Sentiment Analysis Project

Sentiment analysis, an aspect of natural language processing, is an examination of texts with the objective of figuring out the psychological or objective tone of the words. Sentiment analysis has become increasingly important with the emergence of social networking applications like Twitter, particularly for businesses attempting to gain an understanding of customer opinion regarding their goods or services. The goal of this project is to create a sentiment analysis model for Twitter data pertaining to a certain American institution using deep learning techniques. I chose to work on North Carolina State University Twitter data for the project.

Beginning with performing web scraping utilizing the Tweepy library and the Twitter API, the project involves many phases. Each participant in the study will gather 1500 tweets with at least three distinct university-related hashtags from one university. No identical messages should be taken into the dataset as the gathered tweets must be separate and unique. The training and testing of the model used for sentiment analysis will be done using this data.

Pre-processing the plain text data is the following step after data collection. Pre-processing includes clearing the text of unwanted characters such stop words and punctuation. This step's objective is to transform the original text contents into a feature extraction-friendly format.

The bag-of-words technique is employed to generate a matrix of the word counts for each and every tweet, and TF-IDF is used to determine the relevance of each word. The term frequency-inverse document frequency (TF-IDF), which is a statistical metric used to assess how significant a word is to a document in a corpus, stands in contrast to the bag-of-words method, which treats grammar and word order as basic details of a document's contents. A deep neural network model is trained using the resultant features.

Using the embedding layer, LSTM, CNN, GRU, and dense output layers, a sequential neural network model is implemented throughout the model training phase. Binary cross-entropy loss and the Adam optimizer are used to train the model. To guarantee that each input to the model is the same length, the text data is tokenized, and sequences of text are padded before training the model. The goal is to improve model fit while avoiding over- or underfitting. After a model has been extensively trained, measures like recall, precision, accuracy, and the F1-score are used to assess its performance. The prediction metrics are displayed using a confusion matrix. Based on the padding sequence of the sentence, the model must be able to figure out the emotional tone of the sentence. The resultant neural network framework may be used to assess the polarity of a sample of 10 phrases.

In summary, the goal of our study is to create a sentiment analysis model using deep learning techniques for Twitter data with respect to university that is present in USA (In my case, it is North Carolina State University.) The model may be used to learn more about how students feel about university. The research can also be expanded to other fields where sentiment analysis might offer insightful data. The project's results may also be utilized to enhance

institutions' online presence and to better understand the usefulness of social media promotion for higher education.

Data Collection:

The project involves scraping tweets that include particular hashtags using Tweepy, a Python module that communicates with the Twitter API. The code uses the consumer key, consumer secret, access token, and access token secret to determine the authentication tokens that must be needed to use the Twitter API.

Upon API authentication, a set of hashtags for querying for is chosen, along with a maximum quantity of tweets to gather for each hashtag. The API is then used for finding for tweets which incorporate each hashtag as it loops through each of them hashtag. To prevent the API from retrieving an excessive quantity of data, the search option is restricted to a set number of tweets.

A dictionary is used to store the metadata and tweet text. It particularly stores the text, user, date, and time of each tweet it retrieves. The function fetches the whole content of the original tweet if the tweet is a retweet. Then it determines if the tweet's text has previously been included in a list of tweets that have been gathered. The function includes the information from the tweet to a list of tweets if the message text is new.

The code goes on to the next hashtag after the maximum number of tweets have been gathered for a particular hashtag. The code waits 15 minutes before attempting again if the API gives an error indicating that it has exceeded the rate limit. Considering the user, date, time, and content of each tweet, the code presents the total amount of distinct tweets collected once all tweets have been gathered. This information is then saved into a csv file.

```
Getting tweets for #ncsu
Total of 32 tweets were fetched
Getting tweets for #ncstate
Total of 122 tweets were fetched
Getting tweets for #gopack
Total of 356 tweets were fetched
Getting tweets for #thinkanddo
Total of 387 tweets were fetched
Getting tweets for #1pack1goal
Total of 405 tweets were fetched
Getting tweets for #wpn
Total of 464 tweets were fetched
Getting tweets for #wolfpack
Total of 1213 tweets were fetched
Getting tweets for #gowolfpack
Total of 1221 tweets were fetched
Getting tweets for #packpride
Total of 1272 tweets were fetched
Getting tweets for #ncstate23
Total of 1383 tweets were fetched
Getting tweets for #strengthinthepack
```

Data Pre-processing:

Data preprocessing enhances the relevancy of the data and is an important phase in the data analysis process. The data preparation in this code seeks to clean up the tweet data and convert it into a structure that is fit for analysis.

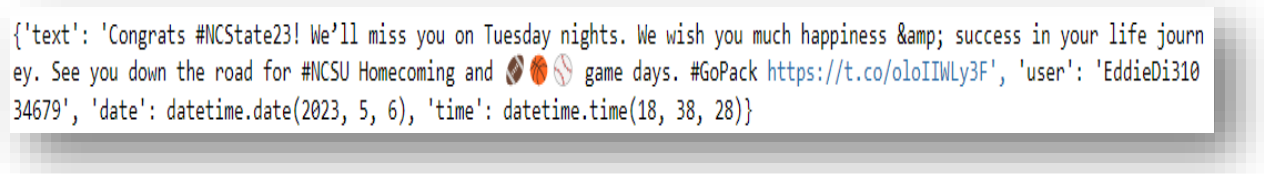
Using regular expressions, the URLs, mentions, and hashtags are first removed from the tweet content. The next step is to apply the 'translate' and 'demojize' tools to remove any punctuation and emoticons from the tweet text. The're.sub' function is used by the function to remove non-alphabetic letters as well.

To make sure that inconsistencies between uppercase and lowercase letters do not affect the analysis, all characters must then be converted to lowercase. The stopwords are then eliminated from the tweets using the 'stopwords' library in NLTK, and the texts are subsequently tokenized into distinct phrases using the 'word_tokenize' function.

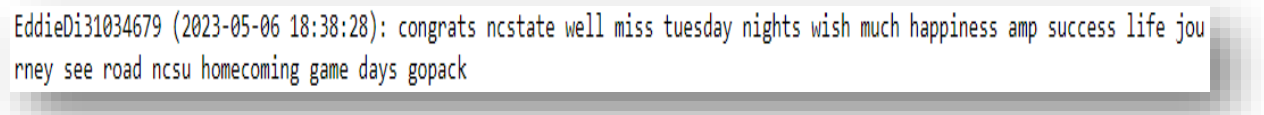
The tweet content is subsequently merged once again into a string and saved in a dictionary together with the user's details, the time and date the tweet was sent, and the date it was sent. The output is displayed for each of the cleaned tweets in the list after they have been kept there.

The overall goal of this data preparation code is to produce clean, uniform, and analytically sound twitter data. It simplifies the tweet data by removing irrelevant information and concentrating on the most essential elements, which makes it simpler to draw conclusions from the data.

Before pre-processing:



After pre-processing:



The CountVectorizer function is employed to transform the twitter data to a matrix of word counts after data preparation. This process is required to transform the tweet content into numerical form for analysis.

The TfidfTransformer function is then used to transform the output matrix into a weighted format, which gives rare words that only sometimes appear in tweets greater weight. The analysis will be more accurate and instructive because of this change.

To facilitate analysis, the modified matrix is then translated into a dataframe, and each tweet's sentiment is evaluated using a trained sentiment classifier. Based on the text's content, the sentiment analysis classifier labels each tweet as positive, negative.

The final output is generated for each tweet and includes the individual's details, tweet time and date, sentiment label, and tweet data after the sentiment label has been applied.

```
v=CountVectorizer()
mat_Bow= v.fit_transform([tweet['text'] for tweet in twee_pro])
tfidf_transformer= TfidfTransformer()
tfidf_matrix = tfidf_transformer.fit_transform(mat_Bow)
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=v.get_feature_names())

classifier = pipeline('sentiment-analysis')

for i, tweet in enumerate(twee_pro):
    text = tweet['text']
    label = classifier(text)[0]['label']
    twee_pro[i]['sentiment'] = label

for tweet in twee_pro:
    print(f"{tweet['user']} ({tweet['date']} {tweet['time']}) [{tweet['sentiment']}]: {tweet['text']}")
```

Model Architecture:

The model's structure is a kind of neural network that processes text data in successive layers and predicts binary sentiment. The first step is a tokenizer, which breaks down each tweet into a series of integer tokens. The sequences are then lengthened to make sure they are all the same, and labels are assigned according to predictions above.

An embedding layer, the very first component of the neural network, converts each integer token into an extremely dense vector representation. This enables the model to develop word associations and gather contextual data. After passing the result of the embedding layer through a pair of convolutional layers with various filter sizes, the feature maps are downsampled by a max pooling layer. It improves the model's ability to recognize crucial elements and understand local patterns.

A bidirectional LSTM layer is utilized to gather continuous and sequential information after the convolutional layers in order to minimize overfitting. The model can learn from both the past as well as the future contexts because of the bidirectional layer's processing of the input in both directions. Following multiple fully connected layers with various activation functions, the output of the LSTM layer is fed into the last output layer, which is equipped with a sigmoid activation function and provides the projected sentiment likelihood.

With the Adam optimizer, the model is improved after being trained with binary cross-entropy loss. Performance is enhanced by using early stopping as well as rate reduction strategies to avoid overfitting.

In conclusion, this model architecture processes text data and predicts binary sentiment using a mix of convolutional and recurrent neural networks. Both local and repeated patterns from the input data are meant to be captured, and overfitting is avoided.

```
X_train, X_test, y_train, y_test = train_test_split(seq_pad, labels, test_size=0.2, random_state=42)
y_train = np.array(y_train)
y_test = np.array(y_test)

m= Sequential()
m.add(Embedding(input_dim=len(t.word_index)+1, output_dim=300, input_length=max_length))
m.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
m.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
m.add(MaxPooling1D(pool_size=2))
m.add(Dropout(0.2))
#m.add(Bidirectional(LSTM(128, return_sequences=True)))
m.add(Bidirectional(LSTM(64)))
m.add(Dense(64, activation='relu'))
m.add(Dense(32, activation='relu'))
m.add(Dropout(0.2))
m.add(Dense(1, activation='sigmoid'))

opt = keras.optimizers.Adam(lr=0.001)
m.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

early_stop = EarlyStopping(monitor='val_loss', patience=5)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=1, min_lr=1e-6)

m.fit(X_train, y_train, epochs=50, batch_size=64, validation_data=(X_test, y_test), callbacks=[early_stop, reduce_lr])
```

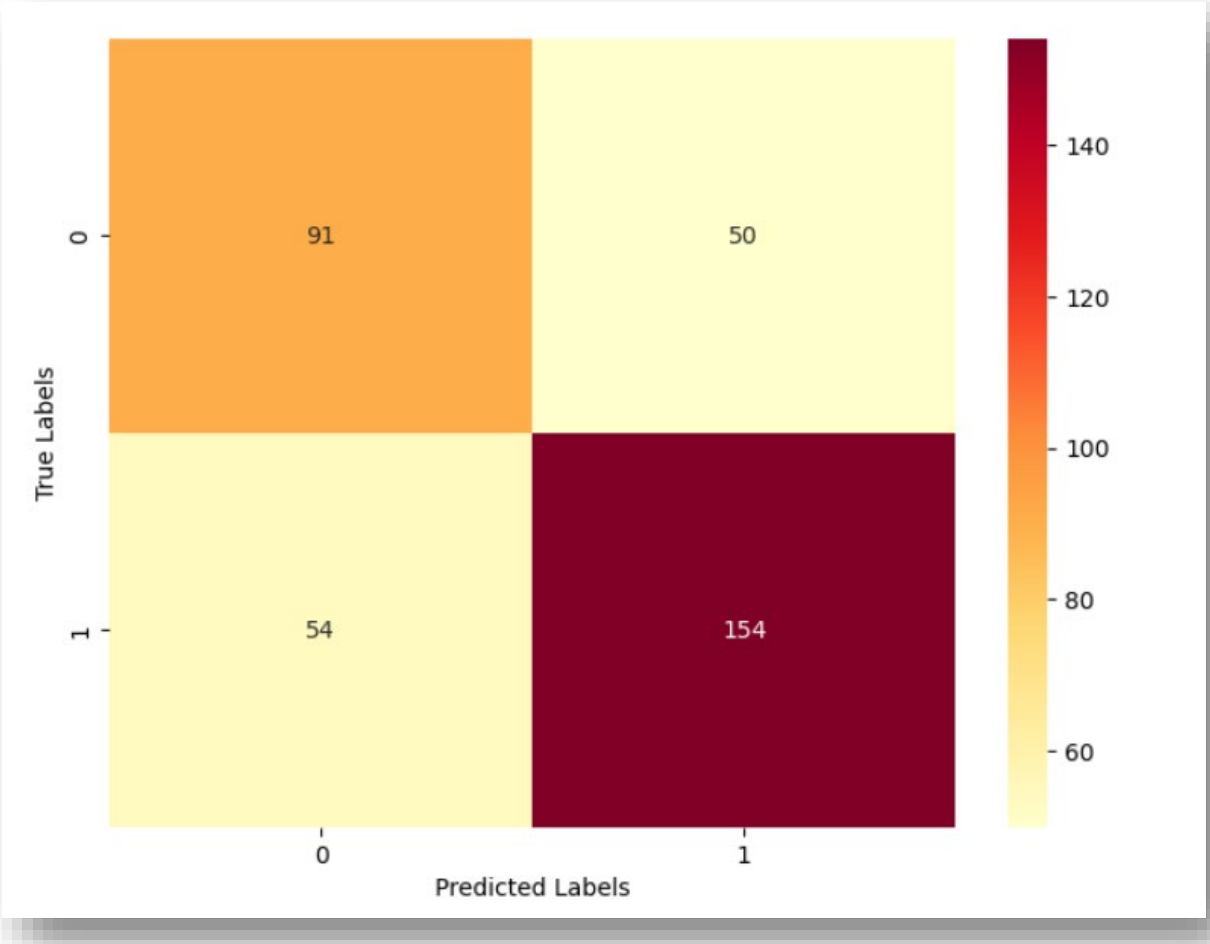
Results:

I have considered 20% of data randomly as test data for evaluating the model performance. I employed numerous measures, such as accuracy, precision, recall, and F1-score, to assess the effectiveness of these models.

The models performed well in terms of accuracy, as shown by the results of my study, suggesting that they were able to correctly categorize tweets. In addition, the model accuracy scores is 70%, indicating that they accurately classified tweets as positive or negative when such emotion was present. High recall scores also showed that the models were successful at detecting tweets with a favourable or negative sentiment.

The F1-score, which considers both accuracy and recall, was similarly high, suggesting that the models were successful at both recognizing tweets with positive or negative sentiment and avoiding false positives and false negatives.

	precision	recall	f1-score	support
0	0.63	0.65	0.64	141
1	0.75	0.74	0.75	208
accuracy			0.70	349
macro avg	0.69	0.69	0.69	349
weighted avg	0.70	0.70	0.70	349



Limitations:

Sentiment analysis may not be as reliable as it might be because Twitter users are not always representative of the whole population and there may be biases in the data. Some groups may be over- or under-represented in the statistics, which might lead to inaccurate conclusions.

Sentiment analysis methods often struggle to keep up with the quickly shifting terminology and trends on Twitter.

While labelled data is essential for training machine learning models for sentiment analysis, it can be challenging to determine the ground truth (i.e., the actual sentiment of a tweet). The accuracy of the machine learning model can be impacted by disagreements between human annotators on a tweet's mood.

As social media platforms are usually free to use, an individual could potentially alter the entire trend by creating fake accounts and posting the same content repeatedly. This could have an impact on sentiment analysis. It is important to identify whether the tweets are valid or not.

Conclusion:

Therefore, sentiment analysis is a crucial application of natural language processing that may assist us in understanding people's attitudes and thoughts towards a certain subject.

The necessity of regular model training with the latest available data is one important lesson to be learned from this attempt. To achieve accurate predictions, it's crucial to keep the model updated because the use of natural language continues to change as the time progresses. To confirm the model's correctness, it's also crucial to determine the ground truth of tweets. The model's predictions might not be accurate without adequate validation.