht, having the roll number CS2121, has been carried out under my supervision in partial fulfilment for the award of the degree of Master of Technology in Computer Science during the session 2022-23 in the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute.

Dr. Ujjwal Bhattacharya,
MSc MPhil PGDCA PhD,
Associate Professor, Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute, Kolkata

Acknowledgements

Writing a report for my research work was once in a lifetime experience. I have read many great books, good research papers and extremely well written articles during my studies and I always admired the sheer brilliance of the authors who were involved in the production of such well documented pieces of knowledge. While admiring their work, I realised that every well-finished thing is not possible single-handedly, the support of many people is behind it.

Of all the people who made this work possible, Dr. Ujjwal Bhattacharya is the first and foremost. His supervision allowed me the freedom to explore the field that I was interested in on my own. His guidance regarding the methodology to pursue while reading research papers, finding different problems and the way to approach them and presenting the work were instrumental. The learnings that I have gathered from him during this dissertation work would remain with me for life. I would like to thank him for his constant support.

I have benefitted from the efforts of my peers during my journey through this Masters program. Our brainstorming sessions together immensely helped me throughout this research work. I would like to thank all my batchmates at Indian Statistical Institute.

Date: 21/06/2023

Balwant

Balwant Singh Bisht
Roll No. CS2121
MTech Computer Science (2021-23)
Indian Statistical Institute

Contents

List of Figures

List of Tables

Abstract

Social media has emerged as a prominent platform for expressing opinions, leading to the development of a unique language known as code-mix text. This form of language incorporates words from multiple languages, such as Hindi and English in India. While sentiment analysis techniques have achieved moderate success in handling English texts, the same level of effectiveness has not been attained when dealing with code-mix text.

In this study, we propose deep learning techniques to address the challenges of sentiment analysis in code-mix Hindi-English text data. Leveraging a pre-trained cross-lingual large language model called XLM-RoBERTa, we employ a transfer learning approach. Four distinct approaches are employed to train the model for sentiment analysis on a Hinglish dataset. The first approach involves training the model using the Hinglish dataset exclusively. The other three approaches utilise mixed datasets, where one includes the augmentation of Spanish-English and Marathi-English datasets with the Hinglish dataset, the second approach solely relies on the mixed dataset without Hinglish data, while the final approach exclude the Spanish-English data. The trained models are evaluated on the same Hinglish dataset, and their performance is compared.

The results indicate that the approach of increasing the training data by arbitrarily combining different kinds of mixed datasets does not yield improvements over previous findings. But combining the data of languages with similar linguistic characteristics can result in better performance. This highlights that the problem associated with scarcity of data for code-mixed languages can be effectively solved by using data of similar languages.

In conclusion, our study emphasises the ongoing challenge of limited data for code-mixed languages. We demonstrate that augmenting the training data with various mixed datasets does not lead to enhanced performance but the data of similar languages can be combined to produce better outcomes. These findings provide valuable insights for future research in sentiment analysis of code-mix text.

Chapter 1

Introduction

In recent years, social media has become an integral part of our daily lives, consuming a significant portion of our time. Numerous studies have indicated that individuals spend anywhere from two to more than ten hours each day navigating various social media platforms. These platforms serve as a means for users to express their thoughts and opinions through posts and comments.

The widespread adoption of social media has also given rise to a unique linguistic phenomenon known as code-mixing. This involves the combination of words from multiple languages, where individuals blend their native tongue with English or other languages while writing their comments or posts.

Within this context, sentiment analysis plays a crucial role in understanding the emotional states of users through their text. This work aims to apply sentiment analysis techniques specifically to Hindi-English code-mix data, with the objective of extracting valuable insights regarding user emotions. By analysing the sentiment expressed in code-mix text, we can gain valuable knowledge about user emotions in this multilingual context. This research focuses on developing effective approaches for sentiment analysis, leveraging the unique characteristics of Hindi-English code-mix data.

The task of extracting knowledge from code-mixed text can be challenging due to various factors such as spelling variations, informal language, and lack of adherence to grammar rules. These factors contribute to data sparsity and make it difficult to compress all words into a single dictionary.
To tackle the issue of spelling variations, it's important to employ techniques like stemming and lemmatization, which can reduce words to their base form and handle variations based on sounds or language-specific rules. This can help in consolidating similar words and improving data coverage.
Informal language poses another challenge, as users tend to use abbreviations, acronyms, or non-standard spellings. To address this, you can build a comprehensive dictionary or lexicon that includes commonly used informal variations along with their formal counterparts. This dictionary can be used to map informal words to their standard equivalents, enabling better understanding and analysis of the text.

Data sparsity can be mitigated by leveraging language resources and tools specific to each language present in the code-mixed text. By incorporating language-specific resources, such as language models or datasets, you can improve the coverage and accuracy of your analysis.
Lastly, dealing with grammar inconsistencies requires robust natural language processing techniques that can handle code-mixed text. It may involve using language-specific grammatical rules, part-of-speech tagging, or dependency parsing to identify and correct errors or irregularities in the text.

Overall, effectively extracting knowledge from code-mixed text involves a combination of linguistic resources, data preprocessing techniques, and language-specific analysis methods to address the challenges posed by spelling variations, informality, data sparsity, and grammar inconsistencies. The subsequent sections of this report are structured as follows. Section 2 provides an overview of the related research conducted in the field. Section 3 outlines the terminologies used in the study. Section 4 presents the methodology used in the study. Section 5 presents the experimental results obtained from the study. Finally, in Section 6, we provide our conclusion based on the findings and discuss potential avenues for future research.

Chapter 2

Related Work

In recent times, the focus of sentiment analysis research has primarily been on single-language text, with English being the predominant language due to the abundance of social media data available in this language. However, with the development of multilingual societies, researchers from diverse linguistic backgrounds recognized the need to address sentiment analysis in code-mixed text.

Initial efforts in this field primarily relied on machine translation techniques to convert the native language present in the text into English before applying machine learning methods. These endeavours achieved moderate success in various languages such as Chinese, French, Spanish, and Italian. Nevertheless, the scarcity of code-mixed data remains a significant challenge. Most research in multilingual sentiment analysis (MSA) involves creating datasets by merging monolingual datasets, training machine learning models on these datasets. However, this approach has yielded mixed results when applied to code-mixed text, mainly due to issues with poor translation and loss of meaning during the conversion process.

The emergence of deep learning techniques has propelled research in the field of code-mixed sentiment analysis, although the accuracy of the results still lags behind those obtained for single-language text. CNN, RNN, LSTM, Bi-LSTM, and other similar architectures have been explored to address this task. The introduction of pre-trained large language models, such as BERT and its advanced variants like XLM-RoBERTa, has further advanced research in MSA.

Researchers have made notable contributions in this domain. Gupta et al. [1] proposed an unsupervised self-training approach using pre-trained BERT models specifically for code-switched data. Their method outperformed supervised models in sentiment analysis for four code-mixed languages: Hinglish (Hindi-English), Spanglish (Spanish-English), Tanglish (Tamil-English), and Malayalam-English. Similarly, Ou and Li [2] leveraged XLM-RoBERTa, a pre-trained multi-language model, to determine sentiment polarity in code-mixed datasets from the Dravidian language group. Their system employed a k-folding approach for ensemble learning and achieved promising results in sentiment analysis for Malayalam-English and Tamil-English code-mixed datasets. Braaksma et al.[3] adopted a two-step fine-tuning method, utilising English BERT-base and Spanish BERT-base models, to improve sentiment classification performance in Spanish-English (Spanglish) datasets. They reported that the large multilingual XLM-RoBERTa model attained the best-weighted F1-score on the development and test data.

Other studies have focused on specific languages and datasets. Kumar and Albuquerque[4] applied cross-lingual XLM-RoBERTa to sentiment analysis in resource-poor Hindi datasets. Their model, trained and fine-tuned using English-language benchmark datasets, achieved favourable performance compared to state-of-the-art approaches. Furthermore, Chakravarthi et al.[5] curated a code-mixed dataset of under-resourced Dravidian languages for sentiment analysis and offensive language identification. They employed machine learning and deep learning techniques, with the XLM technique attaining the highest accuracy of 71%. Thara et al.[6] explored offensive language identification and sentiment analysis for Malayalam-English code-mixed data. Their framework incorporated various word embedding methods and deep learning models, with the best-performing hybrid models achieving a high F1-score of 0.9969. In the following section, we will introduce some terminologies commonly used in natural language processing (NLP) and specifically related to our problem.

Chapter 3

Terminologies

Metrics for code-mix data

Researchers have developed various metrics to measure the complexity and degree of code-mixing in text. Among these metrics, the Code Mixing Index (CMI) has gained popularity. The CMI is defined as follows:

$$\mathrm{CMI} = \begin{cases} 100 * \left[1 - \frac{\max(w_l)}{n-u}\right] & n > u \\ 0 & n = u \end{cases}$$

In the provided metric, wl represents a word belonging to language l, n represents the total number of tokens, and u represents the count of independent tokens. Independent tokens refer to those that contain entity names or hashtags, which remain unchanged regardless of the language used.

A lower value of CMI suggests that the text is primarily written in a single language, whereas a higher value indicates a significant amount of code-mixing in the text.

Tokenization

Tokenization in natural language processing (NLP) refers to the process of breaking down a text or a sequence of words into smaller units called tokens. These tokens can be individual words, subwords, or even characters, depending on the chosen tokenization technique.

The purpose of tokenization is to create a standardised and structured representation of textual data that can be easily processed by NLP algorithms. By dividing the text into tokens, we can analyse and manipulate the data at a more granular level, enabling tasks such as part-of-speech tagging, named entity recognition, and sentiment analysis.

Tokenization techniques vary depending on the specific requirements of the NLP task and the language being processed. Common tokenization methods include whitespace-based tokenization, where tokens are separated by spaces or punctuation marks, and morphological tokenization, which breaks down words into morphemes (meaningful word parts).

Tokenization is a crucial initial step in many NLP pipelines as it forms the foundation for subsequent text processing and analysis. By breaking down text into tokens, NLP models can better understand the linguistic structure and extract meaningful information from the data.

Here's an example of tokenization for the sentence "I love to eat pizza":
Tokens: [I, love, to, eat, pizza]
In this example, the sentence has been tokenized into individual words, resulting in five tokens.

XLM-RoBERTa

XLM-R, which stands for XLM-Roberta, is a transformer-based multilingual masked language model developed by the Facebook AI team. It was released in November 2019 as an enhancement to the original XLM-100 model. The primary objective of XLM-R is to provide an effective solution for non-English natural language processing (NLP) tasks.

One significant improvement in XLM-Roberta compared to its predecessor is the substantial increase in the amount of training data used. It was trained on a vast corpus of 2.5 terabytes of data, covering 100 languages. This extensive training data, filtered from CommonCrawl texts, contributes to the model's enhanced performance.

XLM-R has demonstrated state-of-the-art results on various cross-lingual benchmarks. By leveraging a masked language model approach, it has proven to be a successful alternative for NLP tasks involving languages other than English. What sets XLM-R apart is its compatibility with both monolingual and cross-lingual benchmarks, addressing the challenges associated with multilingualism in NLP.

Overall, XLM-Roberta has emerged as a powerful and versatile model in the field of multilingual NLP, showcasing its effectiveness in a wide range of tasks and languages.

RoBERTa is a transformer-based model that undergoes self-supervised pre-training on a large corpus of raw texts. The pre-training process does not involve any human labelling, making use of publicly available data. The model employs a masked language modelling (MLM) objective, where it randomly masks 15% of the words in a sentence and predicts those masked words by running the entire masked sentence through the model.
Unlike traditional recurrent neural networks (RNNs) or autoregressive models like GPT, RoBERTa considers the entire masked sentence at once rather than processing words sequentially. This allows the model to learn a bidirectional representation of the sentence, capturing contextual information effectively.

The self-supervised pre-training enables RoBERTa to learn a comprehensive internal representation across 100 languages. These learned features can then be utilised for downstream tasks. For instance, if there is a labelled sentence dataset available, one can train a standard classifier using the features extracted from the XLM-RoBERTa model as input.

The architecture of XLM-RoBERTa, as described in Conneau et al.[9], is depicted in Figure 1. It demonstrates the model's design and components, highlighting its ability to capture language features and facilitate transfer learning for various NLP tasks.

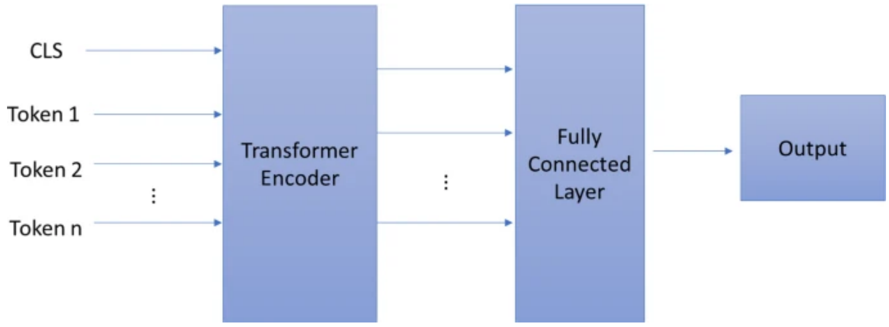


Figure 1. Basic Architecture of XLM-RoBERTa

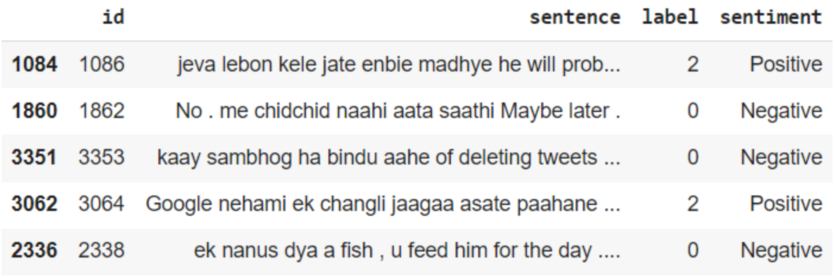
Chapter 4

Methodology

Datasets

Marathi-English Dataset

We downloaded the dataset from the github repository “https://github.com/arundprabhu/”.
The dataset consists of four columns which were not named. They signify the serial number, sentence, label, label. Yes, the last two columns contained the same number which signifies the sentiments of the text. 0, 1, 2 were associated with negative, neutral and positive sentences respectively.

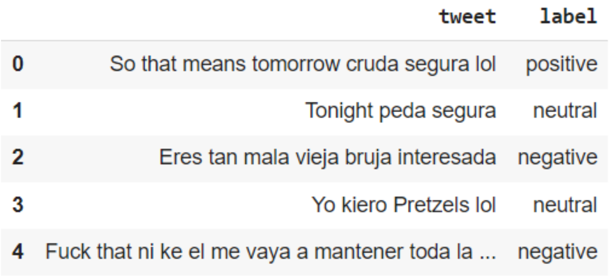


	id	sentence	label	sentiment
1084	1086	jeva lebon kele jate enbie madhye he will prob...	2	Positive
1860	1862	No . me chidchid naahi aata saathi Maybe later .	0	Negative
3351	3353	kaay sambhog ha bindu aahe of deleting tweets ...	0	Negative
3062	3064	Google nehami ek changli jaagaa asate paahane ...	2	Positive
2336	2338	ek nanus dya a fish , u feed him for the day	0	Negative

Figure 2. A subset of Marathi-English Dataset

Spanish-English Dataset

We downloaded the dataset from the github repository “https://github.com/tejasvicsr1/”. The dataset can be classified in two columns which could be clearly seen as the tweet by the user and the sentiment as positive, negative or neutral.

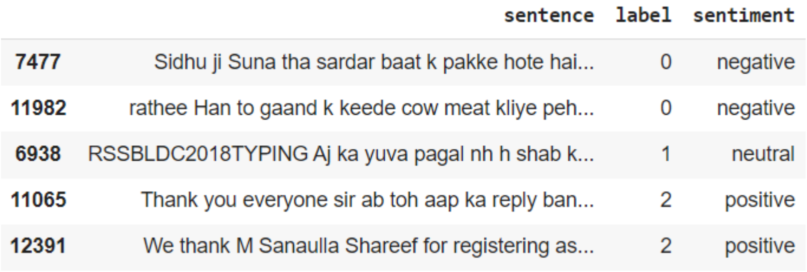


	tweet	label
0	So that means tomorrow cruda segura lol	positive
1	Tonight peda segura	neutral
2	Eres tan mala vieja bruja interesada	negative
3	Yo kiero Pretzels lol	neutral
4	Fuck that ni ke el me vaya a mantener toda la ...	negative

Figure 3. A subset of Spanish-English Dataset

Hindi-English Dataset

We downloaded the dataset from the github repository “https://github.com/rsgoss/”.
The dataset contains 14000 training samples and 3000 validation samples. It has four columns namely id, sentence, label, sentiment. Negative, neutral and positive sentiments have been labelled as 0, 1, 2 respectively. “Id” is the twitter ids of the users.



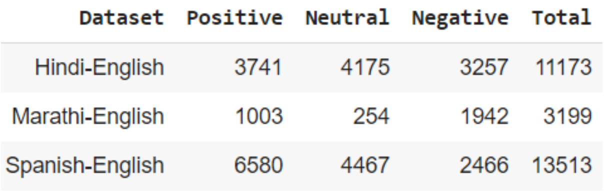
	sentence	label	sentiment
7477	Sidhu ji Suna tha sardar baat k pakke hote hai...	0	negative
11982	rathee Han to gaand k keede cow meat kliye peh...	0	negative
6938	RSSBLDC2018TYPING Aj ka yuva pagal nh h shab k...	1	neutral
11065	Thank you everyone sir ab toh aap ka reply ban...	2	positive
12391	We thank M Sanaulla Shareef for registering as...	2	positive

Figure 4. A subset of Hindi-English Dataset

Data Preprocessing

This step was extremely crucial for our analysis. All our datasets were labelled differently. Since we are considering the problem of sentiment analysis on Hinglish dataset and our analysis requires us to process all datasets in a similar way, we processed every other dataset in the form of Hinglish dataset.
Firstly, we did not require the “id” column in any dataset so it was removed. A “label” column was added to the Spanish-English dataset and the other two columns were named as “sentence” and “sentiment” respectively. One of the last two columns which contained integers 0, 1, 2 as labels in the Marathi-English column was replaced with the respective sentiment column. Finally all datasets were checked for any null or unwanted values, if present were removed. Marathi mix and Spanish mix datasets were splitted in the ratio 80/20 to form train and test set respectively. The Marathi mixed dataset finally had 3199/800 train/test split and the Spanish-English dataset had 2912/728 train/test split.

Table 1. shows the number of the samples of each class in the training datasets.



Dataset	Positive	Neutral	Negative	Total
Hindi-English	3741	4175	3257	11173
Marathi-English	1003	254	1942	3199
Spanish-English	6580	4467	2466	13513

Table 1. Counts of different classes of sentence

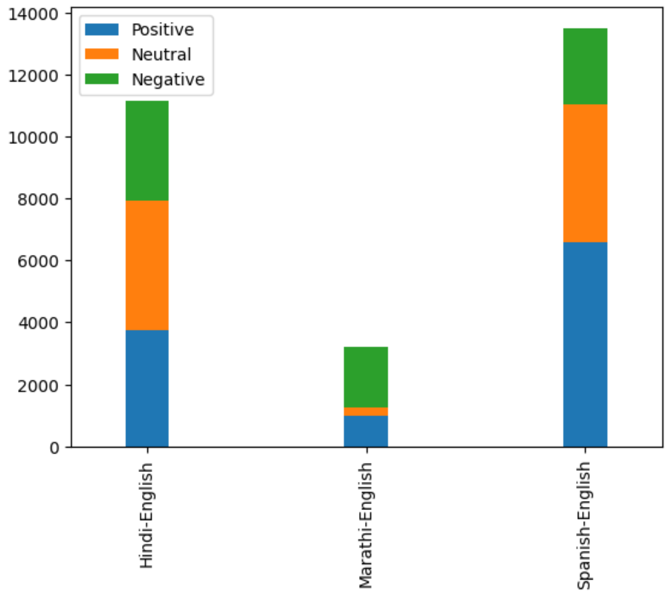
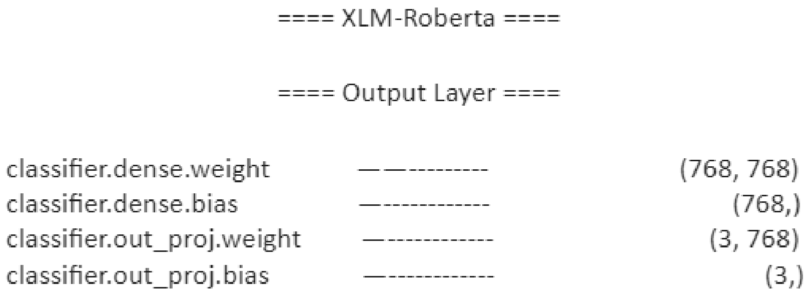


Figure 5. Class Wise distribution of different datasets

Approach

Our architecture incorporates a dense classification network with a single hidden layer.



```
==== XLM-Roberta ====

==== Output Layer ====

classifier.dense.weight          ——---------          (768, 768)
classifier.dense.bias            —------------            (768,)
classifier.out_proj.weight       —------------          (3, 768)
classifier.out_proj.bias         —------------              (3,)
```

We conducted training using three distinct datasets, but the evaluation was performed on a consistent Hindi-English dataset. Let's delve into the specifics of each approach:

1. Approach 1: Training on Hindi-English Dataset. In this approach, the model was trained using the Hindi-English dataset. The objective was to learn patterns and relationships specific to this code-mixed language combination. Subsequently, the trained model was tested on the same type of dataset, assessing its performance in accurately classifying instances within the Hindi-English code-mixed context.

2. Approach 2: Training on mixed Dataset. In the second approach, the model was trained on the combined Hindi- English, Spanish-English and Marathi-English dataset. The intention behind this was to explore the model's ability to generalise and adapt to different code-mixed language pairs. Despite being trained on a different language combination, the model was evaluated on the Hindi-English dataset to assess its cross-lingual performance.

3. Approach 3: Training on mixed Dataset consisting of data samples different from the one used for testing the model. In the third approach, the model was trained on the combined Spanish-English and Marathi-English dataset. This approach aimed to understand how well the model could handle cross lingual training and code-mixed testing scenarios. By training on a completely different kind of code-mix dataset than the one used for testing, the model's performance was evaluated, providing insights into its ability to generalise code-mixing challenges.

4. Approach 4: Training on mixed Dataset consisting of data samples only from Indian languages. In the final approach, the model was trained on the combined Hindi-English and Marathi-English dataset. Both Hindi and Marathi have Indo-Aryan origins which was the motivation to understand how well the model could handle cross lingual training of two languages with similar origins, and code-mixed testing scenarios on one of them.

In all four approaches, the trained models were tested on the same Hindi-English dataset, allowing for a consistent and comparative evaluation across different training scenarios. This setup facilitates an understanding of how the model's training data influences its performance when applied to code-mixed language scenarios.

Evaluation Metrics

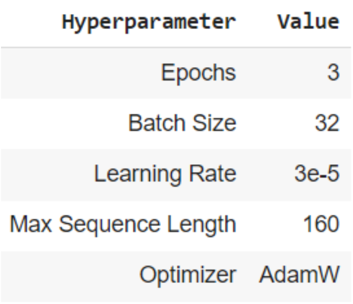
When evaluating a machine learning model, it is essential to assess its performance using appropriate metrics. Different metrics are defined for different types of models, and for our task, we used the following metric:

Accuracy: Accuracy represents the overall performance of a model across all classes. It is calculated as the ratio of the number of correct predictions to the total number of predictions. Accuracy is useful when all classes have equal importance.

Accuracy = (Number of true predictions) / (Total number of predictions)

Experimental Setup

In our experiments conducted in the Google Colab environment, we utilised several libraries and techniques for data preprocessing and model training. Here is a breakdown of the steps and components involved:



Hyperparameter	Value
Epochs	3
Batch Size	32
Learning Rate	3e-5
Max Sequence Length	160
Optimizer	AdamW

Table 2. Hyperparameters

1. Tokenization with XLM-Roberta: The XLM-Roberta tokenizer from the transformers library was imported to tokenize the textual data. We tokenized each sentence, mapping the tokens to their respective word IDs. Each sentence was transformed into a 160-length vector to standardise the input size.
2. TensorDataset and DataLoader: To optimise memory space, the tokenized data was combined into a TensorDataset, which efficiently stores the training inputs. Additionally, the DataLoader was employed to efficiently load the tokenized data in batches during training and validation. A batch size of 32 was used to process the training samples.
3. Importing XLM-Roberta for Sequence Classification: We imported the XLMRoberta model from the transformers library. The model consists of 201 different named parameters, encompassing various layers and components.
4. Embedding Layer: The embedding layer of the XLMRoberta model includes parameters such as word embeddings, position embeddings, token type embeddings, LayerNorm weights, and biases.
5. Transformer Layers: The XLMRoberta model consists of multiple transformer layers. Each layer has different sets of parameters, including attention weights, intermediate dense weights, output dense weights, LayerNorm weights, and biases. The example provided includes the details of the first transformer layer, but the model comprises several such layers.
6. Output Layer: The output layer of the model, named "classifier," comprises parameters for the dense layer weights and biases, as well as the final output projection weights and biases for sequence classification. In the given example, there are three output classes represented by the shape (3, 768).
7. Optimization with AdamW: The AdamW optimizer, where W stands for "Weight Decay fix," was utilised for optimization during training. It is an extension of the Adam optimizer that incorporates weight decay regularisation.

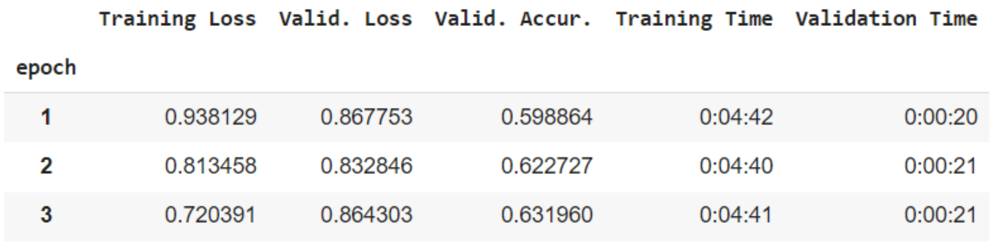
Chapter 5

Results

The performance of the XLM-RoBERTa model on various code-mixed datasets has been summarised in three separate tables, namely Table 3, Table 4, and Table 5. These tables provide insights into the training performance metrics of the model. Figure 6, Figure 7 and Figure 8 depict the training loss vs validation loss curve for Approach 1, Approach 2 and Approach 3 respectively.

Approach 1: Table shows the training and validation performance of a model across three epochs. Here is a summary of the results:

- Training Loss: The model's training loss decreases from 0.938 to 0.720 over the three epochs, indicating that the model is gradually improving its fit to the training data.
- Validation Loss: The validation loss fluctuates between 0.867 to 0.864 across the epochs. Since the fluctuation is not significant, it suggests that the model performance is acceptable based on other results.
- Validation Accuracy: The validation accuracy increases from 0.599 to 0.632, which indicates that the model is making progress in correctly predicting instances in the validation set. However, the improvement is relatively modest.



	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
epoch					
1	0.938129	0.867753	0.598864	0:04:42	0:00:20
2	0.813458	0.832846	0.622727	0:04:40	0:00:21
3	0.720391	0.864303	0.631960	0:04:41	0:00:21

Table 3. Approach 1: Accuracy and Loss

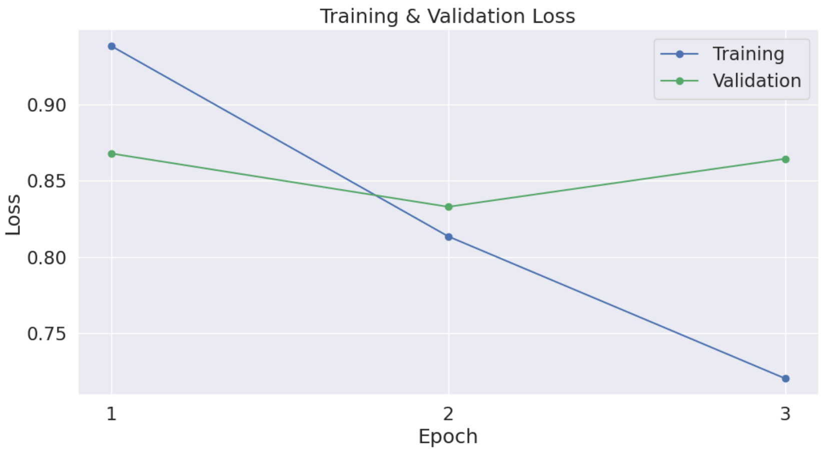
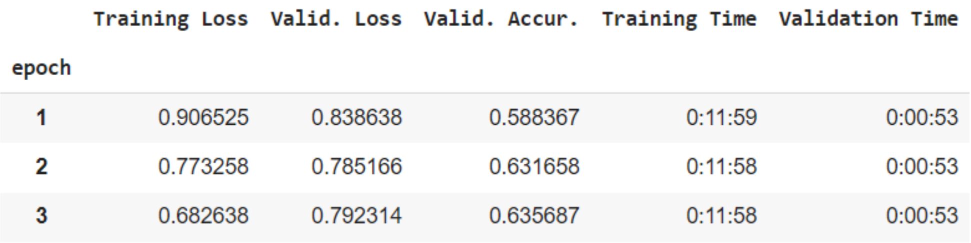


Figure 6. Approach 1: Loss Vs. Epoch

Approach 2: Here is a summary of the provided training and validation results for three epochs:

- Training Loss: The training loss decreases from 0.907 to 0.683 over the three epochs. This indicates that the model is progressively fitting the training data better.
- Validation Loss: The validation loss decreases from 0.839 to 0.792 across the epochs. The decreasing trend suggests that the model is improving its generalisation performance.
- Validation Accuracy: The validation accuracy increases from 0.588 to 0.636 over the three epochs. This shows that the model is getting better at correctly classifying instances in the validation set.



epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.906525	0.838638	0.588367	0:11:59	0:00:53
2	0.773258	0.785166	0.631658	0:11:58	0:00:53
3	0.682638	0.792314	0.635687	0:11:58	0:00:53

Table 4. Approach 2: Accuracy and Loss

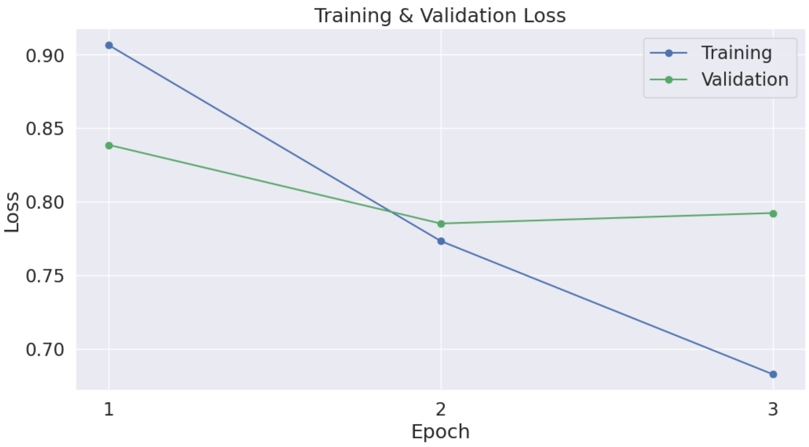
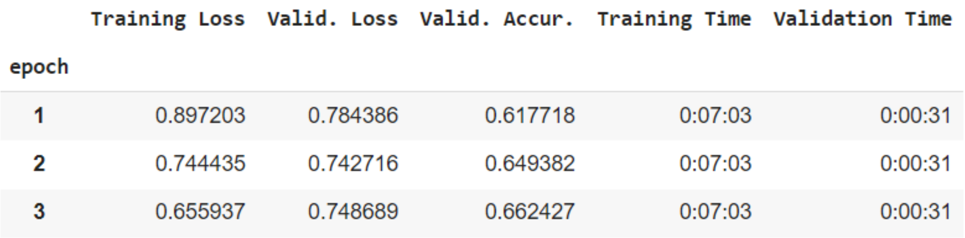


Figure 7. Approach 2: Loss Vs. Epoch

Approach 3: Here is a summary of the provided training and validation results for three epochs:

- Training Loss: The training loss decreases from 0.897 to 0.656 over the three epochs. This indicates that the model is gradually improving its fit to the training data.
- Validation Loss: The validation loss decreases from 0.784 to 0.749 across the epochs. The decreasing trend suggests that the model is improving its generalisation performance.
- Validation Accuracy: The validation accuracy increases from 0.618 to 0.662 over the three epochs. This shows that the model is getting better at correctly classifying instances in the validation set.



epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.897203	0.784386	0.617718	0:07:03	0:00:31
2	0.744435	0.742716	0.649382	0:07:03	0:00:31
3	0.655937	0.748689	0.662427	0:07:03	0:00:31

Table 5. Approach 3: Accuracy and Loss

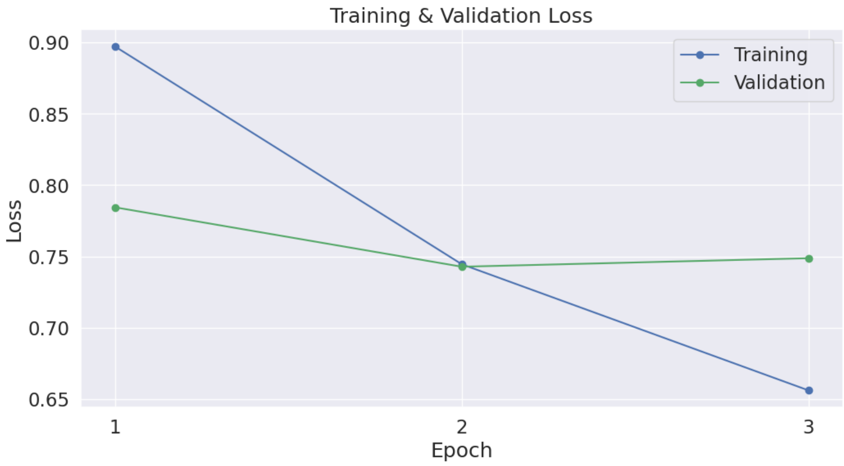
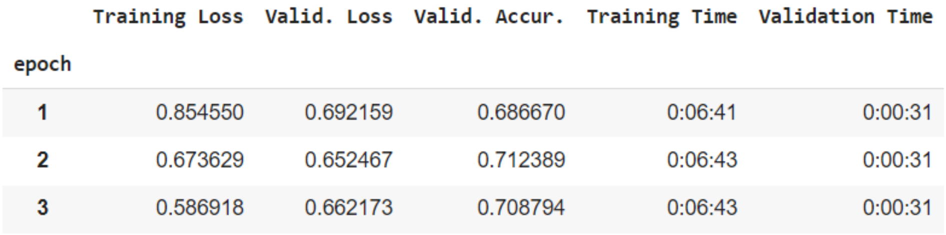


Figure 8. Approach 3: Loss Vs. Epoch

Approach 4: Here is a summary of the provided training and validation results for three epochs:

- Training Loss: The training loss decreases from 0.854 to 0.586 over the three epochs. This indicates that the model is progressively fitting the training data better.
- Validation Loss: The validation loss decreases from 0.692 to 0.662 across the epochs. The decreasing trend suggests that the model is improving its generalisation performance.
- Validation Accuracy: The validation accuracy increases from 0.687 to 0.709 over the three epochs. This shows that the model is getting better at correctly classifying instances in the validation set.



epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.854550	0.692159	0.686670	0:06:41	0:00:31
2	0.673629	0.652467	0.712389	0:06:43	0:00:31
3	0.586918	0.662173	0.708794	0:06:43	0:00:31

Table 6. Approach 4: Accuracy and Loss

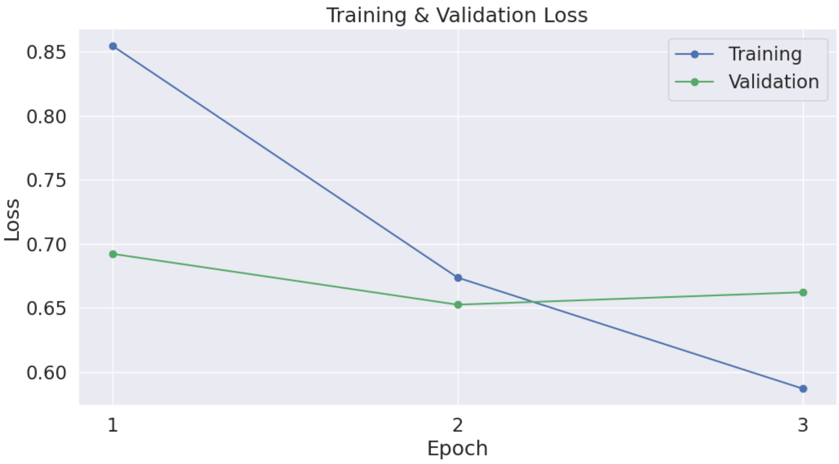
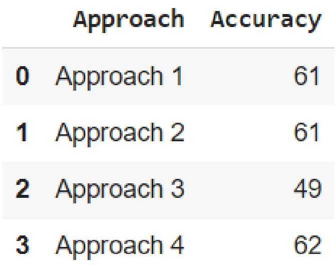


Figure 9. Approach 4: Loss Vs. Epoch

Table 6 presents the accuracy results of the trained model when evaluated on a Hindi-English test set that was not seen during the training phase.



	Approach	Accuracy
0	Approach 1	61
1	Approach 2	61
2	Approach 3	49
3	Approach 4	62

Table 7. Test Accuracy for different Approaches

The results indicate that XLM-R achieved an identical accuracy of 61% in Approach 2 as compared to Approach 1. This similarity in performance can be attributed to the fact that both training datasets included samples of the same kind present in the test set. But this is inconsistent with the observation that the dataset used for training in Approach 4 also had the same characteristics as above. However, it can be concluded that augmenting the training data with different types of mixed languages did not have any impact on the model's performance. But the outcome using Approach 4 clearly outlines the fact that the behaviour of the model for mixed dataset involving languages of similar kind mixed with English results in a better performance. A reasonable reason for this can be the fact that there is a significant difference in the grammatical structure of Spanish and Hindi unlike the grammatical differences between Hindi and Marathi which both share their origins. This discovery indicates that the model's comprehension of code-mixed languages is influenced by the unique linguistic traits present in the data. Furthermore, utilising combined datasets that share similar linguistic characteristics can prove beneficial in addressing challenges associated with languages that have limited available data.

Chapter 6

Conclusion And Future Work

Social media platforms generate vast amounts of unstructured, multi-lingual, and multi-modal data every day. While social intelligence on these platforms continues to grow, the diverse nature of the content poses challenges in analysing and understanding sentiments. This study addresses the issue by utilising the cross-lingual transformer model, XLM-R, as a pre-training model to effectively analyse sentiment at the sentence-level, specifically on tweets, in scenarios with limited resources.

Four different approaches were explored, training the model on distinct datasets with varying multilingual characteristics. The performance of each approach was then evaluated on a Hindi-English dataset. Interestingly, the results revealed that XLM-R achieved the same accuracy of 61% in two distinct approaches. One approach involved training on a similar dataset to the test data, while the other involved augmenting the training data with code-mixed data from two different languages than those present in the test dataset.

The consistency in accuracy when the model encountered similar data suggests its ability to understand the sentiment in the given languages. However, it also highlights that the model interprets different code-mixed languages differently. Furthermore, adding other code-mixed datasets to increase the training data size did not yield satisfactory results, emphasising that the problem of limited availability of training data for mixed languages cannot be addressed through this approach alone.

This observation is further supported by the results of the third approach, where the model did not encounter any data similar to the test samples during training. As expected, the model performed poorly on the test data, achieving an accuracy of only 49%, significantly lower than the previous approaches.

But the results obtained in Approach 4 suggest that the dataset of languages with similar linguistic characteristics can be combined to solve problems associated with a single language. The obtained results address the issue of limited data availability for under-resourced languages. The solution we present here can be applied to various research problems related to languages that share similarities with many other languages. By leveraging the combined data from these languages, we can achieve improved outcomes and better results.

In conclusion, the scarcity of data for mixed languages remains a significant challenge that cannot be effectively resolved by combining different types of code-mixed datasets. But the same technique can be effectively applied in the case of two or more languages with similar linguistic characteristics.

As a future direction, the study aims to employ transfer learning techniques, involving training and fine-tuning the model on a resourceful language such as English, and then utilising the learned model to evaluate performance on less resourceful languages. Previous studies have demonstrated that this approach produces state-of-the-art results when applied to Hindi and Spanish-English datasets. Additionally, the extent of code-mixing will also be an important factor to consider in future research.

Bibliography

1. Akshat Gupta, Sargam Menghani, Sai Krishna Rallabandi, and Alan W Black. 2021. Unsupervised Self-Training for Sentiment Analysis of Code-Switched Data. In Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 103–112, Online. Association for Computational Linguistics.
2. Ou, Xiaozhi and Hongling Li. "YNU@Dravidian-CodeMix-FIRE2020: XLM-RoBERTa for Multi-language Sentiment Analysis." Fire (2020).
3. Braaksma, B., Scholtens, R., Suijlekom, S. van, Wang, R., & Üstün, A. (2020). FiSSA at SemEval-2020 task 9: Finetuned for feelings. ArXiv, 1239–1246.
4. Akshi Kumar and Victor Hugo C. Albuquerque. 2021. Sentiment Analysis Using XLM-R Transformer and Zero-shot Transfer Learning on Resource-poor Indian Language. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 20, 5, Article 90 (September 2021), 13 pages.
5. B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae, ''DravidianCodeMix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text,'' Lang. Resour. Eval., vol. 56, no. 3, pp. 765–806, Sep. 2022.
6. S. Thara and P. Poornachandran, ''Social media text analytics of Malayalam–English code-mixed using deep learning,'' J. Big Data, vol. 9, no. 1, pp. 1–25, Dec. 2022.
7. Mabokela, K. R., Celik, T., & Raborife, M. (Year). "Multilingual Sentiment Analysis for Under-Resourced Languages: A Systematic Review of the Landscape." IEEE Access, Volume 11
8. Jamatia, A., Swamy, S. D., Gambäck, B., Das, A., & Debbarma, S. (2020). Deep Learning Based Sentiment Analysis in a Code-Mixed English-Hindi and English-Bengali Social Media Corpus. International Journal on Artificial Intelligence Tools, 29(05), 2050014.
9. Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451, Online. Association for Computational Linguistics.