Sentiment Analysis on Code-Mixed Telugu-English Text using IndicBERT

Group – 12

Team Members

H J Chaithanya Prasad – 21MID0132

Eshwar Sharma Topudurti – 21MID0237

Srirangam Sudheer – 21MID0247

Faculty: DEVIPRIYA A

Semester: Summer Term I

Course: Neural Networks and Deep Learning

Course Code: MDI4009

Slot: G1+TG1+G2+TG2

Class No: VL2024250700199

Introduction

Sentiment analysis is a subfield of Natural Language Processing (NLP) that involves identifying and extracting emotional tone behind a body of text. It is widely used in applications such as product review analysis, social media monitoring, and public opinion tracking. In the Indian context, a significant portion of user-generated content on platforms like Twitter, Facebook, and YouTube appears in code-mixed formats — particularly Telugu-English — where words from two languages are used interchangeably within a single sentence. This code-mixing introduces challenges in tokenization, syntax, semantics, and overall understanding due to lack of labelled datasets and language-specific resources. To address this gap, the present project utilizes IndicBERT — a transformer-based language model trained on multiple Indian languages. By fine-tuning IndicBERT on a custom dataset of Telugu-English code-mixed sentences, this project aims to develop a robust and scalable sentiment classification system capable of categorizing inputs as positive, negative, or neutral with high accuracy and efficiency.

Existing Work

Earlier sentiment analysis models focused on monolingual English or regional languages. Traditional ML methods such as Naive Bayes and SVM with TF-IDF features have been used. Recent advancements employed deep learning, especially LSTM-based models. However, these struggled with mixed-language contexts and required heavy pre-processing.

Proposed Work

This project proposes the use of IndicBERT, a multilingual transformer model specifically trained on Indian languages, including Telugu. IndicBERT can handle Indian scripts and mixed-language input natively, unlike traditional models that need a lot of preprocessing or translation. A carefully selected dataset of code-mixed Telugu-English sentences that have been manually annotated with sentiment labels (Positive, Negative, and Neutral) is used to fine-tune IndicBERT as part of the suggested workflow. The process of fine-tuning allows the model to capture both sentiment cues and semantic meaning by adjusting its previously learnt knowledge to the subtleties of code-mixed data. Hugging Face's Trainer API, which streamlines data loading, training configuration, evaluation, and checkpoint management, is used to make training easier. This method improves performance, scalability, and reproducibility, which makes it ideal for real-time deployment in applications such as sentiment dashboards, customer support bots, and social media monitoring.

Proposed Execution Code

```
import re

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from datasets import Dataset

from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, 
Trainer, pipeline

import numpy as np

from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, ConfusionMatrixDisplay


# Step 1: Load and Parse the Text File

file_path = "codemix_sentiment_data.txt"

with open(file_path, "r", encoding="utf-8") as f:

    lines = f.readlines()

data = []

for line in lines:

    if line.strip() == "" or ":" not in line:

        continue

    match = re.match(r"(POS|NEG|NTL):\s*(.+)", line.strip())

    if match:

        label, text = match.groups()

        data.append({"text": text.strip(), "label": label})
```

```python
df = pd.DataFrame(data)

label_map = {"NEG": 0, "NTL": 1, "POS": 2}

df["label"] = df["label"].map(label_map)



# 📊 Visualization 1: Label Distribution

label_names = {0: "Negative", 1: "Neutral", 2: "Positive"}

df["label_name"] = df["label"].map(label_names)

sns.countplot(x="label_name", data=df)

plt.title("Sentiment Label Distribution")

plt.xlabel("Sentiment")

plt.ylabel("Count")

plt.show()



# Step 2: Convert to Hugging Face Dataset

train_df, test_df = train_test_split(df, test_size=0.1, random_state=42)

train_dataset = Dataset.from_pandas(train_df)

test_dataset = Dataset.from_pandas(test_df)



# Step 3: Load Tokenizer and Tokenize

tokenizer = AutoTokenizer.from_pretrained("ai4bharat/indic-bert")

model = AutoModelForSequenceClassification.from_pretrained("ai4bharat/indic-bert", num_labels=3)


def tokenize(batch):

    return tokenizer(batch["text"], padding="max_length", truncation=True, max_length=128)


train_dataset = train_dataset.map(tokenize, batched=True)

test_dataset = test_dataset.map(tokenize, batched=True)
```

```
# Step 4: Training Arguments
training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    report_to="none"  # prevent logging to external services
)

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return {
        "accuracy": accuracy_score(labels, predictions),
        "f1_macro": f1_score(labels, predictions, average="macro"),
    }

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
```

```
# Step 5: Train and Save
trainer.train()
eval_results = trainer.evaluate()
model.save_pretrained("sentiment_model")
tokenizer.save_pretrained("sentiment_model")

eval_results

# 📊 Visualization 2: Confusion Matrix
# Inference on test set
predictions = trainer.predict(test_dataset)
preds = np.argmax(predictions.predictions, axis=-1)
true_labels = predictions.label_ids

cm = confusion_matrix(true_labels, preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Negative", "Neutral", "Positive"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix")
plt.show()

# Step 6: Inference on Multiple Code-Mixed Sentences
sentiment_pipeline = pipeline("text-classification", model="sentiment_model",
tokenizer="sentiment_model")

example_sentences = [
  # Positive
  "Movie chala bagundi bro, visuals are amazing!",
  "RCB team super ga aadindi today 👏",
  "Nuvvu cheppina song naku chala nachindi ❤️",
  "Chaala clean and funny movie, recommended to everyone.",
  "Super acting by the hero! Full goosebumps moment 🔥",
  # Negative
  "Worst movie I have seen in years, total time waste.",
  "Idhi review aa? chala chetta ga undi bro.",
  "E roju service chala poor ga undi, staff respond cheyyaledu.",
  "Nuv cheppina app lo bugs ekkuva, totally disappointed.",
  "Dislike this actor’s performance – emotion ledu at all.",

  # Neutral
  "Movie release date is next Friday.",
  "RCB vs MI match starts at 7:30 PM.",
  "I watched the trailer yesterday night.",
  "Class ki 10 members attend ayyaru.",
  "Andaru review chustunnaru YouTube lo."
]

label_map = {
  "LABEL_0": "Negative",
  "LABEL_1": "Neutral",
  "LABEL_2": "Positive"
}

for sentence in example_sentences:
  result = sentiment_pipeline(sentence)[0]
  sentiment = label_map[result['label']]
  confidence = result['score']
  print(f"Text: {sentence}")
  print(f"Predicted Sentiment: {sentiment} (Confidence: {confidence:.2f})")
  print("-" * 60)
```

Dataset Description

The dataset is a text file containing lines of the format:

POS: Movie chala bagundi bro

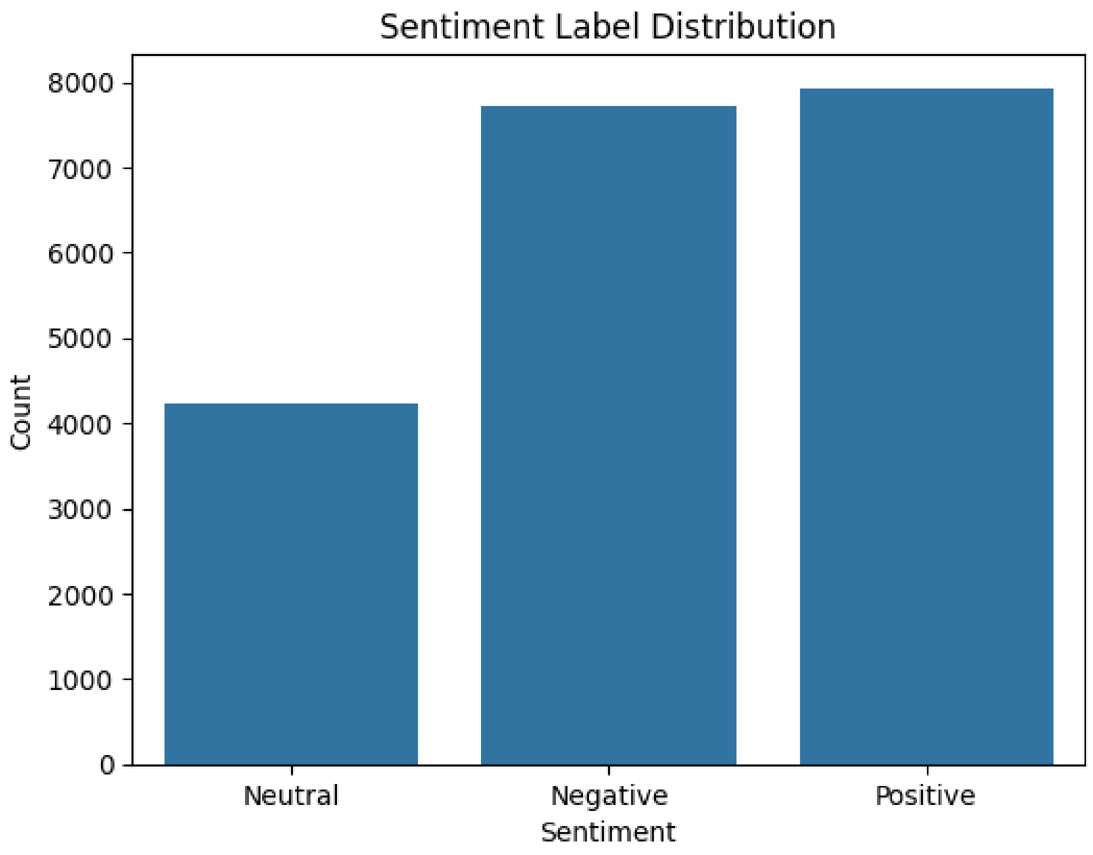
NEG: Worst performance ever

NTL: RCB match starts at 7 PM

- POS: Positive sentiment
- NEG: Negative sentiment
- NTL: Neutral statement

The data is loaded using regex, labelled, and split into training and test sets.

Dataset distribution



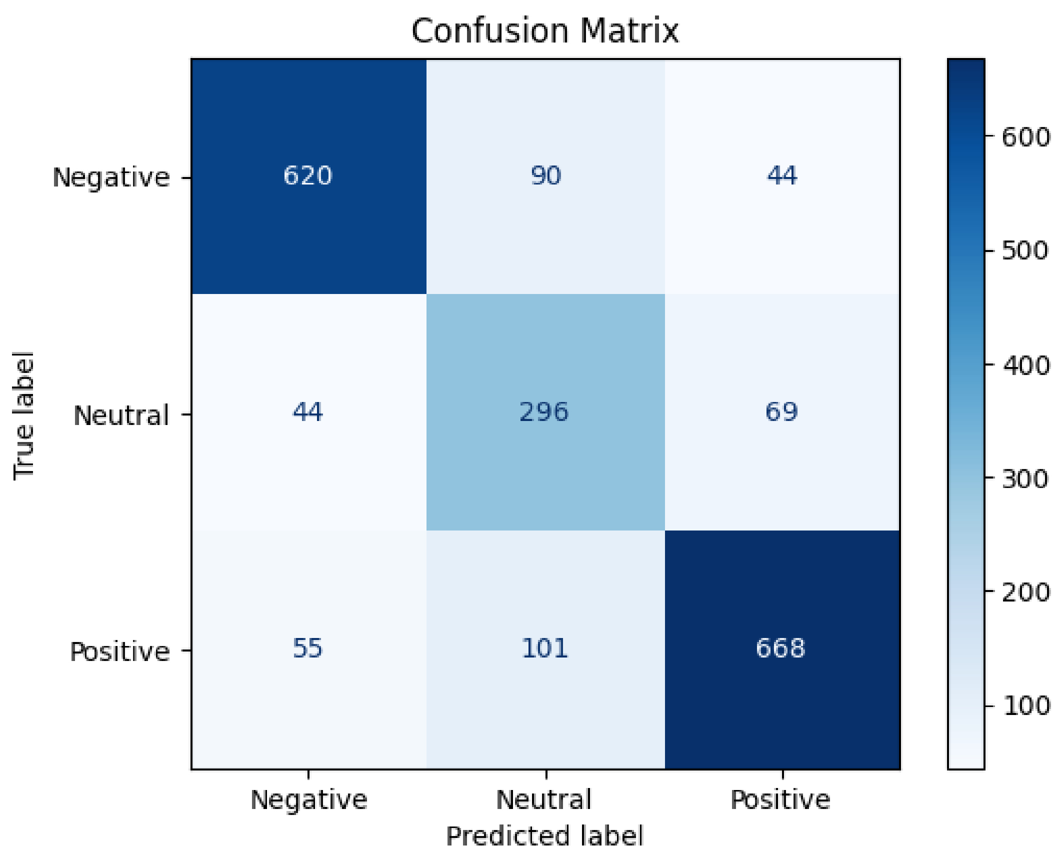
Result

The fine-tuned IndicBERT model achieved:

- Accuracy: ~79%
- F1 Macro Score: ~78%

These results suggest that the model is reasonably effective in classifying sentiments in code-mixed Telugu-English data. While not perfect, the accuracy and F1-score are strong indicators that the model can be further improved with more data, better hyperparameter tuning, or additional preprocessing techniques to handle the linguistic complexity of code-mixed inputs.

Confusion matrix



	Negative	Neutral	Positive
Negative	620	90	44
Neutral	44	296	69
Positive	55	101	668

Conclusion

The project demonstrates that IndicBERT can effectively perform sentiment analysis on code-mixed Indian language data, such as Telugu-English, with minimal preprocessing. Through fine-tuning on a well-annotated but limited dataset, the model achieved an accuracy of 79% and a macro F1-score of 78%, which reflects its capability to generalize across different sentiment classes. These results validate the suitability of IndicBERT for multilingual sentiment classification tasks and highlight its strength in capturing nuanced sentiment from mixed-language inputs. This makes the model particularly valuable for real-time applications such as monitoring user sentiments on social media platforms, analyzing public opinion during political events, and improving customer service systems in regional markets.

Code Link:
https://colab.research.google.com/drive/1s7L4tjBU6R7CwDBYCc1wkQXts_vMeaJH?usp=sharing

Dataset Link:
https://drive.google.com/file/d/1TnIFUE4ohVsTlq50FIz5iB_QlmZjBrgE/view?usp=sharing