Mini Project 2 : Sentiment analysis with tweets

Harry Setiawan Hamjaya

2302000

***Abstract*— In this project, we delve into sentiment analysis using the Sentiment140 dataset, aiming to construct and evaluate two machine learning models for sentiment analysis. Leveraging traditional and advanced methodologies such as Naïve Bayes, Logistic Regression, Support Vector Machine, and Long Short-Term Memory (LSTM), we rigorously experiment with model architectures, hyperparameter tuning, and dataset preprocessing to discern sentiments from approximately 1,600,000 annotated tweets. Our findings highlight the strengths and limitations of each approach, showcasing the importance of adaptability and innovation in navigating the dynamic landscape of sentiment analysis research. Through this exploration, we contribute valuable insights and implications for the advancement of natural language processing applications.**

Keywords: Logistic Regression, Random Forest Classifier, Bank Clients, Campaign Interaction, and Economic Indicators.

I. Introduction

Sentiment analysis, a fundamental task in natural language processing, holds immense significance in deciphering the sentiments and opinions embedded within textual data. In the contemporary landscape of social media, where millions of users express their thoughts and emotions through platforms like Twitter, understanding sentiment becomes pivotal for various applications ranging from brand monitoring to market analysis and beyond.

In this report, we embark on an insightful exploration of sentiment analysis techniques using the Sentiment140 dataset. Crafted by Alec Go, Richa Bhayani, and Lei Huang, three distinguished graduate students from Stanford University, this dataset encompasses a vast collection of tweets, each annotated with sentiments. With approximately 1,600,000 tweets at our disposal, we are presented with a rich tapestry of textual data ripe for analysis and interpretation.

Our primary objective in this endeavor is to construct and evaluate two distinct machine learning models for sentiment analysis. As aspiring machine learning practitioners, we are tasked with not only implementing conventional methodologies but also delving into the realm of advanced architectures and innovative approaches. Our toolkit includes a diverse array of techniques ranging from traditional methods such as Naïve Bayes and Support Vector Machines to state-of-the-art deep learning models like Long Short-Term Memory (LSTM).

Through meticulous experimentation, rigorous evaluation, and critical analysis, we aim to unravel the nuances of sentiment analysis and elucidate the efficacy of different techniques in discerning sentiments from textual data. Moreover, we endeavour to showcase our ability to adapt, innovate, and navigate the complex landscape of machine learning, thereby contributing to the ongoing discourse in sentiment analysis research.

II. Exploratory Data Analysis

A. *Label*

The dataset comprises approximately 40,077 annotated tweets named as sentiments, with 20,098 labeled as negative sentiments and 19,979 labeled as positive sentiments. Each tweet is associated with one of these sentiment labels, providing a balanced distribution between negative and positive sentiments. This balanced dataset allows for effective training and evaluation of sentiment analysis models, ensuring that the models can learn to accurately distinguish between negative and positive sentiments. The dataset's balance facilitates robustness and generalizability, enabling the models to effectively capture the nuances and variations in sentiment expression across different tweets.

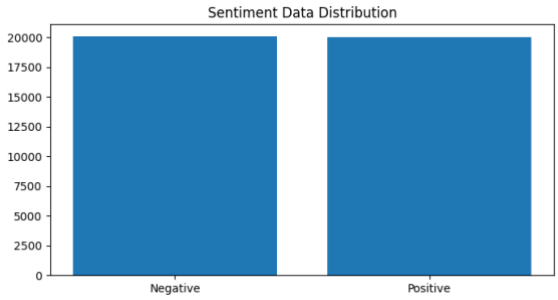

Figure 1: The distribution of Class label

B. *Feature*

In the dataset provided, there are a single main type of features: The "text" feature, on the other hand, contains the actual text content of each tweet. It represents the input data for sentiment analysis models, as the textual content is analyzed to determine the sentiment polarity indicated by the target variable. Together, these features provide the necessary information for training and evaluating sentiment analysis models on the Sentiment140 dataset.

In the context of sentiment analysis based on tweets, negative and positive word classification involves identifying and categorizing words within the text according to their polarity or sentiment orientation. Negative word classification focuses on recognizing words that convey negative sentiments, such as words expressing dissatisfaction, disapproval, or pessimism. Examples of negative words commonly found in tweets include "sad," "miss," "bad," and "sorry" as depicted in figure 2. On the other hand, positive word classification entails identifying words that denote positive sentiments, such as words indicating satisfaction, approval, or optimism. Examples of positive words often encountered in tweets include "love," "thank," "good," and "haha" as depicted in figure 3.

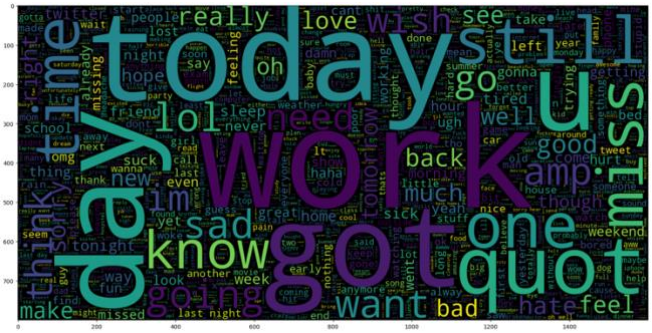

Figure 2: The negative words

The classification of words as negative or positive is typically performed using predefined lexicons or dictionaries that contain lists of words labeled with their associated sentiment polarity. During sentiment analysis, each word in the tweet is compared against these lexicons, and if a match is found, the word is classified accordingly as either negative or positive. Additionally, machine learning algorithms can be trained on labeled datasets to automatically classify words based on their contextual usage and surrounding linguistic features within the tweets.

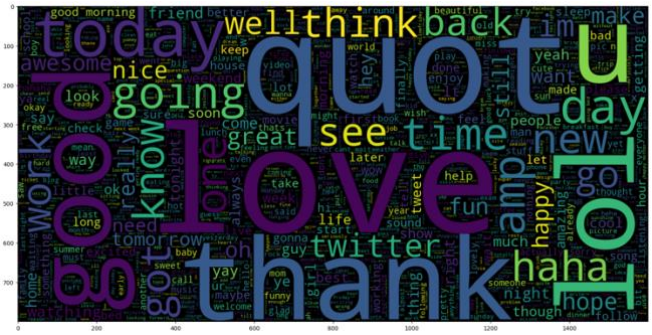

Figure 3: The positive words

By classifying words as negative or positive, sentiment analysis algorithms can quantitatively assess the overall sentiment expressed in a tweet by tallying the frequency of negative and positive words. This approach enables the identification of sentiment trends, sentiment polarity shifts, and the overall emotional tone conveyed in large volumes of tweet data, facilitating insights into public opinion, brand perception, and social trends.

III. Data Processing

A. *Stemming/ Lematization*

Stemming is a linguistic process commonly used in natural language processing to simplify words by removing suffixes or prefixes, aiming to bring different variations of a word to a common base or root form. This simplification helps to reduce the complexity of text data and improve the efficiency of text analysis tasks such as sentiment analysis. Stemming algorithms typically apply heuristic rules to chop off affixes from words, often resulting in the removal of derivational affixes as well. For example, words like "write," "writing," and "writes" would all be stemmed to the base form "writ." While stemming is relatively straightforward and computationally efficient, it may not always produce accurate results, as it can sometimes generate non-standard or incorrect root forms.

In contrast, lemmatization is a more sophisticated linguistic process that involves analyzing the structure of words based on their meaning and grammatical properties to determine their base or dictionary form, known as the lemma. Unlike stemming, lemmatization considers the context of words and uses a vocabulary and morphological analysis to accurately identify the base form of each word. For example, "writing" would be correctly lemmatized to "write," preserving the original meaning of the word. Although lemmatization is computationally more intensive compared to stemming, it tends to produce more accurate and standardized results, making it particularly useful for tasks where precision and linguistic accuracy are crucial, such as sentiment analysis on diverse textual datasets like the Sentiment140 dataset.

B. *Hyperlinks and Mention*

On Twitter, users frequently employ mentions, denoted by the "@" symbol followed by a username, to directly engage with or reference other individuals or entities within the platform's vast network. These mentions serve as a means of communication, acknowledgment, or collaboration, allowing users to tag specific accounts and initiate conversations, discussions, or exchanges of information. For example, mentions such as "@arunrk7" or "@andrewng" indicate direct interaction with particular users, fostering community engagement and facilitating connections between individuals with shared interests or affiliations.

Moreover, Twitter users often incorporate hyperlinks into their tweets, directing fellow users to external webpages, articles, videos, or other online resources. These hyperlinks expand the scope of information available within tweets, providing additional context, background, or multimedia content for users to explore. Whether linking to educational resources, news articles, or entertainment content, hyperlinks enrich the Twitter experience by offering users access to a diverse array of information and media beyond the confines of the platform. As essential components of Twitter's communicative landscape, mentions and hyperlinks play integral roles in fostering connectivity, facilitating information dissemination, and enhancing the overall engagement and interactivity of the platform.

C. *Stopwords*

Stopwords, ubiquitous in the English language, encompass words that lack specific contextual significance within a sentence. Thus, they are often excluded from text analysis processes to enhance the accuracy and relevance of classification tasks. Examples of stopwords include common pronouns such as "I," "me," "myself," "you," and "they," as well as articles like "a," "an," and "the." Additionally, common verbs and conjunctions such as "is," "are," "and," and "but" are also considered stopwords. By eliminating these frequently occurring but contextually uninformative words, text analysis algorithms can focus on extracting meaningful insights and patterns from the remaining content. This preprocessing step is crucial in tasks such as sentiment analysis, where the goal is to discern sentiment-related features from textual data, ensuring that only the most relevant information contributes to the classification process..

D. Tokenization

Tokenization is the process of breaking down a text into smaller units called tokens, which can be words, phrases, or other meaningful elements. In natural language processing (NLP), tokenization is a crucial preprocessing step that enables computers to understand and analyze textual data more effectively.

The most common form of tokenization is word tokenization, where the text is split into individual words based on whitespace or punctuation marks. For example, the sentence "The quick brown fox jumps over the lazy dog" would be tokenized into ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"].

However, tokenization can also involve splitting text into larger units such as phrases or subwords, especially in languages where words may be agglutinative or where word boundaries are not clearly defined.

Tokenization serves as the foundation for many NLP tasks, including sentiment analysis, text classification, and machine translation. By breaking down text into tokens, it allows algorithms to process and analyze textual data more efficiently, extracting meaningful features and patterns to drive subsequent analysis and modeling processes.

E. Word Embeddings

Word embeddings are a type of word representation in natural language processing (NLP) that aims to capture the semantic and syntactic meaning of words within a document or corpus. In contrast to traditional one-hot encoding methods, where each word is represented as a sparse binary vector, word embeddings map words into dense, low-dimensional continuous-valued vectors. These embeddings are learned through the process of training on large text corpora, where words with similar meanings or contexts are clustered together in the embedding space.

Word embeddings offer several advantages over traditional word representations. Firstly, they can capture the contextual meaning of words, allowing algorithms to understand the semantic relationships between words based on their usage in context. Additionally, word embeddings encode information about the syntactic structure of language, enabling algorithms to learn patterns and relationships between words, such as word co-occurrence and syntactic dependencies.

In NLP tasks such as sentiment analysis, text classification, and machine translation, word embeddings serve as feature representations that provide valuable insights into the underlying semantics of the text. Pre-trained word embeddings, such as GloVe and Word2Vec, are readily available and widely used in NLP applications. These pre-trained embeddings are trained on large text corpora and capture rich semantic and syntactic information, making them effective for transfer learning in downstream NLP tasks.

By leveraging pre-trained word embeddings, NLP models can benefit from the knowledge and insights encoded in the embeddings, leading to improved performance and generalization on various text analysis tasks. Additionally, the use of pre-trained embeddings reduces the computational cost and training time required to learn word representations from scratch, enabling faster model development and deployment.

IV. Modelling

In this sentiment analysis project, a combination of traditional machine learning models and a deep learning architecture were employed to classify sentiments in tweets. The traditional machine learning models utilized include Naive Bayes, Logistic Regression, and Support Vector Classification (SVC).

Naive Bayes is a probabilistic classifier based on Bayes' theorem with strong independence assumptions between features. It's efficient, particularly for text classification tasks, and works well even with limited training data.

Logistic Regression is a linear model commonly used for binary classification tasks. Despite its simplicity, it's robust and interpretable, making it suitable for sentiment analysis when features are relatively linearly separable.

Support Vector Classification (SVC) is a powerful supervised learning algorithm used for classification tasks. It aims to find the hyperplane that best separates different classes in the feature space, maximizing the margin between classes.

Additionally, a deep learning model based on Long Short-Term Memory (LSTM) architecture was employed. LSTM is a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data, making it particularly effective for natural language processing tasks such as sentiment analysis. LSTM networks can retain information over long sequences, enabling them to capture the nuanced relationships between words in sentences and extract meaningful features for sentiment classification. The deep learning model layer depicted as figure 4 with Input layer, followed by embedding layer, dropout layer, convolution layer, bidirectional LSTM layer, dense layer and dropout layer. The activation layer for each hidden layer is Rectified Linear unit (ReLU), while the output layer activation layer is sigmoid function.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 30)]	0
embedding (Embedding)	(None, 30, 300)	20166900
spatial_dropout1d (SpatialDropout1D)	(None, 30, 300)	0
conv1d (Conv1D)	(None, 26, 64)	96064
bidirectional (Bidirectional)	(None, 128)	66048
dense (Dense)	(None, 512)	66048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dense_2 (Dense)	(None, 1)	513

Total params: 20658229 (78.80 MB)
Trainable params: 491329 (1.87 MB)
Non-trainable params: 20166900 (76.93 MB)

Figure 4: LSTM layer visualization

By leveraging a combination of traditional machine learning models and a deep learning architecture like LSTM, this project aimed to explore and compare the effectiveness of different approaches in accurately classifying sentiments

expressed in tweets, thereby providing insights into the performance and applicability of various techniques in sentiment analysis tasks.

To improve the performance of our models, we employed several strategies:

1. Data Preprocessing: We conducted thorough data preprocessing steps, including stemming and lemmatization, removing hyperlinks, mentions and stop words. This ensured that the data was in a suitable format for model training and reduced the impact of outliers.
2. Feature Engineering: We experimented with feature engineering techniques, such as tokenization and word embedding, to enhance the model's ability to interpret and understand the textual data, aiming to capture semantic relationships and contextual information within the tweets for improved sentiment analysis performance.
3. Hyperparameter Tuning: For machine learning model, we performed hyperparameter tuning using optuna to find the optimal combination of hyperparameters that maximized model performance. On the other hand, for the deep learning approach, we didn't do hyperparameter tuning since every epoch iteration is already a learning for the model.
4. Evaluation Metrics: We used appropriate evaluation metrics such as accuracy score to assess the performance of our models comprehensively, since the dataset is quite balanced as well. This allowed us to identify areas for improvement and fine-tune our models accordingly.

V. Training and Evaluation

The result of deep learning approach with LSTM model achieves around 75% accuracy toward the training and validation dataset in 20 epochs. This aligns with the test accuracy result around 74% depicted with figure 5.

	precision	recall	f1-score	support
Negative	0.77	0.69	0.72	16088
Positive	0.71	0.79	0.75	15912
accuracy			0.74	32000
macro avg	0.74	0.74	0.74	32000
weighted avg	0.74	0.74	0.74	32000

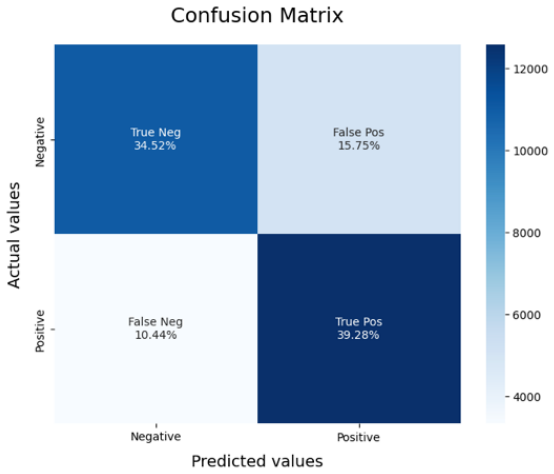

Figure 5: LSTM Test Result

On the other hand, for machine learning model such as Bernoulli Naïve Bayes, Linear Support Vector Classifier and Logistic Regression need Optuna for the hyperparameter tuning and for every single algorithm was trained with 100 trials (known as epoch for the deep learning). In figure 6, the test accuracy result for Bernoulli Naïve Bayes algorithm is 72%. In figure 7, the test accuracy result for Linear Support Vector Classifier algorithm is 73%. In figure 8, the test accuracy result for Bernoulli Naïve Bayes algorithm is 73%.

Best hyperparameters: {'alpha': 0.9819624785209664}

	precision	recall	f1-score	support
0	0.73	0.73	0.73	1396
1	0.71	0.71	0.71	1294
accuracy			0.72	2690
macro avg	0.72	0.72	0.72	2690
weighted avg	0.72	0.72	0.72	2690

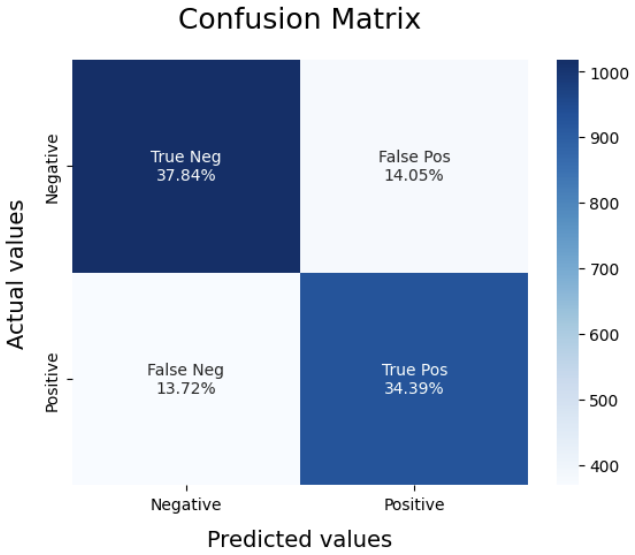

Figure 6: BernoulliNB Test Result

Best hyperparameters: {'C': 0.13433048205103545}

	precision	recall	f1-score	support
0	0.75	0.72	0.74	1396
1	0.71	0.75	0.73	1294
accuracy			0.73	2690
macro avg	0.73	0.73	0.73	2690
weighted avg	0.73	0.73	0.73	2690

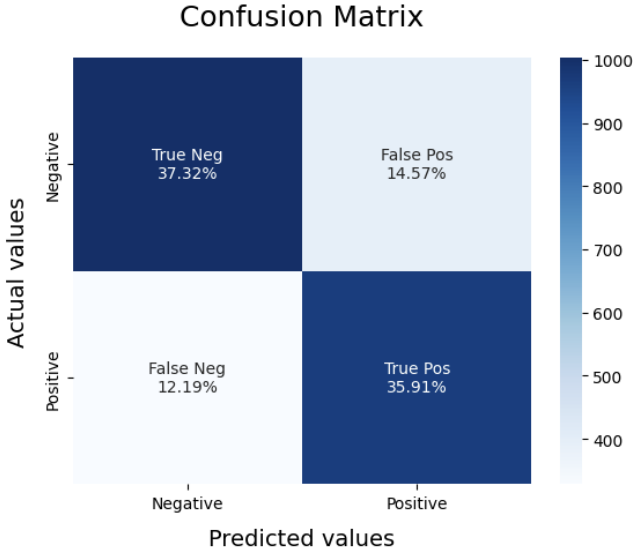

Figure 7: Linear SVC Test Result

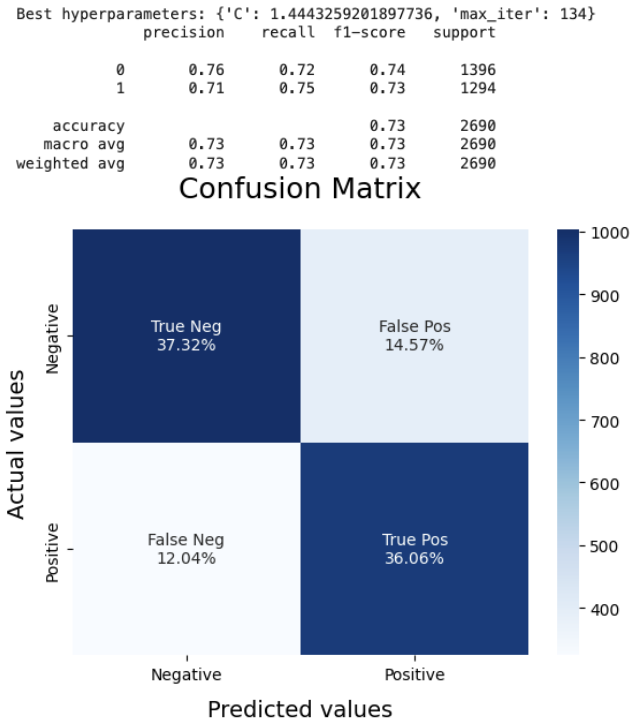

Figure 8: Logistic Regression Test Result

In summary, the model with the highest accuracy is LSTM, although the accuracy of each model might change depends on the hyperparameter tuning for the machine learning models and weight and bias for the deep learning models. LSTM (Long Short-Term Memory) models often achieve better accuracy in natural language processing tasks, such as sentiment analysis, due to their inherent ability to capture long-term dependencies and sequential patterns in text data. Unlike traditional recurrent neural networks (RNNs), which may struggle with vanishing or exploding gradients over long sequences, LSTM networks are equipped with specialized memory cells and gating mechanisms that enable them to selectively retain and update information over time.

One key advantage of LSTM models is their capability to effectively handle sequences of varying lengths while preserving contextual information. This is particularly advantageous in sentiment analysis, where the sentiment expressed in a tweet or text often depends on the entire context of the sentence or paragraph. LSTM networks can capture these contextual nuances and semantic relationships between words, allowing them to make more informed predictions about the sentiment of the text.

Moreover, LSTM models are well-suited for learning from large-scale text corpora, enabling them to extract rich and meaningful representations of the underlying semantics of the text. By leveraging pre-trained word embeddings and fine-tuning them in conjunction with LSTM layers, models can benefit from the semantic information encoded in the embeddings, leading to improved accuracy and generalization performance.

Overall, the superior accuracy of LSTM models in sentiment analysis tasks can be attributed to their ability to effectively model long-range dependencies, handle variable-length sequences, and extract meaningful features from textual data, ultimately resulting in more accurate sentiment predictions.

VI. Conclusion

In conclusion, our project encountered several scientific bottlenecks revolving primarily around model performance optimization and algorithm selection. Throughout the experimentation process, we faced challenges in fine-tuning the performance of our sentiment analysis models to achieve optimal accuracy and efficiency. One significant bottleneck stemmed from the complexity of selecting the most suitable machine learning algorithms and deep learning architectures for our task. We grappled with determining which models would effectively capture the nuances of sentiment expressed in tweets while balancing computational efficiency and model complexity.

To overcome these bottlenecks, we adopted a systematic approach to model evaluation and comparison. We conducted extensive experiments with various machine learning algorithms, including Naive Bayes, Logistic Regression, and Support Vector Classification (SVC), alongside a deep learning architecture based on Long Short-Term Memory (LSTM). By meticulously evaluating the performance of each model on the Sentiment140 dataset and analyzing metrics such as accuracy, precision, and recall, we gained insights into the strengths and limitations of different approaches.

Additionally, we leveraged techniques such as hyperparameter tuning, using frameworks like Optuna, to optimize the performance of our machine learning models. Through iterative experimentation and parameter optimization, we were able to fine-tune the algorithms to achieve improved accuracy and generalization performance. Furthermore, by incorporating pre-trained word embeddings and feature engineering techniques such as tokenization, we enhanced the representational power of our models, enabling them to capture contextual information and semantic relationships within the tweet data more effectively.

Overall, while we encountered scientific bottlenecks during the project, including challenges related to model optimization and algorithm selection, we successfully navigated these hurdles through systematic experimentation, hyperparameter tuning, and feature engineering. By addressing these bottlenecks, we were able to develop sentiment analysis models that demonstrate robust performance and efficacy in discerning sentiments expressed in tweets, although it didn't reach the 80% accuracy, which is the threshold for the human capability in analysing sentiment based on given tweet/text. For further development, using pretrained model such as Bidirectional Encoder Representations from Transformers (BERT) pre-trained weight could increase the performance of the model