(Data Analysis and Management)

Term Paper Report

Analysis On Twitter Sentiment Data

Table of Contents

1. Introduction **1**

1.1. Introduction 1

1.2. Technical Approach 2

2. Applied Techniques **3**

2.1. Data Preprocessing Techniques 3

2.2. Tensorflow based model 4

2.3. Layers of the model applied 5

3. Data Preprocessing **7**

4. Model Applied **9**

4.1. Construct Model 9

4.2. Model Training 10

4.3. Model Evaluation 11

5. Data Visualization 12

6. Future Directions 17

7. Conclusion 17

1. Introduction

1.1. Introduction

Twitter, a global hub for real-time conversations, is a vast source of unfiltered public opinion, covering everything from personal experiences to reactions to world events. With its fast-paced and dynamic nature, Twitter offers invaluable data for sentiment analysis, helping to capture how people feel about topics in real time. However, the informal tone of tweet filled with slang, abbreviations, and even emoji presents challenges in accurately gauging sentiment.

This report focuses on the application of neural networks for sentiment analysis on Twitter data, a widely utilized method in natural language processing. Given the vast amount of data generated on the platform, understanding user sentiments presents significant opportunities for businesses, researchers, and policymakers. Neural networks, with their capability to model complex relationships within data, are particularly effective in identifying and classifying sentiments expressed in tweets as positive, or negative. Through this approach, the analysis aims to provide a deeper understanding of public opinion and the effectiveness of neural networks in sentiment analysis tasks.

1.2. Technical Approach

The implementation of the system will be carried out using the Python programming language within a Jupyter Notebook environment, which is well-suited for machine learning and data science projects. A TensorFlow-based model will be developed for the task.

The ***"Sentiment 140 dataset"*** will be utilized as the primary dataset for this analysis. The data will be split into two parts: 70% will be used for training the model, while the remaining 30% will be allocated for testing purposes. This split will ensure that the model is trained effectively while retaining a portion of the data for an unbiased evaluation of the model's performance.

After training the model, its performance will be evaluated using standard metrics to assess how well it generalizes to unseen data. The evaluation phase will provide insights into the model's accuracy, precision, recall, and other relevant performance indicators.

2. Applied Techniques

2.1. Data Preprocessing Techniques

Data preprocessing is a crucial step in the data analysis that directly impacts the machine learning model's performance. For this, before feeding the model with the data, data preprocessing is needed to do first.

For the natural language processing (NLP)-based analysis, stemming and lemmatization are essential.

Stemming

Stemming is a heuristic process that reduces words to their base form by removing prefixes or suffixes. It often produces root forms that may not be actual words. It uses simple rules, such as chopping off common word endings like *-ing*, *-ed*, or *-s*. The resulting stems may not be linguistically correct words.

Example:

- "running" → "run"
- "better" → "bett"
- "studies" → "studi"

Stemming is faster and computationally simpler. It works well when precision in word forms is not concerned like in search engines.

Lemmatization

Lemmatization is a more sophisticated process that reduces words to their base or dictionary form, known as a lemma. It takes into account the context and the part of speech (POS) of the word. It uses vocabulary and morphological analysis to ensure that the root form is a valid word in the language. The lemmatized form is a real word, based on its meaning and POS.

Example:

- "running" → "run"
- "better" → "good"
- "studies" → "study"

Lemmatization is more accurate than stemming but also more computationally expensive. It's ideal when context and meaning are important, like in text analysis or machine learning applications.

2.2. Tensorflow based model

Tensor flow based model is a kind of neural network. A neural network is a computational model inspired by the structure and function of the human brain. The goal of neural networks is to create an artificial system that can process and analyze data in a similar way to how the human brain works. A neural network is composed of interconnected nodes, called neurons, organized into layers, and a layer is a set of neurons that perform a specific task. Each neuron receives input signals, performs some computation on these inputs, and produces an output signal. These outputs then serve as inputs to neurons in the next layer, creating a network of interconnected neurons.

There are many types of neural networks and among them the Recurrent Neural Network (RNN) is used as our analysis is NLP-based. To train data, the neural network of Tensorflow based models is used.

2.3. Layers of the model applied

1. Input Layer

- To instantiate a tensor representing the input data, “Input” (a Keras function) will be used.
- Variable for the maximum length of a tweet, i.e. the length of the input sequence, is needed to be declared in this function.

2. Embedded Layer

- This layer is to convert integer-encoded representations of words into dense vectors of fixed size.
- ***layer=Embedding(2000,50,input_length=max_len)(input)***
- It maps each word index (up to 2000 words) to a 50-dimensional vector.
- **Dimension:** Each dimension represents a different aspect or feature of the word's meaning or usage.

  E.g. for a word of “King”,

  ➢ Dimension 1: Gender

  ➢ Dimension 2 : Royal status

  ➢ Dimension 3 : Age

3. LSTM Layer (Long Short Term Memory)

- It is a type of recurrent neural network suitable for sequence data (NLP).
- LSTM saves the words and predicts the next words based on the previous words.

4. Dense Layer

- This layer is a fully connected layer and used for learning nonlinear relationships in the data.
- Dense layer reduces the outputs by getting inputs from the previous layer.

5. Dropout Layer

- Dropout layer drops some neurons from previous layers to avoid overfitting problems.
- In overfitting, the model gives good accuracy on training time but not good on testing time.
- ***layer = Dropout(0.5)(layer)***
- This prevents overfitting by randomly dropping 50% of the neurons during training.

6. Output Layer

- This layer is used to output the result.
- It is a dense layer with a single unit, which serves as the output layer for binary classification (positive or negative sentiment).

3. Data Preprocessing

Data preprocessing is essential for sentiment analysis to ensure accurate and meaningful results. It involves cleaning and transforming raw text data into a usable format by removing noise like stop words, punctuation, and irrelevant characters. This process enhances the performance of machine learning models by reducing complexity, improving consistency, and enabling better extraction of relevant features from the data, which leads to more accurate sentiment predictions. The steps used in our analysis are:

1. Slice the Data

- Keep only the first 20,000 rows of the dataset to manage computational resources effectively.
- Ensure the dataset contains a balanced number of positive and negative samples.

2. Clean and Remove Stop Words

- Stop words are common words that are usually filtered out in text processing as they do not contribute to the semantic meaning.
- Use a predefined list of stop words to remove these from the text.

3. Remove Punctuation

- Punctuation marks are removed to focus on the actual words in the text, which can help in reducing noise in the data.

4. Remove Repeating Characters

- Clean text by removing consecutive repeating characters to normalize words (e.g., "loooove" becomes "love").

5. Remove Email Addresses

- Remove any email addresses from the text, as they are typically not relevant for text analysis.

6. Remove URLs

- Strip out URLs to avoid including links in the text data, which do not contribute to the analysis.

7. Remove Numeric Values

- Eliminate numeric values from the text if they are not relevant to the text analysis.

8. Tokenize the Text

- Split the text into individual words or tokens. This process converts the text into a format suitable for further processing.

9. Apply Stemming

- Reduce words to their root form (e.g., "running" becomes "run") using a stemming algorithm. This helps in normalizing words.

10. Apply Lemmatization

- Convert words to their base or dictionary form (e.g., "better" becomes "good") using a lemmatization algorithm. This is more sophisticated than stemming and provides a more accurate base form of words.

11. Separate Input Feature and Label

- Divide the dataset into features (text) and labels (sentiment or category) to prepare for modeling.

4. Model Applied

After data preprocessing is completed, the TensorFlow-based model is applied. The clean, structured, and tokenized data is fed into the model to begin the learning process.

4.1. Construct Model

First, build the model for training and testing. The background theories of the layers in the model are explained in the previous session.

```
def tensorflow_based_model():
    inputs = Input(name='inputs',shape=[max_len])
    layer = Embedding(2000,50,input_length=max_len)(inputs)
    layer = LSTM(64)(layer)
    layer = Dense(256,name='FC1')(layer)
    layer = Activation('relu')(layer)
    layer = Dropout(0.5)(layer)

    layer = Dense(1,name='out_layer')(layer)
    layer = Activation('sigmoid')(layer)
    model = Model(inputs=inputs,outputs=layer)
    return model
```

Fig 1. Model building

4.2. Model Training

The preprocessed dataset is divided into two parts, with 80% allocated for training. During this phase, the model learns the underlying patterns in the data, associating the preprocessed text with sentiment labels. Using TensorFlow, the model will undergo multiple iterations (epochs) where it adjusts its internal parameters based on the training data to minimize the error between predicted and actual sentiment labels.

```
Epoch 1/6
315/315 [==============================] - 62s 193ms/step - loss: 0.5771 - accuracy: 0.6897 - val_loss: 0.5237 - val_accuracy: 0.7407
Epoch 2/6
315/315 [==============================] - 59s 189ms/step - loss: 0.6654 - accuracy: 0.6948 - val_loss: 0.5339 - val_accuracy: 0.7275
Epoch 3/6
315/315 [==============================] - 59s 188ms/step - loss: 0.4942 - accuracy: 0.7613 - val_loss: 0.5195 - val_accuracy: 0.7475
Epoch 4/6
315/315 [==============================] - 60s 192ms/step - loss: 0.4814 - accuracy: 0.7705 - val_loss: 0.5182 - val_accuracy: 0.7436
Epoch 5/6
315/315 [==============================] - 61s 193ms/step - loss: 0.4979 - accuracy: 0.7586 - val_loss: 0.5350 - val_accuracy: 0.7418
Epoch 6/6
315/315 [==============================] - 61s 193ms/step - loss: 0.4618 - accuracy: 0.7783 - val_loss: 0.5408 - val_accuracy: 0.7411
Training finished !!
```

Fig 2. Training the model

We trained our model for only six epochs due to practical constraints, despite knowing that more epochs might improve its accuracy. Here's why we kept it to six:

1. **Storage Limits:** More epochs require more memory and storage, which can quickly fill up, especially with large datasets.
2. **Time Constraints:** Training for more epochs takes a lot longer. With limited time, we needed to keep the training period manageable.
3. **Diminishing Returns:** After a certain point, training more doesn't really make the model much better—it just uses more resources.
4. **Avoiding Overfitting:** Too many epochs can make the model too tailored to the training data, meaning it might not perform well on new data.

These reasons led us to choose a balance between a well-trained model and the practical limits of our resources by sticking to six epochs.

4.3. Model Evaluation

After training, the remaining 20% of the preprocessed dataset, reserved for testing, is used to evaluate the model's performance. The model makes predictions on the unseen test data, and its accuracy, precision, recall, and other performance metrics are calculated.

```
[48]: accr1 = model.evaluate(X_test,Y_test)

375/375 [==============================] - 9s 23ms/step - loss: 0.5314 - accuracy: 0.7423

[49]: print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1]))

Test set
  Accuracy: 0.74
```

Fig 3. Checking model accuracy

After the model is applied and evaluated, the accuracy is ***0.74.***Since the accuracy is acceptable, the model can make further predictions on sentiment analysis. If more input data is used for training and makes more iterations, the accuracy value will rise up more.

5. Data Visualization

Data visualization is essential in data analysis as it allows complex information to be understood at a glance. By transforming raw data into visual representations, such as charts, graphs, and plots, data visualization helps identify patterns, trends, and outliers that might be difficult to detect in tabular or textual formats. It aids in making informed decisions by providing a clear picture of data distributions, relationships, and performance metrics.

In this analysis, visualizations of ***Confusion matrix*** with plot, ***Proportion chart***, and ***ROC Curve*** are used.

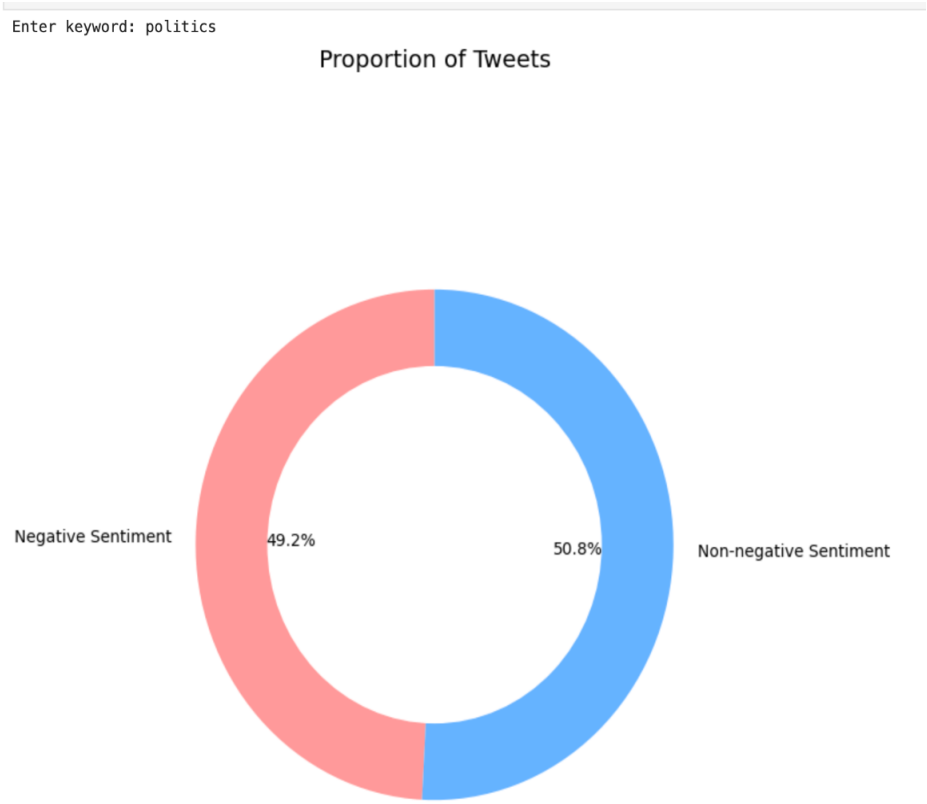

Fig 4. Proportion chart

This chart shows the proportion of the negative sentiment and non-negative sentiment when people talk about “Politics” topic.

We can also see in the bar chart below when we enter “Netflix” as a keyword.

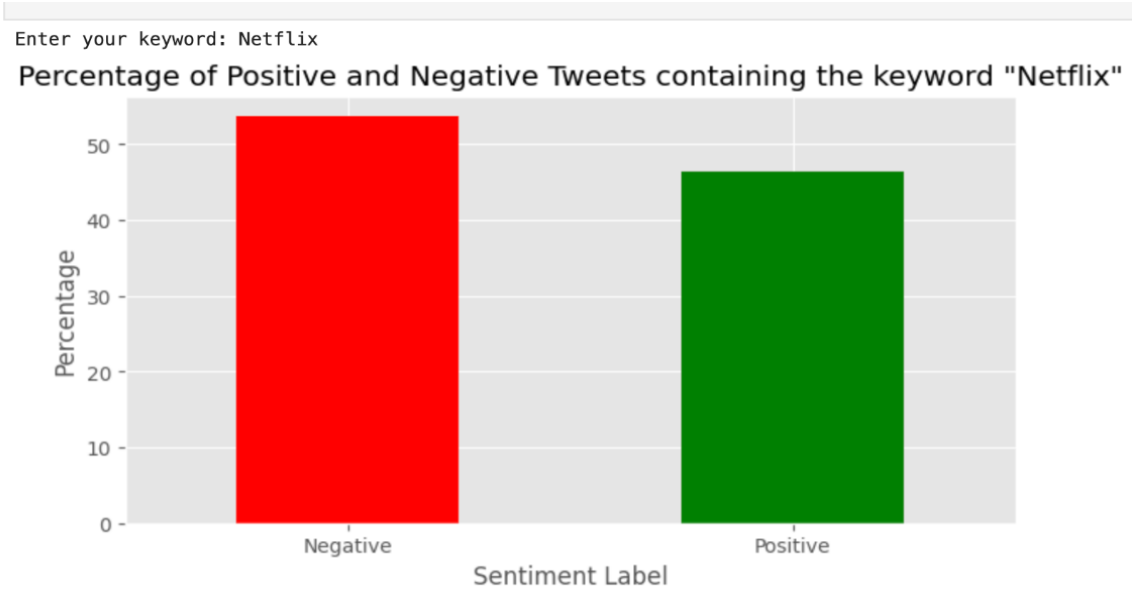

Fig 5. Bar chart on positive or negative tweets containing the keyword “Netflix”

This is the percentage of positive/negative portions when people talk about “Netflix” on twitter.

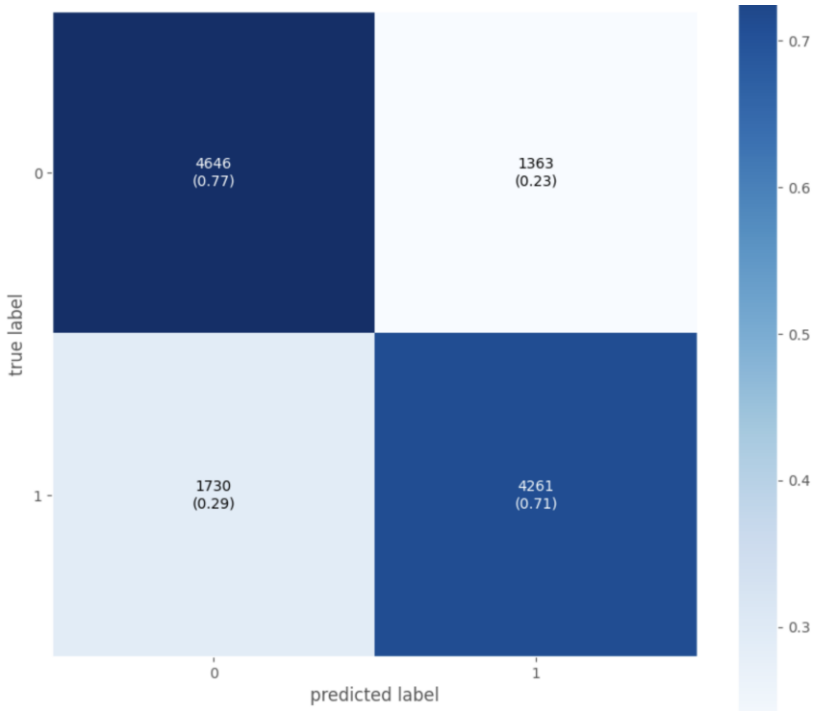

Fig 6. Confusion matrix

The confusion matrix provides a detailed insight into its performance across different classifications. The matrix displays the following results:

- **True Negatives (Top Left, 0-0):** The model correctly identified 4646 instances as negative (**77%** of actual negatives). This shows a good ability to recognize negative cases.
- **False Positives (Top Right, 0-1):** The model incorrectly labeled 1363 instances as positive when they were actually negative (**23%** of actual negatives). This error rate indicates some confusion in distinguishing positives.
- **False Negatives (Bottom Left, 1-0):** The model incorrectly identified 1730 positive instances as negative (**29%** of actual positives). This means it sometimes misses detecting positive cases.
- **True Positives (Bottom Right, 1-1):** It correctly identified 4261 instances as positive (**71%** of actual positives). This is a strong performance for identifying positive cases.

Although there are errors (23% false positives and 29% false negatives), these rates are not excessively high, suggesting that the model maintains a good balance. Particularly in practical applications, it's rare to have a model that makes no errors, so these figures represent a solid performance where the model correctly predicts the sentiment more often than not.

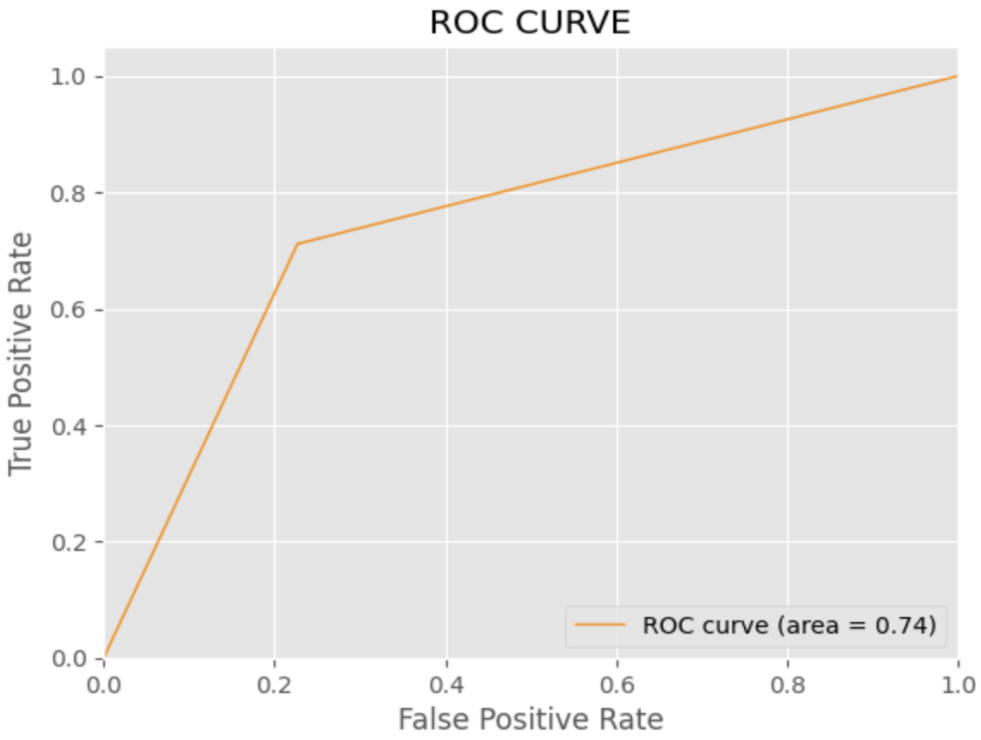

Fig 7. ROC curve

The ROC (Receiver Operating Characteristic) curve in the graph represents the performance of the classification model at various threshold settings.

Area Under the Curve (AUC) = 0.74: This value describes the overall ability of the model to distinguish between the classes. An AUC of 1 indicates a perfect model; 0.5 indicates a model doing no better than random chance. The model's AUC of 0.74 suggests it has a good predictive performance, although there is still some room for improvement.

Model Prediction Results

```
Enter your tweet: cats are very annoying ,i will not ever pet them!
1/1 [==============================] - 0s 21ms/step
[[0.2914941]]
This is negative.


Enter your tweet: my tears drop because I am so happy that winning the prize
1/1 [==============================] - 0s 21ms/step
[[0.9187884]]
This is positive.


Enter your tweet: we broke up last night, he is selfish and I want to kill him!
1/1 [==============================] - 0s 22ms/step
[[0.3399725]]
This is negative.


Enter your tweet: my dream is to become a professional Machine Learning Engineer.
1/1 [==============================] - 0s 20ms/step
[[0.6954182]]
This is positive.
```

Fig 8. Prediction results for input tweets

The scores described in **Fig 8.** are generated by the model based on its learned patterns from the training data, and typically range from 0 to 1. Here's what these values generally mean:

- 0 to 0.5: The model predicts the sentiment of the tweet as negative. The closer the score is to 0, the more confident the model is that the sentiment is negative.
- 0.5 to 1: The model predicts the sentiment of the tweet as positive. The closer the score is to 1, the more confident the model is that the sentiment is positive.

6. Future Directions

- **Advancements in AI and NLP**: Leveraging more sophisticated AI models for better accuracy in interpreting nuances.
- **Cross-Platform Sentiment Analysis**: Integrating data from multiple social media platforms for comprehensive insights.
- **Enhanced Contextual Understanding**: Improving models to better understand context and ambiguous language.
- **Ethical Considerations and Privacy**: Addressing privacy concerns and ethical implications of sentiment analysis.

7. Conclusion

Twitter sentiment analysis is a dynamic tool that taps into the pulse of global conversations, offering insights into public emotions, opinions, and trends. By analyzing the content of tweets, it provides valuable feedback on consumer sentiment, political climate, and public reactions to events, products, and services. Through the lens of Twitter sentiment analysis, businesses, governments, and researchers can make informed decisions, fostering a deeper connection with the global conversation.