IST736 Text Mining - Homework 1

Martin Alonso

## Introduction

The objective of this exercise is to perform sentiment analysis on a data set of texts regarding how people feel about artificial intelligence (AI) on social media; and whether we can accurately categorize whether people have a positive or negative sentiment towards AI.

## Methodology

For this exercise, I will mainly employ the use of the sklearn and nltk packages to understand the data set and predict the outcomes. I will also use pandas, numpy, seaborn, and matplotlib.pyplot to do initial data analysis, attempting to find any patterns in the data prior to modeling it. I’ll also import the re package to clean the tweets.

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import re   
# sklearn package  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.svm import LinearSVC  
from sklearn.metrics import confusion\_matrix  
# nltk package  
from nltk.classify import NaiveBayesClassifier  
from nltk.corpus import subjectivity  
from nltk.sentiment import SentimentAnalyzer  
from nltk.sentiment.util import \*

As for the dataset, it was obtained from Twitter using the twitterscape package [1]. This was done prior to building this report as the data collection requires the use of the command prompt. The data set, however, includes 200 tweets that feature either ‘AI’ or ‘Artificial Intelligence’ within their text. The data was also transformed prior to loading the data, dropping columns regarding the url, html, likes, retweets, and fullname of the user; while also adding an id and label column. The label column is a categorical column that indicates the strengh of the sentiment from 1 to 5. In order: 1 = Purely negative, 2 = Somewhat negative, 3 = Neutral, 4 = Somewhat positive, 5 = Purely positive. This label was manually coded, so expect some slight bias in it.

# Load the data set.   
df = pd.read\_csv('ai\_tweets.csv')  
print(df.shape)

## (201, 5)

print(list(df.columns.values))

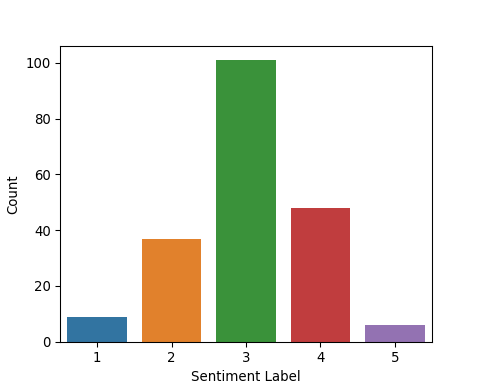
## ['id', 'label', 'user', 'timestamp', 'text']

Having loaded the data, I’ll clean it, removing special characters and converting uppercase letters into lowercase. This will allow me to easily vectorize all words in the data set.

# Transform the data  
df['label'] = df['label'].astype(object)  
cleanText = []  
for f in df['text']:   
 text = re.sub('(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])', '', f).lower()  
 cleanText.append(text)  
cleanText = pd.DataFrame(cleanText)  
df2 = pd.merge(df, cleanText, how='outer', left\_index=True, right\_index=True)  
df2.rename(columns={0:'cleanText'}, inplace=True)

With the data clean, let’s check how the sentiment labels are distributed.

sns.countplot('label', data=df)  
\_ = plt.xlabel('Sentiment Label')  
\_ = plt.ylabel('Count')  
plt.show()



The counts for each sentiment label are almost equally distributed. With that, let’s start performing some sentiment analysis. I’ll first start with building both a training and testing set for the classifiers. After that, I’ll use the sklearn package to build a first classifier, and then move to nltk package to build a second model to compare results.

# Convert label back to int  
df2['label'] = df['label'].astype('int')  
# Split the data into train and test sets.  
np.random.seed(42)  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(df2['cleanText'], df2['label'], test\_size=0.33, random\_state=3)

With tha data sets split, time to use sklearn.

tf = TfidfVectorizer(min\_df=5, max\_df=0.8, sublinear\_tf=True, use\_idf=True, stop\_words='english')  
# Count and transform the training and testing sets  
x\_train\_tfidf = tf.fit\_transform(x\_train)  
x\_test\_tfidf = tf.transform(x\_test)  
# Now to use an svm to create the model.   
model1 = LinearSVC()  
model1.fit(x\_train\_tfidf, y\_train)  
result1 = model1.predict(x\_test\_tfidf)

## Results

## Conclusions

## References

1. Twitterscrape package: <https://github.com/taspinar/twitterscraper>