Predicting Sentiments of Tweets Using the Sentiment140 Dataset

Aditya Niraj Patkar
University of Maryland
College Park, MD
apatkar@umd.edu

***Abstract*—This report focuses on predicting the sentiment of tweets using the Sentiment140 dataset. It explores various architectures, including classical machine learning and deep learning models, for this binary classification problem. The significance of this study lies in understanding the polarized and potentially toxic nature of language on Twitter. The report first highlights the data exploration process, addressing challenges like large dataset size and untidiness, and details the data cleaning steps.**

The project utilized various frameworks and technologies such as CuDF, Dask, PySpark, AWS S3, AWS Sagemaker, Google Colab, Apache Parquet, and Weights & Biases for handling big data, model training, and performance tracking. Several models, including Logistic Regression, Support Vector Machines (SVM), Random Forest, Long Short Term Memory Network (LSTM), and RoBERTa, were trained and evaluated. The report discusses the results of these models, with particular focus on their F1 scores, precision, recall, accuracy, and run-time.

The outcome of this project demonstrates the superior performance of deep learning models over classical ones in sentiment analysis of tweets. It underscores the importance of retaining context especially for the RoBERTa model. It also highlights the critical role of parallelism achieving tools in managing large datasets. The report concludes with insights into possible improvements, such as further data cleaning, more training epochs for RoBERTa, and hyper-parameter tuning. The project's findings are significant for understanding Twitter sentiments and have insights into social media analysis and natural language processing.

I. Introduction

In an era where social media platforms like Twitter significantly influence public opinion, the ability to analyze sentiments expressed in tweets has become increasingly crucial. As discussed, twitter can be quite a polar platform for many with a significant amount of tweets that contain negative sentiment being consumed daily by millions. This project seeks to address this need to analyze sentiments expressed in tweets by hypothesizing that advanced machine learning techniques, particularly deep learning models, can effectively discern and classify the sentiment of tweets. This task builds upon previous work in the field of sentiment analysis, a domain that has seen significant advancements due to the evolution of natural language processing and machine learning technologies.

To understand the unique challenges presented by twitter data, it is imperative to understand what qualities a tweet possesses. Tweets are often concise, especially being limited to 280 characters. they contain frequent use of informal language and slang. the presence of ambiguous or noisy content along with emoticons, emojis and hashtags are some of the unique characteristics of tweets. By utilizing the power of both classical machine learning models and more sophisticated deep learning approaches, this project aims to navigate these challenges and provide a robust tool for understanding Twitter sentiments.

The implications of analyzing sentiment and polarity of tweets are more than a few. A tool that can analyze sentiments of tweets can be useful in different domains. In marketing, it could be utilized to analyze the consumer sentiment towards products or campaigns. Vitally, it could be used for crisis aversion, where recognizing and responding to negative sentiments can be pivotal. In social research, such a tool can help explore people's responses to various events or trends. In politics, it would be useful to gauge public sentiment towards policies or political events. In one such study, content of Twitter messages has been found to plausibly reflect the offline political landscape [1]. This finding suggests that analyzing political sentiment on Twitter can provide valuable insights into public opinion and political trends. By predicting the sentiment of political tweets, researchers can gain a deeper understanding of the political landscape and potentially predict election outcomes.

Another interesting study found that the sentiment expressed in tweets about a company is correlated with the rise and falls in stock prices of that company. This insight, highlighted in [2], suggests that sentiment analysis of Twitter data can be a valuable tool for predicting stock market movements as well. However, as per the study, further research is needed to investigate the causality and temporal dynamics between tweet sentiment and stock market trends.

Studying previous work done in the domain of sentiment and social media analysis helps give vital context about the task at hand. Deep Convolution Neural Networks (CNNs) combined with word embeddings obtained through unsupervised learning on large Twitter corpora have shown improved performance in Twitter sentiment classification compared to classical models, as reported in [3]. Additionally, the Long Short-Term Memory (LSTM) recurrent network has outperformed feature-engineering approaches for Twitter sentiment prediction, as mentioned in [4]. While deep learning models

have shown promising results, their black-box nature limits the ability to understand the underlying reasons for sentiment predictions. Developing interpretable models is also something that needs to be looked at, thus exploring classical machine learning methods coupled with classic features like TF-IDF and n-grams is essential.

The project, informed by previous research mentioned above, seeks to apply a variety of machine learning models to effectively classify tweet sentiments.

II. Methodology

The methodology that we followed was a structured approach, beginning with understanding the Sentiment140 dataset [5], which includes a substantial number of tweets with predefined sentiments. This section covers the initial analysis of the dataset, emphasizing its primary attributes and understanding some of the underlying patterns. The data cleaning process addresses removing noise and normalizing text. Following this, we detail the application of various models: classical machine learning techniques like Logistic Regression, Random Forest and SVM, along with advanced deep learning methods such as LSTM and RoBERTa, including their setup and rationale. Each model's performance is assessed using key metrics like F1 score, recall, precision, accuracy and runtime to ensure a thorough evaluation.

A. Understanding the Data

For the task of predicting the sentiment of tweets, we utilized the Sentiment140 dataset. This dataset includes 1.6 million tweets, each labeled with a sentiment target (positive or negative). The dataset, compiled by Alec Go, Richa Bhayani, and Lei Huang from Stanford, is evenly divided into 800,000 positive and 800,000 negative tweets. The dataset presents challenges such as its large volume and the presence of untidy elements like usernames and URLs. Each tweet, limited to 280 characters, often features informal language, emoticons, and specific Twitter elements like hashtags.

B. Data Cleaning

The data cleaning process for the Sentiment140 dataset was both thorough and essential for the subsequent sentiment analysis. While looking at random examples of tweets, we encountered an anomaly where some tweets exceeded Twitter's 280-character limit, which can be seen clearly in figure 1. This was attributed to the presence of HTML content embedded within the tweets. To address this, BeautifulSoup was used to convert HTML to text, effectively reducing the length of these tweets to fit within the standard Twitter character limit.

Further cleaning involved the use of regular expressions to remove usernames and URLs from tweets. These elements, while common in Twitter data, are irrelevant to the sentiment analysis and could potentially skew the results. Non-ASCII characters were also replaced with a placeholder, and any non-alphabetic characters were removed using regex, which helped in standardizing the text data.

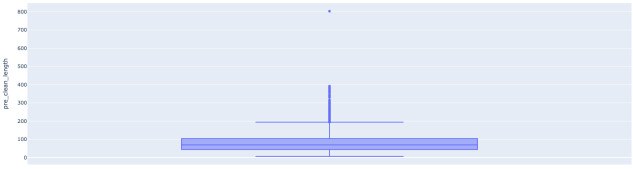

Fig. 1. Distribution of Tweet Length Before Cleaning

Stop-words, commonly used words that do not contribute significantly to the sentiment, were eliminated. In some cases, this led to rows becoming null as they were entirely composed of stop-words, necessitating their removal from the dataset. It is also important to note that the tweets without stop-words were stored in a new column, preserving the original cleaned tweet. It was to see if context aware models like RoBERTa would benefit from the semantic and syntactic structure of the tweets.

The final step in the cleaning process was to recalculate the length of each tweet after cleaning, ensuring that all tweets now conformed to Twitter's character limit and were in a more analyzable format. The difference in lengths of tweets before and after cleaning can be seen in figure 2. It can also be seen that the distribution of length is similar for both the positive (0) and negative (4) labels after cleaning. This comprehensive cleaning process was critical in helping the models converge better.

The data cleaning process was further enhanced by using Dask data-frames for efficiency [6]. Dask, a flexible parallel computing library for analytics, allowed for handling large datasets more effectively. It utilizes all available cores on the CPU of the machine by slicing the data-frame into equivalent number of pieces. It also applies the concept of lazy computing by using the CPU cores only when necessary. Dask was chosen for its immensely faster implementation of the 'apply()' method as compared to pandas. The same data cleaning function required 20min 23s of CPU time using pandas apply(). But by using Dask, this time was slashed to 3min 18s. This choice was crucial considering the massive size of the Sentiment140 dataset.

By leveraging Dask's capabilities, the data cleaning process was not only thorough but also time-efficient, ensuring that the cleaned dataset was ready for the subsequent stages of modeling and analysis without significant delays. The cleaned data file along with all other intermediate artifacts were stored in AWS S3 for easy and fast retrieval and reliability.

C. Data Exploration

After getting the data cleaned, a more detailed look at the data was needed to understand it better. Below is a detailed analysis of the data visualization that was performed along with some takeaways.

1. Word Cloud: In the data exploration phase, separate word clouds were generated for negative and positive tweets. Initially, these word clouds were dominated by stopwords, but

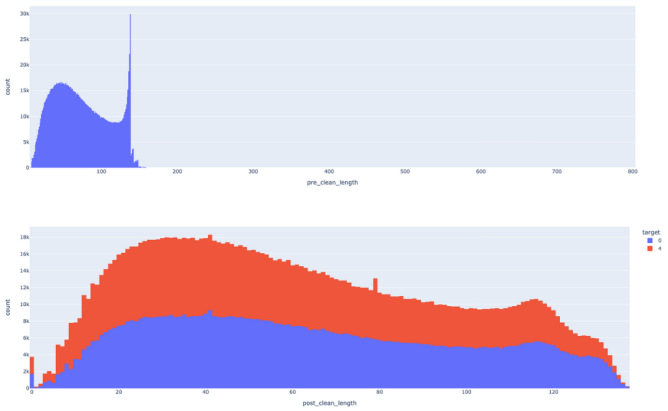

Fig. 2. Distribution of Tweet Length Before and After Cleaning

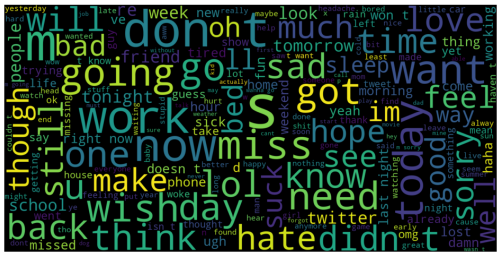

Fig. 3. Word Cloud Without Stop Words for Negative Tweets

after their removal, distinct patterns emerged. Negative tweets frequently included words such as ”hate,” ”sorry,” ”bad,” ”missed,” and ”suck.” Conversely, the word cloud for positive tweets revealed words like ”awesome,” ”good,” ”love,” ”fun,” and ”thank.” These distinct word choices highlight key lexical differences between positive and negative sentiments, offering valuable insights for the tweet classification process. Figures 3 and 4 show these clouds respectively.

2. Word Frequency Analysis: In the word frequency analysis, the frequency of specific words was examined for its potential influence, particularly in classical models using Tf-Idf features. Analyzing the top 50 tokens in positive and negative tweets, it was noted that while certain words might hint at a tweet's sentiment, frequently occurring neutral words like “just” and “day” do not distinctly belong to either sentiment category. This observation indicates the presence of common, non-indicative words within the most frequent tokens, which

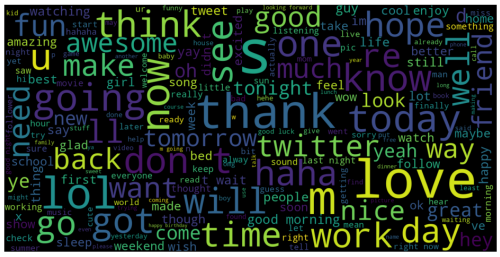

Fig. 4. Word Cloud Without Stop Words for Positive Tweets

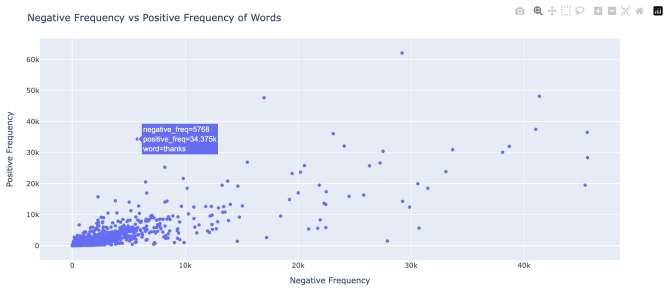

Fig. 5. Frequency of Words in Negative Tweets vs Positive Tweets

is in accordance with the Zipf's law.

Plotting the frequency of words in negative tweets against their frequency in positive tweets yielded some interesting results. While most words had a negative and positive frequency below 10000, some words had a massive difference between their negative and positive frequency. For instance, ”thanks” appeared 34,375 times in positive tweets but only 5,768 times in negative ones. Conversely, ”miss” was found 30,713 times in negative tweets and just 5,676 times in positive ones. These significant frequency disparities underscore the relevance of certain words in classifying tweet sentiment. Figure 5 shows the difference in negative and positive frequencies of different words, concentrating on the word ”thanks.”

D. Classical Machine Learning Methods

In this project, the initial focus was on classical machine learning methods. The techniques implemented included Logistic Regression, Support Vector Machine (SVM), and Random Forest. These methods were chosen for their proven effectiveness in text classification tasks.

The process of implementation was similar for all the methods except the model and features used. The process involved first setting up a Spark Context to leverage PySpark's distributed computing capabilities. The model configuration parameters were set and logged to Weights & Biases, an experiment tracking platform. The data was split into training, validation and test set. A series of steps were performed on the data which was in the form of a Spark DataFrame. including data tokenization, feature extraction (using n-grams, TF-IDF, and/or Word2Vec), and label indexing. A PySpark pipeline was established to streamline these processes. Model training was conducted using PySpark ML to handle the dataset's large size efficiently. Finally, the model was evaluated by calculating evaluation metrics including F1, Recall, Precision and Accuracy.

PySpark played a crucial role in enabling the processing of large-scale data and facilitating the efficient training of models. It basically allows for the distribution of computational tasks across multiple nodes, significantly speeding up the process. While the training for this paper was done on a single machine with 8 cores which were used as nodes, this method is scalable to multiple machines without any code modification. Weights & Biases, on the other hand, was used for experiment tracking.

It enabled logging of hyper-parameters, monitor model performance in real time, and visualize results, which greatly helped in fine-tuning the models and comparing their performance. This combination of tools helped optimize the workflow.

1. Logistic Regression + Hashing TF + IDF: Logistic Regression model was applied using a combination of Hashing Term Frequency (TF) and Inverse Document Frequency (IDF) for feature transformation. Hashing TF converts tweet texts into numerical feature vectors, mapping each word to a fixed-size hashing space, by counting the occurrence of each word, thus reducing dimensionality and computational complexity. IDF is then applied to these vectors to diminish the impact of frequently occurring words that might be less informative. The resulting feature is then fed to the logistic regression model which is a linear classifier model that predicts the sentiment of each tweet, determining the probability of a tweet being positive or negative. The model was trained for maximum 200 iteration with a regularization parameter of 0.001 and elasticnet parameter of 0.001. The hash size was 32 and minimum document frequency for IDF was 5.

2. Logistic Regression + N-Gram + Count Vectorizer + IDF: In this approach, 1-gram, 2-gram, and 3-gram features combined with Count Vectorizer and IDF (Inverse Document Frequency) were paired with Logistic Regression. N-grams are useful in understanding the context and structure of text data. An n-gram is a contiguous sequence of n tokens from a given sample of text. Unigrams consider individual words, while bigrams and trigrams consider pairs and triples of consecutive words. This approach enables the capture of more contextual information compared to unigrams alone. For instance, while a unigram model would treat "not good" as two separate words, a bigram model captures the combined negative sentiment of this phrase. The Count Vectorizer transforms the tweets into a matrix of token counts, capturing the frequency of word combinations up to three words in length. IDF is then applied to this matrix to reduce the weight of more common words. This is combined with a logistic regression model to classify the sentiment. This method provides a more nuanced understanding of the context and sentiment in tweets by considering not just individual words, but also their combinations. The configuration parameters were the same as the model explained above except for the vocabulary size which was set to 5000.

3. Logistic Regression + Word2Vec + 3-gram: In this approach, Word2Vec was combined with 3-gram features to enhance the Logistic Regression model. Word2Vec is a method for representing words in a high-dimensional vector space, capturing semantic relationships between words. By integrating this with 3-gram features, which consider sequences of three consecutive words, we were able to capture both the semantic nuances and the contextual information of word sequences. The window size for Word2Vec was set to 5 while the vector size was set to 65.

4. SVM + N-Gram + Count Vectorizer + IDF: Support Vector Machine (SVM) was used in conjunction with n-gram features, Count Vectorizer, and IDF. SVM is a powerful and versatile machine learning model that works by finding the hyperplane that best separates different classes in the feature space. In the context of sentiment analysis, SVM analyzes the vectorized tweet data (transformed by n-gram, Count Vectorizer, and IDF) to classify a tweet as negative or positive. The strength of SVM is that it is able to handle high-dimensional data, making it highly suitable for text classification tasks.

5. Random Forest + N-Gram + Count Vectorizer + IDF: The Random Forest algorithm was combined with N-Gram features, Count Vectorizer, and IDF. Random Forest, an ensemble learning method, constructs multiple decision trees during training and outputs the mode of the classes of the individual trees. This method is particularly effective due to its ability to handle large datasets with high dimensionality. While the training time for random forest is very high, the inference is very fast as it needs to just check the conditions which are predefined. A key advantage of Random Forest is its interpretability. The model provides insights into which features (words or word combinations) are most influential in determining sentiment, making it a valuable tool for understanding the underlying patterns in the data. The vocab size for count vectorizer was set to 10000 and the maximum depth for the random forest trees was set to 5.

E. Deep Learning Methods

We implemented LSTM (Long Short-Term Memory) and RoBERTa (a robustly optimized BERT approach) [7] models. These advanced techniques were chosen to capture complex patterns in the data. For effective training and evaluation of these models, we utilized Google Colab with A100 GPU support, which provided the necessary computational power. Weights & Biases was used for experiment tracking, hyper-parameter tuning. HuggingFace was used to save the final RoBERTa model. This section offers an insight into the implementation of the said deep learning methods.

1. Long Short-Term Memory (LSTM) Network: An LSTM model was Implemented as the first deep learning method. LSTM, or Long Short-Term Memory, is a type of recurrent neural network that is particularly well-suited for processing sequences of data, like text. This makes it ideal for our task of analyzing tweets, as it can effectively capture the context and sequence of words within a tweet.

The LSTM model has a unique architecture that includes memory cells and three types of gates: input, output, and forget gates. These work together to regulate the flow of information, allowing the model to remember important details over long sequences. This is particularly beneficial in this task, as it enables the model to focus on the most relevant aspects of the text that contribute to the overall sentiment.

Coupled with LSTM, we used GloVe (Global Vectors for Word Representation) embeddings [8] as the numerical vector representations of the tokens. GloVe embeddings are pre-trained word vectors that provide a dense representation of words based on their co-occurrence in a large corpus. For example, two synonyms would have similar vectors. By using

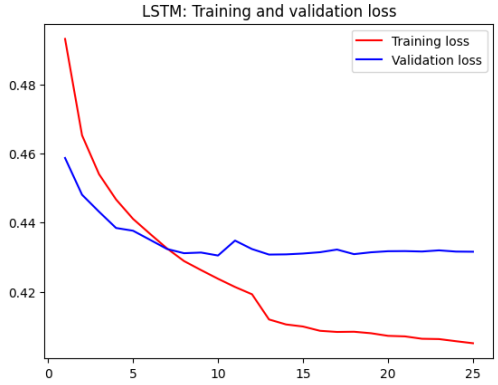

Fig. 6. Training and Validation Loss vs Ppochs for LSTM

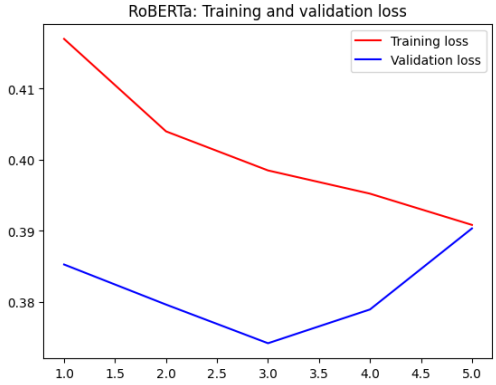

Fig. 7. Training and Validation Loss vs Epochs for LSTM

these embeddings, our LSTM model was able to understand the semantic relationships between words.

For the model architecture, we started with an embedding layer loaded with GloVe embeddings to convert the tweet text into vector representations. We then added an LSTM layer, which processes these embeddings in sequence, capturing the context and relationships between words. To improve the model's generalization, we included dropout layers to reduce overfitting. The architecture also featured dense layers with Leaky ReLU to classify the sentiment of the tweets. The final layer had a sigmoid activation function which denotes the probability of a word being positive.

In terms of the training process, we tuned the hyperparameters of the LSTM model, including the number of units in the LSTM layer and the dropout rate. We used Binary Cross-Entropy as the loss function, which is suitable for binary classification tasks like ours. The Adam optimizer was chosen for its efficiency in updating the model's weights. The learning rate was adjusted using a ReduceOnPlateau scheduler, which would reduce the learning rate once it starts to stagnate. The model was trained for 25 epochs with a minimum learning rate of 0.0001.

Overall, the combination of LSTM with GloVe embeddings was useful in understanding the sentiment expressed in tweets. Leveraging the sequential nature of text and the semantic information encoded in GloVe embeddings, classification results were satisfactory. Figure 6 shows the changes in training and validation loss with respect to training epochs. As the model was trained, the training loss kept going down, but after a point, the validation loss did not go down.

2. RoBERTa: RoBERTa (Robustly Optimized BERT Approach) is an advanced adaptation of BERT (Bidirectional Encoder Representations from Transformers) built by Meta (previously Facebook) AI . It modifies key aspects of BERT, including removing the next-sentence pretraining objective and training with larger batches and learning rates, leading to a much improved performance.

In this project, the RoBERTa model was used in two training scenarios: one with stopwords removed from the tweets and another with the original text. This dual approach was used to compare the impact of stop-words on the model's performance. The training process involved utilizing the HuggingFace library to load the RoBERTa model and tokenizer. The tweets were tokenized and converted into TensorFlow datasets, suitable for training in RoBERTa's framework.

Training was conducted on powerful A100 GPUs, leveraging their computational capabilities for efficient model training. It took around 13 hours for 5 epochs. We used the Adam optimizer with a linear learning rate scheduler, starting with 1e-5, decreasing up to a minimum of 1e-7, and trained the model for five epochs. Binary Cross-Entropy was chosen as the loss function, appropriate for the binary classification task. Throughout the training, model's performance was monitored, particularly focusing on the validation loss. The model with the least validation loss was saved as the best model. Finally, the performance metrics and model outputs were logged and analyzed using Weights & Biases, providing valuable insights into the model's efficacy and areas for improvement.

Figure 7 shows the change in training and validation loss with respect to epochs. While training loss went down, validation loss increased after first 3 epochs. This shows that the model gradually over-fit to the training data. Model corresponding to the least validation loss was saved and loaded.

III. Results

A. Classical Machine Learning Results

Analyzing the results of the classical machine learning models from table I provides a detailed insight into their performance:

1) **Logistic Regression with Hashing TF + IDF**: This model showed good overall performance, with balanced precision, recall, F1 score, and accuracy, boosted by the fastest run-time. It shows the effectiveness of this feature set for sentiment analysis.
2) **Logistic Regression with 1 + 2 + 3 gram + Count Vectorizer + IDF**: This approach outperformed the other methods tried within classical machine learning, achieving the highest scores across all metrics, highlighting the value of combining multiple n-grams. However, this came at the cost of a little increased runtime.

Model	Feature Set	Precision (%)	Recall (%)	F1 Score (%)	Accuracy (%)	AUC (%)	Runtime
Logistic Regression	Hashing TF + IDF	77.03	76.97	76.96	76.58	84.07	**2m 31s**
Logistic Regression	1 + 2 + 3 gram + Count Vectorizer + IDF	**78.17**	**78.03**	**78.05**	**78.06**	**85.62**	6m 14s
Logistic Regression	3 gram + Word2Vec	61.64	63.43	61.66	63.47	67.16	7m 35s
SVM	N-gram + Count Vectorizer + IDF	77.92	77.37	77.64	77.46	85.28	4m 25s
Random Forest	N-Gram + Count Vectorizer + IDF	64.47	60.56	62.45	60.61	67.01	6m 14s

TABLE I
PERFORMANCE METRICS OF CLASSICAL ML MODELS

3) **Logistic Regression with 3 gram + Word2Vec**: This model's performance was notably lower in all metrics, suggesting that 3-gram features combined with Word2Vec might not capture the sentiment as effectively for this specific dataset. Coupling 1 + 2 + 3 grams with Word2Vec can be tried in the future.
4) **SVM with N-gram + Count Vectorizer + IDF**: The SVM model showed competitive performance, especially in terms of AUC, indicating its robustness in distinguishing between classes, though it fell slightly behind the best Logistic Regression model.
5) **Random Forest with N-Gram + Count Vectorizer + IDF**: While not as high-scoring as the other models, Random Forest's interpretability is a key advantage, offering insights into feature importance. Though, this advantage cannot justify the low performance over all the metrics.

The varied performance shows how different models and feature representations can impact the ability to accurately classify sentiments in tweets.

B. Deep Learning Results

1) **LSTM with GloVe Embeddings**: This configuration demonstrated a high recall but lower precision. Its F1 score and accuracy were notable, suggesting a balanced approach to sentiment classification, though it fell short of the RoBERTa model's performance. The LSTM's relatively shorter runtime still makes it a viable option. Figure 8 shows how both training and validation accuracy increased with respect to epochs.
2) **RoBERTa with Stop Words Removed**: This version achieved a good balance between precision and recall. Its accuracy and F1 score were superior to the LSTM model. However, the 11-hour runtime is a significant consideration.
3) **RoBERTa without Stop Words Removal**: This approach recorded the highest precision, F1 score, and accuracy, outperforming all other models. In fact, according to [9], it achieved higher accuracy than the 3rd ranked Sentiment140_XLNET_5E model on the sentiment140 dataset. The increase in these metrics suggests that retaining stop words provided big contextual benefits. Nonetheless, this came at the cost of the longest runtime, over 13 hours. Figure 9 shows the increase in training and validation accuracy with respect to epochs.

Comparing these results with classical models, the deep learning approaches, particularly RoBERTa, demonstrate superior performance. This shows their advanced capability in decoding complex patterns in tweets. However, this high performance is offset by longer training times, which is an important trade-off to consider between computational efficiency and model effectiveness.

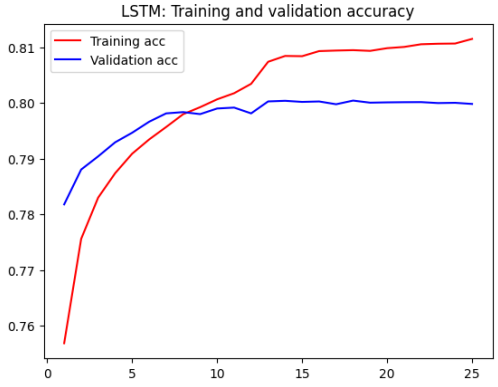

Fig. 8. LSTM: Accuracy vs Epoch

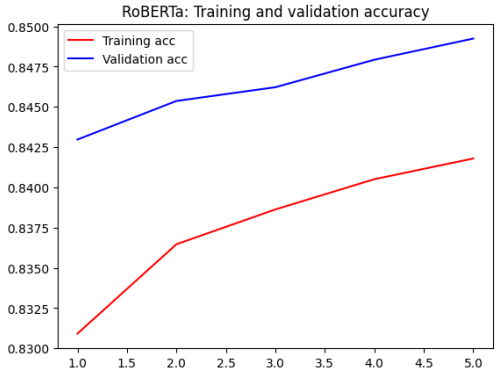

Fig. 9. RoBERTa: Accuracy vs Epoch

IV. CONCLUSION

The project on predicting the sentiments of tweets using the Sentiment140 dataset provided significant insights into the application of machine learning and deep learning models for sentiment analysis. The journey began with a comprehensive data cleaning process, using tools like Dask for efficient handling of the large dataset. The cleaned dataset was then explored through methods like word clouds and frequency analysis, revealing patterns in the language used in positive and negative tweets.

Model	Feature Set	Stop Words	Precision (%)	Recall (%)	F1 Score (%)	Accuracy (%)	Runtime
LSTM	GloVe Embeddings	Yes	75.29	**87.15**	80.79	80.09	**31m 37s**
RoBERTa	Using Default Tokenizer	Yes	77.42	86.26	81.60	80.55	11h 15m
RoBERTa	Using Default Tokenizer	No	**83.27**	86.87	**85.02**	**84.71**	13h 14m

TABLE II
PERFORMANCE METRICS OF CLASSICAL ML MODELS

The application of classical machine learning models, including Logistic Regression, SVM, and Random Forest, showed the effectiveness of traditional methods in sentiment analysis. These models, each with their unique strengths, provided a foundation for understanding the nuances of sentiment classification. The performance of these models varied a lot based on the feature sets used, with combinations of N-grams, TF-IDF, and Word2Vec being highly effective.

The deep learning models, specifically RoBERTa, demonstrated its advanced capability in understanding complex language patterns. The LSTM model, with its sequential data processing ability and the use of GloVe embeddings, was effective in capturing the context of tweets. RoBERTa, on the other hand, showed superior performance, particularly when trained without removing stop words. This indicated the importance of retaining the full context of the tweets for sentiment analysis.

The project's findings show the potential of both classical and deep learning models in analyzing sentiments on social media platforms like Twitter. While deep learning models, especially RoBERTa, exhibited higher accuracy and F1 scores, they required considerably longer training times along with higher computation power and cost. This trade-off between performance and computational efficiency is an important consideration in the practical application of these models.

In conclusion, this study provides a comparative analysis of various machine learning approaches. It demonstrates the strengths and limitations of different models and feature sets in classifying tweet sentiments using the sentiment140 dataset. Future work could explore combining classical and deep learning methods or investigating more efficient training strategies for complex models like RoBERTa in an ensemble approach.

REFERENCES

[1] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 4, no. 1, pp. 178–185, May 2010. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14009

[2] V. S. Pagolu, K. N. Reddy, G. Panda, and B. Majhi, "Sentiment analysis of twitter data for predicting stock market movements," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016, pp. 1345–1350.

[3] D. Stojanovski, G. Strezoski, G. Madjarov, and I. Dimitrovski, "Twitter sentiment analysis using deep convolutional neural network," in *Hybrid Artificial Intelligence Systems*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:42683921

[4] X. Wang, Y. Liu, C. Sun, B. Wang, and X. Wang, "Predicting polarities of tweets by composing word embeddings with long short-term memory," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1343–1353. [Online]. Available: https://aclanthology.org/P15-1130

[5] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[6] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," pp. 126–132, 01 2015.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[8] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[9] "Papers with Code - Sentiment140 Benchmark (Text Classification) — paperswithcode.com," https://paperswithcode.com/sota/text-classification-on-sentiment140, [Accessed 12-12-2023].

APPENDIX

A. GitHub Repository

The source code and additional resources for this project are available in the following GitHub repository:

- GitHub Repository: https://github.com/adityapatkar/SentimentSifter

B. HuggingFace Model

The trained RoBERTa model utilized in this project can be accessed on the HuggingFace model hub at the following link:

- HuggingFace Model: https://huggingface.co/adityapatkar/TweeBERTa