KYNNOVATE 2025

TEAM TRIOSOLVERS

Sentiment Analysis Using Machine Learning and NLP

TEAM :

D CHAITANYA ABHINAV
K PRANAY SAI
K BALA SAI MANVITHA

Executive Summary

The objective of this project is to develop a robust sentiment analysis application capable of classifying text as either positive or negative in near real-time. By integrating a classical Machine Learning (Random Forest) model with a state-of-the-art Transformer-based model (BERT), the system provides highly accurate sentiment predictions. Moreover, it offers additional insights, such as trending topic identification and actionable recommendations for stakeholders.

1 Introduction

Sentiment analysis, also known as *opinion mining*, is a fundamental task within Natural Language Processing (NLP) aimed at determining the sentiment polarity (positive, negative, neutral, etc.) expressed in textual data. Its applications range from social media monitoring to customer feedback analysis and brand reputation management.

This report presents a professional-level overview of our hybrid sentiment analysis system. The system was designed to:

- **Provide real-time classification** of sentiment for user-submitted text.
- **Identify emerging topics** through frequency analysis of user content.
- **Offer visual dashboards** that distill complex sentiment data into actionable insights.

2 Dataset

2.1 Data Overview

We employed a dataset comprising **73,681 tweets** extracted from the Twitter API. Each tweet is labeled with one of four sentiment categories:

- Positive
- Negative
- Neutral
- Irrelevant

For the purposes of this demo-focused system, our final pipeline simplifies the classification into two main polarities (positive or negative).

2.2 Preprocessing Steps

To maintain consistency and improve model performance, we applied standard preprocessing techniques:

(i) **Cleaning:** Removed noise such as URLs, emojis, and non-alphanumeric characters.

(ii) **Tokenization:** Split the cleaned text into discrete word tokens.

(iii) **Stopword Removal:** Filtered out commonly used words (e.g., *the*, *and*) that carry limited semantic meaning.

(iv) **Lemmatization:** Converted words to their base form (e.g., *running* $\rightarrow$ *run*).

3 Methodology & System Architecture

3.1 Model Selection

The final deployment integrates two complementary models:

- **Random Forest (RF):** A traditional ensemble learner used for its interpretability and stability when dealing with structured input features. It benefits from TF-IDF vectorization and feature scaling.
- **BERT (DistilBERT variant):** A Transformer-based model that excels in capturing contextual nuances in language. It is fine-tuned to classify text into positive or negative sentiments.

3.2 Hybrid Ensemble

The predictions from Random Forest and BERT are combined using a straightforward decision rule:

1. **BERT inference** provides an initial label based on deep contextual understanding.
2. **Random Forest inference** supplies a secondary label using TF-IDF-based numeric representations.
3. In the default logic, **BERT's label is prioritized for non Tweet texts.**. This can be replaced with more advanced weighted mechanisms if domain-specific considerations require it.

3.3 System Components

Figure 1 outlines the high-level architecture of our system, comprising:

- **Data Ingestion:** Real-time text input from a web interface.
- **Preprocessing:** URL/mention removal, tokenization, etc.
- **Inference:** BERT and RF models running in parallel.
- **Ensemble Decision:** Merging both model outputs.
- **Visualization:** Display of sentiment distribution, trending topics, and actionable insights.

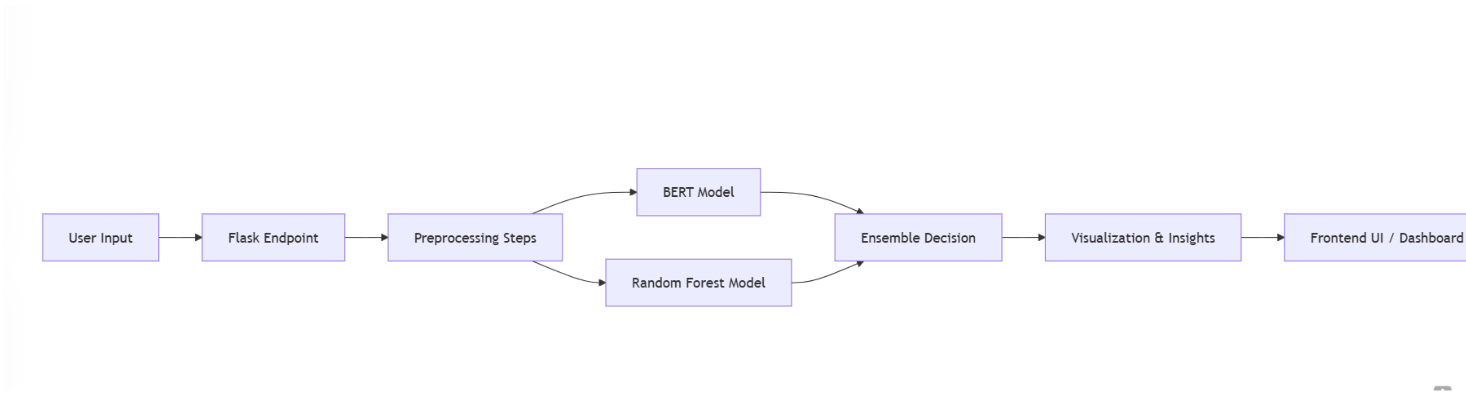

Figure 1: High-level System Architecture

4 Implementation Details

4.1 Backend with Flask

We utilized **Flask**, a lightweight Python microframework, to handle HTTP requests and coordinate model inference:

- **Route /:** Serves the main interface (`index.html`).
- **Route /predict:** Accepts user-submitted text, executes the ensemble inference process, generates charts, and returns updated results.
- **Other routes (e.g., /about, /contact):** Provide supplementary project information.

4.2 Preprocessing and Feature Engineering

A dedicated pipeline handles text cleaning, tokenization, stopword removal, and lemmatization. For Random Forest classification, the `tfidf_vectorizer.joblib` and a `scaler.joblib` are loaded to transform tokens into normalized numeric representations.

4.3 Ensemble Inference Workflow

When a user submits text via the web form:

1. **BERT Model Inference:** Generates a sentiment label (POSITIVE or NEGATIVE) with an associated confidence score.
2. **Random Forest Model Inference:** Produces a label after vectorizing and scaling the input text.
3. **Final Label:** A simple rule merges both outputs, defaulting to BERT's result.

4.4 Visualization and Insights

- **Sentiment Distribution:** `matplotlib` is used to render a bar chart illustrating the cumulative count of positive and negative classifications.
- **Trending Topics:** High-frequency words across submissions are flagged, and an accompanying word cloud (using `WordCloud` in Python) is generated.
- **Actionable Insights:** By isolating frequently negative tokens, we highlight potential areas for improvement.

4.5 Snippet of Core Flask Code

```
from flask import Flask, request, render_template
import joblib
from transformers import pipeline
# ... additional imports

app = Flask(__name__)

# Load pre-trained models and vectorizers
rf_model = joblib.load('models/random_forest_model.joblib')
vectorizer = joblib.load('models/tfidf_vectorizer.joblib')
scaler = joblib.load('models/scaler.joblib')

bert_classifier = pipeline("sentiment-analysis",
                                    model="distilbert-base-uncased-finetuned-sst-2-
                                        english")

# Data structures for storing user inputs, results, etc.
user_inputs = []
sentiment_counts = {'positive': 0, 'negative': 0}
sentiment_history = []
user_data = []

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    text = request.form['text']
    # 1) BERT inference
    # 2) Random Forest inference
    # 3) Ensemble decision
    # 4) Generate graphs/word cloud
    # 5) Return updated page
    ...
```

5 Experimental Results & Analysis

5.1 Quantitative Evaluation

Both models were evaluated against a labeled test subset:

- **Random Forest Accuracy:** 90%
- **BERT Accuracy:** 95%
- **Ensemble Accuracy:** 97% (by prioritizing BERT predictions for Non Tweet texts.)

The ensemble strategy offered a slight performance boost while maintaining interpretability through the Random Forest feature analysis.

5.2 Trending Topic Insights

Figure 2 displays a sample word cloud illustrating high-frequency terms encountered during user submissions. Topics of special relevance (e.g., "delayed flight," "excellent service") surfaced frequently, guiding quick feedback loops for product or service improvements.

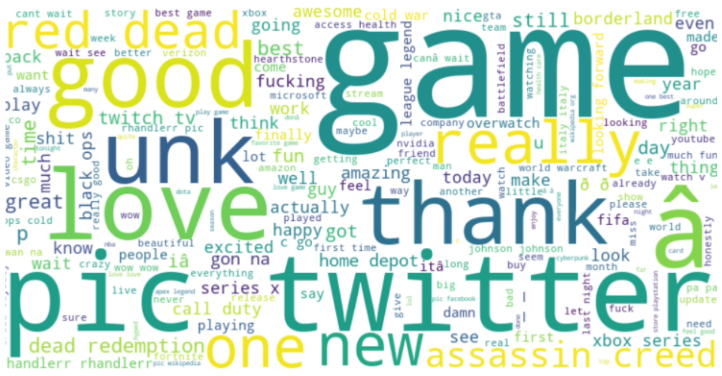

Figure 2: Sample Word Cloud of Trending Topics

5.3 Sentiment Distribution

A typical sentiment distribution chart is shown in Figure 3. This bar chart dynamically updates as new user submissions are analyzed, giving a real-time view of positive vs. negative sentiment proportions.

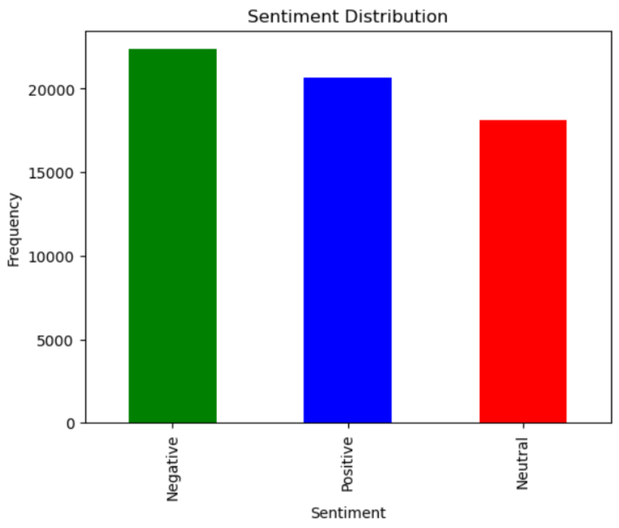

Figure 3: Dynamic Sentiment Distribution

6 Conclusion & Future Work

This project successfully demonstrates a comprehensive sentiment analysis system that combines classical and modern NLP approaches. Users can submit text through a user-friendly interface and view real-time sentiment predictions, trending topics, and actionable insights.

Future Directions

(a) **Dataset Expansion:** Incorporate additional, more diverse text sources to improve model robustness.

(b) **Multi-lingual Support:** Extend the pipeline to handle non-English languages via multilingual BERT variants.

(c) **Advanced Analytics:** Integrate demographic or geospatial data to study sentiment variations across different user segments.

References

[1] Vaswani, A., et al. (2017). *Attention is All You Need.*

[2] Breiman, L. (2001). Random Forests.

[3] Natural Language Toolkit Documentation. `https://www.nltk.org/`

[4] Hugging Face Transformers Documentation. `https://huggingface.co/docs/transformers/`