Keenan Flynn: [kpfxn8@umsystem.edu](mailto:kpfxn8@umsystem.edu)

Mustavi Islam: [mizmc@umkc.edu](mailto:mizmc@umkc.edu)

ICP 2

Use the same data (that we obtained by in source code Data=pd.read\_csv( 'https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv')) and perform the sentiment analysis task on this data using one of the scikit learn classifier for text.

YOUTUBE: <https://youtu.be/xeaQd18E2kA>

ICP Requirements:

1. Data cleaning and preprocessing (at minimum have the following: Removing unnecessary columns or data, Removing Twitter Handles( @user ), Removing punctuation, numbers, special characters, removing stop words, Tokenization, and Stemming, TFIDF vectors, POS tagging, checking for missing values ,train/test split of data).
2. Data Visualization and analysis for critical steps (WordCloud, Bar plots, etc)
3. Model building and successfully executing the model to make prediction.
4. Code quality, WikiReport quality, video explanation

In this ICP, we learned how to apply natural language processing techniques to a dataset of tweets. To do this task, we utilized Python’s NLTK library. We learned how to clean the text data, apply NLP transformations (Tokenizing, Lemmatizing, and POS). Finally we learned how to create a model to do predictions on our transformed data.

The data set consisted of about 30,000 tweets. These tweets had an associated tag. If the tweets had been flagged as hateful or racist, the flag would be 1 else it would be 0. Using a SKLearn model, we wanted to be able to predict this flag.

Text

Description automatically generated

We faced several challenges when doing this ICP. The biggest challenge was cleaning the data. There is not uniform tweet structure and twitter utilizes special characters (@user, #hashtag) to add flavor to tweets. NLTK allowed us to do a cleanse of the data and remove the special characters, punctuation, and numbers. When processing words, some words have more meaning than others. NLTK allowed us to remove these ‘stopwords’ to get a more concise dataset.

Below is the frequency distribution of words before cleaning.

Text

Description automatically generated

The given dataset was large, and words can come in many forms. One challenge associated with this task was that words were not always in their base form. The NLTK WordNet Lemmatizer allows us to transform all of the words into ‘pure’ words. NLTK also allowed us to add Part-of-Speech tags to words which are helpful for differentiating words like ‘bat’ (the animal) from ‘bat’ (to hit with a stick).

Below are screenshots of code that clean the data.

Text

Description automatically generated

Text

Description automatically generated

Below is the frequency distribution of words after cleaning.

Text

Description automatically generated

Text

Description automatically generated

With these transformations done, we were able to make a relatively simple SKLearn model. We used a pipeline to create a model that contains a TF-IDF Vectorizer and a Support Vector Classifier. These analyze frequency of the words and predict the associated tag, either 1 or 0. Below is a picture of the metrics of our model. Graphical user interface, text

Description automatically generated

Text

Description automatically generated