National College of Ireland

Sentimental analysis for Hinenglish-code mixed data using advanced word embeddings

MSc Research Project
Data Analytics

Bhumika Sardana
Student ID: x22105859@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Bhumika Sardana
Student ID:	x22105859@student.ncirl.ie
Programme:	Data Analytics
Year:	2023
Module:	Msc Resesrch Project
Supervisor:	Vikas Tomer
Submission Due Date:	14/12/2023
Project Title:	Sentimental analysis for Hinenglish-code mixed data using advanced word embeddings
Word Count:	6007
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.

Bhumika	
Date:	30th January 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):	

Sentimental analysis for Hinenglish-code mixed data using advanced word embeddings

Bhumika Sardana
x22105859@student.ncirl.ie

Abstract

This study investigates the usefulness of combining machine learning models for sentiment analysis with coded mixed language.It primarily focuses on the Hindi and English languagecommonly referred to as 'Hinglish'

This research addresses the application of novel neural network architectures, such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks.It focuses on better interpretation of text in Hinglish texts.This paper shows how these advanced models and traditional NLP techniques such as TF-IDF vectorization improve the accuracy of emotion classification using comprehensive datasets. This shows that it significantly improves the performance by the combination proposed.This result highlights the superiority of certain embeddings such as her MURIL and BERT in processing code-mixed languages and provides insight into their context understanding abilities.

Overall, this study also shows practical implications for improving sentiment analysis tools.It could help to be more inclusive of linguistically diverse environments.

1 Introduction

In the era of Digital communication, Hinglish, a combination of Hindi and English, has carved out a unique niche for itself .This code-mediated language presents undefinite challenges in the field of computational linguistics, especially sentiment analysis.Traditional sentiment analysis tools designed for monolingual texts fail when dealing with Hinglish's morphological rich vocabulary and undefined structure.

To deal with this complexity, this study uses advanced word embeddings such as MURIL (Multilingual Representation for Indian Languages) and BERT (Bidirectional Encoder Representation from Transformers) for Hindi and English. MURIL(Khanuja 2021) has significantly outperforms multilingual BERT.MURIL excels in contextual skills in untangling the complex web of Indian languages, including Hinglish.In contrast, BERT for Hindi and English provides a nuanced understanding of transformer-based models, which can capture the complexity of bilingual texts.

In addition to these state-of-the-art models, this study also investigates the role of a traditional technique like TF-IDF vectorization.Although, TF-IDF may not provide deep contextual insights like MURIL or BERT but its strength lies in highlighting the meanings of words in a corpus. .It provides a different perspective for this study.It also addresses the effectiveness of these different methods in decoding the emotions embedded in Hinglish texts..Below are few examples of the tweets used in this project:

Tweet 1: Aapne papa ki pic lagaya ho profile pe

Tweet 2: Thankyou so much apne followers ko apna dost mane ke liye

Tweet 3: Yeh tiktok kids joh apne aap ko cool samajh te hai

These examples show the complex structure associated with the code-mixed form are and the necessity of dealing the complex structure with advanced techniques.

1.1 Research objective Question

The objective of this research study is to answer the following question:

To what extent do advanced embeddings like MURIL [1] and hindi-english bert [2], as well as TF-IDF vectorization, enhance the accuracy of sentiment analysis in Hinglish texts, especially in capturing more nuanced semantic contexts? Moreover, how effectively do machine learning algorithms such as CNN and LSTM leverage these embeddings and vectorization techniques for improved performance in sentiment analysis tasks?

The objective is to understand how these advanced embedding and vectorization techniques can capture the unique semantic context inherent in Hinglish, a language that fluently blends Hindi and English in a non-standard manneris to evaluate.

This main aim of this study to understand how the combination of advanced NLP word embedding techniques with two deep learning models can improve sentimental analysis.A popular deep leaning method, Convolutional Neural Networks (CNNs) is known to achieve excellent benchmarks in sentence classification Kim (2014). Recurrent Neural Network and Long short-term memory networks (LSTMs) a type of deep learning model related to timing is often used in text classification Graves and Schmidhuber (2005).We are also aiming for the results of this study may significantly contribute to computational linguistics and provide valuable insights that may reshape India's engagement with the dynamic digital communication environment.

Furthermore, these findings may inform approaches in other regions with similar language phenomena and provide new perspectives for addressing mixed language contexts in sentiment analysis.

2 Related Work

Natural language processing(NLP) has been evolving and has fundamentally changed how we interpret and analyse language,especially in situations involving code-mixed context.The work has been further divided into subsection that have made major contributions to the subject.It focuses on different methods , approaches and challenges faced in the computational linguistics environment.

2.1 Efficiency of Different Embeddings in Multilingual Contexts.

The study by Kumar et al. (2020) establishes new standards in language modeling through the creation of 436 word embedding models incorporating 14 Indian languages, including Hindi. Significant performance advantages are seen when complex embedding techniques

[1]https://huggingface.co/google/muril-base-cased

[2]https://huggingface.co/ganeshkharad/gk-hinglish-sentiment

like BERT and ELMo are used for important natural language processing (NLP) applications like named entity recognition (NER) and part-of-speech (POS) tagging. .This application of transfer learning to low-resource languages underscores the potential for customised embedding strategies to capture semantic relationships amidst linguistic diversity. Although the study emphasizes monolingual environments, the insights highlight the challenges in sentiment analysis for code-mixed languages like Hinglish. However, factors like variability and uneven dataset sizes affect model generalizability and also neglect the nature of hybrid linguistic structures.

Another research conducted by Yadav and Chakraborty (2020) demonstrates progression in the field of computational linguistics. Specifically, this research introduces methods that make use of multilingual as well as crosslingual embeddings for transferring knowledge from monolingual to code-mixed text for sentiment analysis.This approach thereby eliminates the requirement of machine translation system, which are often ineffective when dealing with low-resource languages. The study's results demonstrate a significant advancement over the previous state-of-the-art in English-Spanish code-mixed sentiment analysis, obtaining an F1 score of 0.58 when trained unsupervisedly and 0.62 when trained using a parallel corpus. This indicates the value of training on real code-mixed corpus and utilising subwords. The study comes to the conclusion that zero-shot techniques hold the future to the development of code-mixed data analysis.

The article by Singh and Lefever (2020) contributes to the domain of computational linguistics by tackling the delicate taskof sentiment analysis in code-mixed language data, specifically Hinglish. The study's main method is focused on unsupervised cross-lingual embeddings to analyze sentiment in Hinglish (a blend of Hindi and English) tweets. This approach was chosen to overcome the lack of parallel data for such a code-mixed language. The methodology the researchers devised involved two stages: first, they used monolingual embeddings from sizable collections of Hinglish and English tweets to build baseline models. After that, they investigated two systems that used cross-lingual embeddings: a transfer learning technique and a supervised classifier that were both trained on sentiment data in English before being assessed on code-mixed Hinglish as data. According to our findings, using cross-lingual embeddings improves performance over the single-language baseline system generates cross-lingual embeddings that are so strong that, even in a transfer learning environment, the system achieves an F1-score of 0.556, which is equivalent to the scores of 0.606 and 0.616 for supervised classification. Our results show that there is a potential in use of cross-lingual embeddings to improve sentiment analysis, demonstrating that they can outperform monolingual baselines without needing parallel data for training. Chaitanya et al. (2018) further study significantly advances computational linguistics, particularly in the domain of code-mixed linguistic processing. This research combines word embedding techniques such as Continuous Bag of Words or CBOW & Skip-Gram models to identify various languages in code-mixed data, especially on social media sites. Good cross-validation scores were obtained by the research using machine learning algorithms as SVM, Logistic Regression, K-Nearest Neighbours, Adaboost,Gauss Naive Bayes, and Random Forest.Highest accuracy was achieved by SVM for both CBOW and Skip-Gram at 67.33% The intricate challenge of linguistic recognition in code-mixed data, which is a common occurrence in multilingual countries like India, is tackled creatively in this work. The study's shortcoming, though, is that it only looked at word-level identification, which may not adequately reflect the syntactic and semantic nuances present in code-mixed languages. Moreover, the dependence on certain word embedding models can restrict the applicability of the results in other linguistic settings

and code-mixes.

The article presented by Ravi and Ravi (2016) a significant advancement in the field of sentiment analysis in contexts with mixed languages.Utilising the TF-ID (term frequency-inverse document frequency) representation of features, the study focused on Hinglish, which is a combination of Hindi and English. A total of 840 experiments were conducted to identify the most efficient techniques for predicting sentiment in news and Facebook comments. Using sensitivity, specificity, and Area Under the Curve (AUC) metrics, the results showed that a mixture of TF-IDF, Gain Ratio (GR) feature selection, and Radial Basis Function Neural Network (RBFNN) produced the best results. The study comes to the conclusion that this combination successfully outperforms other classifiers in the datasets for news and Facebook comments, with an emphasis on lexicon-based techniques and phrase parsing for more in-depth sentiment analysis in the future.

2.2 Efficiency of Neural Network Architecture

A study by Das and Singh (2023) expands the field of sentiment analysis, with a particular focus on Hinglish social media content.Analyze sentiment from labeled Hinglish datasets using deep learning techniques such as CNN, LSTM, and Bi-LSTM.The methodology is comprehensive and includes dataset collection, labeling, preprocessing, and semantic similarity analysis with data obtained using the TwitterScraper API.Among the deep learning models tested, CNN had the highest accuracy of 75.25%.This study concludes that CNNs are the most effective for Hinglish sentiment analysis and suggests future research comparing these models with transformer models such as BERT, RoBERTa, and ALBERT.The strength of this study lies in the application of a variety of deep learning models to a challenging and relatively unexplored dataset.However, because it focuses on a specific dataset, it may not be representative of the entire range of use of the Hinglish language, and is a limitation that even the best-performing models still have significant room for improvement in accuracy.

In this study, Souza and Filho (2022) investigate several applications of trained and optimised BERT-based models for sentiment analysis with a particular focus on Brazilian Portuguese. This study approach is new as it combines his BERT output-wide technology and variation and aggregation procedures to allow for complete document embedding. Evidently, his BERtimbau embedding outperforms multilingual m-BERT in terms of performance. The findingsdemonstrate the usefulness of his BERT model for sentiment analysis and point to the need for more research on the model that makes use of deep learning and optimisation.

In this paper, Wadhawan and Aggarwal (2021) performs the study of emotion recognition in data with mixed Hindi and English codes and then compares transformer-based models with traditional word representations.Their research indicates that BERT has achieved an accuracy of 71.43% which is much higher than baseline model SVM classifier with an RBF kernel.This increase can be because of the use of embeddings trained on a combination of English and Hindi data.It has helped to enhance the semantic coverage and it also allows more efficient word vector correlation.Additionally, FastText embedding outperformed Word2Vec in terms of performance.This is probably because terminology used into the code is processed more effectively.In order to lessen noise and contextual ambiguity, this study stresses on the difficulty presented by complex nature of language and the requirement for cleaner, more precise data.

The study by Mathur et al. (2018) significantly contributes to computational linguist-

ics.It presents a novel Multi-Input Multi-Channel Transfer Learning (MIMCT) model for identifying offensive content in code-switched Hinglish tweets.It makes use of a CNN-LSTM architecture to establish a new benchmark in classifying Hinglish offensive text. The F1 score for the classification of non-offensive, abusive, and hate-inducing tweets in the Hinglish Offensive Tweet (HOT) dataset using transfer learning (TFL) with Glove, Twitter Word2vec, and FastText embeddings was found to be 0.823.The MIMCT model outperforms baseline model thus establishing its effectiveness.But, the study's limitation lies in its dependence on transfer learning and its focus on Twitter data. The research proves to be inline with our research.It helps to understand to how deep learning models can be used in code-switched language processing

The paper by Minaee et al. (2019) introduces a novel model that blends GloVe embeddings with CNN and Bi-LSTM networks. Sentiment analysis using geographical and temporal data characteristics is a hallmark of this ensemble technique.It performs far better than only LSTM model and separate CNN models in terms of accuracy. As an illustration of its efficacy in sentiment analysis tasks, the ensemble model's curation rate on the IMDB review dataset was 90%, compared with 89.3% for CNN & 89% for LSTM. This shows how combining two neural network architecture along with embedding can improve the accuracy in context of sentimental analysis.

2.3 Code-mixing and comparative evaluations

The article by Patil et al. (2023) offers a comparision of different pre-trained BERT models in the context of codemixed language. The goal of the research is to evaluate the performance of several models such as Multilingual BERT, HingBERT, HingRoBERTa, HingRoBERTa-Mixed, Albert, BERT, and RoBERTa.It also makes use of five different dataset.The results indicate that HingRoBERTa and HingRoBERTa-Mixed models consistently achieved the highest F1 scores. For instance, on the Emotions dataset, HingRoBERTa-Mixed achieved an F1 score of 0.98285, and on the Sentiment dataset, it achieved an F1 score of 0.74898.The reason why these two models were successful as they were trained on romanized Hindi-English code-mixed data. On the other hand, ALBERT performed the poorest.It can primarily because of its training on smaller English-only dataset. The study concludes that HingBERT-based models significantly improve the performance on code-mixed datasets for tasks related to sentiment analysis, emotion recognition or hate speech detection.Therefore, contributing significantly to the fields of sentiment analysis and opinion mining.

In the research paper by Dave et al. (2021), the researchers use a detailed approach to identify desired speech in coded texts.The study of text representation involves the use of MuRIL and TF-IDF approaches.MuRIL is a model based on transformers trained on 17 Indian languages and provides a 768-dimensional vector representation.In contrast, TF-IDF focuses on the arrangement of letters in N-grams.Linear SVM and logistic regression classifier are used for classification.The validity of the research methodology is demonstrated by weighted F1 values of 0.92 for English, 0.75 for Malayalam English, and 0.57 for Tamil English in the test dataset.

The study by Awatramani et al. (2021)significantly contributes to understanding sentiment analysis in Hinglish byaddressing the challenges of mixed-case language processing. Employing a range of methodologies, including Lexicon-Based, Rule-Based, and Machine Learning approaches, the study stands out for its effective use of Support Vector Machine (SVM) and Logistic Regression (LR), achieving an F1-score of 0.86 and 86% accuracy.

This highlights the effectiveness of these methods in classifying sentiments in Hinglish text. The study's key strength is its comprehensive examination of different techniques to address the unique challenges of sentiment analysis in Hinglish. However, its focus primarily on Hinglish might limit its direct applicability to other mixed-case languages, and the machine learning techniques used may require significant adaptation for different linguistic contexts. The researchers suggest future exploration in deep learning techniques and expanding the Hindi Romanized dictionary to enhance model performance, indicating potential avenues for further research and application in other mixed-case languages.

The paper by Jada et al. (2021) details the performance of various transformer models like MuRIL, BERT, XLM-Roberta, and DistilBERT in sentiment analysis tasks for Dravidian languages. The study uses a soft voting classifier and then combines these fine-tuned models, and finally reports weighted F1-scores on different test data sets. For instance, it achieved F1-scores of 0.626, 0.708, and 0.609 for Tamil, Malayalam, and Kannada respectively. This demonstrates the models' effectiveness in sentiment analysis across these languagesThe limitations mentioned include the challenge of handling the class imbalance in datasets, where the majority of texts fall into the positive category, leading to weaker performance in minority classes. Additionally, the F1-score for the 'Mixed Feeling' class was lower compared to the 'Non-Tamil' class in the Tamil validation set. Another issue was the highest F1-score for the 'Not-Malayalam' label despite more samples belonging to the positive class in the Malayalam dataset.

The article by Yadav et al. (2021)uses Bi-LSTM and ensemble classification techniques to analyze social media code-mixed data.It focuses on text sentiment analysis.In this study, various algorithms such as multinomial naive Bayes, support vector classifiers, and stochastic gradient descent are used. The models were first tested separately and then an ensemble classifier was proposed consisting of baseline models.Also, Bi-LSTM was proposed to provide consistent classification.The ensemble classifier achieved the highest average accuracy of 0.74.The overall accuracy of the Bi-LSTM model was 0.73, with F1 values of 0.60 for positive emotions, 0.65 for negative emotions, and 0.53 for neutral emotions.These results prove the usefulness of the model in sentiment analysis of code-mixed data.

The article by Mukherjee (2020)adds to the context of sentiment analysis in code-mixed by introducting a new approach.In this study, the author has introduce his DLACMT (Deep Learning Architecture for Coded Mixed Text) model, which uses lazy fusion of character and word features to improve text classification.Upon Comparing this model with the baseline model, DLACMT achieved a high accuracy of 69.845% over baseline's accuracy of 66.494% .The F1 score of models were 0.6613 and 0.606 respectively. This demonstrates the effectiveness of his DLACMT model, especially when using his Adamx optimizer and categorical cross-entropy loss function.It also demonstrates its robustness and stability for sentiment analysis in difficult bilingual contexts.

The research by Patra et al. (2018) makes significant contribution in the context of sentiment analysis in Indian code-mixed languages, especially Hindi-English and English-Bengali. The main method focuses on using GloVe word embeddings and TF-IDF scores of n-grams, with classifiers like ensemble voting and linear SVM for classification. Another approach that was followed was to apply a simple deep learning method using FastText for word embeddings and CNNs combined with Bi-LSTM for feature extraction, with a softmax layer for prediction. The results achieved were of the highest macro average F1-scores of 0.569 for the Hindi-English dataset and 0.526 for the Bengali-English dataset. For two-way classification, the highest F1-scores were 0.707 for positive, 0.666 for

negative, and 0.663 for neutral sentiments in Hindi-English, and 0.641 for positive, 0.677 for negative, and 0.621 for neutral in Bengali-English.The main highlight of this research is the innovative use of a combination of embedding and machine learning techniques to address the challenges of sentiment analysis in code-mixed languages.

The research by Senevirathne et al. (2020) demonstrates various deep learning models such as RNN, LSTM, GRU, Bi-LSTEM and other hybrid combinationation with CNN , as well the capsule network in textual Sinhala sentimental analysis . These models are applied to categorize emotional experiences at the document level.Traditional language-independent features like bag-of-words, word n-grams, and TF-IDF were used but they didn't perform well for Sinhala sentiment analysis.However, word embedding techniques (Word2Vec and fastText) were then utilized as input features, with fastText performing better for Sinhala sentiment analysis.From the results, the Bi-LSTM model surpasses the other models with a weighted accuracy of 63.81

Despite the advancements in NLP, there is a significant research gap in applying these advanced embeddings specifically to Hinglish sentiment analysis. This section highlights this gap, emphasizing the need for research that explores the integration of these embeddings with machine learning algorithms.Additionally, CNN and LSTM effectiveness in using these technique need further investigation.Table 1 provides the tabular representation of literature review.

Table 1: Summary of Research Papers

Sr. No	**Paper Title/Author**	**Year**	**Models (Language)**	**Results (F1 score/Accuracy)**
1	"A Passage to India": Pre-trained Word Embeddings for Indian Language	2020	hi-cbow, hi-fasttext, hi-sg (Hindi)	Hindi (hi-cbow): Approx 0.88, Hindi (hi-fasttext): Just below 0.92, Hindi (hi-sg): Just below 0.8
2	Unsupervised Sentimental Analysis for code-mixed data	2020	LSTM with custom FastText, MUSE-SUP (Multilingual)	LSTM with custom FastText embeddings: F1-score of 58.40 (unsupervised), LSTM with MUSE-SUP embeddings: F1-score of 64.00
3	Sentiment analysis for Hinglish code-mixed tweets by means of cross-lingual word embeddings	2020	VecMap SeedDict(Hinglish)	Macro-Average F1-score of 0.635
4	Word level language identification in code-mixed data using Word Embedding Methods for Indian Languages	2018	SVM for both CBOW and Skip-Gram (Indian Languages)	67.33%

Continued on next page

Table 1 continued from previous page

Sr. No	Paper Title/Author	Year	Models (Language)	Results (F1 score/Accuracy)
5	Sentiment classification of Hinglish text	2016	Hinglish	AUC 0.8601
6	Sentiment recognition of Hinglish code mixed data using deep learning methods.	2023	CNN(Hinglish)	Accuracy CNN: 75.25%
7	Bert for sentiment analysis: Pre-trained and fine-tuned alternatives.	2022		Not specified
8	Towards emotion recognition in Hindi-english code-mixed data: A transformer based approach	2021	BERT, CNN(Word2Vec), CNN(FastText), LSTM(Word2Vec), LSTM(Fast Text), Bi-LSTM(Word2Vec), Bi-LSTM(FastText) (Hindi-English)	Accuracy BERT: 71.43%, CNN(Word2Vec): 62.22, CNN(FastText): 62.24, LSTM(Word2Vec): 63.83, LSTM(Fast Text): 63.69, Bi-LSTM(Word2Vec): 66.65, Bi-LSTM(FastText): 67.34
9	Did you offend me? Classification of offensive tweets in Hinglish language.	2019	Hinglish	F1 score: HOT dataset: 0.823
10	Deep-sentiment: Sentiment analysis using ensemble of CNN and Bi-LSTM Models	2019		IMDB review dataset: 90% accuracy
11	Comparative study of pre-trained Bert Models for Code-Mixed Hindi-English Data	2023	HingRoBERTa, HingRoBERTa-Mixed(Hindi-English	For HingRoBERTa, the F1 score is 0.73270. For HingRoBERTaMixed, the F1 score is 0.71087
12	Hope speech detection in code mixed text using TF-IDF char N-grams and muril.	2021	Code-mixed	English: 0.92, Malayalam English: 0.75, Tamil English: 0.57
13	Sentiment analysis of mixed-case language using natural language Processing.	2021	Mixed-case	0.86
14	Transformer based sentiment analysis in Dravidian languages.	2021	Dravidian	Tamil: 0.626, Malayalam: 0.708, Kannada: 0.609

Continued on next page

Table 1 continued from previous page

Sr. No	Paper Title/Author	Year	Models (Language)	Results (F1 score/Accuracy)
15	Bi-LSTM and ensemble based bilingual sentiment analysis for a Code-mixed Hindi-English Social Media Text	2021	Ensemble classifier, Bi-LSTM(Hindi-English)	Ensemble classifier: 0.74, Bi-LSTM: 0.73 (Positive: 0.60, Negative: 0.65, Neutral: 0.53)
16	Deep learning technique for sentiment analysis of Hindi-English Code-Mixed Text using Late Fusion of Character and Word Features.	2020	DLACMT(Hindi-English)	0.6613
17	Sentiment analysis of code-mixed Indian languages: An overview of SAIL$_{C}ode - mixedsharedtask@icon-$ 2017	2018	Indian languages	Hindi-English: 0.569, Bengali-English: 0.526
18	Sentiment Analysis for Sinhala Language using Deep Learning Techniques	2020	RNN, LSTM, GRU, Bi-LSTM,other hybrid combinations(Singhala)	Bi-LSTM : 0.6318

3 Methodology

The methodology for sentiment analysis on Hinenglish text. Instead of using CRISP-DM, this project utilizes the Knowledge Discovery in Databases (KDD) approach.CRISP-DM is an industry-based approach designed to meet project deployment needs of business application.The main goal of this research project is to focus on knowledge acquisition while addressing diverse data selection, preprocessing, feature extraction, and model selection with accurate outcome evaluation.There is no fixed algorithm or implementation to identify sentiment in Hindi-English tweets with mixed codes. The KDD (Cross-Industry Standard Process for Data Mining) framework, involves five phases: Data Selection, Data Preparation, Data Tranformation, Data Mining and Evaluation.Here's a detailed explanation of each step:

3.1 Data Selection

In the Data Understanding phase of the Hinenglish sentiment analysis project, the primary focus is on the L3Cube-HingCorpus and HingBert dataset Nayak and Joshi (2022), which provides a comprehensive and intricate collection of Hinenglish text. This dataset1 , comprising around 1,355,497 cases, is meticulously labeled at the word level, with tags 'en' for English and 'hi' for Hindi, offering detailed insights into the linguistic

structure of Hinglish—a blend of Hindi and English.

The dataset's composition reveals 387,121 English words and 968,376 Hindi words, indicating a higher prevalence of Hindi. This imbalance is crucial for understanding the typical language patterns in Hinglish contexts. An example from the dataset illustrates the real-world use of code-mixing, combining Hindi and English in a single sentence, which is essential for analyzing the context and semantics in such texts.

Table 2: Wordcount of different language in dataset

Language labels	**word count**
English	387121
Hindi	968376

3.2 Data Preparation

The Data used for this project is by L3Cube-HingCorpus and HingBert Dataset [3] Nayak and Joshi (2022).It was collected using Twint from various tweets.As the objective was to collect hinglish data .It was preprocesed to remove the non-English characters.It was then passed through word-level language classifier model in order to understand the language used in sentence.Each word was labelled to Hindi or English based on the content and final sentence was labelled as code-mixed if containing both language labels. But for our study, we require sentences. Each row containing a blank space after a word indicated the end of sentence. Hence, based on this information word from each row were concatenated and a sentence was built.Also, it was observed that sentences contained many Hindi or English words, which was written with different spellings for example 'aap' which means 'you' in English, can be written as 'aapne', 'apne', 'aap', 'ap' words sounds same and have the same meaning.Decontraction was performed for English words meaning words ending with ' 't' was decontracted to 'not', ' 're' was decontracted to are and so on.

3.3 Feature Extraction

For feature extraction, two approaches are employed: tokeniser based and vector based.Word embeddings to be used are explained in detail as below.

1. HinEnglish BERT : The creation of this model[4] is based on the fine-grained task of sentiment classification.It is based on three different emotional tones: negative, neutral, and positive. This innovative approach is an evolution of the pre-trained rohanrajpal/bertbase-multilingual-cases-sentiment model .However, it is important to note the inherent limitations of the model. Its dependence on somewhat limited training data sets and language-specific coding frameworks.These shape and to some extent limit the applicability of the model.

2. MuRIL: MuRIL Khanuja et al. (2021) has undergone significant training and refinement to become a reliable and proficient language model for Indians.It supports around 17 languages including English and other Indian languages including 16 others.MuRIL outperforms multilingual BERT on all benchmark datasets of his 4,444

[3]https://github.com/l3cube-pune/code-mixed-nlp/tree/main/L3Cube-HingLID
[4]https://huggingface.co/ganeshkharad/gk-hinglish-sentiment

languages in India.The pre-training phase of MuRIL involves the use of Masked Language Modeling (MLM) and Translated Language Modelling(TLM)

3. TF-IDF Vectorization: TF-IDF (Term Frequency - Inverse Document Frequency) is a numerical statistic that aims to reflect the importance of words to documents in a collection or corpus.For Machine learning models to understand, text data is converted to a numerical representation.This method is especially useful for highlighting the most relevant words in sentiment analysis. During vectorization, the size of the vector will be equal to the number of unique words in the corpus. The vectors will be assigned to each word based on the word frequency in every sentence. These numerical vectors are understandable by machine learning and deep learning models.

3.4 Data Transformation

When it comes to deep learning for text analysis, consistent input shape and size are very important.Our dataset contains sentences of varying lengths, so a standardization process is necessary. Padding is an important technique that converts text of varying length to a uniform length. Our project uses advanced NLP models such as Hindi BERT and MuRIL.They are based on the BERT framework and use padding as part of preprocessing.This ensures consistent sequence length.Additionally, we use TF-IDF vectorization to transform the text into a numerical matrix that emphasizes word meanings and is suitable for machine learning models.LSTM models apply explicit padding to standardize the length of the input.Additionally, an important part of the data transformation process is splitting the data into training and testing sets.This was achieved using sklearn train_test_split().The data is divided into 80:20 ratio, where in 80% is for training and 20% is for testing.This is a necessary step to evaluate the performance of the deep learning model.

3.5 Modeling

The primary models used are Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN). CNN has achieve great advancemen in sentence classification (Yoni et al .2014)as used by Minaee et al. (2019), each offering unique strengths in handling text data. To deploy and construct these deep learning models powerful tools like TenserFlow and Keras are used.

1. LSTM Models: LSTM is a type of recurrent neural network (RNN) particularly suited for sequence prediction problems, like language modeling and translation. In the context of Hinenglish sentiment analysis, LSTMs can effectively remember and utilize context from earlier in the text, which is crucial for understanding the sentiment of a sentence. This is particularly beneficial for handling the nuances of code-mixed language, where context plays a key role in interpretation.Figure shows working of one cell of LSTM

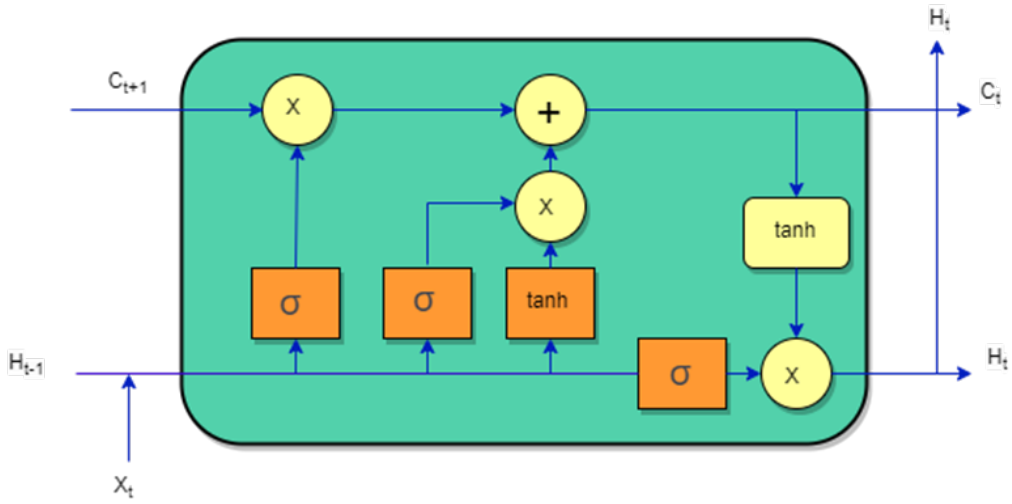

Figure 1: Working cell of one LSTM

2. Convolutional Neural Networks (CNN): While primarily known for image processing, CNNs have also been effective in NLP tasks. In the context of Hinenglish text, CNNs can capture spatial hierarchies and patterns in sentences, identifying indicative features for sentiment analysis, like specific combinations of words and phrases.To understand the sentiment in the text, CNN can abstract and emphasize these feature by applying various filters and pooling layers.

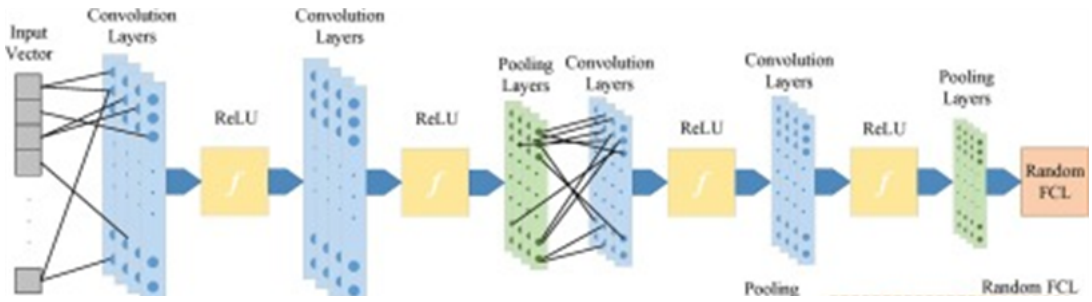

Figure 2: CNN 1D

3.6 Evaluation

A test set is utilized to evaluate the performance of individual sentiment analysis models in a comprehensive manner during the evaluation phase. The main performance metric used is Accuracy.It measures the ability of model to properly separate sentiment.

An important test for assessing the level of accuracy in how models interpret and classify emotions within this language mix is using this accuracy. We also provide thorough performance comparisons to help you make an informed model choice.

This comparative analysis allows you to see which model performs better than other specific tasks in Hinglish sentiment analysis. These insights lead to further improvements in model design and model training, ensuring that the selected model performs well.It has the ability to effectively handle the complexities of Hinglish sentiment analysis.

4 Design Specification

The study project's design specifications are shown in Figure 2, where a Hinglish dataset is cleaned and pre-processed (stemming, stop word removal, tokenization) before being divided into two sets: a 20% test set and an 80% training set. Text may be converted into numerical vectors by feature extraction techniques like vectorization or word embeddings. These are then fed as input to the models used.To standardize input dimensions, features are tokenized and padded for deep learning models like CNN and LSTM. Eventually, the best method for sentimental analysis in Code Mixed Hindi-English tweets is identified by a comparative analysis of the output of all the models.

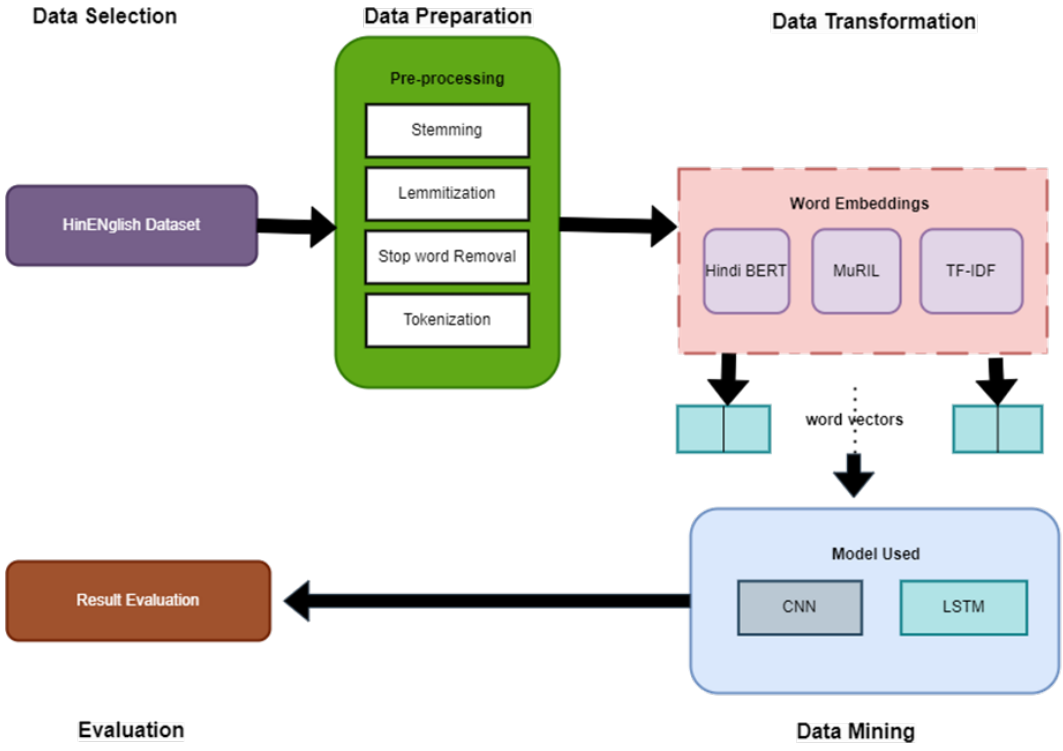

Figure 3: Design Specification

To understand in depth, a brief overview of the algorithm is provided in the form of pseudocode

```
Algorithm 1: One Iteration Through an LSTM Cell

Procedure Iteration(c_t_minus_1, h_t_minus_1, x_t)
    Inputs:
        c_t_minus_1: previous cell state
        h_t_minus_1: previous hidden state
        x_t: current input

    Begin
        1. Calculate the forget gate output f_t.
           (Typically, f_t = sigmoid(W_f * [h_t_minus_1, x_t] + b_f))

        2. Update the cell state.
           c_t = c_t_minus_1 * f_t (Element-wise multiplication) [Equation 3]

        3. Calculate the input gate output m_t and candidate cell state ~c_t.
           (Typically, m_t = sigmoid(W_m * [h_t_minus_1, x_t] + b_m) and
           ~c_t = tanh(W_c * [h_t_minus_1, x_t] + b_c))

        4. Update the cell state with the output from the input gate.
           c_t += ~c_t * m_t [Equation 6]

        5. Calculate the output gate o_t.
           (Typically, o_t = sigmoid(W_o * [h_t_minus_1, x_t] + b_o))

        6. Compute the new hidden state.
           h_t = o_t * tanh(c_t) [Equations 7 and 8]

        7. Return the updated cell state c_t and the hidden state h_t.
    End

End Procedure
```

Figure 4: Pseudo code for one iteration of LSTM cell

Figure pseudocode describes how an LSTM cell operates throughout a single iteration, updating its hidden state and state whenever a new input is received. A forget gate first uses a sigmoid function to determine whether prior data should be discarded. Subsequently, the input gate modulates the new candidate state, which is a mix of the current input and the prior concealed state, and multiplies the old state by the forget gate's output to update the cell state. The current output, which also becomes the new hidden state, is then generated by the output gate selecting which portion of the cell state to employ. The revised cell state and the new concealed state are then produced, prepared for the following iteration.With the help of this process LSTM is able to capture long term dependencies by selectively adding or forgetting information

```
# input_sequence is a 1D array representing the input sequence

# Step 1: Convolution
# Apply a 1D convolutional layer with specified number of filters, kernel size, and activation function
convolved_features = conv1d(input_sequence, num_filters, kernel_size, activation_function)

# Step 2: Flattening
# Flatten the convolved features to create a single long vector
flattened_vector = flatten(convolved_features)

# Step 3: Fully Connected Layer 1
# Pass the flattened vector through a fully connected layer with specified number of neurons and activation function
fc1_output = fully_connected(flattened_vector, num_neurons_fc1, activation_fc1)

# Step 4: Dropout
# Apply dropout with a specified probability to reduce overfitting
dropout_output = apply_dropout(fc1_output, dropout_rate)

# Step 5: Fully Connected Layer 2
# Pass the result through another fully connected layer with specified number of neurons and activation function
fc2_output = fully_connected(dropout_output, num_neurons_fc2, activation_fc2)

# Compile the CNN
# Use specified optimizer, loss function, and metrics
compile_cnn(optimizer, loss_function, metrics)

# CNN Summary
cnn_summary()
```

Figure 5: 1D CNN

The procedure for building a 1-dimensional Convolutional Neural Network (1D CNN) using sequence data is described in the Figure. The input sequence is first run through a convolutional layer, which uses an activation function and a series of filters to turn it into a set of convolved features. The next step is to flatten these characteristics into a single long vector so that it may be used as input for the fully linked layers that follow. After applying an activation function to the flattened vector, the first fully connected (dense) layer produces an output. This output is then sent via a dropout layer, which randomly zeros a part of the features in order to reduce overfitting. The final output is produced by feeding the dropout's output into a second fully connected layer that has its own activation function.

5 Implementation

5.1 Annonation

After cleaning the data, 44,453 tweets in Hindi and English with mixed codes remained in the dataset.Annotations are the most important part of an emotion or emotion analysis problem.So, I have used the SentimentIntensityAnalyzer from the NLTK library, which is used to calculate sentiment scores for sentences in a data frame.Based on the sentiment score, a binary label is assigned, such as 1 for positive and 0 for negative.There was a major issue of class imbalance.Hence, the lower value i.e of positive was considered as threshold and implemented for class imbalancing.Figure 6 & 7 shows word clouds representing positive and negative emotions and Table 3 shows data before and after class balance

Figure 6: Wordcloud for Positive sentiments

Figure 7: Wordcloud for Negative sentiments

Table 3: Data before and after class balance

Labels	Before Class balancing	After Class balancing
Negative	28822	15631
Positive	15631	15631

5.2 Data Pre-processing

The dataset consisted of 15631 positive words and 28822 negative words.All punctuation and special characters have been removed to clean up the text.English words were also decontracted.Additionally, stop words were removed from the sentences using Rana's open source repository (accessed on December 1, 2023) [5].In doing so, the author has compiled his list of over 1,036 stop words, which includes both Hindi and English stop words.

5.3 Model Implementation

Next step, in our implementation after performing data cleaning, pre-processing and feature extraction is to train the supervised machine and deep learning models to compare the performance of model on code-mixed hindi-english tweets.'Jupyter Notebook' and 'Google Collab' have been used to perform all the implementations.Python libraries have been used extensively to perform all the work.

1. LSTM: The LSTM model used for sentiment classification has 32 LSTM units to capture temporal context. The 'tanh' activation induces non-linearity within these units. This feeds into a 16-unit dense block also with 'tanh' activation to learn abstract representations. Final softmax activated output makes binary predictions. A dropout of 0.3 prevents overfitting. Binary crossentropy loss is minimized using

[5]https://github.com/TrigonaMinima/HinglishNLP/blob/master/data/assets/stop$_h$*inglish*

the Adam optimizer by default. The choice of LSTM units, dropout and dense layers aims to balance model complexity, learning representative features and regularization.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 12)                672

 flatten_2 (Flatten)         (None, 12)                0

 dense_4 (Dense)             (None, 32)                416

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_5 (Dense)             (None, 2)                 66

=================================================================
Total params: 1,282
Trainable params: 1,218
Non-trainable params: 64
_________________________________________________________________
```

Figure 8: LSTM Model Summary

2. CNN: Multiple 1D convolutional layers make up the structure of the CNN model used for sentiment analysis. These layers consist of 128, 64, and 32 filters. The convolutional layers employ a mix of 'tanh' and'relu' activations function.The goal of this arrangement is to extract from the text a variety of geographical and temporal aspects. In order to reduce dimensionality and increase computing performance, the model has MaxPooling layers after each convolutional layer. After flattening the output from the convolutional layers, the model has a dense layer with 2 units and 'tanh' activation. The model uses categorical crossentropy as the loss function

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 30, 128)           256

 max_pooling1d (MaxPooling1D  (None, 15, 128)          0
 )

 activation (Activation)     (None, 15, 128)           0

 conv1d_1 (Conv1D)           (None, 15, 64)            8256

 max_pooling1d_1 (MaxPooling  (None, 7, 64)            0
 1D)

 activation_1 (Activation)   (None, 7, 64)             0

 conv1d_2 (Conv1D)           (None, 7, 32)             2080

 max_pooling1d_2 (MaxPooling  (None, 3, 32)            0
 1D)

 activation_2 (Activation)   (None, 3, 32)             0

 flatten_1 (Flatten)         (None, 96)                0

 dense_3 (Dense)             (None, 2)                 194

=================================================================
Total params: 10,786
Trainable params: 10,786
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: CNN model summary

6 Evaluation

The table 4 below details the results of evaluating several machine learning and deep learning models for the analysis of Code Mixed Hindi-English tweets. The accuracies obtained vary widely. The LSTM model that Bert and the team constructed performed exceptionally well, achieving a flawless accuracy score of 100.00%. The CNN model with Bert, on the other hand, drastically underperformed, registering an accuracy of just 20.97%. With an accuracy of 66.69%, the LSTM model that used TFIDF for feature extraction was only moderately successful; in contrast, the CNN model outperformed the former with an accuracy of 82.61%. Amazingly, the LSTM and CNN models combined with Muril achieved an accuracy of 100.00%, which was exactly the same as the LSTM Bert model as shown in figure 10. These findings demonstrate how different model setups and feature extraction techniques can vary in their efficiency for Code mixed Hindi-English tweets.

Table 4: Results of all the models

Sr. no	**Model**	**Feature Extraction**	**Accuracy**
1.	CNN	Hindi-English BERT	20.97186
		MuRIL	100
		TF-IDF	82.6879
2.	LSTM	Hindi-English BERT	100
		MuRIL	100
		TF-IDF	66.687977

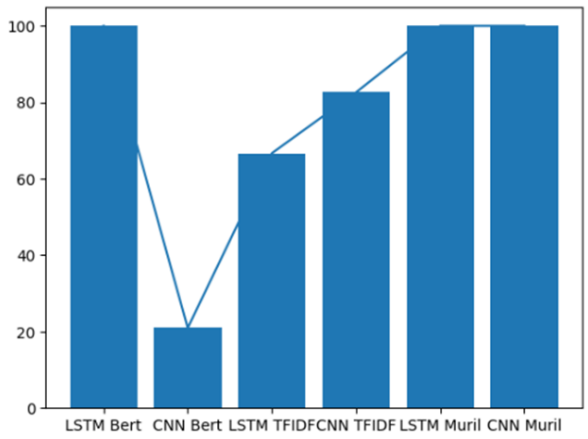

Figure 10: Model performance bargraph

7 Conclusion and Future Work

The main focus of this research was to investigate the combination of embeddings and deep learning models for sentimental analysis.The study maskes use of two Deep learning models that is CNN and LSTM. The dataset was adopted from L3Cubecorupus and appropriate preprocessing techniques were performed.Three techniques were used for feature extraction, which includes MuRIL, TF-IDF, and Hindi-English BERT.

In terms of performance, MuRIL was able to achieve exceptional accuracy for both CNN and LSTM.The Application of MuRIL in understanding the complexities of Hinglish can mark as a significant contribution .Its proficiency with LSTM is useful in recognizing temporal nuances in the language .

HindiEnglish BERT showed wide variation in performance, achieving an accuracy of 20.97 for CNN and 100 for LSTM.Furthermore, the TF-IDF vectorization technique was able to achieve satisfactory performance on both CNN and LSTM models.It reached 82.68% and 66.68% accuracy, respectively.

In the future, the performance gap between CNN and LSTM in BERT for Hindi and English requires further investigation.Future research should aim to expand the diversity of the dataset. It could be expanded to delve into more advanced NLP models. Using these techniques to other mixed-coded languages can be a significant contribution.Moreover, the models themselves can be improved to meet future expectations.A thorough evaluation of their robustness and potential biases could lead to unbiased sentiments suitable for different linguistic contexts. This will contribute in the development of analytical tool.

References

Awatramani, P., Daware, R., Chouhan, H., Vaswani, A. and Khedkar, S. (2021). Sentiment analysis of mixed-case language using natural language processing.
URL: *https://ieeexplore.ieee.org/document/9544554*

Chaitanya, I., S, T., Gupta, S. K. and Madapakula, I. (2018). Word level language identification in code-mixed data using word embedding methods for indian languages.
URL: *https://ieeexplore.ieee.org/document/8554501*

Das, S. and Singh, T. (2023). Sentiment recognition of hinglish code mixed data using deep learning methods.
URL: *https://ieeexplore.ieee.org/document/10048879*

Dave, B., Bhat, S. and Majumder, P. (2021). $\text{Irnlp}_d aiict@lt - edi - eacl2021 : Hopespeechdetectionincodemixedtextusingtf - idfcharn - gramsandmuril, ACLAnthology.$
URL:*https://aclanthology.org/2021.ltedi-1.15/*

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures, *Neural Networks* **18**(5-6): 602–610.
URL: *https://doi.org/10.1016/j.neunet.2005.06.042*

Jada, P. K., Reddy, D. S., Yasaswini, K., K, A. P., Chandran, P., Sampath, A. and Thangasamy, S. (2021). Transformer based sentiment analysis in dravidian languages, *The Information Retrieval Anthology* .
URL: $https://ir.webis.de/anthology/2021.fire_conference - 2021w.98/$

Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D. K., Aggarwal, P., Nagipogu, R. T., Dave, S., Gupta, S., Gali, S. C. B., Subramanian, V. and Talukdar, P. (2021). Muril: Multilingual representations for indian languages, *arXiv* .
URL: *https://arxiv.org/abs/2103.10730*

Kim, Y. (2014). Title of the paper, *in* A. Moschitti, B. Pang and W. Daelemans (eds), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1746–1751. SIGDAT.
URL: *https://aclanthology.org/D14-1181/*

Kumar, S., Kumar, S., Kanojia, D. and Bhattacharyya, P. (2020). A passage to india: Pre trained word embeddings for indian languages, *ACL Anthology* .
URL: *https://aclanthology.org/2020.sltu-1.49/*

Mathur, P., Sawhney, R., Ayyar, M. and Shah, R. (2018). Did you offend me? classification of offensive tweets in hinglish language, *ACL Anthology* .
URL: *https://aclanthology.org/W18-5118/*

Minaee, S., Azimi, E. and Abdolrashidi, A. (2019). Deep-sentiment: Sentiment analysis using ensemble of cnn and bi-lstm models, *arXiv.org* .
URL: *https://arxiv.org/abs/1904.04206*

Mukherjee, S. (2020). Deep learning technique for sentiment analysis of hindi-english code-mixed text using late fusion of character and word features.
URL: *https://ieeexplore.ieee.org/document/9028928*

Nayak, R. and Joshi, R. (2022). L3cube-hingcorpus and hingbert: A code mixed hindi-english dataset and bert language models, *arXiv* .
URL: *https://arxiv.org/abs/2204.08398*

Patil, A., Takawane, G., Phaltankar, A., Patwardhan, V. and Joshi, R. (2023). Comparative study of pre-trained bert models for code-mixed hindi-english data.
URL: *https://ieeexplore.ieee.org/abstract/document/10126273*

Patra, B. G., Das, D. and Das, A. (2018). Sentiment analysis of code-mixed indian languages: An overview of $\text{sail}_code - mixedsharedtask@icon - 2017, arXiv.org$.
URL:*https://arxiv.org/abs/1803.06745*

Ravi, K. and Ravi, V. (2016). Sentiment classification of hinglish text.
URL: *https://ieeexplore.ieee.org/document/7507974/*

Senevirathne, L., Demotte, P., Karunanayake, B., Munasinghe, U. and Ranathunga, S. (2020). Sentiment analysis for sinhala language using deep learning techniques, *arXiv preprint* .
URL: *https://arxiv.org/abs/2011.07280*

Singh, P. and Lefever, E. (2020). Sentiment analysis for hinglish code-mixed tweets by means of cross-lingual word embeddings, *ACL Anthology* .
URL: *https://aclanthology.org/2020.calcs-1.6/*

Souza, F. and Filho, J. (2022). Bert for sentiment analysis: Pre-trained and fine-tuned alternatives, *arXiv.org* .
URL: *https://arxiv.org/abs/2201.03382*

Wadhawan, A. and Aggarwal, A. (2021). Towards emotion recognition in hindi-english code-mixed data: A transformer based approach, *arXiv.org* .
URL: *https://arxiv.org/abs/2102.09943*

Yadav, K., Lamba, A., Gupta, D., Gupta, A. and Karmakar, P. (2021). Bi-lstm and ensemble based bilingual sentiment analysis for a code-mixed hindi-english social media text.
URL: *https://ieeexplore.ieee.org/document/9342241*

Yadav, S. and Chakraborty, T. (2020). Unsupervised sentiment analysis for code-mixed data, *arXiv.org* .
URL: *https://arxiv.org/abs/2001.11384*