ITCS-5111:Natural Language Processing

Sentiment Analysis Using Python

Gopichand Tadapaneni,
Student ID: 801272664.

1. Introduction and Objective:

The aim of this project is to use machine learning to predict whether tweets related to a certain university, in our case it is University of Illinois Chicago, have positive, negative or neutral sentiments. To achieve this goal, we will collect tweets using web scraping methods and preprocess the data by cleaning and converting the text into a numerical format suitable for machine learning. We will then use feature extraction methods such as bag-of-words and TF-IDF to create features that can be used to train sequential neural network models like LSTM, CNN, and GRU.

We will evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score, and visualize the results using a confusion matrix. We will also experiment with different hyperparameters such as the number of features and epochs to optimize the model. Finally, we will save the best model checkpoint based on the validation loss using the ModelCheckpoint callback in Keras.

The overall objective of this project is to develop a machine learning model that can accurately classify tweets about specific universities as having positive, negative or neutral sentiments. This can be useful for gaining insights into public perception of universities and helping universities to improve their reputation.

2.Process of Data Collection:

To collect the data for this project, we will use the Tweepy library and the Twitter API to scrape tweets from Twitter. The first step is to obtain the necessary Twitter API keys and tokens, which are required to authenticate and access the Twitter API.

Once we have obtained the API keys and tokens, we can use the Tweepy library to create a Twitter API object and search for tweets using specific keywords or hashtags related to the universities we are interested in. We will use the following steps to scrape tweets from Twitter:

1. Import the necessary libraries and modules in Python:
2. Authenticate and access the Twitter API using the Tweepy library
3. Define the search query to search for tweets using specific keywords or hashtags related to the universities we are interested in:
4. Use the Twitter API to search for tweets using the search query
5. we will remove any duplicate tweets
6. Store the tweets in a Pandas DataFrame

By following these steps, we can scrape tweets from Twitter using the Tweepy library and the Twitter API. We can repeat this process for each university we are interested in, and collect at least 1500 tweets from University of Illinois Chicago.

3. Data Processing:

The preprocessing step is an important part of the data cleaning process that involves removing any unnecessary characters or text from the raw data to prepare it for analysis. In this project, we will preprocess the collected tweets data by performing the following steps:

1. Removing URLs: We will remove any URLs from the tweets using regular expressions, as they do not provide any useful information about the sentiment of the tweet.
2. Removing mentions: We will remove any mentions of other Twitter users from the tweets, as they do not contribute to the sentiment of the tweet.
3. Removing hashtags: We will remove any hashtags from the tweets, as they do not directly affect the sentiment of the tweet.
4. Removing punctuation: We will remove any punctuation marks such as commas, periods, exclamation marks, and question marks from the tweets, as they can cause noise in the data.
5. Removing stop words: We will remove any stop words such as "the", "and", "a", "an", etc., from the tweets, as they are commonly used words that do not add much meaning to the tweet.
6. Converting text to lowercase: We will convert all the text to lowercase to avoid treating the same word with different cases as separate words.
7. Tokenizing the text: We will split the text into individual words or tokens, so that each word can be considered as a separate feature.

By performing these preprocessing steps, we will clean and transform the raw tweet data into a format that is suitable for machine learning analysis. This will help improve the accuracy of the machine learning model in predicting the sentiment of tweets related to specific universities.

4. Feature Extraction:

In this step, extracted features from the preprocessed tweet text using the bag-of-words approach and TF-IDF using the scikit-learn library.

First, CountVectorizer is used to convert the preprocessed text into a sparse matrix of word counts. This involves learning the vocabulary of the preprocessed text and counting the number of times each word appears in each tweet. The resulting sparse matrix X has one row per tweet and one column per unique word in the vocabulary.

Next, TfidfTransformer is used to weigh the importance of each word in the vocabulary using TF-IDF (term frequency-inverse document frequency). This involves calculating the term frequency (TF) of each word in each tweet, which is the number of times the word appears in the tweet divided by the total number of words in the tweet. The IDF is calculated as the logarithm of the total number of tweets divided by the number of tweets containing the word. The TF-IDF weight is then calculated as the product of the TF and IDF values. The resulting TF-IDF matrix tfidf has the same shape as X.

From python notebook, shape of the TF-IDF matrix is `(3331, 12866)`.

Limitation:TF-IDF matrix has more columns (i.e., more unique words in the vocabulary), this can increase the dimensionality of the feature space and potentially make it more difficult for a machine learning model to learn the relationship between the features and the labels. This problem is known as the "curse of dimensionality".

There are several ways to address the curse of dimensionality when working with high-dimensional data:

1. Feature selection: Select a subset of the most informative features based on some criteria, such as mutual information, chi-squared test, or correlation. This can reduce the dimensionality of the feature space and improve the performance of the model.
2. Dimensionality reduction: Use techniques such as principal component analysis (PCA), singular value decomposition (SVD), or t-distributed stochastic neighbor embedding (t-SNE) to project the high-dimensional data into a lower-dimensional space while preserving the most important information. This can also reduce the dimensionality of the feature space and improve the performance of the model.
3. Regularization: Use regularization techniques such as L1 or L2 regularization to penalize the model for using too many features or for using features that are not relevant to the prediction task. This can encourage the model to focus on the most important features and reduce the impact of irrelevant or noisy features.

In general, selecting too few features can lead to underfitting, where the model is not able to capture the complexity of the data, while selecting too many features can lead to overfitting, where the model becomes too specific to the training data and does not generalize well to new data. Therefore, it's important to strike a balance between selecting enough features to capture the relevant information and not selecting too many features that add noise or complexity to the model.

One way to choose a value for k is to use cross-validation to estimate the performance of the model for different values of k. You can split the data into training and testing sets using a specific random seed, and then perform k-fold cross-validation on the training set to estimate the performance of the model for different values of k. You can then choose the value of k that maximizes the performance on the validation set.

Feature selection on the TF-IDF matrix using the chi-squared test:

perform feature selection on the TF-IDF matrix using the chi-squared test. We create an instance of SelectKBest called selector, and specify the chi2 scoring function and the number of top features to select (k=3000 in this case). We then apply the fit_transform() method to the TF-IDF matrix and the labels to obtain the selected features in a new matrix X_new.

Then splitted the selected features X_new and the tweet_sentiment(labels) into training and testing sets using the train_test_split() function.

5. Model Training:

The model architecture used in this project is based on sequential neural networks, specifically LSTM, CNN, and GRU. Sequential neural networks are ideal for analyzing sequential data such as text, as they can take into account the order of the data and capture complex patterns that may exist within the data.

The model architecture consists of an embedding layer, one or more hidden layers of LSTM, CNN, or GRU, and a dense output layer. The embedding layer is used to map the individual words or tokens in the preprocessed tweet data to a dense vector representation. The hidden layers of LSTM, CNN, or GRU are used to analyze the sequence of these dense vectors and extract relevant features from the data.

The output layer consists of a single neuron with a sigmoid activation function, which outputs a value between 0 and 1 that represents the predicted sentiment of the tweet. A value closer to 0 indicates a negative sentiment, while a value closer to 1 indicates a positive sentiment.

The model is trained using binary cross-entropy loss and the Adam optimizer. We will experiment with different hyperparameters such as the number of features used for training the model, the number of epochs used for training the model, and the batch size used for training the model.

The use of sequential neural networks allows us to capture the sequential nature of text data and extract relevant features that can be used to predict the sentiment of tweets related to University of Illinois Chicago.

6. Results and evaluation metrics:

- Accuracy: 0.70
- Precision: 0.49
- Recall: 0.70
- F1-score: 0.58

7. limitations or potential improvements to the model:

While the model architecture used in this project is suitable for predicting the sentiment of tweets related to specific universities, there are some limitations and potential improvements that could be considered:

1. **Limited training data:** The model is trained using a limited amount of data, which may not be representative of all possible variations in language and sentiment. Collecting more data and increasing the diversity of the dataset could help improve the accuracy of the model.
2. **Unbalanced dataset:**

```
neu_count = tweet_sentiment.count('neutral')
neg_count = tweet_sentiment.count('negative')
pos_count = tweet_sentiment.count('positive')

print(neu_count)
print(neg_count)
print(pos_count)
```

Output:

```
2371
385
575
```

The dataset was unbalanced, meaning that there are more tweets with one sentiment than the other. This could affect the model's ability to accurately predict the less frequent sentiment. Balancing the dataset by oversampling or undersampling the less frequent sentiment could help improve the accuracy of the model.

3. **Hyperparameters tuning:** The model's hyperparameters such as the number of features, epochs, and batch size could be further optimized to improve the model's performance.
4. **Use of pre-trained word embeddings:** The use of pre-trained word embeddings such as Word2Vec or GloVe could help improve the model's performance by providing more context and meaning to the words in the tweets.
5. **Use of attention mechanisms:** Attention mechanisms could be used to help the model focus on the most important parts of the tweet when making predictions.

In summary, while the model architecture used in this project is suitable for predicting the sentiment of tweets related to specific universities, there are still some limitations and potential improvements that could be considered to further improve the accuracy of the model.

8.Conclusion:

This project involved the use of the Twitter API and Tweepy library to collect tweets related to specific universities. The collected data was preprocessed by removing unnecessary characters and stop words, and features were extracted using the bag-of-words approach and TF-IDF. The model architecture used a sequential neural network with LSTM, CNN, and GRU layers to predict the sentiment of the tweets.

The model's performance was evaluated using various metrics, and potential improvements and limitations were discussed. This project highlights the application of machine learning to analyze social media data and predict sentiment, which can be useful in many fields such as marketing, politics, and public opinion research.