Nishtha Singhal
2017302
RMSSD

Assignment:

Sentiment Analysis for Fake News

19th November 2019

OVERVIEW

Training: Trained a Naive Bayes classifier using any 800 news headlines from the dataset.

Evaluation: Using the trained model, report the accuracy of the remaining 60 headlines.

Preprocessing steps used:

1. Removed non-English entries in the given dataset.
2. Read modified `.csv` files and converted into `Pandas data frames`.
3. Dropped the first two columns of the dataset as they were not required.
4. Removed punctuations in the headlines by `clean()`.
5. Tokenized the sentences into a list of words by `nltk.tokenize.word_tokenize()`.
6. Removed stop words by `remstop()` using `nltk.corpus.stopwords`.
7. Tagged each word with the category of part of speech they belong to using `nltk.pos_tag()`.
8. Collected the prepositions, nouns, pronouns, participle and verbs
9. Lemmatized the words collected using `nltk.stem.WordNetLemmatize.lemmatize()`.
10. Converted words to lowercase.
11. I make feature words out of the words that occur more than twice so that there aren't any classifications on words that are irrelevant.

Please scroll ahead.

Assumptions, if any:

I assumed we do not need to run the classifier on the non-English headlines and thus removed 3 entries leaving us with 857 total headlines.

I trained the classifier on 800 entries and tested on 57.

Observations, if any:

- `["IN", "NN”, "NNP", "PRP", "PRP$”, "RP", "VBP"]`

  The prepositions, nouns, pronouns, participle and verbs in the headlines were the most descriptive of the headlines in our context and relevance.

- There wasn’t much difference found in lemmatizing and stemming the words as reflected by the classifier accuracy, however, lemmatizing seemed to be a slightly more accurate method.

Accuracy value:

92.98245614035088

Attached below is the code and output for the trained model.

- In the folder:
  - **dataset folder** contains the modified datasets used as mentioned above.
  - **.ipynb** and **.py** files of the running code
  - Pickle dump of the trained model of the Naive Bayes classifier used, as **naivebayes.pickle**

```
import nltk, string, random, re, pickle
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
datafake = pd.read_csv("/content/drive/My Drive/data/politifact_fake - politifact_fake.csv
datareal = pd.read_csv("/content/drive/My Drive/data/politifact_real - politifact_real.csv
datafake.drop(columns=['id','news_url'],inplace=True)
datareal.drop(columns=['id','news_url'],inplace=True)
```

```
ps = PorterStemmer()
lem = WordNetLemmatizer()
stopword = set(stopwords.words( 'english' ))
```

```
def clean(data):
  return re.sub(r'[^(a-zA-Z)\s]','', data)
def remstop(data):
  global stopword
  return [w for w in data if w not in stopword]
def tagpos(data):
  return nltk.pos_tag(data)
```

```
documents=[]

def addindocs(data,flag):
  global documents
  for p in data.title:
    documents.append(((clean(p).lower()),flag))

addindocs(datareal, "true")
addindocs(datafake,"false")
random.shuffle(documents)
```

```
words=[]
allowedpos=["IN","NN" ,"NNP","PRP","PRP$" , "RP","VBP"]

def addinwords(data):
  global allowedpos, words
  for p in data.title:
    tokenized = word_tokenize(clean(p))
    stopremoved=remstop(tokenized)
    postagged = nltk.pos_tag(tokenized)
    for w in postagged:
      if w[1] in allowedpos:
        words.append(lem.lemmatize(w[0].lower()))

addinwords(data_real)
addinwords(data_fake)
```

```
words = nltk.FreqDist(words)
```

```
wordfeatures=[]
for word in words:
  if(words[word]>2):
    wordfeatures.append(word)

def docfeatures(doc):
    docwords = word_tokenize(doc)
    # for i,word in enum?(word)
    features = {}
    for word in wordfeatures:
        features[word] = (lem.lemmatize(word) in docwords)
    return features
```

```
featuresets = [(docfeatures(d), c) for (d,c) in documents]
trainset, testset = featuresets[:800], featuresets[800:]
```

```
classifier = nltk.NaiveBayesClassifier.train(trainset)
print(nltk.classify.accuracy(classifier, testset)*100)
```

```
92.98245614035088
```

```
classifier.show_most_informative_features(20)
```

```
Most Informative Features
              transcript = True             true : false  =     20.7 : 1.0
                   after = True            false : true   =     16.0 : 1.0
                    that = True            false : true   =     10.1 : 1.0
                    week = True             true : false  =      9.9 : 1.0
                    will = True            false : true   =      8.8 : 1.0
            presidential = True             true : false  =      8.5 : 1.0
                 million = True            false : true   =      8.2 : 1.0
                  before = True            false : true   =      8.2 : 1.0
                    care = True             true : false  =      7.8 : 1.0
                  health = True             true : false  =      7.8 : 1.0
                      if = True            false : true   =      7.5 : 1.0
                     you = True            false : true   =      7.3 : 1.0
                      is = True            false : true   =      7.2 : 1.0
                     out = True            false : true   =      6.9 : 1.0
                     are = True            false : true   =      6.5 : 1.0
                  senator = True             true : false  =      6.5 : 1.0
                congress = True             true : false  =      5.8 : 1.0
                     all = True            false : true   =      5.6 : 1.0
                     our = True            false : true   =      5.6 : 1.0
                      be = True            false : true   =      5.5 : 1.0
```

```
currclassifier = open( "/content/drive/My Drive/data/naivebayes.pickle", "wb" )
pickle.dump (classifier, currclassifier)
currclassifier.close ()
```

```
```