Assignment 7

Archana Molasi, Dwipam Katariya, Krishna Mahajan, Sanjana Agrawal

****Theory****

***Levels of Sentiment Analysis*:**
A sentiment is a complex, multi dimensional entity.
The levels of sentiment analysis are:
Document level: The entire document is considered a single entity and a single sentiment of the document is derived.
Sentence level: Each sentence is analyzed for positive, negative or neutral sentiment.
Aspect and Entity level: In this type of analysis, we directly look at the sentiment instead of sentence structures or other language constructs.

***Sentiment lexicon and issues*:**

Some words are easily classified as positive or negative sentiment words.
eg: good, wonderful, great are positive sentiment words
terrible, bad, miserable are negative sentiment words.
Apart from these words, there are phrases which are used to express sentiments.

Some of the issues associated with sentiment analysis are:
Sarcastic comments: It is difficult to analyse the sentiment of a sarcastic sentence.
Eg: What a great day? I lost my job.
Although the sentence has a positive sentiment word, it expresses a negative sentiment.
Sentiment with no sentiment words: Many sentences without sentiment words can also express an opinion.
Eg: This washing machine uses a lot of water.
Neutral sentences: Some sentences are just a form of question or suggestions which use a positive sentiment word.
Eg: Does anyone know a delicious, easy breakfast to make?
This is just a question and intends to express no sentiment, although it uses positive sentiment word.
Opposite orientation of sentiment word: Another common issue is when a sentiment word expresses an opposite sentiment.
Eg: Life sucks vs this vaccum cleaner sucks all the dust.
The sentiment word sucks has an opposite sentiment in both the sentences.

**** Sentiment Analysis Project Overview ****
- As part of our research on sentiment analysis we worked on [Kaggle competition on Movie sentiment sentiment analysis](https://www.kaggle.com/c/word2vec-nlp-tutorial).

****Data Overview****
- sentiment analysis project
- 100,000 movie reviews
- 25,000 Train reviews & 25,000 Test reviews
- Another 50,000 unlabeled reviews

- Evaluation metric was AUC ROC

****Data Exploration****
- we started with Data Exploration and basic Data Understaning.
- The dataset had three columns id,sentiment,review
- The Movie reviews were like typical paragraphs in English with also punctuations,numbers and emojis
- The good thing about the Dataset was both the classes were balanced i,e 50% positive sentiment and 50% negative sentiments.So any good classifier we build should have accuracy lot more than 50%
- Also during data exploration we observed that reviews had HTML tags as typical in any online data.

```
In [4]: train.head()
```

Out[4]:

	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...
3	"3630_4"	0	"It must be assumed that those who praised thi...
4	"9495_8"	1	"Superbly trashy and wondrously unpretentious ...

```
In [6]: #Sample review
        train["review"][0]
```

```
Out[6]: '"With all this stuff going down at the moment with MJ i\'ve started l
        istening to his music, watching the odd documentary here and there, wa
        tched The Wiz and watched Moonwalker again. Maybe i just want to get a
         certain insight into this guy who i thought was really cool in the ei
        ghties just to maybe make up my mind whether he is guilty or innocent.
         Moonwalker is part biography, part feature film which i remember goin
        g to see at the cinema when it was originally released. Some of it has
         subtle messages about MJ\'s feeling towards the press and also the ob
        vious message of drugs are bad m\'kay.<br /><br />Visually impressive
         but of course this is all about Michael Jackson so unless you remotel
        y like MJ in anyway then you are going to hate this and find it borin
        g. Some may call MJ an egotist for consenting to the making of this mo
        vie BUT MJ and most of his fans would say that he made it for the fans
         which if true is really nice of him.<br /><br />The actual feature fi
        lm bit when it finally starts is only on for 20 minutes or so excludin
        g the Smooth Criminal sequence and Joe Pesci is convincing as a psycho
        pathic all powerful drug lord. Why he wants MJ dead so bad is beyond m
        e. Because MJ overheard his plans? Nah, Joe Pesci\'s character ranted
         that he wanted people to know it is he who is supplying drugs etc so
         i dunno, maybe he just hates MJ\'s music.<br /><br />Lots of cool thi
        ngs in this like MJ turning into a car and a robot and the whole Speed
         Demon sequence. Also, the director must have had the patience of a sa
        int when it came to filming the kiddy Bad sequence as usually director
        s hate working with one kid let alone a whole bunch of them performing
         a complex dance scene.<br /><br />Bottom line, this movie is for peop
        le who like MJ on one level or another (which i think is most people).
         If not, then stay away. It does try and give off a wholesome message
         and ironically MJ\'s bestest buddy in this movie is a girl! Michael J
        ackson is truly one of the most talented people ever to grace this pla
```

****Data Cleaning****
- After Data Exploration and Data Understanding we started with Data Cleaning
- We used python's library BeautifulSoup to remove HTML tags
- We then removed numbers,punctuation ,emojis using regular expression
- We did typical text cleaning steps like lowercasing all the words,Removing Stopwords
- And finally using NLTK library
  - Tokenized all the words in the review

- And lemmatized all the tokens
- Finally joined all the Tokens to retrieve the cleaned reviews
- So we wrote python function of this steps for all the 100K reviews and store all the reviews in list of list format
for training,test & unlabeled reviews

```
#Putting it all together
def my_tokenizer(s):
    s = BeautifulSoup(s,"lxml")
    s = s.get_text()
    s = re.sub("[^a-zA-Z]"," ",s)
    s = s.lower()
    tokens=nltk.tokenize.word_tokenize(s)
    tokens =[wordnet_lemmatizer.lemmatize(t) for t in tokens]
    tokens = [token for token in tokens if token not in stopwords]
    return " ".join(tokens)
```

****Data Modeling****
- Now after all the Exploration and Data Cleaning ,for main part of sentiment Analysis i,e Data Modeling we used three different approaches and 2 different classifiers such logistic,random forest

****First Approach****
- First approach was Bag of words , one gram model
- So in this approach our feature space was 5000 most frequent words
- So now each review was 5000 feature long with each entry equal to number of occurence of that word
- we used scikit learn CountVectorizer function for this but we have also coded it ourselves
- Now when we run our classifier on this approach we got following results
  - Logistic Regression 0.55
  - Random Forest 0.59
- Also we did parameter tuning to some extent for each of this algorithms

****Sample Code for Bag of words models****

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer
        vectorizer = CountVectorizer(analyzer = "word",tokenizer = None,
                                     preprocessor = None,stop_words = None,max_fe

        train_data_features = vectorizer.fit_transform(clean_train_reviews)
        train_data_features = train_data_features.toarray()

        test_data_features = vectorizer.fit_transform(clean_test_reviews)
        test_data_features = test_data_features.toarray()
```

****Second Approach****
- Now in the Second Approach we used Term Frequency Inverse Document Frequency one gram model
- Again we used maximum of 5000 features
- we used Scikit learn Tf-IDf vectorizer module for this

- Now when i run my classifier i got the following scores
  - Logistic Regression 0.87
  - Random FOrest 0.77

****Sample Code for Tf-IDF models****

```
#performing TF-IDF Vectorization on Training Data
corpustr = clean_train_reviews #corpusTraining

#Making TFidf Vectorizer object
vectorizertr = TfidfVectorizer(stop_words='english',
                               ngram_range = ( 1 , 1 ),analyzer="word",
                               max_df = .57 , binary=False , token_pattern=
                               sublinear_tf=False,max_features =5000)

#Fitting the object to Training & Testing Data
tfidftr=vectorizertr.fit_transform(corpustr).todense()
corpusts = clean_test_reviews
tfidfts=vectorizertr.transform(corpusts)
```

****Future Approaches****

- Term Frequency – Inverse document frequency(TF-IDF) ,Two gram model
- Vector Averaging (Unsupervised learning using Google's word2Vec algorithm)
- Bag of Centroids (Unsupervised Learning using Google's word2Vec algorithm and K-means clustering)
- Ensemble of Naïve Bayes and Logistic Regression

**** Work Division** (no of hours)**

Work Division	Theory	Data Exploration	Data Cleaning	Data Modeling
Archana	3	1	1	1
Dwipam	1	3	1	1
Krishna	1	1	1	2
Sanjana	1	1	3	1

**** References****

- Kaggle Discussion
- http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf
- Sentiment-Analysis-tutorial-AAAI-2011.pdf