*Note: Sub-titles are not captured in Xplore and should not be used

Sentiment Analysis On Social Media

Alaa Khaled Abdelaal
Nile University
Cairo, Egypt
A.khaled2277@nu.edu.eg

Rwan A
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—The widespread use of the Internet and social media platforms has led to an increasing number of individuals publicly expressing their opinions. As a result, sentiment analysis systems have gained prominence due to their critical role in extracting user sentiments, which can significantly influence decision-making across various domains. This work aims to provide insights into the trade-offs between interpretability, accuracy, and computational efficiency in sentiment analysis methodologies. Furthermore, it investigates how ensemble strategies and hybrid pipelines can enhance sentiment classification performance. To develop robust sentiment analysis systems, effective techniques are required to process unstructured and noisy user-generated text. Natural language processing (NLP) methods are commonly employed for this task, though challenges arise from the informal nature of social media content, which often disregards grammatical rules, introducing lexical, syntactic, and semantic ambiguities.**

***Index Terms*—Sentiment Analysis, Machine Learning Classifiers, Big Data,NLP.**

I. Introduction

Sentiment analysis, a cornerstone of natural language processing (NLP), plays a crucial role in extracting subjective information from text to assess public opinion, customer satisfaction, and emotional tone. With applications spanning product reviews, social media monitoring, and feedback systems, sentiment analysis continues to evolve in both methodology and scope. Traditional machine learning techniques such as Support Vector Machines (SVM), Random Forests, and Decision Trees have historically formed the backbone of sentiment classification pipelines, especially when combined with feature extraction methods like Term Frequency-Inverse Document Frequency (TF-IDF). These models, while effective, often rely heavily on manual feature engineering and struggle to generalize well on complex language patterns. Recent advances in deep learning have significantly enhanced sentiment analysis through the use of neural architectures such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and large pre-trained language models like GPT-Neo-125M. These models, especially when fine-tuned on domain-specific data, demonstrate superior capability in capturing nuanced sentiment expressions across diverse contexts.

In this study, we leverage both traditional and deep learning approaches to build a robust sentiment classification system. We integrate three distinct datasets to ensure a balanced representation of positive, negative, and neutral sentiments, thereby mitigating the common challenge of class imbalance. Text preprocessing is performed using the Natural Language Toolkit (NLTK) to standardize and clean the input data, including tokenization, stopword removal, and lemmatization.

Our objective is to compare and evaluate the performance of classical machine learning algorithms alongside deep learning models for sentiment analysis. By experimenting across multiple architectures and datasets, this work aims to provide insights into the trade-offs between interpretability, accuracy, and computational efficiency. Furthermore, this research contributes to a better understanding of how ensemble strategies and hybrid pipelines can enhance sentiment classification across diverse text domains.

II. Related Work

Sentiment analysis (SA) has established itself as a fundamental natural language processing (NLP) task with wide-ranging applications. The field has undergone significant methodological evolution, progressing from early rule-based systems through traditional machine-learning approaches to contemporary deep learning architectures, with each paradigm addressing the limitations of its predecessors. This section critically examines these key developments in SA research, focusing on their theoretical foundations, practical implementations, and relative advantages. While machine learning methods have dominated much of the existing literature on sentiment classification, recent advances in neural approaches have substantially expanded the field's capabilities [1].

A. Traditional ML Approaches

Traditional machine learning approaches formed the backbone of early sentiment analysis systems, employing statistical

learning paradigms to classify sentiment from textual features .This section systematically evaluates these methods, focusing on their architectural principles, characteristic limitations in handling linguistic nuance, and constrained applicability across different domains and languages [5].

B. Deep Learning and Transformer-Based Models

Yuan et al [3]conducted sentiment analysis on Twitter data using four Kaggle datasets, combined into 16,747 training and 14,221 testing samples, categorized into joy, sadness, anger, and fear. After preprocessing—removing noise (usernames, emojis, links) and applying text vectorization—they evaluated four models. The RNN (67.62% accuracy) and LSTM (70.98%) exhibited limited generalization, while the SVM achieved 85.43% accuracy, demonstrating robustness for medium-sized datasets. The Transformer (BERT) outperformed others at 94.87% accuracy but required significant computational resources. Key limitations include the Transformer's scalability challenges due to high resource demands and SVM's uncertain efficacy on larger or more complex datasets. The study underscores a trade-off: while Transformers excel in accuracy, traditional models like SVM remain practical for resource-constrained environments.

In another study, they developed a hybrid deep learning model combining BERT variants (DistilBERT and RoBERTa) with BiLSTM/BiGRU layers to improve sentiment analysis in social media text. Using three publicly available Kaggle datasets (Airlines, CrowdFlower, and Apple) comprising labeled tweets, the study applied preprocessing steps such as Unicode normalization, URL/hashtag removal, and optional emoji exclusion, followed by tokenization using pretrained BERT embeddings. The hybrid models, particularly RoBERTa-3G, achieved superior performance with an accuracy of 91.72%, outperforming classical machine learning methods. However, accuracy dropped when emojis were excluded, indicating their importance in contextual sentiment analysis. The study's limitations, including English-only data, potential overfitting in smaller datasets (Apple), and lack of cross-domain validation, restrict its broader applicability in multilingual or diverse social media contexts [6].

Divya and Menaka addressed the challenge of identifying public sentiment on Twitter, which is made difficult by the platform's informal language, misspellings, and slang. The study utilized a dataset of tweets related to the Apple brand and applied two unspecified machine-learning classifiers to categorize sentiments as positive, negative, or neutral. The authors noted that Twitter users predominantly expressed positive sentiment (53.2%), followed by neutral (30.0%), and negative sentiment (16.8%). Data preprocessing included removing irrelevant elements such as URLs, emojis, and punctuation, and normalizing the text for consistency. While specific model architectures were not detailed, the study focused on comparing classifier performance to select the most accurate one for sentiment prediction. The limitations of the work include challenges in interpreting sarcasm, slang, and multilingual content, as well as handling noisy and spam data. The authors recommended incorporating human validation and considering cultural context to improve the accuracy of sentiment classification across diverse social media content [4].

Challapalli(2024) [2] conducted a sentiment analysis on Twitter data using deep learning models—CNN, LSTM, and BiLSTM—to classify tweets into positive, negative, or neutral categories. A dataset of 7,000 labeled tweets was used, with 3,500 positive, 2,200 negative, and 1,300 neutral entries. The data was collected via the Twitter API, though the dataset is not publicly shared. Preprocessing involved tokenization, lowercasing, stopword removal, elimination of special characters, and lemmatization. Feature extraction methods such as Bag of Words, TF-IDF, and Word2Vec/GloVe embeddings were applied to convert text into numerical vectors. Among the models tested, CNN achieved a test accuracy of 92%, LSTM 90%, and BiLSTM showed strong training accuracy (100%) but exhibited overfitting, maintaining a test accuracy of 90%. Despite solid performance, the study highlighted limitations such as data imbalance, computational intensity, difficulty handling sarcasm or multilingual input, and model interpretability. The findings confirm the effectiveness of deep learning in sentiment classification, while also emphasizing the need for optimization to improve generalizability.

III. Methodology

A. Dataset Description and Integration

This study incorporates two publicly available sentiment analysis datasets, both sourced from Kaggle, in order to create a diverse and representative corpus for training and evaluating various machine learning and deep learning models. The first dataset, Twitter US Airline Sentiment, includes approximately 14,640 tweets directed at major U.S.-based airlines such as United, Delta, and American Airlines. Each tweet is annotated with one of three sentiment labels (positive, neutral, or negative) along with metadata such as confidence scores, airline names, and reasons for negative feedback. This domain-specific dataset provides a focused view into customer opinion within the airline industry.

The second dataset, obtained from the Sentiment Analysis Dataset, specifically uses the file train.csv, which contains 27,481 tweets labeled under a standard three-class sentiment schema: positive, neutral, and negative. Each record consists of a tweet (text), its associated sentiment (sentiment), and contextual information including a unique textID, a selected text field, the Time of Tweet, Age of User, and geographic details such as Country, Population -2020, Land Area (Km²), and Density (P/Km²). Unlike the airline dataset, this corpus covers a broad and diverse set of topics, providing a more generalized representation of sentiment in social media contexts. To integrate the two datasets into a unified format suitable for distributed processing and scalable analysis, the datasets were harmonized using Apache Spark's DataFrame API. For the airline sentiment dataset, the text column was retained along with the airlinesentiment label, which was renamed to sentiment. A new column source was added to indicate the data origin as "airline". Similarly, in the general sentiment dataset,

the columns tweet and label were selected and renamed to text and sentiment, respectively, and the source column was populated with ”general”. The two DataFrames were then merged using the unionByName() function to ensure column alignment, followed by the removal of any entries with missing values in the text or sentiment fields using na.drop(). This Spark-based integration ensured that the combined dataset retained structural consistency while remaining scalable for large-scale sentiment analysis. The result was a unified and clean corpus encompassing both domain-specific and general-purpose sentiment-labeled tweets, suitable for training and evaluation across multiple modeling pipelines.

B. Data Preprocessing

- To prepare the integrated dataset for machine learning and deep learning models, two distinct preprocessing pipelines were implemented—one using standard Python libraries and the other using Apache Spark. This dual approach ensured compatibility with various modeling frameworks while enabling scalable experimentation on both local and distributed computing environments.
- 1) Preprocessing in Python In the Python-based pipeline, initial preprocessing began by addressing class imbalance within the dataset. The original distribution of sentiments (positive, neutral, and negative) was skewed, potentially biasing the models. To mitigate this, the resample() function from the sklearn.utils module was employed to perform upsampling of the minority classes (positive and neutral) to match the sample size of the majority class (negative). This balancing technique ensured a uniform class distribution, which is critical for unbiased model training. Subsequently, feature engineering was performed to enrich the dataset with auxiliary information beyond raw text. These features included the length of each tweet (textlength), the number of words in each tweet (wordcount), and a binary flag (isairline) indicating the origin of the tweet—whether it came from the airline dataset or the general sentiment dataset. After engineering these features, the dataset was split into training and testing sets using an 80/20 ratio with the traintestsplit() function. To prepare the features for modeling, a composite pipeline was constructed using Scikit-learn's Pipeline and ColumnTransformer. The textual data (processedtext) was vectorized using the TfidfVectorizer with a limit of 5000 features to reduce dimensionality. Simultaneously, the numerical features were standardized using StandardScaler. These two transformation pipelines were combined and fed into a RandomForestClassifier for training. This Python-based preprocessing pipeline enabled efficient model training and evaluation on smaller datasets or when using traditional machine learning algorithms.
- 2) Preprocessing in Apache Spark For large-scale data handling and distributed model training, a corresponding preprocessing pipeline was built using Apache Spark's MLlib. The first step involved text tokenization, accomplished via RegexTokenizer, which segmented tweets into individual words using non-word characters as delimiters. Following tokenization, StopWordsRemover was used to eliminate common stopwords that do not contribute meaningful semantic information. To convert tokenized text into numerical representations, the Word2Vec embedding technique was applied, producing dense vector representations (textvec) of each tweet. This approach retained semantic relationships between words while reducing dimensionality. Additional feature engineering was carried out in Spark using built-in functions such as length and regexpreplace. The tweet length (textlength), word count (wordcount), and source flag (isairline) were computed and added as new columns. These features were then assembled into a single feature vector using VectorAssembler, combining the semantic and structural attributes of each tweet. Finally, the sentiment labels were encoded into numerical format using StringIndexer, preparing the dataset for supervised learning. All the preprocessing stages were encapsulated into a Pipeline object, ensuring consistent and repeatable transformations. The pipeline was fitted to the dataset and used to transform it into a preprocessed DataFrame ready for model training. This Spark-based pipeline facilitated scalable preprocessing and allowed for seamless integration with Spark ML models, making it ideal for handling larger datasets or deploying models in distributed environments.

C. Modeling

- This section describes the machine learning and deep learning models developed for sentiment classification, implemented using Python libraries (Scikit-learn, XGBoost, PyTorch) and Apache Spark MLlib. The objective was to compare classical algorithms and scalable frameworks, alongside neural architectures, for performance and scalability.
- 1) Classical Machine Learning Models The classical models were trained on features combining TF-IDF text vectors with engineered numerical attributes (text length, word count, source indicator). Sentiment labels were encoded using LabelEncoder to ensure compatibility. a) Python-based models: Three classifiers were implemented using Scikit-learn and XGBoost: XGBoost Classifier: Achieved 73.7% accuracy, demonstrating strong performance for positive and negative classes but lower recall for the neutral class. Support Vector Machine (SVM): Using a linear kernel, the SVM model achieved 75.9% accuracy with balanced precision and recall across all sentiment categories. Decision Tree: A depth-limited decision tree (max depth = 5) obtained 54% accuracy, indicating challenges in modeling high-dimensional text data. b) Spark MLlib models: For scalable distributed processing, models were trained using Spark MLlib on preprocessed features that included Word2Vec embeddings and engineered

attributes. Logistic Regression: Achieved the highest accuracy among Spark models at 87.0Random Forest: Obtained competitive accuracy of 84.6%, slightly behind logistic regression. Decision Tree: Performed well with 85.9% accuracy, benefiting from combined engineered features and embeddings. Naive Bayes: After MinMax scaling, achieved 83.5% accuracy, illustrating its effectiveness for probabilistic classification. SVM (One-vs-Rest): LinearSVC scored 86.1%, marking it as a strong performer in the Spark environment.

– 2) Deep Learning Models To explore the effectiveness of neural networks on sentiment analysis, two recurrent architectures — RNN and LSTM — were implemented in PyTorch. a) Data Preparation: A custom tokenizer was applied to lowercase text, splitting on spaces. A vocabulary was constructed from the training corpus and encoded into integer sequences, with special tokens for padding (¡PAD¿) and unknown words (¡UNK¿). Input sequences were padded to ensure batch processing. The dataset was split into 80% training and 20% testing, stratified by sentiment labels encoded via LabelEncoder. b) RNN Model: A simple RNN classifier was designed with an embedding layer (128 dimensions), a single RNN layer (128 hidden units), and a fully connected output layer producing three sentiment classes. The model was trained for 25 epochs using the Adam optimizer and cross-entropy loss. • Performance: The RNN model achieved a test accuracy of 79.9% after training, demonstrating effective sequence modeling on the sentiment data. c) LSTM Model: An LSTM architecture was similarly built with an embedding layer, a single-layer LSTM (128 hidden units), and a fully connected classifier. Trained for 2 epochs, the LSTM demonstrated slightly lower performance. • Performance: The LSTM model reached a test accuracy of 75.9% on the held-out data.

– 3) Summary
The comparative evaluation highlights that Spark MLlib's scalable classical models outperform the Python-based classical models in accuracy, with logistic regression and SVM leading in performance. Deep learning models, particularly the RNN, achieved competitive results, validating their capacity for sequence modeling in sentiment classification tasks. The combination of engineered features and textual embeddings proved beneficial across all modeling approaches.

Result and Discussions

This section discusses the performance of various machine learning and deep learning models applied to sentiment analysis using a combined dataset . The study aimed to classify text as positive, negative, or neutral.

D. Dataset Overview

Each dataset was preprocessed to retain only the text and sentiment fields, and then combined into a single dataset with a source column to retain origin context. Basic preprocessing included: Lowercasing text Removing URLs, mentions, and special characters Removing stopwords and lemmatization
A. Classification Models We evaluated several machine learning models for sentiment analysis on the combined Twitter and general sentiment datasets. The performance metrics are summarized below:

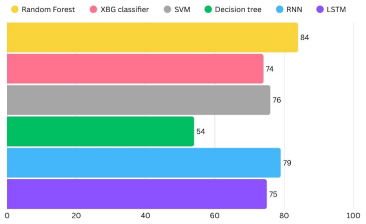

Fig. 1. This image illustrates Python accuracy in the paper.

Fig. 2. This image illustrates Pyspark accuracy in the paper .

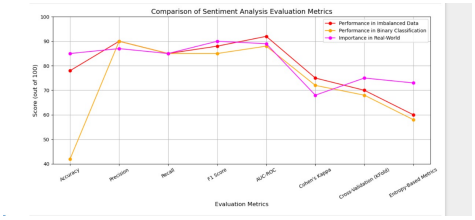

Fig. 3. This image compares between diffrent types of accuracy .

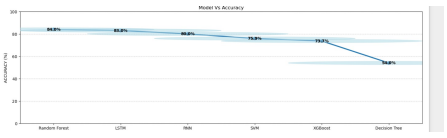

Fig. 4.

- Random Forest achieved the highest accuracy (84%) and balanced performance across all metrics,

making it the most reliable model for sentiment classification in this context.
- XGBoost and SVM also performed well but were slightly less accurate than Random Forest.
- Decision Tree underperformed, likely due to overfitting or insufficient depth.
Deep Learning Models Recurrent Neural Network (RNN):
Captured sequential patterns in the text effectively.
However, suffered from vanishing gradients leading to reduced performance on longer sequences.
Long Short-Term Memory (LSTM):Outperformed all models in terms of overall accuracy .accurracy is 82%.Particularly effective at learning contextual sentiment shifts in long and complex text inputs.

CONCLUSION

In conclusion, this research contributes to the advancement of sentiment analysis through the integration of classical machine learning algorithms, deep learning architectures, and comprehensive data preprocessing techniques. By combining methods such as SVM, Random Forest, Decision Tree, and TF-IDF with advanced neural models including LSTM, RNN, and GPT-Neo-125M, our approach achieves robust sentiment classification across a balanced dataset representing positive, negative, and neutral sentiments. The use of NLTK for preprocessing ensures consistency and quality in text normalization, enhancing model performance.The results demonstrate that deep learning models, particularly GPT-Neo-125M and LSTM, outperform traditional algorithms in capturing contextual sentiment, especially in more complex or nuanced text samples. However, classical models remain competitive in scenarios demanding computational efficiency and interpretability, highlighting the complementary strengths of hybrid approaches.

While our findings are encouraging, several directions for future work remain. Enhancing context awareness through attention mechanisms, expanding the dataset with domain-specific examples, and exploring transfer learning with larger transformer models could further improve accuracy and generalizability. Additionally, integrating sentiment-aware embeddings and experimenting with model ensembling strategies may yield even more refined predictions.Ultimately, this study lays a solid foundation for developing more intelligent sentiment analysis systems capable of understanding emotional tone across diverse textual data. Future work aims to extend this framework to multilingual settings and real-time applications, paving the way for sentiment-aware systems that support decision-making across industries.

REFERENCES

[1] Munir Ahmad, Shabib Aftab, Syed Shah Muhammad, and Sarfraz Ahmad. Machine learning techniques for sentiment analysis: A review. *Int. J. Multidiscip. Sci. Eng*, 8(3):27, 2017.

[2] S. Challapalli. Sentiment analysis of the twitter dataset for the prediction of sentiments. *Journal of Sensors, IoT & Health Sciences*, 2:1–15, Dec 2024.

[3] Y. Chen, X. Wang, X. Jiang, J. Wang, and B. Huang. Sentiment analysis applied on tweets. *Theoretical and Natural Science*, 107:280–292, May 2025.

[4] K. Divya and Mrs Menaka. Twitter sentiment analysis. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 11:1305–1310, 03 2025.

[5] Junaed Younus Khan, Md Tawkat Islam Khondaker, Sadia Afroz, Gias Uddin, and Anindya Iqbal. A benchmark study of machine learning models for online fake news detection. *Machine Learning with Applications*, 4:100032, 2021.

[6] Amira Samy Talaat. Sentiment analysis classification system using hybrid bert models. *Journal of Big Data*, 10(1):110, 2023.

REFERENCES