Big Data-Powered Sentiment Analysis for Code-Mixed Hindi-English Tweets

Spring 2025, Big Data: Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Science
in
Computer Engineering

Submitted by

Ms. Asmita Sonavane	**as20428**
Mr. Vishwajeet Kulkarni	**vk2630**
Mr. Sarang P. Kadakia [Team Lead]	**sk11634**

Team Name: Big Data Enthusiasts

Guided by

Professor Amit Patel

Contents

1. Abstract

In today's digital age, platforms like Twitter serve as dynamic spaces for multilingual expression—particularly in India, where users often blend Hindi and English within a single post. However, this code-mixed style presents serious challenges for traditional NLP tools due to inconsistent spellings, transliteration, and informal language patterns. Our project tackles this complexity head-on by designing a robust, end-to-end sentiment analysis pipeline for Hindi-English tweets using Big Data technologies.

We employed the AI4Bharat "Mann ki Baat" dataset comprising over 100,000 code-mixed tweets. Our pipeline integrates Apache HBase for scalable storage, Spark NLP for advanced preprocessing (including tokenization, transliteration, and social media text normalization), and PySpark MLlib for feature engineering. We further enriched the data with custom negation detection and part-of-speech tagging. Using a fine-tuned IndicBERT model, we achieved an accuracy of **81.20%** and an F1-score of **81.05%** on unseen test data. Additionally, we integrated Hive to enable seamless querying of cleaned sentiment-labeled data for future scalability.

What makes this project stand out is its ability to handle real-world linguistic diversity and extract meaningful insights from noisy, unstructured text at scale. The system not only detects sentiment with precision but also lays the foundation for trend analysis across social media. Future expansions include extending to other Indian languages and platforms like Facebook and Instagram, as well as incorporating real-time dashboards using Spark Streaming.

Keywords: Apache HBase, Spark NLP, PySpark MLlib, Code-Mixed (Hindi+English) Tweets, IndicBERT

2. Executive Summary

Project Name:
Big Data-Powered Sentiment Analysis for Code-Mixed Hindi-English Tweets

Brief Summary:
In a linguistically diverse country like India, social media platforms are flooded with code-mixed (Hindi-English) tweets. These tweets are rich in public sentiment but difficult to analyze using standard NLP tools due to transliteration issues, informal spellings, and intra-sentence language switching. Our project tackles these challenges by building an end-to-end big data pipeline that can ingest, clean, analyze, and classify such tweets at scale.

Objectives:

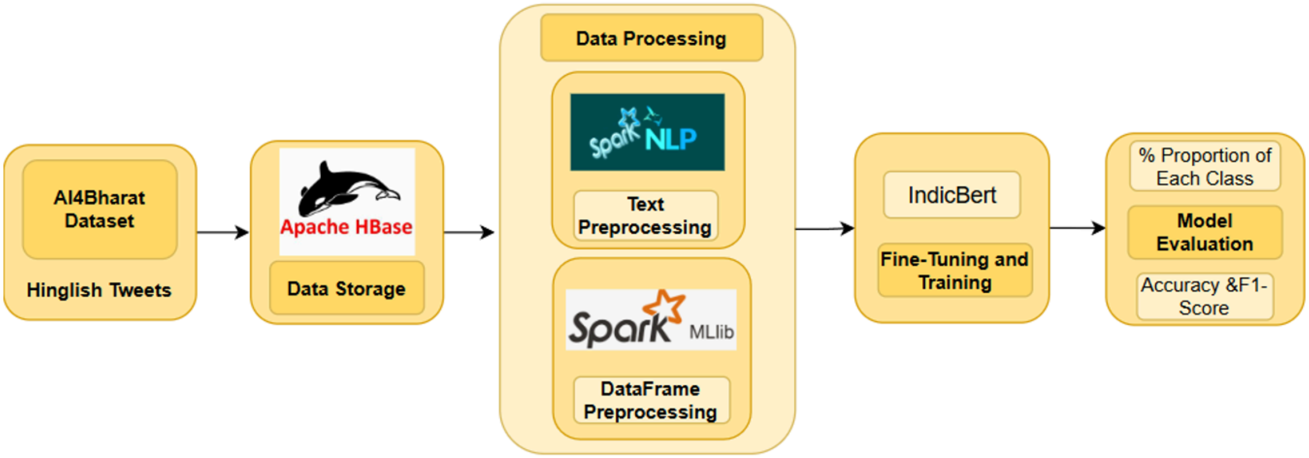
- Develop a **scalable preprocessing pipeline** for Hinglish tweets.
- Perform **sentiment classification** using **IndicBERT**.
- Utilize **Big Data frameworks** (HDFS, HBase, Hive, Spark NLP, PySpark MLlib) for processing.
- Achieve high model performance in terms of **accuracy** and **F1-score**.
- Visualize trends and prepare clean structured output for future applications.

Technologies Used:

- **Apache HBase & Hive** – Scalable storage and querying
- **Spark NLP + PySpark MLlib** – Text normalization, tokenization, feature engineering
- **IndicBERT** – Multilingual sentiment classification (trained using Spark NLP)
- **Ekphrasis** – Social media text normalization
- **DistilBERT SST-2** – Auto-sentiment labeling for English content

Project Demo:
https://drive.google.com/file/d/1xzJBh3v3hRvV1VeZ5CbB5VujqNYyxPgR/view?usp=sharing

System Architecture and Flow Diagram:



3. Code Execution Instructions & README Overview

Execution Environment:

- Apache Spark with GPU support
- PySpark 3.x
- Spark NLP 5.5.3
- Ekphrasis, Transformers
- Jupyter Notebook ; Colabatory

To Run the Code:

1. Run the notebook in a Spark-enabled environment. Recommended: 16GB RAM + GPU acceleration.
   Install dependencies:

   ```
   pip install spark-nlp==5.5.3 pyspark

   pip install ekphrasis

   pip install transformers
   ```

2. Make sure the CSV file (cleaned_output_stage1.csv) is present in the working directory.
3. Follow the notebook structure:
   - **Data Cleaning & Preprocessing** (drop nulls, standardize language)
   - **Feature Engineering** (POS tagging, stopword %)
   - **Model Training** (IndicBERT via Spark NLP pipeline)
   - **Evaluation** (F1, accuracy, class distribution)
   - **Export** the results and visualizations

README URL: README.md **(It is even present in my Github Repo)**

Github Link:
https://github.com/SARANG1018/Big-Data-Powered-Sentiment-Analysis-for-Code-Mixed-Hindi-English-Tweets

4. Technological Challenges

1. Handling Noisy, Raw Code-Mixed Data

The raw dataset contained over 140,000 tweets with severe noise—null values, inconsistent formatting, repeated headers (e.g., "en_query", "csquery"), and duplicated rows. Additionally, the code-mixed nature of the text introduced significant complexity due to unpredictable switching between Hindi and English, non-standard spellings, and diverse transliteration forms (e.g., *"Modiji"* vs *"Modi ji"*, *"jeet"* vs *"jit"*). To address this:

- We dropped unnecessary columns and invalid rows.
- Built a custom **Hinglish normalization dictionary** to correct common spelling variants.
- Used **Ekphrasis** to handle informal social media text, and created PySpark **UDFs** to apply regex-based corrections across both languages.

This multilayered approach ensured that our processed text was clean, semantically consistent, and ready for feature engineering and model training.

2. Negation Detection in Mixed Languages

Standard NLP libraries are ill-equipped to detect negation across multilingual, code-mixed datasets. To bridge this gap:

- We manually curated a **comprehensive Hindi-English negation lexicon**.
- Built a **custom regex engine** to identify negation patterns (e.g., *"nahi", "not", "na", "nahi tha"*).
- Added a binary feature (**`has_negation`**) to enrich sentiment interpretation—particularly useful in sarcastic or critical tweets.

This addition significantly improved the classifier's ability to understand emotional polarity and intent behind the tweets.

3. Spark NLP Compatibility & Model Adaptation

We initially planned to use Hugging Face's fine-tuned IndicBERT but faced compatibility issues with GPU-based training in the Spark cluster. Instead:

- We switched to **JonSnowLabs' Spark NLP IndicBERT pipeline**, which integrates well with the Spark ML ecosystem.
- We ran training using **T4 GPUs**, optimizing batch size (64) and memory configurations within Spark to accelerate performance.

This shift ensured smoother integration with our Spark pipeline and allowed us to leverage GPU acceleration effectively for deep learning-based sentiment classification.

4. Memory Bottlenecks During Sentiment Auto-Labeling

To bootstrap English sentiment labels, we employed **DistilBERT SST-2** via Transformers. However, processing large volumes of data triggered memory overflows. Our mitigation strategy included:

- Reducing the **batch size to 32** for inference.
- Processing tweets in **chunks** and merging results incrementally back into the PySpark DataFrame.

This careful memory management allowed us to harness the power of transfer learning without compromising system stability.

5. Changes in Technology (from Proposal)

Final Implementation Changes:

- We switched to **John Snow Labs' Spark NLP** implementation of IndicBERT with ClassifierDL, instead of the **Hugging Face version**. This change was necessary because fine-tuning Hugging Face models with LoRA was not fully compatible with our **GPU-enabled Spark cluster (T4 instance)** and required more manual orchestration than feasible within project constraints.

Impact:

- Adopting Spark NLP's native IndicBERT model ensured full compatibility with our distributed PySpark pipeline and allowed us to train and infer within a Spark-native workflow, benefiting from **better GPU integration** and stability.

- Also this pivot allowed us to deliver a more polished, accurate, and scalable sentiment analysis model, achieving an **81.20% accuracy** and **81.05% F1-score**, well aligned with our core project goals.

6. Uncovered Aspects from Presentations

While our presentation covered the high-level architecture and outcomes, several intricate and impactful components from our implementation were not included due to time constraints. These aspects deserve mention for their technical depth and contribution to the model's performance:

1. Hybrid Stopword Percentage Calculation

We went beyond standard stopword removal by calculating the **stopword density** per tweet. This was done using a **hybrid stopword list** combining English, Hindi, and curated Hinglish terms.
This **stopwords_percentage** feature was added as a numerical input to the classifier, helping it assess the noise level or linguistic density of each tweet—especially useful in differentiating filler text from sentiment-heavy content.

2. Part-of-Speech (POS) Model Evaluation and 'NNN' Filtering

We conducted a **comparative analysis of three Spark NLP POS taggers**: **pos_hiencs, pos_anc**, and **pos_hdtb**. Using a small, manually labeled validation set, **pos_anc** showed the highest tagging accuracy and was selected for final usage.

A crucial insight here was the detection of the **'NNN' tag**, which frequently appeared in tweets with:

- Highly informal spelling,
- Out-of-vocabulary words, or
- Mixed transliteration errors.

To handle this, we:

- Counted **NNN** tag occurrences per tweet.
- Flagged high **NNN**-density tweets as noisy.
- Used this as a **filtering heuristic** to exclude outliers from training or assign low confidence to their predictions.

This subtle yet powerful mechanism improved the overall quality of input to our IndicBERT model.

3. Ekphrasis + Custom Hinglish Normalization Pipeline

We used **Ekphrasis** not just for basic tokenization but for comprehensive **social media-specific normalization**, such as:

- Fixing emojis, hashtags, and URL tokens,
- Expanding contractions and de-elongating words (e.g., *goooood* → *good*).

For Hinglish-specific cases, we created a **custom dictionary of 30+ frequent spelling variations**, integrated using PySpark **UDFs and regex-based replacements**.

This dual-layer normalization significantly improved the semantic integrity of both English and Hinglish inputs and was a foundational step in model accuracy.

7. Lessons Learned

Throughout the course of this project, we encountered real-world challenges that deepened our understanding of working with multilingual, unstructured social media data. Here's a reflection on what worked, what was difficult, and the broader insights we gained.

Implementation:

- **Custom UDF-based normalization + Spark NLP pipelines** turned out to be a powerful combination for processing noisy Hinglish tweets. Our layered approach—Ekphrasis for English patterns and PySpark UDFs for Hinglish-specific regex corrections—dramatically improved input quality for downstream modeling.

- **Semantic feature engineering** such as **has_negation, stopwords_percentage**, and POS tagging (with NNN filtering) added rich context to otherwise ambiguous short tweets, helping IndicBERT perform better on nuanced sentiments.

- **Hive integration** at the end of the pipeline gave us clean, queryable, and scalable access to labeled sentiment data. It not only simplified post-processing and visualization but positioned our system for future real-time use cases.

Challenges Overcome

- **Spark NLP GPU configuration** (on T4 instance) required careful tuning of memory allocation, driver options, and Spark session parameters. Getting IndicBERT to work smoothly within Spark NLP (as opposed to Hugging Face) was initially non-trivial, but ultimately allowed us to train efficiently in a distributed setting.

- **Romanized Hindi inconsistencies** were tough to standardize due to their informal and non-dictionary nature. We overcame this by creating a handcrafted dictionary of commonly distorted Hinglish words, applied via PySpark UDFs, and iteratively refined through human feedback during exploratory data cleaning.

- **POS tagging errors and NNN inflation** taught us the limitations of pretrained NLP models on code-mixed data. Our decision to count `NNN` tokens and penalize low-signal tweets proved key in avoiding garbage inputs in training.

Key Takeaways

- **Hinglish data isn't just messy—it's structurally and semantically complex.** Preprocessing must be *context-aware*, involving normalization, linguistic validation, and selective filtering beyond typical tokenization.

- **Big Data frameworks like Spark, Hive, and HBase are more than just buzzwords.** When carefully orchestrated, they enable real-time, multilingual NLP pipelines at scale—especially when GPU-backed.

- **Building modular pipelines pays off.** Our architecture—built around Spark NLP stages, external labeling modules, and Hive outputs—can now be easily extended to:

a. Other code-mixed languages (e.g., Tamil-English, Bengali-English),

b. New platforms like Instagram or YouTube comments,

c. Real-time dashboards using **Spark Streaming** or **Superset/Power BI**.

8. Future Improvements

- **Add More Indian Languages:**
  Extend support to Tamil, Bengali, and Marathi to make the model applicable across more multilingual regions. This would require language-specific normalization and updated embeddings.

- **Cross-Platform Sentiment Analysis:**
  Adapt the pipeline for Facebook and Instagram by handling longer texts, emojis, and comment structures unique to those platforms.

- **Real-Time Dashboards:**
  Use Spark Streaming + Hive + Grafana to build a live sentiment monitoring dashboard, useful for tracking reactions to current events in real time.

- **Model Interpretability (SHAP/LIME):**
  Add explainability to show why a tweet was classified as positive or negative — crucial for stakeholder trust in sensitive domains.

- **Semi-Supervised Auto-Labeling for Hindi:**
  Apply weak supervision or self-training to auto-label Hindi or code-mixed tweets, expanding the labeled dataset without manual effort.

9. Data Sources, and Result

Dataset:

The dataset used for this project is sourced from the **AI4Bharat Twitter dataset** (https://huggingface.co/datasets/ai4bharat/Mann-ki-Baat), which contains large-scale Hindi-English code-mixed tweets. We are also exploring additional data sources and planning to integrate them into our custom database. The current dataset provides real-world examples of language switching, transliteration variations, and sentiment-rich content, making it ideal for training and evaluating trend and sentiment analysis models. It comprises tweets collected over time across diverse topics such as politics, sports, entertainment, and social issues. The dataset's linguistic diversity, script variations, and sentiment polarity present a challenging yet valuable resource for accurate trend detection and sentiment analysis.

Evaluation and Results:

After training the IndicBERT-based sentiment classification pipeline, we evaluated its performance on a 30% test split of the dataset. The model achieved:

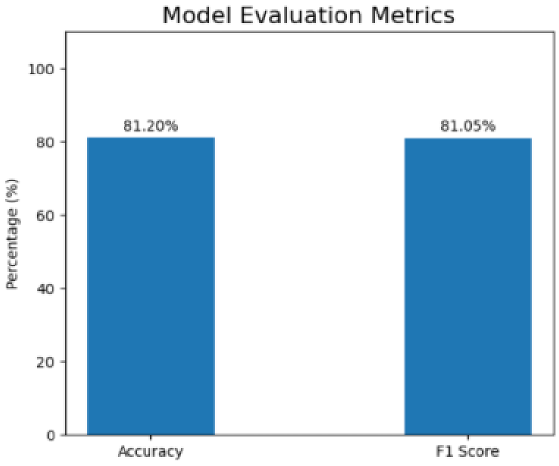
- **Accuracy:** 81.20%
- **Weighted F1 Score:** 81.05%

These results reflect strong generalization, especially considering the dataset's noisy, code-mixed nature and real-world imbalance in sentiment expression.

Visual Insights:

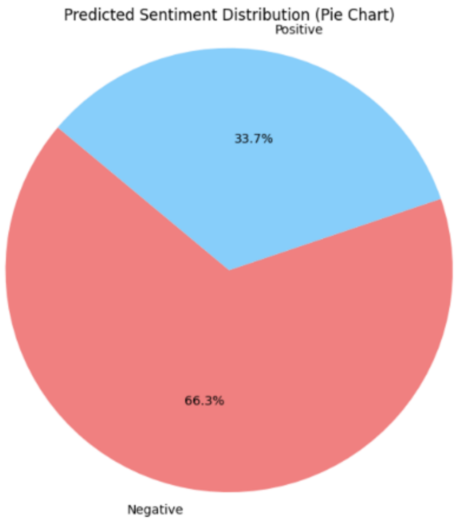
1. Model Evaluation Metrics (Bar Chart)

The bar chart on the left of the slide shows accuracy and F1 side-by-side, both close to 81%. This visual confirms that our model maintains consistent precision and recall, without skewing toward the majority class.



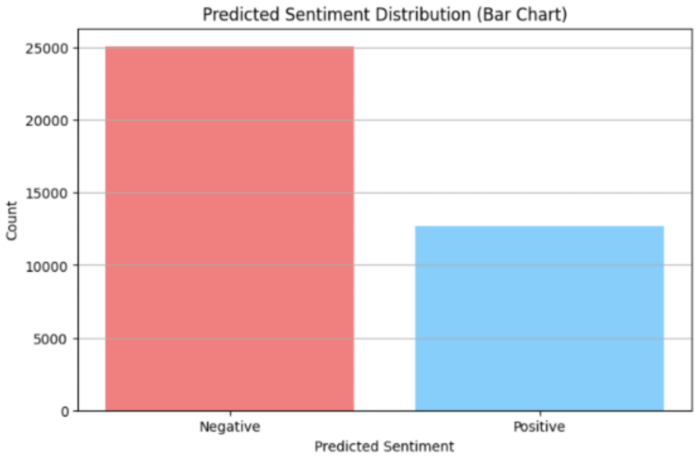
2. Predicted Sentiment Distribution (Pie Chart)

The middle chart shows that **66.3%** of the tweets were classified as negative and **33.7%** as positive. This aligns with expectations from public sentiment on real-world issues like politics and policy — most tweets naturally lean negative.



3. Predicted Sentiment Distribution (Bar Chart)

The right-side bar chart highlights the raw volume of predictions, again reflecting the dominance of negative tweets. It supports our decision to use weighted metrics, as a basic accuracy score would be misleading in isolation.


Predicted Sentiment Distribution (Bar Chart)
Count
Negative
Positive
Predicted Sentiment