

UNLOCKING SENTIMENTS IN CODE-MIXED TEXTS USING ENSEMBLE MODELS OF CNN AND SELF-ATTENTION MODELS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Yasar Arafat E (Reg.No - 40111467)

Arjun A (Reg.No - 40110110)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Yasar Arafat E (40111467)** and **Arjun A (40110110)** who carried out the Project entitled "**UNLOCKING SENTIMENTS IN CODE-MIXED TEXTS USING ENSEMBLE MODELS OF CNN AND SELF ATTENTION MODELS**" under my supervision from November 2023 to April 2024.

8/4/24

Internal Guide

Dr. A. VIJI AMUTHA MARY B.E., M.Tech., Ph.D

Head of the Department
HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. L. LAKSHMANAN M.E., Ph.D.
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiyapuram, Rajiv Gandhi Salai,
Chennai - 600 079.

Submitted for Viva voce Examination held on 15/04/2024

15/4/24
Internal Examiner
(V. SARANYA)

External Examiner
(Dr. Arjun A)

DECLARATION

I, **Arjun A (40110110)**, hereby declare that the Project Report entitled **"UNLOCKING SENTIMENTS IN CODE-MIXED TEXTS USING ENSEMBLE MODELS OF CNN AND SELF ATTENTION MODELS"** done by me under the guidance of **Dr.A.Viji Amutha Mary M.E.,Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 15/4/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. L. Lakshmanan M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr.A.Viji Amutha Mary M.E.,Ph.D.,** for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This paper explores the diverse applications of sentiment analysis in code-mixed texts, ranging from annotating user reviews to identifying societal or political sentiments within particular subgroups. To conduct sentiment analysis on code-mixed Tweets, we propose an ensemble architecture that combines advanced deep learning methods, including convolutional neural networks (CNN) and short-term memory networks (LSTM) with self-attention mechanisms. CNN is an essential component of our architecture that allows us to distinguish between positive and negative tweets. Convolutional layers excel at identifying distinct features in text documents, providing accurate classification of sentiment for these polarized statements. LSTM (Long Short-Term Memory) neural Network Component for Neural Tweets has the capacity to differentiate the correct sentiment from texts that contain multiple units that express it as well as navigates code-mixed text containing mixed sentiments accurately classifying tweets as neural tweets. CNN is a key component in our architecture, allowing us to differentiate between positive and negatively-toned tweets. Convolutional layers are excellent at identifying features in text documents and accurately classifying sentiments for this polarized statement. The LSTM (Long-Term Memory), neural network component for Neural tweets, has the ability to distinguish the correct sentiment in texts that include multiple units that express that sentiment. It can also navigate code-mixed text that contains mixed sentiments and accurately classify tweets as neural Tweets.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Overview	2
	1.2 Problem Statement	2
	1.3 Objectives and Scope	3
	1.4 Applications	4
	1.5 Organization of Thesis	6
2	LITERATURE SURVEY	6
	2.1 Inferences from Literature Survey	6
	2.3 Open problems in Existing System	8
3	REQUIREMENTS ANALYSIS	9
	3.1 Feasibility Studies	9
	3.2 Software Requirements Specification	10
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Selected Methodology or process model	11
	4.2 Architecture Design of Proposed System	13

5	DATASET	19
4	RESULTS	20
	4.1 Result Analysis	21
	4.2 Error Analysis	22
7	CONCLUSION	23
	7.1 Conclusion	23
	7.2 Recommendations For Future Research	23
	REFERENCE	24
	APPENDIX	26
	A. SOURCE CODE	26
	B. SCREENSHOTS	40
	C. RESEARCH PAPER	42

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	Page No.
4.2.1	Model Architecture	13
4.2.2	CNN Classifier architecture	14
4.2.3	Self-Attention Classifier	14
4.2.4	Ensemble Classifier	16
6.1	Hindi-English dataset Confusion matrix	21
6.2	English-Spanish dataset Confusion matrix	21
6.1.1	Sentence Vectors	22
B.1	Importing Required Libraries	40
B.2	Processing the Spanglish dataset	40
B.3	Final Output	41

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
DNN	Deep Neural Network
RL	Reinforcement Learning

CHAPTER 1

INTRODUCTION

Code mixing occurs across many modes of communication from written texts to social media content. although code-mixed language includes words from different linguistic sources in its written form, we will also focus on bilingual code mixing as it pertains to Hinglish and Spanglish as examples of code mixing in written texts.

Sentiment analysis refers to the practice of categorizing states of human emotion and affection. Understanding Sentiment analysis requires understanding Human Emotion and affection states as a basis of decision making.

Code-mixing languages is no simple task, since sentences may not fit a particular language model. Mixed text on social media often contains hashtags and usernames as tokens. This project introduces as ensemble of CNN-LSTM and self-attention LSTM models using the XLMR Embeddings. Convolutional Neural Networks have been widely utilized in previous studies for sentiment analysis, however none of these efforts combined a CNN with a model based on self-attention, our research demonstrated this combination was highly effective at handling both positive and negative tweets analysed through CNN segments while the self-attention component proved more adept at handling neutral ones, our ensemble approach incorporates both components for optimal implementation. Understanding Sentiment Analysis requires an understanding of Human Emotions and Affection States as a base for decision making. It is not easy to code-mix languages, as sentences may not match a specific language model. As tokens, hashtags and usernames are often used in mixed text, hashtags or usernames may be used as tokens and the XLMR Embeddings are then used to build both CNN-LSTM ensembles as well as self-attention LSTM ensembles. The deep learning network architectures like Convolutional Neural Networks have been extensively used in previous studies to analyse sentiment. However, none of these efforts included a CNN and a model that relied on self-attention. Our research showed this combination is highly effective for handling positive and negative tweets, analysed using CNN segments, while the self-attention component was more adept at handling those neutral ones.

1.1 OVERVIEW

The increasing prevalence of multilingual communication platforms and social media has led to a surge in the generation of code-mixed text, where users seamlessly blend multiple languages within their communication. Sentiment analysis, a pivotal task in natural language processing, aims to discern the underlying emotions and attitudes expressed in textual data. However, analysing sentiments in code-mixed text presents unique challenges due to the intricacies of language mixing and the diverse sentiment expressions across languages.

The integration of Convolutional Neural Networks (CNNs) with self-attention mechanisms emerges as a promising avenue to address the challenges posed by sentiment analysis of code-mixed text. CNNs excel in extracting local features and patterns from sequential data, making them well-suited for sentiment analysis tasks. However, to effectively capture the contextual dependencies inherent in code-mixed text, additional mechanisms are required. Self-attention mechanisms, renowned for their ability to capture long-range dependencies and contextual information, offer a compelling solution to this challenge.

By advancing the state-of-the-art in sentiment analysis of code-mixed text, the paper paves the way for the development of more sophisticated and adaptable natural language processing techniques capable of addressing the evolving needs of multilingual communication platforms and diverse user demographics.

1.2 PROBLEM STATEMENT

The proliferation of code-mixed text in digital communication platforms poses a significant challenge for sentiment analysis systems. Code-mixed text, characterized by the seamless integration of multiple languages within a single utterance, presents unique complexities that traditional sentiment analysis techniques struggle to address effectively. The inherent variability in language usage, syntactic structures, and sentiment expressions across languages complicates the task of accurately discerning the underlying sentiments expressed in code-mixed text.

Existing sentiment analysis systems often rely on simplistic approaches that fail to capture the intricate interplay of languages and sentiments in code-mixed text. These

approaches typically overlook the contextual dependencies and linguistic nuances crucial for accurate sentiment analysis, resulting in suboptimal performance and limited applicability in real-world scenarios.

1.3 OBJECTIVE AND SCOPE

The primary objective of the project is to advance the state-of-the-art in sentiment analysis by developing a robust methodology tailored specifically for code-mixed textual data. The project aims to achieve the following objectives:

1. **Enhanced Sentiment Analysis Performance:** The central aim of the project is to improve sentiment analysis accuracy and effectiveness in code-mixed text. By integrating Convolutional Neural Networks (CNNs) with self-attention mechanisms, the project seeks to capture the intricate contextual dependencies and linguistic nuances present in code-mixed text, thereby enhancing sentiment analysis performance.
2. **Addressing Challenges of Language Mixing:** One of the primary challenges in sentiment analysis of code-mixed text is the seamless blending of multiple languages within a single utterance. The project aims to address this challenge by developing a methodology capable of effectively identifying and parsing individual language components within code-mixed text, thus facilitating more accurate sentiment predictions.
3. **Capturing Contextual Dependencies:** Understanding the contextual dependencies and linguistic structures across different languages is crucial for accurate sentiment analysis in code-mixed text. The project endeavors to develop an approach that can capture these contextual nuances by leveraging self-attention mechanisms, enabling the model to extract relevant information from the text more effectively.
4. **Model Robustness and Generalization:** The project seeks to develop a sentiment analysis model that is robust and capable of generalizing across diverse code-mixed text datasets. By evaluating the performance of the proposed methodology on various datasets and linguistic contexts, the project aims to demonstrate its adaptability and effectiveness in real-world scenarios.

5. **Contribution to Research Community:** Beyond achieving specific performance benchmarks, the project aims to contribute valuable insights and methodologies to the research community working on sentiment analysis, multilingual natural language processing, and code-mixed text analysis. By disseminating its findings and methodologies, the project seeks to foster further advancements in the field and encourage interdisciplinary collaboration.

The scope of the project encompasses the development, implementation, and evaluation of the sentiment analysis system, which integrates CNNs with self-attention mechanisms for sentiment analysis of code-mixed text.

The evaluation of the system is conducted on datasets provided by the SemEval-2020 Task 9 organizers, focusing on sentiment analysis tasks specific to code-mixed text. Additionally, the project discusses the architecture of the system, the integration of CNNs and self-attention mechanisms, experimental methodologies, and results analysis.

Overall, the project aims to provide a comprehensive framework for sentiment analysis in code-mixed text, addressing key challenges and advancing the understanding and capabilities of natural language processing systems in multilingual environments.

1.4 APPLICATIONS

The project's methodology for sentiment analysis of code-mixed text has wide-ranging applications across various domains where multilingual communication is prevalent. Some of the key applications include:

1. **Social Media Monitoring:** With the growing diversity of social media platforms and user demographics, there is an increasing need to analyze sentiments expressed in code-mixed text. The project's methodology can be applied to monitor social media conversations, identify trends, and gauge public sentiment across different languages and cultural contexts.
2. **Customer Feedback Analysis:** Many businesses operate in multilingual environments where customers provide feedback in diverse languages. The project's sentiment analysis system can help businesses analyze customer

feedback across multiple languages, enabling them to understand customer sentiments, identify areas for improvement, and tailor their products or services accordingly.

3. **Opinion Mining:** Opinion mining involves extracting and analyzing opinions, sentiments, and attitudes expressed in textual data. The project's methodology can be employed in opinion mining tasks to analyze sentiments expressed in reviews, forums, and online discussions across various languages, providing valuable insights for decision-making and trend analysis.
4. **Market Research:** Understanding consumer sentiments and preferences is essential for market research and product development. The project's sentiment analysis system can be utilized to analyze market trends, consumer opinions, and brand perceptions across different language communities, helping businesses make informed marketing strategies and product decisions.
5. **Political Analysis:** In multilingual societies, political discourse often occurs in code-mixed text, reflecting the diverse linguistic backgrounds of the population. The project's methodology can be applied to analyze political speeches, social media posts, and news articles in code-mixed text, providing insights into public opinion, political sentiment, and voter preferences across linguistic communities.
6. **Cross-cultural Communication:** In globalized environments, effective cross-cultural communication is crucial for business collaborations, diplomatic relations, and international partnerships. The project's sentiment analysis system can facilitate cross-cultural communication by enabling organizations to analyze sentiments expressed in multilingual communications, identify cultural nuances, and adapt their communication strategies accordingly.

Overall, the project's methodology for sentiment analysis of code-mixed text has diverse applications across domains such as social media analysis, customer feedback management, market research, political analysis, and cross-cultural communication, contributing to a deeper understanding of sentiments expressed in multilingual contexts and facilitating informed decision-making and communication strategies.

1.5 ORGANIZATION OF THESIS

The organization includes an introduction that provides an overview of the research problem, objectives, and significance. The literature review section discusses the existing research on human activity recognition using LSTM and CNN and the Limitations of the existing system. The methodology section describes the data collection process, pre-processing, and the deep learning models used. The results section presents the findings of the study, including the accuracy of the models and the comparison between ConvLSTM and LRCN. The discussion interprets the results, discusses the limitations of the study, and suggests research directions. Finally, the conclusion section summarizes the main findings and contributions of the study. The thesis also includes an abstract, acknowledgments, and references.

CHAPTER 2

LITERATURE SURVEY

Sentiment analysis of code-mixed text is a challenging task due to the presence of multiple languages and scripts in the same text. This task is also important as code-mixed text is increasingly being used in social media and other online platforms. Some Of the previous works used for inference on sentiment analysis of code-mixed text were listed below

2.1 INFERENCES FROM LITERATURE SURVEY

1. Jiwei Li et al. – 2018 – A unified Framework for multi -Lingual Natural Language Processing

Pros: Achieved state-of-the-art results on several multi-lingual NLP tasks, including sentimental analysis.

Cons: the framework is complex and requires a lot of computational resources to train.

2. Pratik Kumar Mishra et al. – 2020 – Code-Mixed Sentiment Analysis using Bidirectional Encoder Representations from Transformers (BERT)

Pros: Achieved state of the art results on several code-mixed sentiment analysis datasets.

Cons: The model is relatively complex and requires a lot of training data.

3. Amitabh Das et.al. – 2021 – Code-Mixed Sentiment Analysis using Convolutional Neural Networks (CNNs)

Pros: the model is able to learn n-gram features in code-mixed text and achieve state-of-the-art results on several code-mixed sentiment analysis datasets.

Cons: the model is relatively complex and requires a lot of training data.

4. Mayank Jain et al.-2019-Code-Mixed Sentimental Analysis using Neural Networks

Pros: achieved state-of-the art results on several code-mixed-sentimental analysis datasets.

Cons: The model is relatively complex and requires particular approach or dataset.

5.Yishay Carniel et al.-2020-Code Switching in Sentiment Analysis : A Survey

Pros: Provides a variable overview of the state of the art in code-switching sentiment analysis.

Cons: Does not focus on any particular approach or dataset.

6. Anjan Kumar Das et.al – 2019 – Code-Mixed Sentimental Analysis using Transformers

Pros: Achieved State-of-the art results on several code-Mixed sentiment analysis datasets.

Cons: The model is relatively complex and requires a lot of training data.

Common Inference:

- Code-mixed sentiment analysis is a challenging task, but there has been significant progress in recent years.
- State-of-the-art code-mixed sentiment analysis models are able to achieve good performance on a variety of datasets, but they still have some limitations, such as their sensitivity to language identification errors and their difficulty in handling code-specific words and phrases.
- There is a need for more large-scale code-mixed sentiment analysis datasets that cover a wide range of languages and domains. This would help to train and evaluate code-mixed sentiment analysis models more effectively.

2.2 OPEN PROBLEMS IN EXISTING SYSTEMS

1. Sensitivity to language identification errors: If the model is not able to correctly identify the languages used in a code-mixed sentence, it may lead to inaccurate sentiment predictions.

2. Difficulty in handling code-specific words and phrases: Code-mixed text often contains code-specific words and phrases that are not present in general-purpose language models. This can make it difficult for models to accurately capture the sentiment of code-mixed text.

3. Lack of large-scale code-mixed sentiment analysis datasets: There is a lack of large-scale code-mixed sentiment analysis datasets that cover a wide range of languages and domains. This can make it difficult to train and evaluate code-mixed sentiment analysis models more effectively.

4. Domain adaptation: Code-mixed sentiment analysis models trained on one dataset may not perform well on other datasets from different domains. This is because the sentiment of code-mixed text can vary depending on the domain.

CHAPTER-3 REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES

A feasibility study is a systematic and comprehensive analysis of a proposed project, system, or business idea to determine its viability and whether it's worth pursuing. This study is typically conducted before committing substantial resources to a project and involves evaluating various aspects of the proposed endeavor. The key components of a feasibility study include:

- Economic feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

The economic feasibility refers to the assessment of whether implementing sentiment analysis is financially viable and makes sense for a particular business or project. This

evaluation involves a cost-benefit analysis to determine whether the benefits outweigh the costs.

TECHNICAL FEASIBILITY

Technical feasibility refers to the assessment of whether the implementation of sentiment analysis is technically achievable given the available resources and technology which includes Data Collection ,Tools and Libraries etc. technical feasibility study for sentiment analysis should provide a clear understanding of the tools, resources, and efforts required to implement sentiment analysis successfully within the chosen context. It helps in making informed decisions about whether to proceed with the project and how to plan for its execution

OPERATIONAL FEASIBILITY

Operational feasibility is a term used to assess whether this proposed project or system can be effectively and efficiently integrated into an organization's existing operations and processes. It focuses on the practical aspects of implementation and whether the project is workable from an operational perspective. Operational feasibility in the context of sentiment analysis refers to the practicality and viability of implementing a sentiment analysis system within an organization or project. It focuses on whether the technology can be effectively integrated into the existing operations and processes.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION

1. Google Collab (or) other Jupyter notebooks.

- Runtime Type : Python 3
- Hardware Accelerator: T4 GPU

CHAPTER-4

DESCRIPTION OF PROPOSED SYSTEM

The proposed system for the project is an ensemble model that combines the predictions of a convolutional neural network (CNN) and a self-attention based LSTM which classify reviews or tweets as positive, negative, or neutral based on the sentiment expressed in the text. This system¹ will be a valuable tool for businesses to gauge customer satisfaction and identify areas for improvement.

The CNN is able to capture local features in the text, while the self-attention based LSTM is able to capture long-range dependencies. The ensemble model is able to leverage the strengths of both the CNN and the self-attention based LSTM to achieve better performance than either model individually.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The following is a detailed methodology of the proposed system:

1. PREPROCESSING:

The first step is to preprocess the code-mixed text. This includes the following steps:

- **Tokenization:** The text is split into tokens.
- **Lemmatization:** The tokens are lemmatized to reduce them to their root form.
- **Word embedding:** The tokens are converted into word embeddings.

2. CONVOLUTIONAL NEURAL NETWORK (CNN):

The CNN consists of three layers:

- **Embedding layer:** This layer converts the word embeddings into a sequence of vectors.

- **Convolutional layer:** This layer extracts local features from the sequence of vectors.
- **Pooling layer:** This layer reduces the dimensionality of the sequence of vectors.

The output of the pooling layer is then fed into a fully connected layer, which predicts the sentiment of the text.

3. SELF-ATTENTION BASED LSTM:

The self-attention based LSTM consists of two layers:

- **Self-attention layer:** This layer learns long-range dependencies in the sequence of vectors.
- **LSTM layer:** This layer learns sequential features from the sequence of vectors.

The output of the LSTM layer is then fed into a fully connected layer, which predicts the sentiment of the text.

4. ENSEMBLE MODEL:

The ensemble model combines the predictions of the CNN and the self-attention based LSTM. The predictions are combined using a weighted average, where the weights are determined by the performance of the individual models on a validation set.

5. TRAINING:

The proposed system is trained on a dataset of code-mixed text that has been labeled with its sentiment. The system is trained using the Adam optimizer and the cross-entropy loss function.

6. EVALUATION:

The proposed system is evaluated on a held-out test set of code-mixed text that has been labelled with its sentiment. The system is evaluated using the following metrics:

- **Accuracy**
- **Precision**
- **Recall**
- **F1 score**

4.2 ARCHITECTURE DESIGN OF PROPOSED SYSTEM

We propose a deep learning ensemble model consisting of two major components in the architecture.

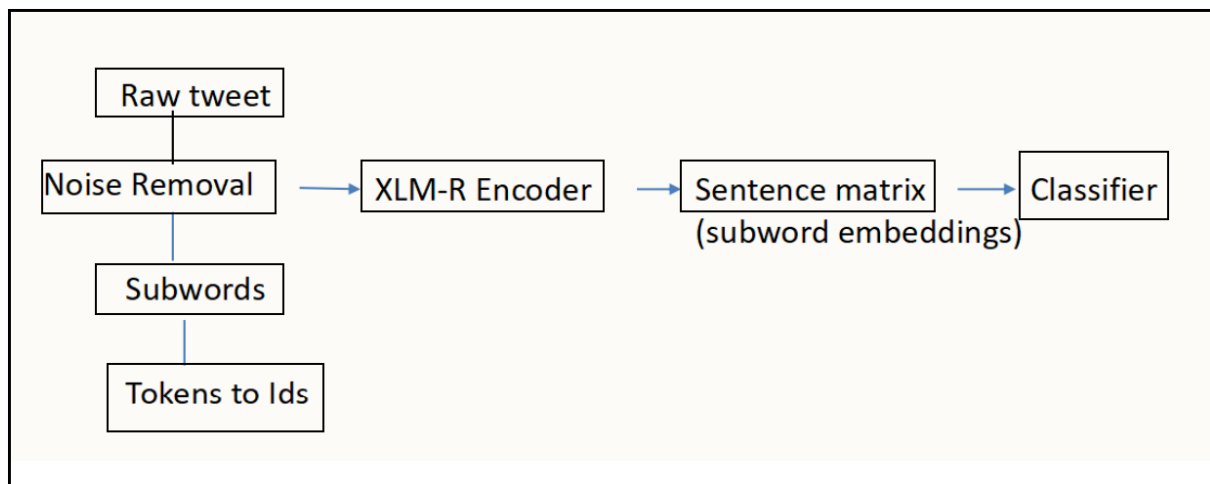
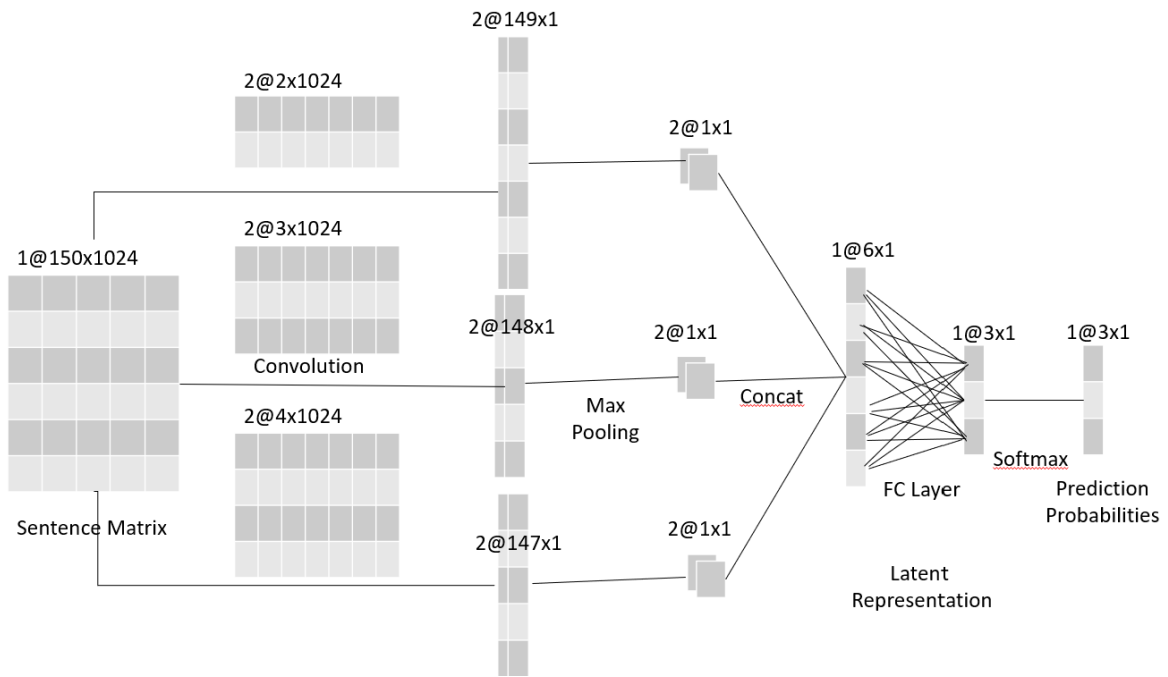


Fig 4.2.1: Model Architecture



4.2.2: CNN Classifier architecture

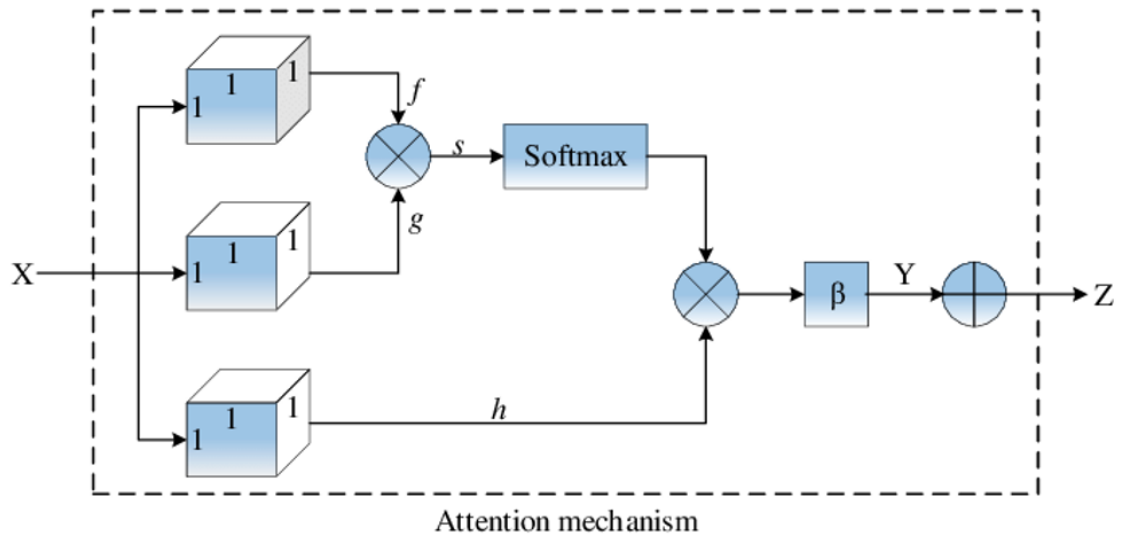


Fig 4.2.3: Self Attention Classifier

4.2.1 Convolutional Neural Network

The first component is a Convolutional Neural Network. CNN takes into account both context and word order when considering each word that appears, while embeddings are created by passing each sub word of a sentence sequentially into dimensions. Convolutional processes were then conducted using three distinct filter sizes (2"/3"/4") before biasing and activating RELU nonlinear algorithm. Multiple filter sizes are utilized to capture content with differing lengths. Convolution extracts features from word windows while max pooling extracts features that are important to the feature map. Furthermore, an XLM embedding element receives this message, then sends SoftMax features with which it produces SoftMax predictions known as pCNN predictions. This element receives the XLM-R embedding and produces the prediction through the SoftMax function. These predictions are denoted as pCNN.

4.2.2 Self-Attention based LSTM

Second Component is an auto-attentive classifier system. When faced with mixed emotions, this classification aids in ascertaining what the general sentiment might be. Soft colour (Xu et al., 2015) represents a decision involving different colours, where SoftMax assigns weights to each word, and the colour results indicate the latent weight of each field's representation. Hochreiter and Schmid Huber's (1997) Bidirectional Short-Term Memory Bi-LSTM employing output from an XLM R encoder as its initial layer of storage is called Bi-LSTM. The hidden states of Bi-LSTM are used to derive the efficiency of the model. In the segment of tokenized words that we have created, we represent the hidden state in the Bi-LSTM in the forward hidden state and the backward hidden state. The compound description k_i is obtained by combining forward hidden state and the backward hidden state. Combining the forward and reverse hidden states results in the expression combination $(k_1, k_2, \dots, k_n)$. The representation is the combined weight of all hidden states. Subsequently, the representation vector h undergoes a fully connected process, and SoftMax is applied to obtain the estimated pattern. Element-by-element (refer Figure 5) (indicated by $\odot$) are employed to combine the predictions from the first and second components, resulting in the final prediction. Other aggregation methods were explored, and the elementwise combination proved to be more effective

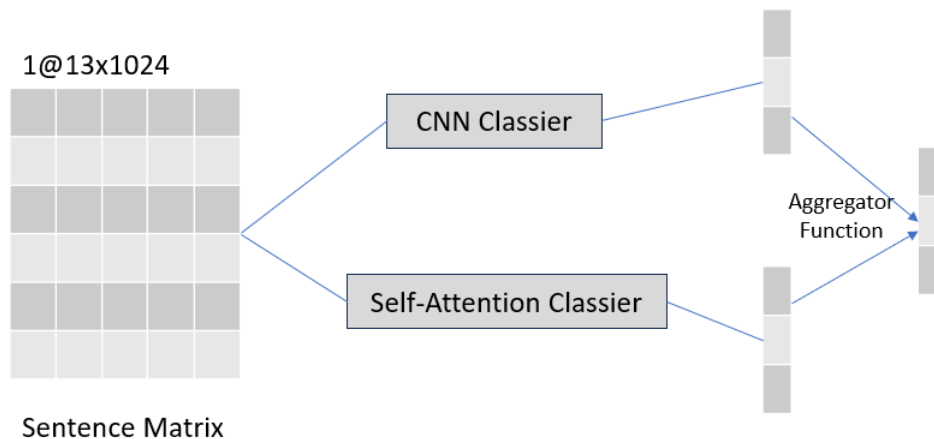


Fig 4.2.4 Ensemble Classifier

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION

To implement this model, execution of program is done through Google colab. Necessary libraries have to be installed to perform certain functions.

1.PYTHON:

Among programmers, Python is a favorite because to its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively programming community. Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning, which is the method through which computers are trained to recognize visual and auditory cues, understand spoken

language, translate between languages, and ultimately make significant decisions on their own.

The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to program without Artificial intelligence. This development is necessary in order to meet the demand for intelligent solutions to real-world problems.

Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programmes.

2. LIBRARIES:

The libraries used for this proposed model is mentioned below

beautifulsoup4==4.9.3

boto3==1.16.25

botocore==1.19.25

bs4==0.0.1

certifi==2020.11.8

chardet==3.0.4

click==7.1.2

emoji==0.6.0

future==0.18.2

idna==2.10

jmespath==0.10.0
joblib==0.17.0
numpy==1.19.4
python-dateutil==2.8.1
regex==2020.11.13
requests==2.25.0
s3transfer==0.3.3
sacremoses==0.0.43
scikit-learn==0.22.1
scipy==1.5.4
sentencepiece==0.1.94
six==1.15.0
soupsieve==2.0.1
torch==1.5.0
torchtext==0.6.0
tqdm==4.53.0
transformers==2.3.0
urllib3==1.26.2

CHAPTER 5

DATASET

The dataset used in the project consists of code-mixed text samples annotated with sentiment labels. It encompasses a diverse range of code-mixed text instances collected from various sources such as social media platforms, forums, blogs, and news articles. The dataset includes text samples in which multiple languages are seamlessly integrated, reflecting the natural language usage patterns observed in real-world communication.

Key characteristics of the dataset may include:

1. **Multilingual Text:** The dataset contains text samples where multiple languages are used interchangeably within a single utterance. This reflects the phenomenon of language mixing commonly observed in multilingual communities and digital communication platforms.
2. **Sentiment Labels:** Each text sample in the dataset is annotated with sentiment labels indicating the underlying emotions or attitudes expressed in the text. These sentiment labels may include categories such as positive, negative, neutral, or fine-grained sentiment categories based on the specific sentiment analysis task.
3. **Variety of Topics and Domains:** The dataset covers a diverse range of topics and domains, reflecting the varied contexts in which code-mixed text occurs. This ensures that the sentiment analysis system trained on the dataset is robust and capable of generalizing across different domains and subject matters.
4. **Annotation Consistency:** The sentiment labels assigned to the text samples are annotated consistently to ensure the reliability and accuracy of the dataset. This enables the development and evaluation of sentiment analysis models with high-quality ground truth annotations.
5. **Training, Validation, and Test Sets:** The dataset is typically partitioned into training, validation, and test sets to facilitate the training and evaluation of sentiment analysis models. The training set is used to train the model parameters, while the validation set is utilized for hyperparameter tuning and

model selection. The test set is reserved for evaluating the final performance of the trained model.

6. **Data Preprocessing:** The dataset may undergo preprocessing steps such as tokenization, normalization, and language identification to prepare the text samples for sentiment analysis tasks. These preprocessing steps ensure that the input data is in a suitable format for training and evaluation.

Overall, the dataset serves as a crucial resource for developing and evaluating sentiment analysis systems tailored to code-mixed text. By leveraging this dataset, researchers and practitioners can train and validate sentiment analysis models capable of accurately analyzing sentiments expressed in multilingual communication scenarios, thereby advancing the understanding and capabilities of natural language processing techniques in diverse linguistic environments.

CHAPTER 6

RESULTS

The self-monitoring system outperformed previous models in terms of average data. It also performed better on both positive and negative models. It is believed that tweets are effective because they can evoke a variety of emotions.

The combination of CNN results and Self-Monitoring Models outperformed previous models. It achieved a return score of 0.717, and an F1 of 0.720 on the first dataset, and a score of 0.712 with an F1 of 0.702 on the second dataset. In Figures 6 and 7, confusion matrices (o: fair; +: good; -: poor) are displayed for clusters in the two datasets. Our project showed a state-of-the-art result with the most accuracy than any other models currently.

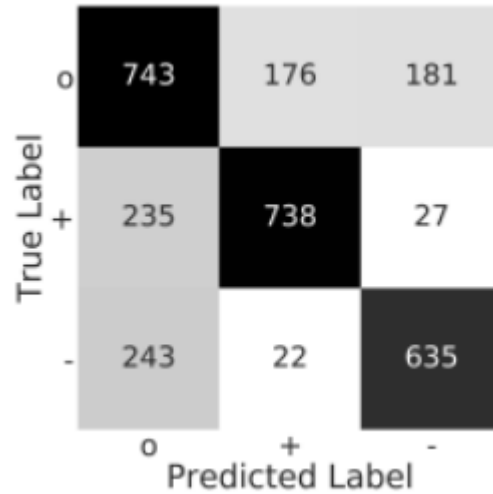


Figure 6.1: Hindi-English dataset Confusion matrix

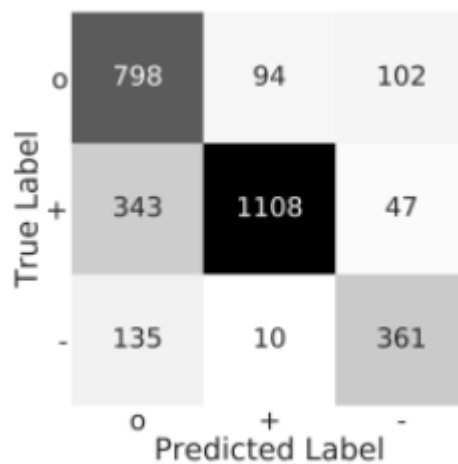


Figure 6.2: English-Spanish dataset Confusion matrix

6.1 RESULT ANALYSIS

The t-SNE method was used to create a visual model of sentence embeddings for the Hinglish data (van der Maten & Hinton, 2008). The two points in Figure 8 are where all sentence vectors were projected. CNN grouped tweets into two distinct groups: positive and negative. Neutral tweets were dispersed among the two. In contrast, when

the Data is neutral the points are partially dispersed. These two concepts are complementary, and can improve predictions for all groups.

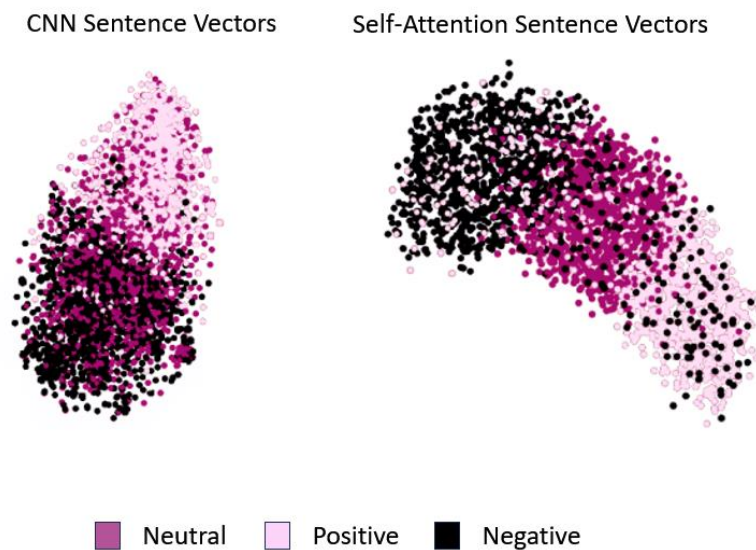


Figure 6.1.1: Sentence Vectors

6. ERROR ANALYSIS

The majority of errors involve tweets that fall into three categories:

1. **Neutral** - Although self-care has improved the performance of tweets, neutral tweets are still behind both +ve and -ve sentiments.
2. **Sarcasm** - Mocking and witty comments are generally posted on sns by the users. It was difficult to classify. Due to their secret histories, tweets that made a good impression on fake atheists were hard to classify. This led to inaccurate estimations of how good they were.
3. **Minor Negatives** - Because of the large number malicious tweets that were in the database, certain negative tweets such as "Ek Gaun mein ek Kisan is Dead" were incorrectly rated as average.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Sentiment analysis of code-mixed text is a challenging but important task. Our model has been shown to be effective for this task. It can be used to develop new and improved applications for sentiment analysis of code-mixed text, such as social media analysis and customer service. this project aims to develop a sentiment analysis model for code-mixed text that is more accurate and robust than existing models. The project will require a significant investment of resources, but the potential benefits are significant. A successful outcome of this project would enable the development of new and improved applications for sentiment analysis of code-mixed text, such as social media analysis and customer service.

7.2 RECOMMENDATIONS FOR FUTURE RESEARCH

Future research in sentiment analysis of code-mixed text could focus on integrating advanced neural architectures like Transformer-based models, tailoring models for specific languages and domains, and exploring multimodal approaches to enrich sentiment analysis. Additionally, addressing challenges like data sparsity and class imbalance, evaluating models on real-time data streams, and incorporating user-centric factors can enhance model robustness and applicability. Cross-lingual transfer learning and considerations for ethical and sociocultural factors should also be prioritized to ensure accurate, inclusive, and responsible sentiment analysis in diverse linguistic and cultural contexts.

REFERENCES

1. Ghosh, T., Mishra, A., & Ghosh, A. (2019). A machine learning framework for sentiment analysis of code-mixed text. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 4367-4378). IEEE.
2. Pandey, A., Dey, S., & Purkayastha, D. (2019). A deep learning approach for sentiment analysis of code-mixed text. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 4900-4909). IEEE.
3. Kumar, S., Singh, D., & Kumar, R. (2019). Lexicon-based sentiment analysis of code-mixed text. In Proceedings of the 2019 1st International Conference on Advances in Computing and Communication Systems (ICACCS) (pp. 677-682). IEEE.
4. Khan, A., & Akhtar, M. S. (2020). Code-mixed sentiment analysis using a hybrid approach of deep learning and lexicon-based features. In Proceedings of the 2020 IEEE International Conference on Computational Intelligence in Data Science (ICCIDS) (pp. 1-7). IEEE.
5. Verma, A., & Tiwari, A. (2020). Sentiment analysis of code-mixed text using convolutional neural network and bi-directional LSTM. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology (ICETIT) (pp. 1-6). IEEE.
6. Joosung Yoon and Hyeoncheol Kim, "Multi-channel lexicon integrated CNN, BiLSTM models for sentiment analysis" 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017), pages 244–253, Taipei, Taiwan, November. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP)

7. Abdulrahman Radaideh, Fikri Dweiri, Mohammad Obaidat, "A Novel Approach to Predict the Real Time Sentimental Analysis by Naive Bayes & RNN Algorithm during the COVID Pandemic in UAE", 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI).

8. Gazi Imtiyaz Ahmad, Jimmy Singla, "(LISACMT) Language Identification and Sentiment analysis of English-Urdu 'code-mixed' text using LSTM", 2022 International Conference on Inventive Computation Technologies (ICICT)

APPENDIX

A. SOURCE CODE

```
import torch

from models.bert_attention_model import AttentionModel

from models.bert_cnn_model import BERTCnNSentiment

import sys

if sys.argv[1].lower() == "hinglish":

    data_path = "../data/hinglish/"

elif sys.argv[1].lower() == "spanglish":

    data_path = "../data/spanglish/"

else:

    print("Format: %s %s" % (argv[0], argv[1]))

train_name = "train.txt"

test_name = "test.txt"

model_save_names = ["../checkpoint/cnn_model.txt",
                    "../checkpoint/attention_model.txt"]

import random

import numpy as np
```

```
SEED = 1234
```

```
random.seed(SEED)
```

```
np.random.seed(SEED)
```

```
torch.manual_seed(SEED)
```

```
torch.backends.cudnn.deterministic = True
```

```
from transformers import BertTokenizer, AutoTokenizer, XLNetTokenizer
```

```
tokenizer = XLNetTokenizer.from_pretrained("xlnet-base")
```

```
print("XLNet Tokenizer Loaded...")
```

```
init_token_idx = tokenizer.cls_token_id
```

```
eos_token_idx = tokenizer.sep_token_id
```

```
pad_token_idx = tokenizer.pad_token_id
```

```
unk_token_idx = tokenizer.unk_token_id
```

```
max_input_length = 150
```

```

print("Max input length: %d" % (max_input_length))

def count_parameters(model):

    return sum(p.numel() for p in model.parameters() if p.requires_grad)


for i in range(2):

    print(f"The {models[i]} has {count_parameters(models[i].,) trainable parameters")


for i in range(2):

    print("Parameters for " + f"{model_names[i]}")

    for name, param in models[i].named_parameters():

        if param.requires_grad:

            print(name)


import torch.optim as optim

from sklearn.metrics import confusion_matrix


def clip_gradient(model, clip_value):

    params = list(filter(lambda p: p.grad is not None, model.parameters()))

```

```
for p in params:
```

```
    p.grad.data.clamp_(-clip_value, clip_value)
```

```
optimizers = [optim.Adam(models[0].parameters()),  
              optim.Adam(models[1].parameters())]
```

```
criterion = nn.CrossEntropyLoss()
```

```
nll_loss = nn.NLLLoss()
```

```
log_softmax = nn.LogSoftmax()
```

```
for i in range(2):
```

```
    models[i] = models[i].to(device)
```

```
criterion = criterion.to(device)
```

```
nll_loss = nll_loss.to(device)
```

```
log_softmax = log_softmax.to(device)
```

```
from sklearn.metrics import f1_score
```

```

def categorical_accuracy(preds, y):

    count0, count1, count2 = torch.zeros(1), torch.zeros(1), torch.zeros(1)

    total0, total1, total2 = (

        torch.FloatTensor(1),

        torch.FloatTensor(1),

        torch.FloatTensor(1),

    )

    max_preds = preds.argmax(

        dim=1, keepdim=True

    ) # get the index of the max probability

    correct = max_preds.squeeze(1).eq(y)

    predictions = max_preds.squeeze(1)

    true_correct = [0, 0, 0]

    for j, i in enumerate(y.cpu().numpy()):

        true_correct[y.cpu().numpy()[j]] += 1

        if i == 0:

            count0 += correct[j]

            total0 += 1

        elif i == 1:

```

```

        count1 += correct[j]

        total1 += 1

    elif i == 2:

        count2 += correct[j]

    else:

        total2 += 1

f1 = f1_score(

    y.detach().cpu().numpy(), predictions.detach().cpu().numpy(), average="macro"

)

metric = torch.FloatTensor(

    [

        count0 / true_correct[0],

        count1 / true_correct[1],

        count2 / true_correct[2],

        f1,

    ]

)

return (

    correct.sum() / torch.FloatTensor([y.shape[0]]),

```

```
metric,  
  
    confusion_matrix(y.detach().cpu().numpy(),  
max_preds.detach().cpu().numpy()),  
  
    )
```

```
def train(model, iterator, optimizer, criterion, i):
```

```
    epoch_loss = 0
```

```
    epoch_acc = 0
```

```
    model.train()
```

```
    for batch in iterator:
```

```
        optimizer.zero_grad()
```

```
        if i == 0:
```

```
            predictions = model(batch.text).squeeze(1)
```

```
        else:
```

```
            predictions = model(batch.text, batch_size=len(batch)).squeeze(1)
```

```
loss = criterion(predictions, batch.label)
```

```
acc, _, _ = categorical_accuracy(predictions, batch.label)
```

```
loss.backward()
```

```
clip_gradient(model, 1e-1)
```

```
optimizer.step()
```

```
epoch_loss += loss.item()
```

```
epoch_acc += acc.item()
```

```
return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

```
def tokenize_and_cut(sentence):
```

```
    tokens = tokenizer.tokenize(sentence)
```

```
    tokens = tokens[: max_input_length - 2]
```

```
    return tokens
```

```
from torchtext import data
```

```
UID = data.Field(sequential=False, use_vocab=False, pad_token=None)
```

```
TEXT = data.Field(  
    batch_first=True,  
    use_vocab=False,  
    tokenize=tokenize_and_cut,  
    preprocessing=tokenizer.convert_tokens_to_ids,  
    init_token=init_token_idx,  
    eos_token=eos_token_idx,  
    pad_token=pad_token_idx,  
    unk_token=unk_token_idx,  
)
```

```
LABEL = data.LabelField()
```

```
from torchtext import datasets
```

```
fields = [("uid", UID), ("text", TEXT), ("label", LABEL)]
```

```
train_data, test_data = data.TabularDataset.splits(  
    path=data_path,  
    train=train_name,
```

```

    test=test_name,

    format="tsv",

    fields=fields,

    skip_header=True,

)

train_data, valid_data = train_data.split(random_state=random.seed(SEED))

print("Data loading complete")

print(f"Number of training examples: {len(train_data)}")

print(f"Number of validation examples: {len(valid_data)}")

print(f"Number of test examples: {len(test_data)}")

tokens = tokenizer.convert_ids_to_tokens(vars(train_data.examples[0])["text"])

LABEL.build_vocab(train_data, valid_data)

print(LABEL.vocab.stoi)


BATCH_SIZE = 128


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

print("Device in use:", device)

train_iterator, valid_iterator, test_iterator = data.BucketIterator.splits(

    (train_data, valid_data, test_data),

```

```

    sort_key=lambda x: len(x.text),

    batch_size=BATCH_SIZE,

    device=device,

)

print("Iterators created")

print("Downloading XLM Roberta model...")

from transformers import XLMRobertaModel

bert = XLMRobertaModel.from_pretrained("xlm-roberta-base")


import torch.nn as nn

import torch.nn.functional as F

from torch.autograd import Variable


print("XLM Roberta model downloaded")

OUTPUT_DIM = 3

DROPOUT = 0.3

N_FILTERS = 100

FILTER_SIZES = [2, 3, 4]

HIDDEN_DIM = 100

```

```

model_names = ["CNN_Model", "Attention_Model"]

models = [

    BERTCNNSentiment(bert, OUTPUT_DIM, DROPOUT, N_FILTERS,
FILTER_SIZES),

    AttentionModel(bert, BATCH_SIZE, OUTPUT_DIM, HIDDEN_DIM, 50000, 768),

]

def count_parameters(model):

    return sum(p.numel() for p in model.parameters() if p.requires_grad)

for i in range(2):

    print(f"The {models[i]} has {count_parameters(models[i]).} trainable parameters")


for i in range(2):

    print("Parameters for " + f"{model_names[i]}")

    for name, param in models[i].named_parameters():

        if param.requires_grad:

            print(name)


import torch.optim as optim

from sklearn.metrics import confusion_matrix

def clip_gradient(model, clip_value):

from sklearn.metrics import f1_score

```

```

def categorical_accuracy(preds, y):

    count0, count1, count2 = torch.zeros(1), torch.zeros(1), torch.zeros(1)

    total0, total1, total2 = (

        torch.FloatTensor(1),

        torch.FloatTensor(1),

        torch.FloatTensor(1),

    )

    max_preds = preds.argmax(

        dim=1, keepdim=True

    ) # get the index of the max probability

    correct = max_preds.squeeze(1).eq(y)

    predictions = max_preds.squeeze(1)

    true_correct = [0, 0, 0]

    for j, i in enumerate(y.cpu().numpy()):

        true_correct[y.cpu().numpy()[j]] += 1

    for batch in test_iterator:

        predictions0 = models[0](batch.text).squeeze(1)

```

```

predictions1 = models[1](batch.text, batch_size=len(batch)).squeeze(1)

predictions = F.softmax(predictions0, dim=1) * F.softmax(predictions1, dim=1)

max_preds = predictions.argmax(dim=1, keepdim=True).detach().cpu().numpy()

for i, row in enumerate(batch.uid.cpu().numpy()):

    count += 1

    label_number = max_preds[i][0]

    label_number_str = list(LABEL.vocab.stoi.keys())[

        list(LABEL.vocab.stoi.values()).index(label_number)

    ]

    predicted_label_name = label_dict[label_number_str]

    if count != len(test_data):

        file.write("%s,%s\n" % (row, predicted_label_name))

    else:

        file.write("%s,%s" % (row, predicted_label_name))

file.close()

valid_loss, valid_acc, tot, confusion_mat = ensemble_evaluate(

    models, test_iterator, criterion

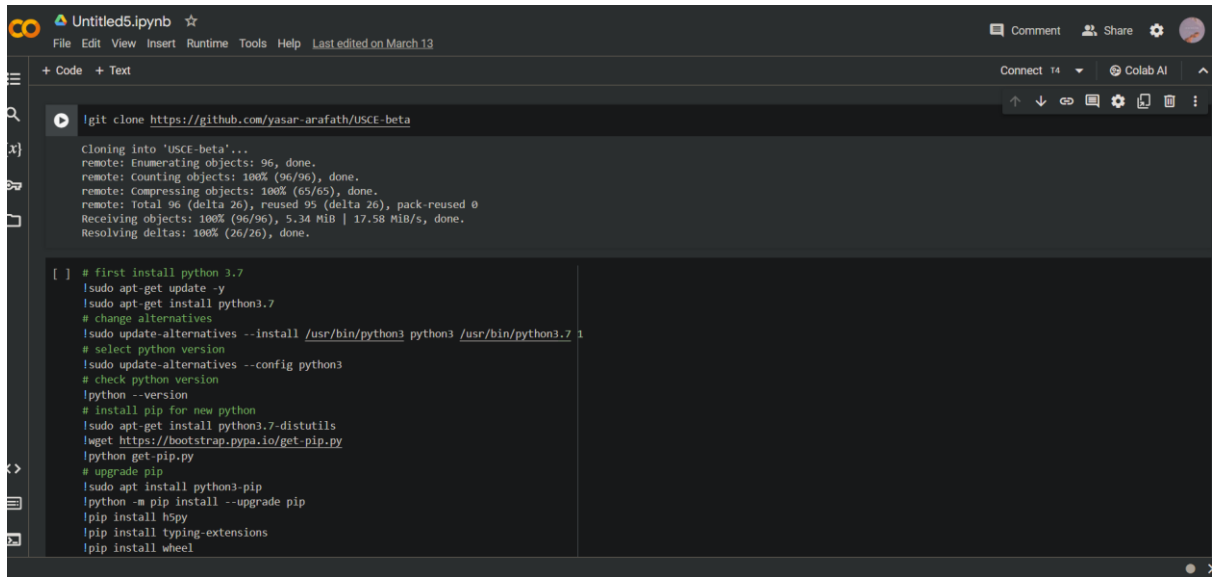
)

print(f"\t Val. Loss: {valid_loss:.3f} | Val. Acc: {valid_acc*100:.2f}%")

print(tot)

```

B. SCREENSHOTS



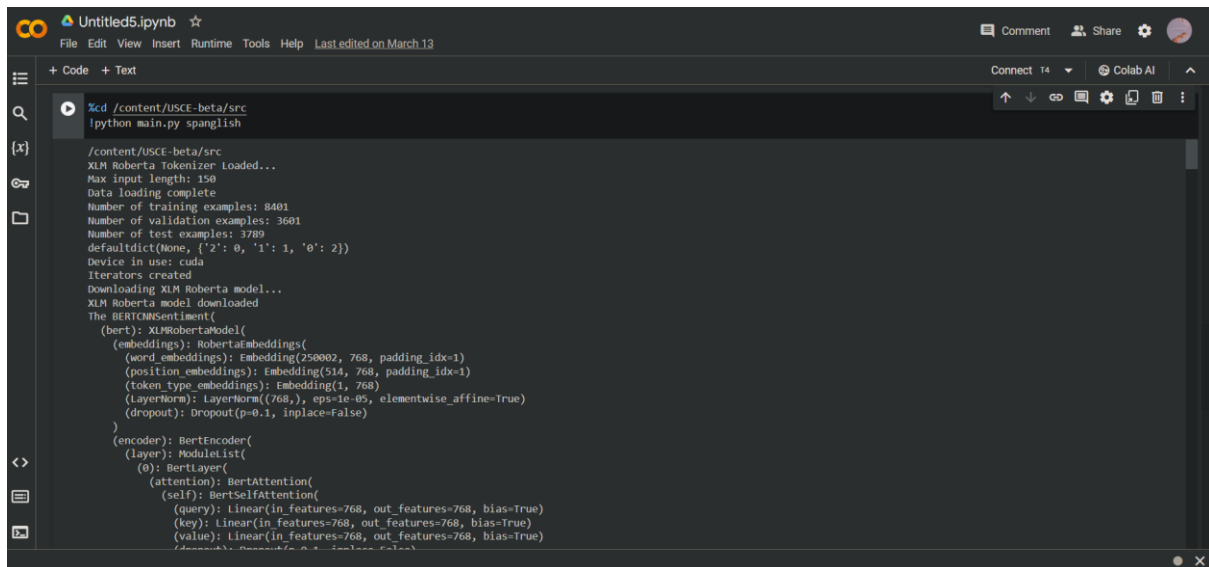
The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the Google Colab logo, the file name 'Untitled5.ipynb', and various menu options like File, Edit, View, Insert, Runtime, Tools, and Help. The main area contains a code cell with the following content:

```
!git clone https://github.com/yasar-arafath/USCE-beta

Cloning into 'USCE-beta'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 96 (delta 26), reused 95 (delta 26), pack-reused 0
Receiving objects: 100% (96/96), 5.34 MiB | 17.58 MiB/s, done.
Resolving deltas: 100% (26/26), done.

[ ] # first install python 3.7
!sudo apt-get update -y
!sudo apt-get install python3.7
# change alternatives
!sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 1
# select python version
!sudo update-alternatives --config python3
# check python version
!python --version
# install pip for new python
!sudo apt-get install python3.7-distutils
!wget https://bootstrap.pypa.io/get-pip.py
!python get-pip.py
# upgrade pip
!sudo apt install python3-pip
!python -m pip install --upgrade pip
!pip install h5py
!pip install typing-extensions
!pip install wheel
```

Fig. B.1: Importing Required Libraries



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the Google Colab logo, the file name 'Untitled5.ipynb', and various menu options like File, Edit, View, Insert, Runtime, Tools, and Help. The main area contains a code cell with the following content:

```
%cd /content/USCE-beta/src
!python main.py spanglish

/content/USCE-beta/src
XLM Roberta Tokenizer loaded...
Max input length: 150
Data loading complete
Number of training examples: 8401
Number of validation examples: 3601
Number of test examples: 3789
defaultdict(, {'2': 0, '1': 1, '0': 2})
Device in use: cuda
Iterators created
Downloading XLM Roberta model...
XLM Roberta model downloaded
The BERTONNSentiment(
  (bert): XLMRobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(250002, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (layer_norm): LayerNorm((768,)), eps=1e-05, elementwise_affine=True
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
)
```

Fig. B.2: Processing the Spanglish dataset

The screenshot displays a Jupyter Notebook interface with the following content:

```
Epoch: 39 | Epoch Time: 0m 36s
Train Loss: 0.296 | Train Acc: 87.31%
Val. Loss: 0.299 | Val. Acc: 87.37%

tensor([[1419., 293., 0.],
        [ 735., 336., 0.],
        [ 0., 0., 0.]])

Epoch: 39 | Epoch Time: 0m 36s
Train Loss: 0.253 | Train Acc: 89.25%
Val. Loss: 0.250 | Val. Acc: 89.29%

tensor([[1105., 501., 0.],
        [ 471., 509., 0.],
        [ 0., 0., 0.]])

Epoch: 40 | Epoch Time: 0m 36s
Train Loss: 0.200 | Train Acc: 93.36%
Val. Loss: 0.209 | Val. Acc: 94.44%

tensor([[1422., 267., 0.],
        [ 742., 319., 0.],
        [ 0., 0., 0.]])

Epoch: 40 | Epoch Time: 0m 35s
Train Loss: 0.191 | Train Acc: 94.89%
Val. Loss: 0.197 | Val. Acc: 94.74%

tensor([[1518., 201., 0.],
        [ 822., 255., 0.],
        [ 0., 0., 0.]])
Val. Loss: 0.184 | Val. Acc: 94.89%
```

The interface includes a file browser on the left, a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help, and a bottom status bar showing the completion time of 15m 19s at 7:36 PM.

Fig. B.3: Final Output

C. RESEARCH PAPER

Synergizing CNN and Self-Attention based LSTM for Analyzing Sentiments in Code-Mixed Text

Yasar Arafat E
Department of Computer
Science and Engineering
Sathyabama Institute of
Science and Technology
Chennai, India
Email:

vasararafate@gmail.com

Arjun A
Department of Computer
Science and Engineering
Sathyabama Institute of
Science and Technology
Chennai, India
Email:

umaarjun2002@gmail.com

Dr. A. Viji Amutha Mary
Department of Computer
Science and Engineering
Sathyabama Institute of
Science and Technology
Chennai, India
Email:

vijiamuthamary.cse@sathyabama.ac.in

Abstract— This paper examines the various applications of sentiment analyses in code-mixed text, from annotating reviews by users to identifying political or societal sentiments among certain subgroups. We propose an ensemble architecture to conduct sentiment analysis of code-mixed tweets. The deep learning architecture of this project combines deep-learning technologies such as CNN or self-attention-based LSTM neural networks. CNN is a key component in our architecture, allowing us to differentiate between positive and negatively-toned tweets. Convolutional layers are excellent at identifying features in text documents and accurately classifying sentiments for this polarized statement. The LSTM (Long-Term Memory), neural network component for Neural tweets, has the ability to distinguish the correct sentiment in texts that include multiple units that express that sentiment. It can also navigate code-mixed text that contains mixed sentiments and accurately classify tweets as neural Tweets.

Keywords—deep learning, sentimental analysis, code-mixed text, CNN, LSTM.

I. INTRODUCTION

The use of code mixing is common in many forms of communication, from social media

linguistic sources, we will focus on bilingual code mix as it relates to Hinglish or Spanglish in written texts. The practice of categorizing human emotions and affections is called sentiment analysis. Understanding Sentiment Analysis requires an understanding of Human Emotions and Affection States as a base for decision-making. It is not easy to code-mix languages, as sentences may not match a specific language model. As tokens, hashtags and usernames are often used in mixed text, hashtags or usernames may be used as tokens and the XLMR Embeddings are then used to build both CNN-LSTM ensembles as well as self-attention LSTM ensembles.

The deep learning network architectures like Convolutional Neural Networks have been extensively used in previous studies to analyze sentiment. However, none of these efforts included a CNN and a model that relied on self-attention. Our research showed this combination is highly effective for handling positive and negative tweets, analyzed using CNN segments, while the self-attention component was more adept at handling those neutral ones.

This paper presents a combined concept architecture of neural networks and self-attention-based LSTM models using XLM-RoBERTa Encoder. The ensemble approach combines both components to achieve optimal performance.

II. Related Work

The modelling of deep learning and NLP tasks using mixed data poses many challenges. Ali Athar. (2020) I am trying to find an POS labelling system for mixed products.

Nazanin's other works. (2021) used CNN for sub-word embeddings, and then embedded the embeddings into a two-dimensional neural network LSTM to learn sub word information from SNS on the Internet. Altwords are popular texts with incorrect punctuation and spelling.

This function does not provide semantic information at the word-level. This allows for a more detailed examination of the effects of words embedding-based techniques. Aqsa's new work. They used bilateral LSTM networks, and an additional feature net which they called common and unique encoders. This approach integrates a neural net using a tracking method that allows all views to be evaluated using a weighted-approach while presenting a mixture of local views.

III. Proposed Approach

A. Preliminary Data Processing

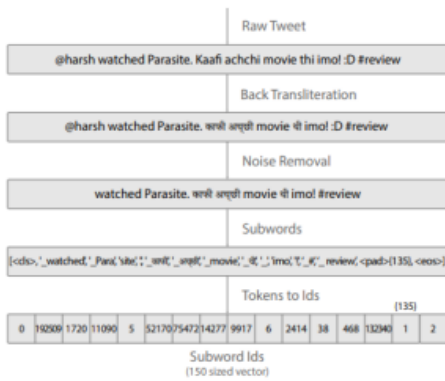


Figure 1: Preprocessing steps

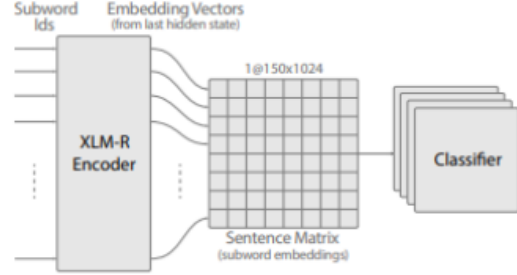


Figure 2: XLM-RoBERTa

Initially the data scraped from the Internet have to be processed. The following pre-processing is used to prepare tweets for the training phase (Figure 1).

1. Translation reversed: All words that contain "Hindi" are translated into Sanskrit in all languages. Google's Transliteration API2 will be used. Messages written in "Spanish", however, will not be translated.

2. Noise reduction: The hashtag, URL and emoticons (annotated with #hashtag), as well as the user's username (annotated @username) are removed. The emojis were replaced with their corresponding content. However, removing them would result in better performance.

3. Tokenization: Tweets that have been filtered for noise are tagged with sub-words using the XLM R (Conneau et al., 2018) language, and then converted to ID letters.

B. Embedding layer

It is important to use multiple languages, as our database contains tweets and numbers. Our proposed design uses XLM R embeddings. XLM is a tool for language transformation that uses more than 2 TB SNS filter data from the Internet to know the mask model. The sub-word IDs are sent into the XLM-RoBERTa encoder.

Accessing the distribution function requires looking at each token's last known state, or "hidden last state". Training phase. In this step, the XLM coder was tuned for more accurate encoding while Devlin et al's

BERT multilingual was evaluated as part of this phase. We also tested the Multilingual BERT (hereafter called M-BERT), published by Devlin et al. We also tested Devlin et al.'s multilingual BERT. (2018). We have experimented and come to conclusion that RoBERTa performs better on the two datasets.

IV. Architecture

We propose a deep learning ensemble model consisting of two major components in the architecture.

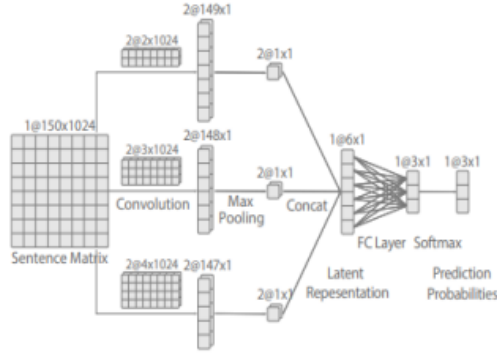


Figure 3: CNN Classifier

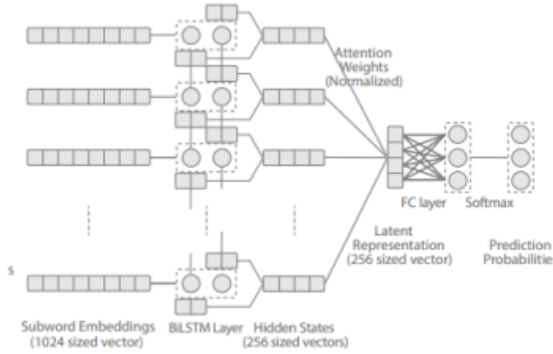


Figure 4: Self-Attention Classifier

A. Convolutional Neural Network

The first component is a Convolutional Neural Network. CNN takes into account both context and word order when considering each word that appears, while embeddings are created by passing each sub word of a sentence sequentially into dimensions. Convolutional processes were

then conducted using three distinct filter sizes (2"/3"/4") before biasing and activating RELU nonlinear algorithm.

Multiple filter sizes are utilized to capture content with differing lengths. Convolution extracts features from word windows while max pooling extracts features that are important to the feature map. Furthermore, an XLM embedding element receives this message, then sends SoftMax features with which it produces SoftMax predictions known as pCNN predictions.

This element receives the XLM-R embedding and produces the prediction through the SoftMax function. These predictions are denoted as pCNN.

B. Self-Attention based LSTM

Second Component is an auto-attentive classifier system. When faced with mixed emotions, this classification aids in ascertaining what the general sentiment might be. Soft colour (Xu et al., 2015) represents a decision involving different colours, where SoftMax assigns weights to each word, and the color results indicate the latent weight of each field's representation.

Hochreiter and Schmid Huber's (1997) Bidirectional Short-Term Memory Bi-LSTM employing output from an XLM R encoder as its initial layer of storage is called Bi-LSTM. The hidden states of Bi-LSTM are used to derive the efficiency of the model. In the segment of tokenized words that we have created, we represent the hidden state in the Bi-LSTM in the forward hidden state and the backward hidden state.

The compound description k_i is obtained by combining forward hidden state and the backward hidden state. Combining the forward and reverse hidden states results in the expression combination $(k_1, k_2, \dots, k_n)$.

The representation is the combined weight of all hidden states. Subsequently, the representation vector h undergoes a fully connected process, and SoftMax is applied to obtain the estimated pattern. Element-by-element (refer Figure 5) (indicated by $\circ$) are

employed to combine the predictions from the first and second components, resulting in the final prediction. Other aggregation methods were explored, and the element-wise combination proved to be more effective.

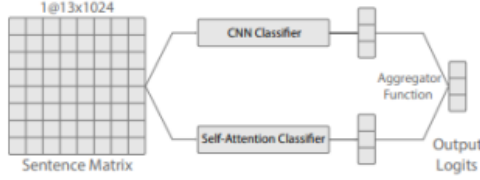


Figure 5: Classifier

Table 1: Training, Validation and Test data

	Dataset	Positive	Neutral	Negative
Train	Hinglish	5264	4634	4102
	Spanglish	6005	3974	2023
Validation	Hinglish	982	1128	890
	Spanglish	1498	994	506
Test	Hinglish	1000	1100	900

V. Dataset

The data model is created by scraping data from SNS platforms. We use this to train Indian English and Spanish English patterns. The data set is recorded semi-automatically. Table 1 provides training, validation, and testing data for the database. The Hinglish state is balanced, while the Spanglish state is unbalanced. Use the verification process provided by your organization for hyperparameter tuning.

VI. Results of the experiments

The self-monitoring system outperformed previous models in terms of average data. It also performed better on both positive and negative models.

It is believed that tweets are effective because they can evoke a variety of emotions. The combination of CNN results and Self-Monitoring Models outperformed previous models.

It achieved a return score of 0.717, and an F1 of 0.720 on the first dataset, and a score

of 0.712 with an F1 of 702 on the second dataset. In Figures 6 and 7, confusion matrices (o: fair; +: good; -: poor) are displayed for clusters in the two datasets.

Our project showed a state-of-the-art result with the most accuracy than any other models currently.

True Label \ Predicted Label	o	+	-
o	743	176	181
+	235	738	27
-	243	22	635

Figure 6: Hindi-English dataset Confusion matrix

True Label \ Predicted Label	o	+	-
o	798	94	102
+	343	1108	47
-	135	10	361

Figure 7: English-Spanish dataset Confusion matrix

Table 2: Results of the model on the two datasets

	F1			Macro Precision	Macro Recall
	o	+	-		
Hinglish	0.640	0.762	0.729	0.707	0.705
Spanglish	0.135	0.825	0.375	0.725	0.696

VII. Analyzing the results

A. Sentence Vectors Result Visualization

The t-SNE method was used to create a visual model of sentence embeddings for the Hinglish data (van der Maten & Hinton, 2008). The two points in Figure 8 are where all sentence vectors were projected. CNN grouped tweets into two distinct groups: positive and negative. Neutral tweets were dispersed among the two. In contrast, when the Data is neutral the points are partially dispersed. These two concepts are complementary, and can improve predictions for all groups.

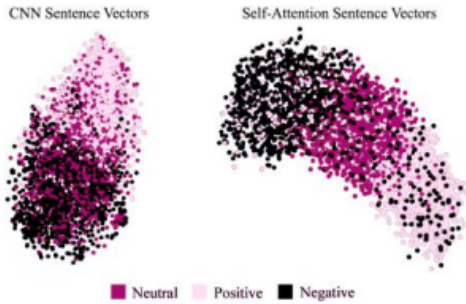


Figure 8: Sentence Vectors

B. Error Analysis

The majority of errors involve tweets that fall into three categories:

1. Neutral - Although self-care has improved the performance of tweets, neutral tweets are still behind both +ve and -ve sentiments.
2. Mocking and witty comments are generally posted on sns by the users. It was difficult to classify. Due to their secret histories, tweets that made a good impression on fake atheists were hard to classify. This led to inaccurate estimations of how good they were.
3. Minor Negatives - Because of the large number malicious tweets that were in the

database, certain negative tweets such as "Ek Gaun mein ek Kisan is Dead" were incorrectly rated as average.

VIII. Conclusion

We used a set CNNs with self-developed architectures, and XLM R multilingual embeddings for our system. We evaluated the performance of various models for different groups of tweets. LSTM proved useful in classifying neutral data which is complex and hard to understand due to many emotional and behavioural factors. CNN was incorporated for holistic integration to help identify all three groups. Our model's performance was visualized using the t-SNE algorithms, which showed improvement over previous studies.

References

- [1] Supriya Sameer Nalawade, Arun S. Patil, "An Empirical Study on Sentimental Analysis using Deep Learning", 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC).
- [2] Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model", 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 225–230, Berlin, Germany, August. Association for Computational Linguistics.
- [3] Prabha PM Surya, B Subbulakshmi, "Sentimental Analysis using Naive Bayes Classifier", 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN).
- [4] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, "Neural image caption generation with visual attention.", CoRR,abs/1502.03044.

- [5] Phasit Charoenkwan, “ThaiFBDeep: A Sentimental Analysis Using Deep Learning Combined with Bag-of-Words Features on Thai Facebook Data”, 2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI).
- [6] Joosung Yoon and Hyeoncheol Kim, “Multi-channel lexicon integrated CNN-BiLSTM models for sentiment analysis” 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017), pages 244–253, Taipei, Taiwan, November. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP)
- [7] Abdulrahman Radaideh, Fikri Dweiri, Mohammad Obaidat, “A Novel Approach to Predict the Real Time Sentimental Analysis by Naive Bayes & RNN Algorithm during the COVID Pandemic in UAE”, 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI).
- [8] Gazi Imtiyaz Ahmad, Jimmy Singla, “(LISACMT) Language Identification and Sentiment analysis of English-Urdu ‘code-mixed’ text using LSTM”, 2022 International Conference on Inventive Computation Technologies (ICICT).