Categorizing Twitter Sentiment Analysis

Introduction

Twitter is one of the most popular social media platform. It is estimated that 500 millions tweets are sent every day. This is an average of 6,000 tweets per second. Tweet sentiment analysis is a great tool to analyze opinions. It can help companies and entities in many different ways, for instance, it can help companies understand how their customers feel about their brand by analyzing negative and positive words. It can also help filtering out comments that can be considered as racist, bullying, and threats. Sentiment analysis is the process of identifying and classifying opinions, judgments, or feelings about a specific topic. The process of sentiment analysis involves different aspects, such as text analytics, linguistics, sentiment connotation, and an accepted language processing to dig subjective information. Sentiment analysis is commonly known for the term "opinion mining." Usually, the statements or tweets are classified into 3 categories: positive, negative and neutral. There is not a particular accetable way to classfify tweets. Some data scientist might choose to assign a label base on the number of positive and negative words,and othes might do it base on emoticons.

Dataset Description

The dataset contains a total of 1,600,000 tweets. There are a total of 6 variables.

Target: 0 = negative and 1 = positive ID: The ID of the tweet Date: The date of the tweet (year, month, day, and time) Flag: The query User: The name of the user that tweeted Text: The text of the tweet

The sentiment category for this dataset was automatically created. The approached used was that any tweet with a positive emoticon were positive, and tweets with a negative emoticon were negative. The data was collected through the twitter API, and it is available at the Kaggle Website (https://www.kaggle.com/kazanova/sentiment140).

NLP Problem:

Can I categorize or classify tweets correctly? Which algorithm is beter to classify tweets Word2vec or Tf-Idf? Does using Word2vec improves the model preformance? If so, Does the context window has any effect on the accuracy score ?

Prospect NLP Solution:

Tweets visualization, develop a tokenization, covert tweets to a matrix of Tf-Idf features, supervised learning model, and performance evaluation.

NLP Solution Approach

Data Inspection and Visualization

In order to inspect and visualize the dataset, we are going to be using 4 libraries. Pandas, in order to create a pandas dataframe. We'll use Matplotlib to plot the value count of the sentiment.The purpose of doing this is to checked if the dataset is biased towards a particular sentiment. We'll use SNS to plot the most common hashtags, and wordcloud to plot the most common terms used per sentiment.

Data Preparation

In order to get the dataset ready for the analysis, we'll first develop a tokenizer to:

```
- Convert all letters to lower case since most of the algorithms
consider textual data as sensitivity.

- Turn the tweets into tokens or words in order to remove the
stopwords and puntuation.

- Eliminate unwanted characters, such as HTML tags, punctuation
marks, contractions, special characters, white       spaces etc.

- Remove stop words which don't give any additional value to the
document vector, and will help to increase
the accuracy score.

- Remove empty strings.
```

Data Analysis

Linear support Vector Classifier

Linear support Vector Classifier is a powerful algorithm considered an extension of perceptron, which goal is to maximize the distance between the decision boundary or hyperplane. In this scenario we'll use it to classify tweets that are either sentiment positive or sentiment negative. It was originally created to work on binary data, but nowadays it is capable of performing multi-class classification. This supervised learning method can be used for classification and regression. The reason for the popularity of this algorithm is that it is capable of solving linear and non linear problems thanks to its kernels.

```
The advantages of Linear Support Vector Classifier are:
```

```
    - Effective in high dimensional spaces.
    - Still effective in cases where the number of dimensions is
greater than the number of samples.
    - Uses a subset of training points in the decision function
(called support vectors), so it is also memory
efficient.
    - Versatile: different Kernel functions can be specified for
the decision function. Common kernels are
provided, but it is also possible to specify custom kernels.

The disadvantages of Linear support Vector Classifier are:

    - If the number of features is much greater than the number
of samples, avoid over-fitting in
       choosing regularization term.
    - SVMs do not directly provide probability estimates, these
are calculated using an expensive
       five-fold cross-validation.
```

In order to use Linear SVC, we will import different functions from the scikit-learn library. Some of the functions that we'll need are LinearSVC and metrics to set our classifier, classification_report and confusion_matrix to measure the quality of the predictions and describe the performance of the model.

Feature Extraction

We'll use feature extraction to conver tokens or words into numbers. Tf-Idf stands for "Term Frequency — Inverse Document Frequency". Tf-Idf is a widely used technique in Natural Language Processing. This method is used to compute the weight of each word in a document or text. The weight signifies the importance of the word in the document. The text feature has a total of , but we will be using only 2,500 out of those.

In order to implement Tf-Idf, we'll use scikit-learn library and TfidfVectorizer for feature extraction. Tf-Idf allow us to our tokenizer, but we decided to implement separately to visualize the data cleaning process.

Data Manipulation and Visualization

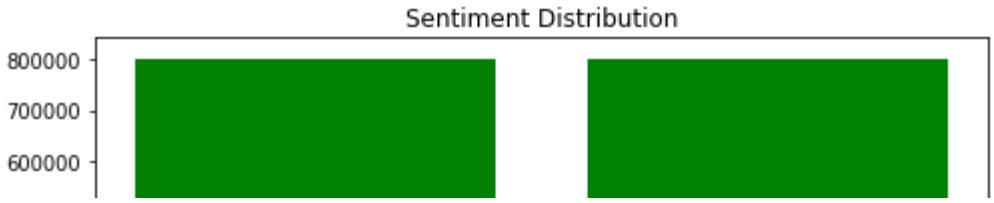
In [1]:
```
#Importing Relevent libraries
import pandas as pd
from nltk.corpus import stopwords
import re
import nltk

import numpy as np
from sklearn.metrics import accuracy_score

import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot

from sklearn.metrics import classification_report, confusion_matrix
from collections import Counter
```

In [2]:
```
#Loading Dataset and assigning Columns Name
df_columns = ["target", "ids", "date", "flag", "user", "text"]
df = pd.read_csv('/Users/carlinsoler/Downloads/training.1600000.processed.noe
target_label = {0: "NEGATIVE", 4: "POSITIVE"}
def sentiment(label):
    return target_label[int(label)]
df['target_label'] = df.target.apply(lambda x: sentiment(x))

#Obtaining number of tweets per sentiment
tweets0 = [_ for i, _ in enumerate(df.text) if df.target[i] == 0]
tweets4 = [_ for i, _ in enumerate(df.text) if df.target[i] == 4]

print(f'Tweets Dataset has {len(df.text)} tweets')
print(f'Tweets Dataset has {len(tweets0)} negative tweets')
print(f'Tweets Dataset has {len(tweets4)} positive tweets')
######### END OF SOLUTION #########
```

```
Tweets Dataset has 1600000 tweets
Tweets Dataset has 800000 negative tweets
Tweets Dataset has 800000 positive tweets
```

In [3]:
```
# Visualizing Sentiment in Dataset
count_target= Counter(df.target_label)
plt.figure(figsize=(8,4))
plt.bar(count_target.keys(), count_target.values(), color = 'green')
plt.title('Sentiment Distribution')
######### END OF SOLUTION #########
```

Out[3]:
```
Text(0.5, 1.0, 'Sentiment Distribution')
```



In [4]:
```
#Sanity Check
print ("First 20 tweets: \n", df.text[0:20])
```

```
First 20 tweets:
 0     @switchfoot http://twitpic.com/2y1zl - Awww, t...
1     is upset that he can't update his Facebook by ...
2     @Kenichan I dived many times for the ball. Man...
3       my whole body feels itchy and like its on fire
4     @nationwideclass no, it's not behaving at all....
5                       @Kwesidei not the whole crew
6                                         Need a hug
7     @LOLTrish hey  long time no see! Yes.. Rains a...
8                @Tatiana_K nope they didn't have it
9                             @twittera que me muera ?
10          spring break in plain city... it's snowing
11                           I just re-pierced my ears
12    @caregiving I couldn't bear to watch it.  And ...
13    @octolinz16 It it counts, idk why I did either...
14    @smarrison i would've been the first, but i di...
15    @iamjazzyfizzle I wish I got to watch it with ...
16    Hollis' death scene will hurt me severely to w...
17                                 about to file taxes
18    @LettyA ahh ive always wanted to see rent  lov...
19    @FakerPattyPattz Oh dear. Were you drinking ou...
Name: text, dtype: object
```

Interpretation of results: As we can observe, the tweets contain many symbols, slang language, misspelled terms, and informal abbreviations. These abbreviations and misspelled terms can be difficult to pre-process since they can vary from user to user. We will proceed to clean the dataset without removing relevant information.

In [5]:
```
#Removing digits from text
df['text'] = df['text'].str.replace('\d+', '')
#Sanity Check
print ("First 20 tweets: \n", df.text[0:20])
```

```
First 20 tweets:
 0     @switchfoot http://twitpic.com/yzl - Awww, tha...
1     is upset that he can't update his Facebook by ...
2     @Kenichan I dived many times for the ball. Man...
3       my whole body feels itchy and like its on fire
4     @nationwideclass no, it's not behaving at all....
5                       @Kwesidei not the whole crew
6                                         Need a hug
7     @LOLTrish hey  long time no see! Yes.. Rains a...
8                @Tatiana_K nope they didn't have it
9                             @twittera que me muera ?
10          spring break in plain city... it's snowing
11                           I just re-pierced my ears
12    @caregiving I couldn't bear to watch it.  And ...
```

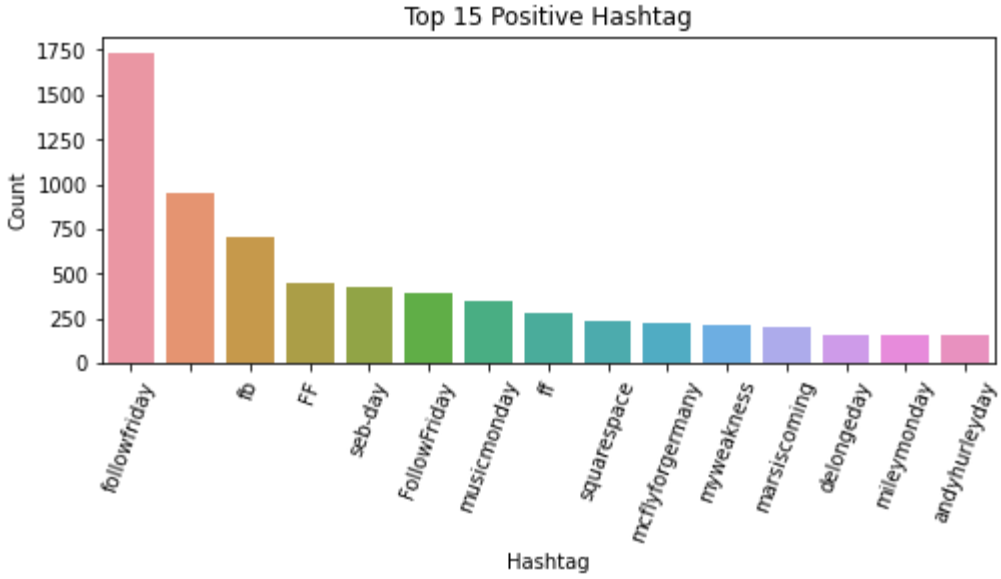
```
13    @octolinz It it counts, idk why I did either. ...
14    @smarrison i would've been the first, but i di...
15    @iamjazzyfizzle I wish I got to watch it with ...
16    Hollis' death scene will hurt me severely to w...
17                                  about to file taxes
18    @LettyA ahh ive always wanted to see rent  lov...
19    @FakerPattyPattz Oh dear. Were you drinking ou...
Name: text, dtype: object
```

In [6]:

```
#Top 15  positive hashtag
words = ' '.join([word for word in df['text'][df['target'] == 4]])
#Collect all hashtags
positive_tag = [htag for htag in words.split() if htag.startswith('#')]
#Remove hashtag symbol (#)
positive_tag = [positive_tag[i][1:] for i in range(len(positive_tag))]
#Count frequency of each word
positive_tag_freqcount = nltk.FreqDist(positive_tag)
positive_tag_df = pd.DataFrame({'Hashtag' : list(positive_tag_freqcount.keys(
                        'Count' : list(positive_tag_freqcount.values())})
```

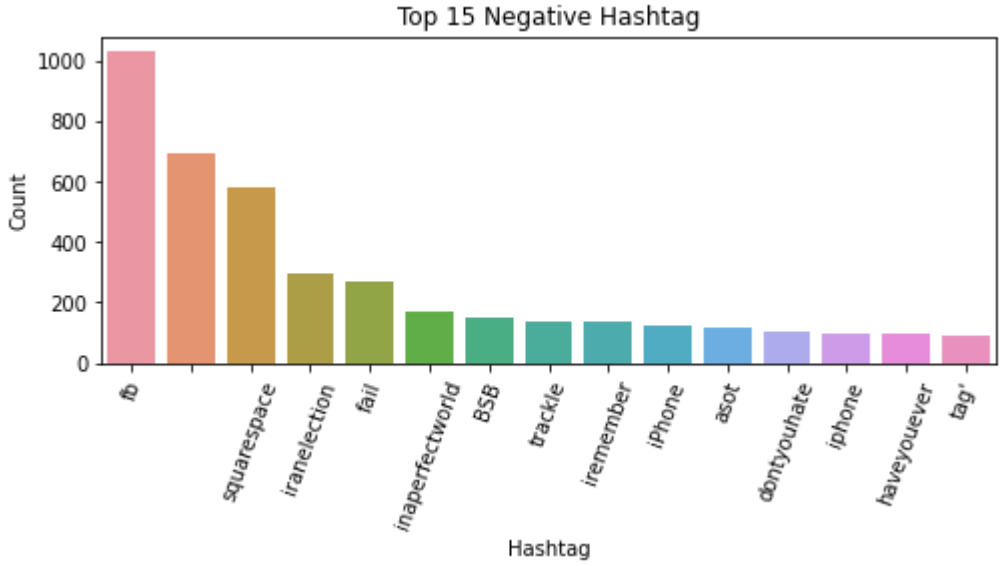
In [7]:

```
#Plotting hashtag
most_frequent = positive_tag_df.nlargest(columns="Count", n = 15)
plt.figure(figsize=(8,3))
ax = sns.barplot(data=most_frequent, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.xticks(rotation=70)
plt.title("Top 15 Positive Hashtag")
plt.show()
######### END OF SOLUTION #########
```



In [8]:
```
#Top 15 negative hashtag
neg_words = ' '.join([word for word in df['text'][df['target'] == 0]])
#Collect all hashtags
negative_tag = [htag for htag in neg_words.split() if htag.startswith('#')]
#Remove hashtag symbol (#)
negative_tag = [negative_tag[i][1:] for i in range(len(negative_tag))]
#Count frequency of each word
negative_tag_freqcount = nltk.FreqDist(negative_tag)
negative_tag_df = pd.DataFrame({'Hashtag' : list(negative_tag_freqcount.keys(
                        'Count' : list(negative_tag_freqcount.values())})
```

In [9]:
```
#Plotting hashtag
most_frequent = negative_tag_df.nlargest(columns="Count", n = 15)
plt.figure(figsize=(8,3))
ax = sns.barplot(data=most_frequent, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.xticks(rotation=70)
plt.title("Top 15 Negative Hashtag")
plt.show()
######### END OF SOLUTION #########
```



Interpretation of results: Hashtags give us a good idea of the topic that is being discussed. As mentioned by Marie Ennis-O'Connor in her article "A Simple Guide to Analyzing Hashtags", hashtags help to measure the level of interest and sentiment. It also helps identify trends and key influencers. In the analysis above, we can see that most common hastags for positive tweets are #followfriday, #fb and #ff. The most common negative hastags for negatives tweets are #fb, #squarespace, and #iran election. Both sentiments have high number of empty hashtags, which we can infer that the # sign was used alone to represent the # sign.

Data Pre-Processing

In [10]:
```
#Removing Stopwords, symbols and punctuation
from string import punctuation
stop_words = list(re.split('', punctuation)) + stopwords.words('english') + [
def mytokenizer(text, lem=False):
    text = re.sub("@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+", ' ', str(text).lo
    tokens = []
    for token in text.split():
        if token not in stop_words:
                tokens.append(token)
    return " ".join(tokens)
df.text = df.text.apply(lambda x: mytokenizer(x))
#Sanity Check
print ("First 20 tweets after tokenizer: \n", df.text[0:20])
```

```
First 20 tweets after tokenizer:
 0          awww bummer shoulda got david carr third day
1     upset update facebook texting might cry result...
2     dived many times ball managed save rest go bounds
3                     whole body feels itchy like fire
4                                    behaving mad see
5                                          whole crew
6                                            need hug
7     hey long time see yes rains bit bit lol fine t...
8                                                 nope
9                                           que muera
10                     spring break plain city snowing
11                                        pierced ears
12             bear watch thought ua loss embarrassing
13                counts idk either never talk anymore
14    would first gun really though zac snyder douch...
15             wish got watch miss iamlilnicki premiere
16    hollis death scene hurt severely watch film wr...
17                                        file taxes
18       ahh ive always wanted see rent love soundtrack
19              oh dear drinking forgotten table drinks
Name: text, dtype: object
```

In [11]:
```
#Removing empty rows and re-setting index
df = df[df.astype(str)['text'] != '[]']
df = df.reset_index(drop=True)
print(f'After pre-processing, tweets Dataset has {len(df)} tweets')
######### END OF SOLUTION #########
```

```
After pre-processing, tweets Dataset has 1600000 tweets
```

Interpretation of results: As we can observed, the tweets look cleaner. The symbols and contractions have been removed. Also, all of those words that didn't improved the dataset were removed using the stop_words functions. Unfortunately,like mentioned before, words that have extra letters, words that were not separated by space, and misspelled words are very difficult to remove. After cleaning the dataset, we might have empty columns, so we deleted those empty columns to avoid future errors when running the models or visualizing the results.

In [12]:

```
#Top Words for Negative Sentiment
from wordcloud import WordCloud
top_words = ' '.join([word for word in df['text'][df['target'] == 0]])
wordcloud = WordCloud(background_color='white',max_words=80, width = 600, hei
print('          Top 80 Words Used in Negative Tweets')
plt.figure(figsize= (10,6))
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
######### END OF SOLUTION #########
```

Top 80 Words Used in Negative Tweets

In [13]:

```
#Top Words for Negative Sentiment
from wordcloud import WordCloud
top_words_pos = ' '.join([word for word in df['text'][df['target'] == 4]])
wordcloud_pos = WordCloud(background_color='white', width = 600, height = 400
print('          Top 80 Words Used in Positive Tweets')
plt.figure(figsize= (10,6))
plt.imshow(wordcloud_pos, interpolation = 'bilinear')
plt.axis('off')
plt.show()
######### END OF SOLUTION #########
```

Top 80 Words Used in Positive Tweets

Interpretation of results: When we interpret these wordclouds, it is very important to look at terms from the smaller to the bigger. The big terms are very similar between both wordclous, but the smallest terms give us a better understanding of the common and relevant terms used on each sentiment. Please note that we selected a high number of words for the wordcloud in order to see the middle terms or the terms that give us a better idea of the sentiment.

Data Preparation

In [14]:
```
text = df.text.astype('U')
sentiment = df.target.astype('U')
```

In [15]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
y = pd.Series(sentiment)
tfidf_original= TfidfVectorizer(dtype=np.float32)
X_tfidf_original= tfidf_original.fit_transform(text).todense()
print(f'N data points= {X_tfidf_original.shape[0]}, M features the dataset ha
```

```
N data points= 1600000, M features the dataset has in total= 318841
```

In [16]:
```
tfidf = TfidfVectorizer(dtype=np.float32, max_features = 2500)
X_tfidf = tfidf.fit_transform(text).todense()
print(f'N data points = {X_tfidf.shape[0]}, M features we will use in our mod
```

```
N data points = 1600000, M features we will use in our model = 2500
```

Interpretation of Results: The original dataset has a total of 318,841 features or words. We used 2,500 features for the analysis since that is enough to obtain a decent accuracy score. In other words, more words or features won't improve or will improve very little the accuracy score.

In [17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2
print('Training Set Partition: X=',(X_train.shape))
print('Testing Set Partition: X=',(X_test.shape))
```

```
Training Set Partition: X= (1280000, 2500)
Testing Set Partition: X= (320000, 2500)
```

In [18]:
```
%%time
from sklearn.svm import LinearSVC
svm_lin = LinearSVC(class_weight='balanced')
y_pred_svc = svm_lin.fit(X_train, y_train).predict(X_test)
print(f'Linear SVC accuracy= {accuracy_score(y_test, y_pred_svc)*100:.1f}%')
######### END OF SOLUTION #########
```

```
Linear SVC accuracy= 76.0%
CPU times: user 1min 48s, sys: 1min 48s, total: 3min 36s
Wall time: 4min 8s
```

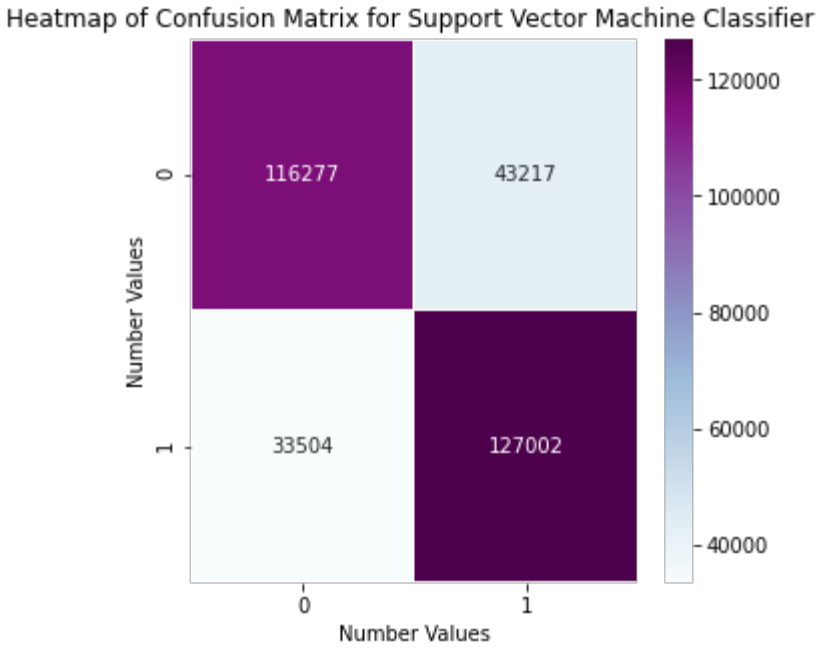
Evaluation

In [19]:
```
from sklearn import metrics
print(metrics.classification_report(y_test,y_pred_svc))
######### END OF SOLUTION #########
```

```
              precision    recall  f1-score   support

           0       0.78      0.73      0.75    159494
           4       0.75      0.79      0.77    160506

    accuracy                           0.76    320000
   macro avg       0.76      0.76      0.76    320000
weighted avg       0.76      0.76      0.76    320000
```

Interpretation of results: The accuracy score is 76.0% after fitting the model on the testing set. This means that we were able to predict the right sentiment 76% of the times or with an accuracy of 76% overall. The model had a decent generalization performance taking into consideration that these were random tweets. Now we will interpret the recall and precision scores. The recall score means the proportion of correct classifications from cases that are actually positive. For instance, of the 800,000 tweets that were label as negative on the testing set, 584,000 were classified correctly or as negative. The precision score means the proportion of correct classifications from cases that are predicted as positive. For instance, 600,000 tweets were classified as positive.

In [20]:
```
#Plotting Confusion Matrix
conf_mat_df_svc = pd.DataFrame(confusion_matrix(y_test,y_pred_svc), columns=r
index=range(2))
#HeatMap
fig, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat_df_svc, linewidth = 0.2, annot = True, cmap = 'BuPu',fmt
plt.title('Heatmap of Confusion Matrix for Support Vector Machine Classifier
plt.xlabel('Number Values')
plt.ylabel('Number Values')
plt.show()
######### END OF SOLUTION #########
```



Interpretation of results: The confusion matrix gives us a better idea of which digits were erroneously classified by others. For instance, 43,216 tweets with a negative sentiment were classified as positive and 33,504 tweets were classified as positive when they were negative.

In [21]:
```
def top_freq_term (_ax, _clf, _top_M, _fnames):
    coef = _clf.coef_.ravel()
    top_pos_coef= np.argsort(coef)[-_top_M:]
    top_neg_coef= np.argsort(coef)[:_top_M]
    top_coef = np.hstack([top_neg_coef, top_pos_coef])

    colors= ['red' if c < 0 else 'blue' for c in coef[top_coef]]
    _ax.barh(np.arange(2 * _top_M), coef[top_coef], color = colors)
    _ax.set_yticks(np.arange(0, 1+2* _top_M))
    _ax.set_yticklabels(_fnames[top_coef])
    plt.title ('Top Commom Terms')
    plt.tight_layout()
```

In [ ]:
```
TOP_M=10

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12,4))

svm_lin.fit(X_tfidf,y)
fnames = np.array(tfidf.get_feature_names())
top_freq_term (ax1, svm_lin, TOP_M, fnames)

svm_lin.fit(X_tfidf_original,y)
fnames = np.append(np.array(tfidf.get_feature_names()))
top_freq_term (ax2, svm_lin, TOP_M, fnames)
######### END OF SOLUTION #########
```

Conclusion

The end goal of this project was to classify and predict tweets sentiment using Tf-Idf and Linear SVC. We were able to successfully accomplish this goal by using TF-Idf to build the model. We later analyzed the results of the classifier in terms of its performance. We computed two different Tf-Idf: one using the entire dataset and another one using only 2,500 words. The objective was to demostrate how a high number of features is not necessary for a good analysis and how the terms can vary. Like mentioned before, do to the nature of the dataset and its pre-processing challanges, a accuracy score of 76% is decent. In conclusion and as explained on the paper written by Andrew L. Mass on his article "Learning word vectors for sentiment analysis.", the semantic analysis is not enough to predict word meaning in context. That is the reason why the results obtained for both sentiments are so different. The sentiment connotation is key in order to find the meaning of words in context. This model still have room for improvement. We can use a different vectorizing technique or word embeddings, such as word2vec to improve the model performance.Please note that this computationally expensive.

References

KazAnova, Μαριος Μιχαηλιδης “Sentiment140 Dataset with 1.6 Million Tweets.” Kaggle, 13 Sept. 2017, www.kaggle.com/kazanova/sentiment140.

Scott, William. “TF-IDF for Document Ranking from Scratch in Python on Real World Dataset.” Medium, Towards Data Science, 21 May 2019, towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089.

O'Connor, Marie Ennis, et al. “A Simple Guide to Analyzing Hashtags.” LinkedIn, Linkedin, 5 Jan. 2015, www.linkedin.com/pulse/guide-analyzing-hashtags-marie-ennis-o-connor/.

Shung, Koo Ping. “Accuracy, Precision, Recall or F1?” Medium, Towards Data Science, 10 Apr. 2020, towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

Maas, Andrew L., et al. "Learning word vectors for sentiment analysis." Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1. Association for Computational Linguistics, 2011.