Twitter Sentiment Analysis using NLP: A Data-Driven Approach

Om Tulsani[1], Prashant Kulkarni[2], and Shubhangi Tidake[3]
[1]*M. Sc. (Data Science) Student, Symbiosis Skill & Professional University, Pune*
[2]*Assistant Professor, Symbiosis Skill & Professional University, Pune*
[3]*Assistant Professor, Symbiosis Skill & Professional University, Pune*

***Abstract* - In the age of digital communication, social media platforms like Twitter have evolved into rich sources of real-time public opinion and sentiment. This study presents a systematic approach to sentiment analysis of Twitter data using Natural Language Processing (NLP). The research aims to classify tweets into positive and negative sentiments through the development of a sentiment classifier trained on preprocessed Twitter data. The methodology spans data acquisition, preprocessing, visualization via word clouds, model training using deep learning techniques, and final deployment strategies. The findings demonstrate that NLP-based sentiment analysis can effectively interpret public sentiment, offering insights for applications in marketing, politics, and public opinion mining. The developed model achieved 94% accuracy on test data, demonstrating high effectiveness in binary sentiment classification.**

***Keywords* - Deep Learning, Natural Language Processing, Sentiment Analysis, Text Classification, Twitter.**

I. INTRODUCTION

Social media has become a primary medium through which individuals express opinions, share feedback, and engage in public discourse. Among these platforms, Twitter stands out due to its real-time nature, brevity of content, and vast user base. With over 500 million tweets posted daily, it presents an invaluable corpus of text data that reflects public sentiment on topics ranging from product reviews to political events.

Sentiment analysis, a subdomain of Natural Language Processing (NLP), seeks to classify text based on emotional tone—commonly positive, negative, or neutral. In the context of Twitter, where messages are brief and often informal, sentiment analysis poses unique challenges, such as handling emojis, hashtags, slang, abbreviations, and sarcasm.

This research aims to develop an NLP-based sentiment classifier capable of analyzing Twitter data and categorizing sentiments effectively. The process encompasses several stages: data acquisition, cleaning, visualization, tokenization, model training, evaluation, and deployment.

II. LITERATURE REVIEW

A crucial first step in this project involved understanding the current landscape of Twitter sentiment analysis through an extensive review of existing studies. Previous research emphasizes the use of traditional machine learning models such as Naïve Bayes, Logistic Regression, and Support Vector Machines (SVMs), as well as more recent advances using deep learning architectures like LSTMs and Transformers [1].

Notably, datasets used in sentiment analysis research typically contain tweets labeled as positive, negative, or neutral. These datasets often originate from manually annotated corpora or public repositories such as Sentiment140 and Kaggle competitions [2].

For this study, a comprehensive dataset of tweets was acquired that included user-generated content with predefined sentiment labels. The dataset provided a balanced mix of positive and negative tweets to ensure unbiased training of the sentiment classifier.

III. RESEARCH AND METHODOLOGY

The methodology employed in this research involved a multi-phase pipeline designed to handle the intricacies of Twitter data and build a robust sentiment analysis model.

A. Data Preprocessing

Given the noisy and unstructured nature of tweets, preprocessing is critical. The preprocessing steps included:

Cleaning special characters and punctuation: All unnecessary symbols, mentions (@user), hashtags, and links were removed to simplify the text.

Stopword removal: Common English stopwords (e.g., "the," "is," "and") were filtered out using standard NLP libraries to reduce noise.
Emoji and emoticon handling: Emojis were either removed or converted to textual representation using emoji libraries.
Case normalization: All text was converted to lowercase to ensure consistency.
Tokenization: Tweets were broken down into individual words (tokens) using the Keras tokenizer.
Padding: To ensure uniform input size for the neural network, sequences were padded to a fixed length.

```
def clean_text(text):
    if not isinstance(text, str):  # Handle non-string inputs
        return text

    text = text.lower()  #  Convert to lowercase
    text = remove_html(text)  #  Remove HTML
    text = clean_url(text)  #  Remove URLs
    text = remove_punctuation(text)  #  Remove punctuation
    text = remove_stopwords(text)  #  Remove stopwords
    text = remove_emojis(text)  # Remove emojis
    text = lemmatize_text(text)  #  Lemmatization

    return text
```

B. Word Cloud Visualization

To better understand the content and frequency of words in positive and negative tweets, word clouds were generated. These visualizations revealed dominant words and emotional patterns in the dataset, aiding in feature selection and enhancing the interpretability of results.
Separate word clouds were created for each sentiment class to extract discriminative keywords. For example, positive tweets often included words such as "love," "great," "amazing," and "happy," while negative tweets featured words like "hate," "bad," "worst," and "sad."

C. Model Architecture

A sequential neural network was constructed using the Keras API with TensorFlow backend. The architecture included:
Embedding layer: For converting words into dense vector representations.
LSTM layer: For capturing temporal dependencies and context.
Dropout layer: To prevent overfitting.
Dense output layer: With softmax activation for sentiment classification.
The model was compiled using the categorical crossentropy loss function and Adam optimizer.

```
from sklearn.model_selection import train_test_split

X = df['message_lemmatized'] # feature matrix ( in this case it's just a vectore cause it's only one column)
y = df['label'] # target column

X_train1 , X_test , y_train1 , y_test = train_test_split(X,y , random_state = 42 , test_size = 0.2  , shuffle = True)
X_train , X_val , y_train , y_val = train_test_split(X_train1 , y_train1 , random_state = 42 , test_size = 0.15  , shuffle = True)
```

IV. EXPERIMENTAL RESULTS

A. Model Performance

Once the model was trained, it was subjected to rigorous evaluation using the reserved testing dataset. Model performance was assessed using standard classification metrics—accuracy, precision, recall, and F1-score—to provide a holistic view of its predictive capabilities.
The evaluation revealed that the model achieved over 94% accuracy on the test dataset, indicating high generalizability and effectiveness. The categorical crossentropy loss steadily decreased across epochs, while validation accuracy stabilized without overfitting.

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")
```

B. Confusion Matrix Analysis

To gain deeper insight into the model's behavior, a confusion matrix was generated. It revealed the number of correctly and incorrectly classified samples in each class. This visualization helped identify any bias or imbalance in prediction, especially in sentiment-sensitive contexts.

```
y_pred = model.predict(X_test)
cm = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1))
sns.heatmap(cm, annot=True, fmt='d')
```

The matrix demonstrated that the model had minimal false positives and false negatives, indicating balanced precision and recall.

The performance metrics are summarized in Table I.

TABLE I PERFORMANCE METRICS FOR SENTIMENT CLASSIFICATION

Metric	Positive Sentiment	Negative Sentiment
Precision	0.94	0.93
Recall	0.93	0.94
F1-Score	0.935	0.935

These results underscore the classifier's effectiveness in capturing both types of sentiment with nearly symmetrical accuracy and recall.

V. DEPLOYMENT AND PREDICTION PIPELINE

The culmination of the sentiment analysis project was to establish a reusable and automated prediction pipeline that could process new tweets and predict their sentiments in real time.

A. Model Serialization

To facilitate later use, the trained model and tokenizer were serialized and saved for future deployment scenarios.

```
model.save('sentiment_model.h5')
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)
```

B. Prediction System

A modular Python function was created to load the model and tokenizer, preprocess new tweets, and output the sentiment label. The pipeline ensured that all preprocessing steps—cleaning, tokenizing, and padding—were reapplied consistently.

This system can be integrated into social media monitoring dashboards or feedback analysis tools to monitor sentiment trends dynamically.

```
[ ]  def predict_sentiment(new_text):
        model = load_model('sentiment_model.h5')
        tokenizer = pickle.load(open('tokenizer.pkl', 'rb'))
        cleaned = clean_text(new_text)
        seq = tokenizer.texts_to_sequences([cleaned])
        padded = pad_sequences(seq, maxlen=maxlen)
        pred = model.predict(padded)
        return "Positive" if np.argmax(pred) == 1 else "Negative"
```

VI. DISCUSSION

The high-performing model demonstrated that deep learning methods, when paired with well-preprocessed text, can offer reliable sentiment classification, even on noisy, short-form text like tweets. Several insights emerged during this research:

Preprocessing was critical: Removing noise such as hashtags, mentions, and emojis significantly improved model accuracy.

Visualization guided design: Word clouds helped identify sentiment-heavy keywords, which aligned with classifier predictions.

Model selection matters: LSTM networks, with their ability to capture temporal word relationships, outperformed traditional classifiers previously experimented with.

Scalability: The saved pipeline ensures that the model can be extended or deployed in production environments with minimal overhead.

However, the model was limited to binary sentiment classification (positive/negative). Extending this to a multiclass classification including neutral or even emotions like joy, anger, or sarcasm, would be a worthwhile direction for future exploration.

VII. CONCLUSION

This research successfully implemented a sentiment analysis pipeline on Twitter data using advanced NLP techniques. Starting from data cleaning to model training and deployment, each phase was carefully designed and executed, resulting in a classifier capable of achieving 94% accuracy on test data.

In real-world applications, such a classifier can be utilized in brand monitoring, political opinion mining, stock market sentiment analysis, and customer service improvement. As public opinion increasingly influences societal and commercial decisions, tools like these become indispensable.

Moving forward, future work could involve incorporating transformer-based models such as BERT for improved contextual understanding, expanding the sentiment categories beyond binary classification, and including real-time data collection and continuous model updating for dynamic sentiment landscapes.

ACKNOWLEDGMENT

The author would like to thank the School of Data Science at SSPU, Pune, for providing the necessary resources and support for this research. Special appreciation goes to the faculty members who provided guidance throughout the project development.

REFERENCES

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1-135, 2008.

[2] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *CS224N Project Report*, Stanford University, 2009.

[3] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter Sentiment Analysis: The Good the Bad and the OMG!," in *Proc. 5th International AAAI Conference on Weblogs and Social Media*, Barcelona, Spain, 2011, pp. 538-541.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

[5] J. Brownlee, *Deep Learning for Natural Language Processing*, Machine Learning Mastery, 2017.

[6] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proc. 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014, pp. 1746-1751.

[7] D. Tang, B. Qin, and T. Liu, "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification," in *Proc. 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 1422-1432.

[8] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proc. 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, WA, 2013, pp. 1631-1642.