Sentiment Analysis

Deep Learning and Reinforcement Learning :

Week 9

Course Final Project

Name : Lau Eng Hui

Role : Data Scientists

Table of Contents

List of Figures

Chapter 1 : Introduction

1.1 Objective

The objective of using Twitter sentiment analysis is to analyze and understand the overall sentiment or opinion of a particular topic or brand on Twitter by collecting and analyzing tweets containing specific keywords or hashtags. This can help businesses and organizations to gain insights into customer opinions and attitudes, identify areas for improvement, and make data-driven decisions.

1.2 Benefit

Twitter sentiment analysis is a powerful tool that can be used to quickly gather and analyze large amounts of data related to public opinion on a particular topic. By analyzing the sentiment of tweets related to a specific subject, individuals and organizations can gain valuable insights into how people feel about the topic in question. This can be particularly useful for businesses looking to gauge consumer sentiment towards their products or services, or for politicians and policymakers looking to understand public opinion on key issues. Overall, Twitter sentiment analysis can provide a wealth of information that can be used to make more informed decisions and take more effective action.

Chapter 2 : Data Description

2.1 Features

a. Tweet_Id : Unique ID for Twitter user

b. Entity: Company

c. Sentiment : Subjective information in an expression

d. Tweet content: A message posted to Twitter containing text

2.2 Size

a. Training datasets

- The dataset contains 74682 rows and 4 columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74682 entries, 0 to 74681
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Tweet_Id       74682 non-null  int64
 1   Entity         74682 non-null  object
 2   Sentiment      74682 non-null  object
 3   Tweet content  73996 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
```

Figure 2.1: Training Dataset Information

b. Testing datasets

- The dataset contains 1000 rows and 4 columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Tweet_Id       1000 non-null   int64
 1   Entity         1000 non-null   object
 2   Sentiment      1000 non-null   object
 3   Tweet content  1000 non-null   object
dtypes: int64(1), object(3)
memory usage: 31.4+ KB
```

Figure 2.2: Testing Dataset Information

2.3 Inspiration

- Predict the sentiment of a tweet content.
- Training datasets has missing value on “Tweet content” column.

Chapter 3 : Exploratory Data Analysis and Data Cleaning

3.1 Exploratory Data Analysis

Before modelling a machine learning, EDA should be done to understand the data and summarize their main characteristics. The relationship between each dependant variables and independent variable (target) is determined by data visualization using bar chart and histogram from seaborn library.

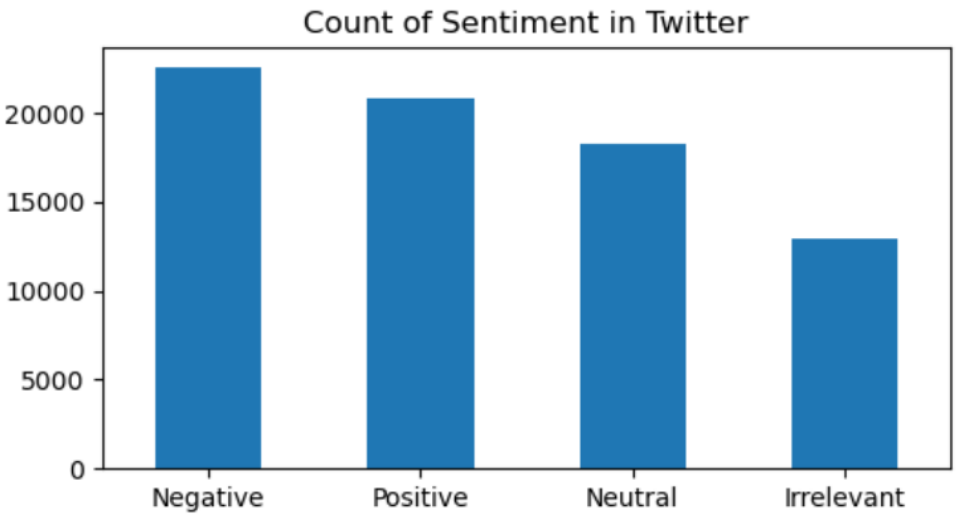


Figure 3.1: Bar chart to show the count of each sentiment in the training datasets.

3.2 Data Cleaning and Pre-processing

Several actions are taken to clean and prepare the data for analysis. These actions include:

a. Removing any missing values in the datasets. In the training dataset, the 'Tweet content' column have some missing values, the rows with missing values will be removed.
b. Extracting the useful information such as 'Sentiment' and 'Tweet content' columns
c. Ensuring that all data is in the correct format and type for analysis.
d. For 'Tweet content' column, the strings will be cleaned by removing the punctuations, stop word and lemmatizing every word in the string.

Chapter 4 : Model Training

a. Bag of Words Vectorization Model

In this model, Term Frequency - Inverse Document Frequency (TFIDF) Vectorization will be used to find meaning of sentences consisting of words and cancels out the incapabilities of Bag of Words technique. It is good for text classification or for helping a machine read words in numbers.

After the vectorization, the dataset will be feed into Logistic Regression model. The model will be trained and used to predict the sentiment. Here are the results shown below:

```
# Train machine learning using Logistic Regression algorithms
lr = LogisticRegression(n_jobs=-1)

lr.fit(X_train_bow,y_train)
yhat_lr = lr.predict(X_test_bow)

# Show evaluation metrics
score_bow = evaluate_metrics(y_test, yhat_lr)
score_bow
```

```
{'Accuracy': 0.917, 'Precision': 0.917, 'Recall': 0.916, 'F1': 0.916}
```

```
lr_labels = lr.classes_

plot_graph(y_test, yhat_lr, lr_labels, "Bag of Words")
```

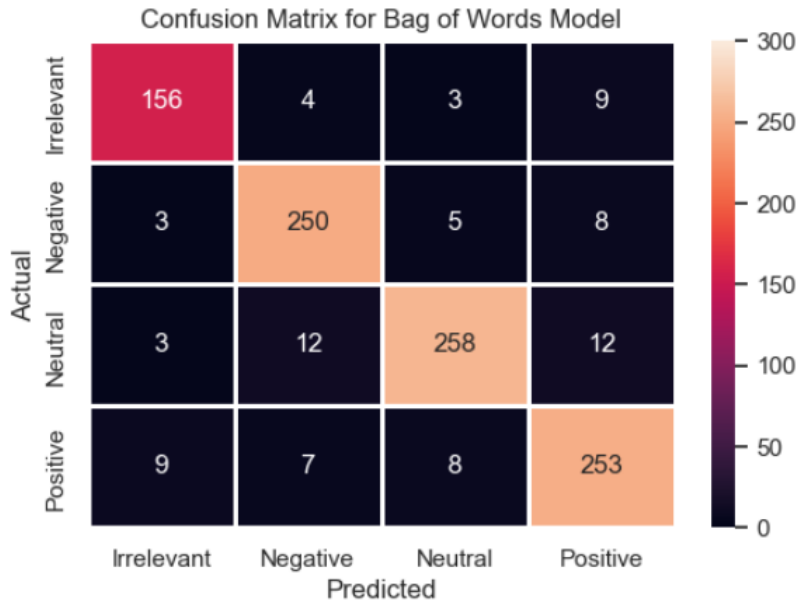


Figure 4.1: Bag of Words Vectorization Confusion Matrix and Evaluation Metrics

b. LSTM neural network

A deep learning-based model is employed for the training of the sentiment analysis model. For NLP tasks, we generally use RNN-based models since they are designed to deal with sequential data. Here, I trained an LSTM (Long Short Term Memory) model using TensorFlow with Keras. The steps to perform sentiment analysis using LSTM-based models are as follows:

1. Import Tokenizer from Keras.preprocessing.text and create its object. Fit the tokenizer on the entire training text (so that the Tokenizer gets trained on the training data vocabulary). Generated text embeddings using the texts_to_sequence() method of the Tokenizer and store them after padding them to an equal length. (Embeddings are numerical/vectorized representations of text. Since we cannot feed our model with the text data directly, we first need to convert them to embeddings)
2. Build model - Add Input, LSTM, and dense layers to it. Add dropouts and tune the hyperparameters to get a decent accuracy score. Generally, we tend to use ReLU or LeakyReLU activation functions in the inner layers of LSTM models as it avoids the vanishing gradient problem. At the output layer, we use Softmax or Sigmoid activation function.

```
: yhat_LSTM = [np.argmax(x) for x in model.predict(X_test_LSTM)]
  score_LSTM = evaluate_metrics(y_test_LSTM, yhat_LSTM)
  score_LSTM

  32/32 [==============================] - 0s 11ms/step

: {'Accuracy': 0.765, 'Precision': 0.759, 'Recall': 0.762, 'F1': 0.76}

: LSTM_labels = le.classes_
  plot_graph(y_test_LSTM, yhat_LSTM, LSTM_labels, 'LSTM')

: <AxesSubplot:title={'center':'Confusion Matrix for LSTM Model'}, xlabel='Predicted', ylabel='Actual'>
```

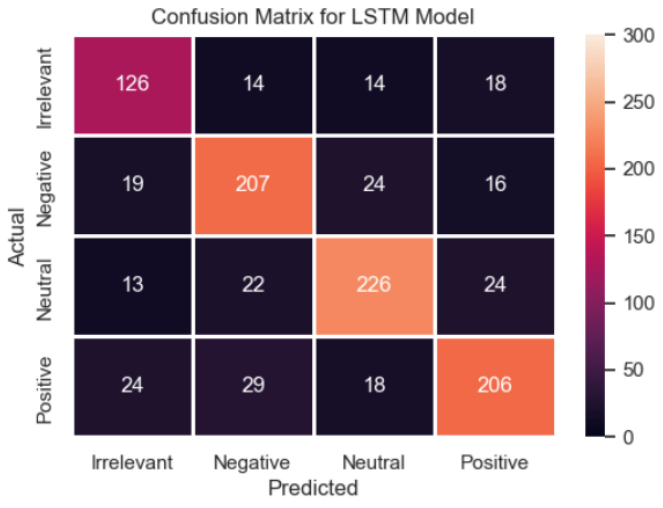


Figure 4.2: LSTM Confusion Matrix and Evaluation Metrics

c. Transformers

Transformer-based models are one of the most advanced Natural Language Processing Techniques. They follow an Encoder-Decoder-based architecture and employ the concepts of self-attention to yield impressive results. Though one can always build a transformer model from scratch, it is quite tedious a task. Thus, we can use pre-trained transformer models available on Hugging Face. Hugging Face is an open-source AI community that offers a multitude of pre-trained models for NLP applications. These models can be used as such or can be fine-tuned for specific tasks.

Here are the results shown below:

```
score_trans = evaluate_metrics(y_test,a)
print(score_trans)

plot_graph(y_test, a,['Negative', 'Positive'], "Transformer")
```

```
{'Accuracy': 0.8066298342541437, 'Precision': 0.826, 'Recall': 0.809, 'F1': 0.805}
<AxesSubplot:title={'center':'Confusion Matrix for Transformer Model'}, xlabel='Predicted', ylabel='Actual'>
```

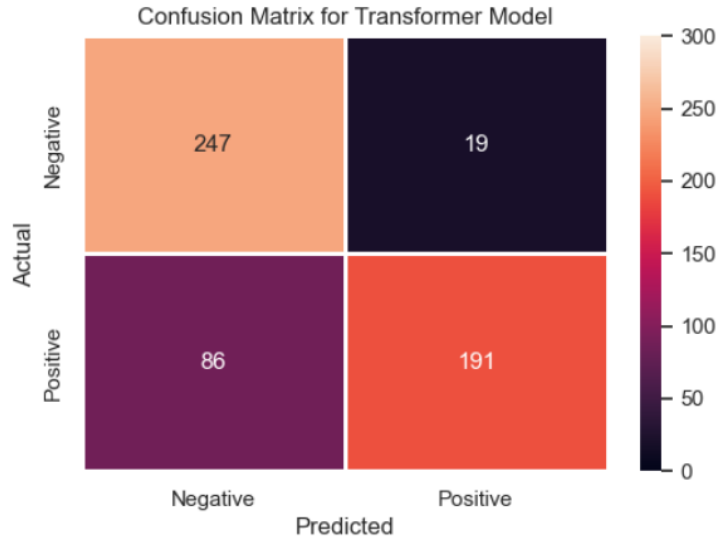


Figure 4.3: Transformer Confusion Matrix and Evaluation Metrics

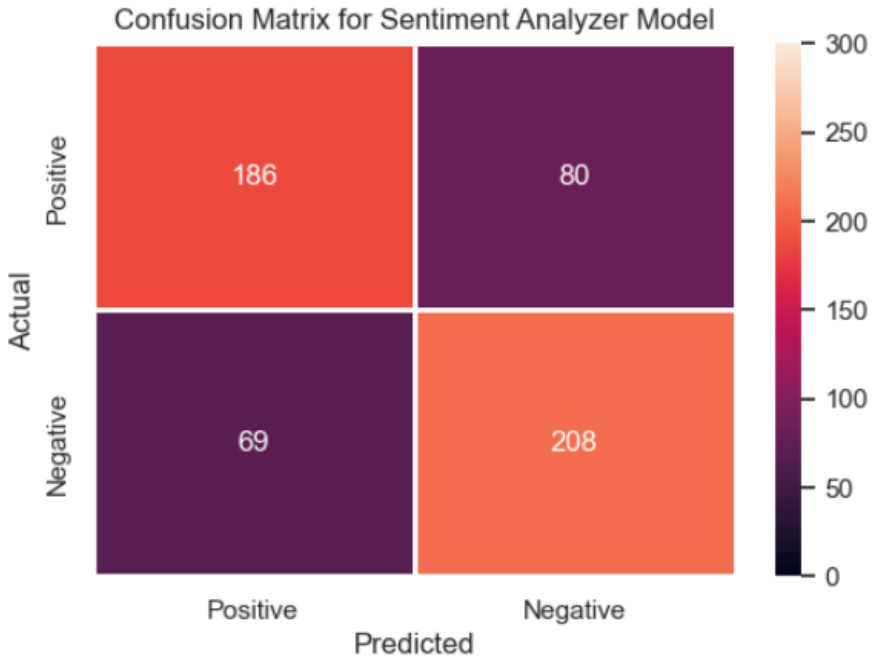
d. Sentiment Intensity Analysis

Sentiment Intensity Analyzer is a rule-based sentiment analyzer in which the terms are generally labeled as per their semantic orientation as either positive or negative. First, we will create a sentiment intensity analyzer to categorize our dataset. Then, we use the polarity scores method to determine the sentiment.

Here are the results shown below:

```
score_sia = evaluate_metrics(y_test_sia, trans_list)
print(score_sia)
plot_graph(y_test_sia, trans_list,['Positive','Negative'], 'Sentiment Analyzer')
```

```
{'Accuracy': 0.7255985267034991, 'Precision': 0.726, 'Recall': 0.725, 'F1': 0.725}
```



Chapter 5 : Results and Discussion

Overall score for each model:

	Accuracy	Precision	Recall	F1
Bag of Words	0.917000	0.917	0.916	0.916
LSTM	0.765000	0.759	0.762	0.760
Transformer	0.806630	0.826	0.809	0.805
Sentiment Intensity Analyzer	0.725599	0.726	0.725	0.725

Overall, the Bag of Words Vectorization got the highest score in terms of every metric such as accuracy, precision score, recall score and F1 score.