Text Classification Simulation

July 28, 2019

Team Members : - Anmol More : 11915043 - Dharani Kiran Kavuri : 11915033 - Shubhendu Vimal : 11915067

References : - https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html - https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a - https://machinelearningmastery.com/clean-text-machine-learning-python/ - https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk

```
In [1]: import wikipedia
        import pandas as pd
        from nltk import sent_tokenize
        from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer
        import string

        #import libraries for feature processing
        from sklearn import preprocessing
        from sklearn import metrics
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score

        #import necessary ml algorithms
        from sklearn.linear_model import SGDClassifier
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.ensemble import RandomForestClassifier
        from xgboost import XGBClassifier
```

Data Collection

Collect data from wikipedia pages

```
In [2]: #Ref : https://pypi.org/project/wikipedia/
        topics = ['Brexit', 'Donald Trump', 'Game of Thrones', 'Bitcoin']
        for topic in topics :
            wiki_page = wikipedia.WikipediaPage(title = topic)
```

```
            with open(topic + ".txt", "w") as text_file:
                text_file.write(wiki_page.content)
```

Prepare a labeled dataframe by readling sentences from text corpus and combine all dataframes. Steps - - read back text files stored after scrapping wikipedia - Use nltk to get sentences - prepare dataframe

```
In [3]: def prepare_df(topic) :
            file_reader = open(topic + ".txt","r",encoding="utf-8")
            text = file_reader.read()
            file_reader.close()

            df = pd.DataFrame(columns=['text', 'topic'])
            sentences = sent_tokenize(text)
            for sentence in sentences :
                df = df.append({'text': sentence, 'topic': topic}, ignore_index=True)
            return df

        brexit_df = prepare_df("Brexit")
        trump_df = prepare_df("Donald Trump")
        got_df = prepare_df("Game of Thrones")
        bitcoin_df = prepare_df("Bitcoin")

        print(brexit_df.sample(3))
        print(trump_df.sample(3))
        print(got_df.sample(3))
        print(bitcoin_df.sample(3))
```

```
                                                  text   topic
213  The deal was voted against 391 to 242, a loss ...  Brexit
98   According to the BBC, "The prime minister ackn...  Brexit
368  === Scotland ===\n\nAs suggested by the Scotti...  Brexit
                                                  text         topic
411  The official counts were 304 and 227 respectiv...  Donald Trump
204  In response to mounting complaints, Trump's te...  Donald Trump
221  His first published book in 1987 was Trump: Th...  Donald Trump
                                                  text            topic
27   He introduced gray tones into a black-and-whit...  Game of Thrones
84   The only exceptions were Peter Dinklage and Se...  Game of Thrones
157  At the beginning of the fourth season Engelen'...  Game of Thrones
                                                  text    topic
288  In July 2017, billionaire Howard Marks referre...  Bitcoin
20   During its 30 months of existence, beginning i...  Bitcoin
104  The PoW requires miners to find a number calle...  Bitcoin
```

Check sampled data from combined dataframe

```
In [4]: df = pd.concat([brexit_df, trump_df, got_df, bitcoin_df])
        df.sample(5)
```

Out[4]:

	text	topic
347	S. J. Clarkson has been announced to direct an...	Game of Thrones
258	=== Racial views ===\n\nTrump has a history of...	Donald Trump
496	Several states immediately challenged the DACA...	Donald Trump
87	In April 2017 an online petition aimed at ment...	Donald Trump
608	Mueller also investigated the Trump campaign's...	Donald Trump

Data Cleaning and Preparation

In [5]:
```
def clean_text(text_line) :
    tokens = word_tokenize(text_line)
    tokens = [w.lower() for w in tokens]  #lowercase the word tokens

    table = str.maketrans('', '', string.punctuation)
    tokens = [token.translate(table) for token in tokens]
    words = [word for word in tokens if word.isalpha()]  #remove all alpha numeric

    stop_words = stopwords.words('english')  #remove all stop words
    words = [word for word in words if not word in stop_words]

    porter = PorterStemmer()
    words = [porter.stem(word) for word in words]
    return ' '.join([str(word) for word in words])

df['clean_text'] = df['text'].apply(lambda line: clean_text(line))
```

In [6]:
```
df.sample(10)
```

Out[6]:

	text	topic	\
120	The main reason people voted Remain was that "...	Brexit	
286	Itzkoff also wrote that critics fear that "rap...	Game of Thrones	
150	In October 2016, European Commission President...	Brexit	
296	=== Popular culture ===\n\nTrump has been the ...	Donald Trump	
50	MPs also voted on four options on 1 April 2019...	Brexit	
527	He has repeatedly criticized the Joint Compreh...	Donald Trump	
143	Trump also acquired a partially completed buil...	Donald Trump	
194	Additionally, the plan appears to breach stand...	Brexit	
387	There may be an interim deal between the time ...	Brexit	
300	There were an estimated 24 million bitcoin use...	Bitcoin	

	clean_text
120	main reason peopl vote remain risk vote leav e...
286	itzkoff also wrote critic fear rape becom perv...
150	octob european commiss presid juncker said eu ...
296	popular cultur trump subject comedian flash ca...
50	mp also vote four option april second round in...
527	repeatedli critic joint comprehens plan action...
143	trump also acquir partial complet build atlant...
194	addit plan appear breach standard wto rule

```
387  may interim deal time uk leav eu final relatio...
300  estim million bitcoin user primarili use bitco...
```

Split data in train and test set in 70:30 ratio

```
In [7]: X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['topic'], tes

        #specify own encoding scheme so, it becomes easier to identify later
        encoder = preprocessing.LabelEncoder()
        encoder.fit(topics)
        y_train = encoder.fit_transform(y_train)
        y_test = encoder.fit_transform(y_test)

        encoded_topics = list(encoder.inverse_transform([0,1,2,3]))
        print(encoded_topics)
```

```
['Bitcoin', 'Brexit', 'Donald Trump', 'Game of Thrones']
```

Very train and test sets are correcly divided. It would do random shuffle by default

```
In [8]: print("Training dataset size :",X_train.shape)
        print("Test dataset size :",X_test.shape)

        print("Labeled dataset size :",y_train.shape)
        print("Labeled dataset size :",y_test.shape)
```

```
Training dataset size : (1192,)
Test dataset size : (512,)
Labeled dataset size : (1192,)
Labeled dataset size : (512,)
```

Run ML Algoithms

Steps - - Create count vector and vectorized tfidf - Create word vector for each of matrix - Fit ML algorithms on both - Check confusion matrix for accuracy

```
In [9]: #Ref : https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.te
        count_vect = CountVectorizer()
        X_train_count_vect = count_vect.fit_transform(X_train)
        X_test_count_vect = count_vect.transform(X_test)
        print("Check train and test set size :")
        print(X_train_count_vect.shape, X_test_count_vect.shape)
```

```
Check train and test set size :
(1192, 4315) (512, 4315)
```

```
In [10]: #Try multiple combinations, go with default for now
         # use_idf = False,
         # smooth_idf = True,
         # sublinear_tf = False,
         # ngram_range=(1,1)
         tfidf_vector = TfidfVectorizer()
         X_train_tfidf_vect = tfidf_vector.fit_transform(X_train)
         X_test_tfidf_vect = tfidf_vector.transform(X_test)
         print("Check train and test set size :")
         print(X_train_tfidf_vect.shape, X_test_tfidf_vect.shape)

Check train and test set size :
(1192, 4315) (512, 4315)
```

Navie Bayes

```
In [11]: MBModel = MultinomialNB().fit(X_train_count_vect, y_train)
         predicted = MBModel.predict(X_test_count_vect)
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on count vector :", accuracy_score(y_test, predicted))

         MBModel = MultinomialNB().fit(X_train_tfidf_vect, y_train)
         predicted = MBModel.predict(X_test_tfidf_vect)
         print("\n\n")
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on tfidf vector :", accuracy_score(y_test, predicted))
```

	precision	recall	f1-score	support
Bitcoin	0.95	0.94	0.95	87
Brexit	0.92	0.98	0.95	125
Donald Trump	0.95	0.93	0.94	177
Game of Thrones	0.94	0.93	0.93	123
micro avg	0.94	0.94	0.94	512
macro avg	0.94	0.94	0.94	512
weighted avg	0.94	0.94	0.94	512

```
accurancy score on count vector : 0.94140625
```

	precision	recall	f1-score	support
Bitcoin	0.99	0.83	0.90	87
Brexit	0.95	0.93	0.94	125
Donald Trump	0.80	0.98	0.88	177
Game of Thrones	0.97	0.80	0.87	123

```
      micro avg       0.90      0.90      0.90       512
      macro avg       0.93      0.88      0.90       512
   weighted avg       0.91      0.90      0.90       512

accurancy score on tfidf vector : 0.896484375
```

Random Forest

```
In [12]: rf_model = RandomForestClassifier().fit(X_train_count_vect, y_train)
         predicted = rf_model.predict(X_test_count_vect)
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on count vector :", accuracy_score(y_test,predicted))

         rf_model = RandomForestClassifier().fit(X_train_tfidf_vect, y_train)
         predicted = rf_model.predict(X_test_tfidf_vect)
         print("\n\n")
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on tfidf :", accuracy_score(y_test,predicted))
```

```
                precision    recall  f1-score   support

       Bitcoin       0.67      0.98      0.79        87
        Brexit       0.88      0.94      0.91       125
   Donald Trump       0.93      0.77      0.85       177
Game of Thrones       0.94      0.80      0.86       123

      micro avg       0.86      0.86      0.86       512
      macro avg       0.86      0.87      0.85       512
   weighted avg       0.88      0.86      0.86       512

accurancy score on count vector : 0.85546875



                precision    recall  f1-score   support

       Bitcoin       0.93      0.91      0.92        87
        Brexit       0.89      0.95      0.92       125
   Donald Trump       0.84      0.92      0.88       177
Game of Thrones       0.96      0.78      0.86       123

      micro avg       0.89      0.89      0.89       512
      macro avg       0.90      0.89      0.89       512
   weighted avg       0.90      0.89      0.89       512

accurancy score on tfidf : 0.890625
```

```
/Users/anmol/anaconda3/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarni
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/Users/anmol/anaconda3/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarni
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

SGD Classifier

```
In [13]: sgd_model = SGDClassifier().fit(X_train_count_vect, y_train)
         predicted = sgd_model.predict(X_test_count_vect)
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on count vector :", accuracy_score(y_test,predicted))

         sgd_model = SGDClassifier().fit(X_train_tfidf_vect, y_train)
         predicted = sgd_model.predict(X_test_tfidf_vect)
         print("\n\n")
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on tfidf :", accuracy_score(y_test,predicted))
```

```
                precision    recall  f1-score   support

       Bitcoin       0.93      0.91      0.92        87
        Brexit       0.92      0.92      0.92       125
  Donald Trump       0.91      0.90      0.91       177
Game of Thrones      0.90      0.93      0.91       123

     micro avg       0.91      0.91      0.91       512
     macro avg       0.92      0.91      0.91       512
  weighted avg       0.91      0.91      0.91       512

accurancy score on count vector : 0.9140625



                precision    recall  f1-score   support

       Bitcoin       0.96      0.93      0.95        87
        Brexit       0.97      0.93      0.95       125
  Donald Trump       0.90      0.96      0.93       177
Game of Thrones      0.94      0.92      0.93       123

     micro avg       0.94      0.94      0.94       512
     macro avg       0.94      0.93      0.94       512
  weighted avg       0.94      0.94      0.94       512

accurancy score on tfidf : 0.9375


/Users/anmol/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py
```

```
  FutureWarning)
```

Gradient Boosted Trees

```
In [14]: xgb_model = XGBClassifier().fit(X_train_count_vect, y_train)
         predicted = xgb_model.predict(X_test_count_vect)
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on count vector :", accuracy_score(y_test,predicted))

         xgb_model = XGBClassifier().fit(X_train_tfidf_vect, y_train)
         predicted = xgb_model.predict(X_test_tfidf_vect)
         print(metrics.classification_report(y_test, predicted, target_names=encoded_topics))
         print("accurancy score on tfidf vector :", accuracy_score(y_test,predicted))
```

```
                precision    recall  f1-score   support

       Bitcoin       1.00      0.83      0.91        87
        Brexit       0.98      0.90      0.94       125
  Donald Trump       0.74      0.98      0.85       177
Game of Thrones      0.98      0.73      0.84       123

     micro avg       0.88      0.88      0.88       512
     macro avg       0.93      0.86      0.88       512
  weighted avg       0.90      0.88      0.88       512

accurancy score on count vector : 0.875
                precision    recall  f1-score   support

       Bitcoin       1.00      0.84      0.91        87
        Brexit       0.97      0.90      0.93       125
  Donald Trump       0.75      0.97      0.85       177
Game of Thrones      0.95      0.73      0.83       123

     micro avg       0.87      0.87      0.87       512
     macro avg       0.92      0.86      0.88       512
  weighted avg       0.89      0.87      0.87       512

accurancy score on tfidf vector : 0.873046875
```

Conclusion

Ensemble method seems to performing best for our corpus

In general Tfidf vector format performs better than count vector. However we don't have a clear winner. Also ML algorithsm performs much better than LTM models built in R