https://doi.org/10.46335/IJIES.2026.11.5.2 e-ISSN: 2456-3463
Vol. 11, No. 5, 2026, PP. 9-12

International Journal of Innovations in Engineering and Science, www.ijies.net

Twitter Sentiment Analysis Full-Stack Deployment

Dr. Pooja Raundale[1], Prof. Samuel Jacob[1], Suyash Gawande[2], Aditya Shirsat[2]

[1]Assistant Professor, Professor (Dept. of Computer Engineering)
Sardar Patel Institute of Technology, Mumbai, India – 400058
pooja@spit.ac.in, samuel.jacob@spit.ac.in
[2]Student, Dept. of Computer Engineering
Sardar Patel Institute of Technology, Mumbai, India – 400058

suyash.gawande24@spit.ac.in, aditya.shirsat24@spit.ac.in

Received on: 7 Aug,2026 ***Revised on: 12 September,2026*** ***Published on: 15 September,2026***

Abstract – *Social media platforms like Twitter provide a continuous stream of public opinion in the form of vast amounts of informal text, which presents challenges for automated analysis. This paper outlines a complete end-to-end system for performing sentiment analysis on Twitter data. The system utilizes robust preprocessing techniques, an LSTM-based deep learning classifier, and is deployed with a full-stack architecture featuring a Python backend and a React frontend. We focus on practical aspects such as dataset curation, tokenization, model training, and evaluation. The methodology is detailed, along with an experimental evaluation that includes an ablation study comparing LSTM with CNN and transformer-based models. A user study was also conducted to assess the system's usability. The paper acknowledges limitations like handling sarcasm and domain bias and suggests future enhancements, including transformer model fine-tuning and multilingual support.*

Keywords – Sentiment Analysis, Twitter, LSTM, Deep Learning, Natural Language Processing, Full-Stack Deployment, Evaluation, Ablation Study.

INTRODUCTION

Social media, especially Twitter, offers a rich source of user-generated content that reflects public sentiment on numerous subjects. The microblogging format of Twitter leads to concise and often informal text containing colloquialisms, emoticons, and hashtags, making it a unique challenge for sentiment analysis. This analysis has practical applications in areas like brand monitoring, political analysis, and social research.

While traditional methods such as bag-of-words and TF-IDF have achieved reasonable success [1], [2], the sequential nature of language is better captured by models that can learn temporal dependencies. Recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, have demonstrated improved performance in these scenarios [3]. More recently, transformer-based models like BERT and RoBERTa have set new state-of-the-art benchmarks [4], although they can be resource-intensive for real-time applications. This paper presents a production-focused approach that balances model effectiveness with deployability. Our contributions are:

1. A practical preprocessing pipeline tailored for Twitter text.
2. An LSTM-based classification model trained and packaged for inference.
3. A full-stack deployment pattern: Python REST API that serves model predictions and a React/TypeScript client that visualizes results.
4. A thorough experimental section including comparisons with CNN and BERT baselines, runtime profiling, and an ablation study.
5. A 25-participant user study assessing usability and perceived accuracy.

RELATED WORK

Initial approaches to sentiment analysis relied on classical supervised learning algorithms with manually engineered features [1]. A foundational study by Go et al. [2] specifically for Twitter utilized distant supervision through emoticons to generate labeled data. The advent

https://doi.org/10.46335/IJIES.2026.11.5.2 e-ISSN: 2456-3463
Vol. 11, No. 5, 2026, PP. 9-12

of deep learning brought forward the use of CNNs and RNNs for sentence classification, as demonstrated in the works of Kim [5] and Hochreiter & Schmidhuber [3] respectively. Transformer architectures, such as BERT, have recently achieved significant advancements across a wide range of NLP tasks [4]. Numerous practical systems integrate retrieval, classification, and visualization components for production workloads [6]. Our work positions itself by favoring an LSTM architecture optimized for latency and memory while presenting empirical contrasts to transformer-based methods.

CASE STUDY: SYSTEM OVERVIEW

We implemented an end-to-end Twitter Sentiment Analysis system with four main components:

- **Data Ingestion –** pipelines to collect and clean tweets (public datasets and sample streaming).
- **Preprocessing –** normalization, tokenization, and sequence padding.
- **Model Training –** LSTM classifier with embeddings.
- **Deployment –** Python backend exposing a REST API and a React frontend for visualization.

DESIGN PATTERNS AND PREPROCESSING

A. Data Sources

We used a combination of public benchmarks (Sentiment140, Kaggle Twitter datasets) and a curated in-house sample for domain-specific evaluation. Public datasets provide large-scale labeled examples; in-house sampling enables qualitative checks.

B. Cleaning and Normalization

The preprocessing steps apply deterministic transformations:

6. Remove URLs: regex pattern http[s]?://\S+.
7. Normalize mentions: replace @username with @USER.
8. Remove or preserve hashtags: strip the leading # but retain token (e.g., #COVID → COVID).
9. Expand common contractions (e.g., “can’t” → “cannot”).
10. Lowercase and trim whitespace.
11. Optional emoji handling: map common emojis to textual sentiment tokens.

C. Tokenization and Vocabulary

We use Keras Tokenizer with a vocabulary cap (e.g., 30k tokens). Out-of-vocabulary tokens are mapped to an <OOV> token. Sequences are padded/truncated to a fixed length (typically 50 tokens for tweets).

MODEL ARCHITECTURE

A. LSTM Classifier

The primary model architecture is as follows:

- **Embedding layer:** input dimension = vocabulary size, output dimension = 128.
- **LSTM layer:** 128 units, return sequences = false.
- **Dense layers:** 64 neurons (ReLU), dropout = 0.5.
- **Output:** Softmax over 3 classes (positive, neutral, negative).

Training used Adam optimizer, categorical cross-entropy loss, batch size 64, and early stopping based on validation loss.

B. Baselines

We compared the LSTM against:

- **CNN:** 1D convolutional layers with max-pooling (Kim-style).
- **BERT finetuned:** DistilBERT base fine-tuned on the same datasets.

METHODOLOGY

A. Problem Formulation

Given a tweet text t, the goal is to predict label $y \in \{0, 1, 2\}$ representing negative, neutral, and positive. The classifier is a function $f\theta$ parameterized by θ:

$$\hat{y} = arg\ maxc\ \ f\theta(t)c \quad (1)$$

Training minimizes categorical cross-entropy:

$$L(\theta) = -\ (1/N) \sum i=1N \sum c=13\ yi,c\ log\ f\theta(ti)c \quad (2)$$

B. Algorithm (Training Loop)

Algorithm 1: Training the LSTM classifier

Load dataset D = {(ti, yi)}i=1N
Preprocess tweets ∀ i : xi = preprocess(ti)
Tokenize and pad sequences ∀ i : si = tokenize(xi)
Initialize model parameters θ
For epoch = 1 to E:
  For batch in make_batches(s):
    Compute predictions ŷ = fθ(batch)
    Compute loss L
    Update θ ← θ − η∇ θL
  Validate on holdout set; implement early stopping

IMPLEMENTATION AND DEPLOYMENT

The deployment pattern follows a stateless model server with a lightweight frontend that consumes predictions via AJAX. The server hosts the trained model and

https://doi.org/10.46335/IJIES.2026.11.5.2 e-ISSN: 2456-3463
Vol. 11, No. 5, 2026, PP. 9-12

tokenizer artifacts (e.g., sentiment_lstm_model.h5, tokenizer.json).

A. Backend

The backend is a Python Flask/FastAPI server that:

- Loads the Keras model and tokenizer at startup.
- Exposes /predict endpoint accepting JSON: {"text":"..."}.
- Returns {"label": "...", "confidence": 0.xx}.

Key implementation details include using a thread- or process-based worker (e.g., Gunicorn) for concurrency, caching of frequent inputs (Redis), and input validation to prevent abuse.

B. Frontend

The UI (React + TypeScript + Tailwind) provides:

- Input textbox for tweets or batch file upload.
- Visualizations: sentiment distribution chart, recent predictions table, and raw confidence scores.
- Export options for reports (CSV).

EXPERIMENTS AND RESULTS

A. Experimental Setup

We evaluated models on a stratified split: 70% training, 10% validation, 20% test. Where public datasets were used, we followed standard preprocessing to ensure reproducibility.

Hardware:

- Training: NVIDIA T4 GPU, 16 GB RAM.
- Inference: commodity CPU (Intel Xeon), 8 cores.

B. Metrics

We report Accuracy, Precision, Recall, and F1-score (macro-averaged). For latency, we measure end-to-end time from HTTP request to response.

C. Main Results

Table 1 summarizes test-set results.

Table 1 – Model Performance on Held-Out Test Set

Model	Accuracy	Precision	Recall	F1
LSTM	0.86	0.85	0.84	0.845
CNN	0.82	0.81	0.80	0.805
Distil BERT	0.90	0.89	0.88	0.885

D. Ablation Study

Key observations from our ablations:

- Emoji mapping yields a 1.5% absolute F1 improvement.
- Embedding size beyond 128 gives diminishing returns.
- Sequence length 50 suffices for most tweets.

E. Runtime and Latency

Average inference latencies (single request):

- LSTM (CPU): 60–120 ms
- DistilBERT (CPU): 500–900 ms

Under concurrent load (500 users), the LSTM service scaled well.

USER STUDY AND QUALITATIVE EVALUATION

We conducted a 25-participant user study (students and non-technical users). We measured usability (System Usability Scale adapted), perceived accuracy, and trust.

Summary:

- SUS-like score: 78/100 (good usability).
- Perceived accuracy: 4.1/5.
- Requests: better handling of sarcasm and multi-lingual tweets.

Qualitative feedback emphasized that clear confidence scores and example clarifications improved trust.

DISCUSSION AND ERROR ANALYSIS

Errors were analyzed and categorized:

- **Sarcasm/Irony:** false positives/negatives where literal sentiment misleads model.
- **Contextual Dependencies:** tweets referring to prior conversation lacking context.
- **Domain Shift:** topic-specific vocabulary (e.g., finance, medicine) causing misclassification.

Mitigation strategies include data augmentation, context windows, and domain-specific fine-tuning. Ethical considerations include user privacy and bias detection.

FUTURE WORK

Future improvements include:

- Fine-tuning transformer models with distillation to retain speed.
- Multilingual modeling and code-switched text handling.
- Sarcasm detection module.
- Real-time streaming architecture using Twitter API.
- Aspect-based sentiment analysis.

https://doi.org/10.46335/IJIES.2026.11.5.2 e-ISSN: 2456-3463
Vol. 11, No. 5, 2026, PP. 9-12

CONCLUSION

We presented a full-stack Twitter sentiment analysis system balancing accuracy, latency, and usability. An LSTM-based model provides a practical trade-off point for real-time systems; transformer models (DistilBERT) yield better accuracy but at higher inference cost. Our comprehensive evaluation highlights the strengths and avenues for improvement. The modular architecture allows future upgrades.

ACKNOWLEDGMENT

Acknowledgment to person or the organization supported to the author for the research work. We thank Sardar Patel Institute of Technology for providing computational resources and guidance for this research work.

REFERENCES

[1] *B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval, 2008.*
[2] *A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," Stanford, 2009.*
[3] *S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, 1997.*
[4] *J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, 2019.*
[5] *Y. Kim, "Convolutional Neural Networks for Sentence Classification," EMNLP, 2014.*
[6] *D. Jurafsky and J. H. Martin, "Speech and Language Processing," Draft, 2021.*
[7] *N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," EMNLP, 2019.*
[8] *A. Dos Santos and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," COLING, 2014.*
[9] *T. Young et al., "Recent Trends in Deep Learning Based Natural Language Processing," IEEE Computational Intelligence Magazine, 2018.*

APPENDIX – DATASET STATISTICS

Dataset composition used for training and evaluation is summarized below.

Table 2 – Dataset Composition (Example)

Split	#Samples	Class distribution (P:N:Neu)
Train	100,000	45:35:20
Validation	10,000	45:35:20
Test	20,000	45:35:20

APPENDIX – TOKENIZATION AND HYPERPARAMETERS

Table 3 – Selected Hyperparameters

Parameter	Value
Vocabulary size	30,000
Embedding dim	128
LSTM units	128
Batch size	64
Epochs	10 (early stop)
Learning rate	1e-3 (Adam)
Sequence length	50

APPENDIX – SAMPLE API RESPONSE

`POST /predict`

Request:

```
{ "text": "I love the new phone! Battery life is amazing :)" }
```

Response:

```
{
  "label": "positive",
  "confidence": 0.9421,
  "scores": {
    "positive": 0.9421,
    "neutral": 0.0353,
    "negative": 0.0226
  }
}
```