NLP – Homework3 SUID: 387781563
Name: Chaithanya Chikkannaswamy NetID: cchikkan

Homework 3

1. Data Pre-processing

For this homework, I have considered the below dataset for training purpose.

http://help.sentiment140.com/for-students

I'm reading the traindata.csv file which contains the dataset I have taken. This dataset contains polarity of the tweets which is 0 for negative sentences, 2 for neutral sentences, 4 for positive sentences. Based on the values of the polarity I'm checking the condition and creating 3 lists as positive, negative and neutral sentences.

```
get_text = open("testdata.manual.2009.06.14.csv", 'r')
df = pd.read_csv('testdata.manual.2009.06.14.csv' , header = None, prefix="var")
```

```
pos_sents = []
neg_sents = []
neu_sents = []

for idx,val in enumerate(df.var0):
    if(val == 4):
        pos_sents.append(df.var5[idx])
    elif(val == 2):
        neu_sents.append(df.var5[idx])
    else:
        neg_sents.append(df.var5[idx])
```

```
print(len(pos_sents))
print(len(neg_sents))
print(len(neu_sents))
```

```
182
177
139
```

Code to define the positive, negative and neutral lists

```
In [52]: print(pos_sents)
```

```
['@stellargirl I loooooooovvvvvveee my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.',
'Reading my kindle2...  Love it... Lee childs is good read.', 'Ok, first assesment of the #kindle2 ...it fucking rock
s!!!', "@kenburbary You'll love your Kindle2. I've had mine for a few months and never looked back. The new big one i
s huge! No need for remorse! :)", "@mikefish  Fair enough. But i have the Kindle2 and I think it's perfect  :)", "@ri
chardebaker no. it is too big. I'm quite happy with the Kindle2.", 'Jquery is my new best friend.', 'Loves twitter',
'how can you not love Obama? he makes jokes about himself.', 'House Correspondents dinner was last night whoopi, barb
ara &amp;amp; sherri went, Obama got a standing ovation', 'Watchin Espn..Jus seen this new Nike Commerical with a Puppet
Lebron..sh*t was hilarious...LMAO!!!', "#lebron best athlete of our generation, if not all time (basketball related)
I don't want to get into inter-sport debates about  __1/2", 'i love lebron. http://bit.ly/PdHur', '@Pmillzz lebron I
S THE BOSS', "@sketchbug Lebron is a hometown hero to me, lol I love the Lakers but let's go Cavs, lol", 'lebron and
zydrunas are such an awesome duo', '@wordwhizkid Lebron is a beast... nobody in the NBA comes even close.', 'download
ing apps for my iphone! So much fun :-) There literally is an app for just about anything.', 'good news, just had a c
all from the Visa office, saying everything is fine.....what a relief! I am sick of scams out there! Stealing!', 'htt
p://twurl.nl/epkr4b - awesome come back from @biz (via @fredwilson)', 'In montreal for a long weekend of R&amp;amp;R. Muc
h needed.', 'Booz Allen Hamilton has a bad ass homegrown social collaboration platform. Way cool!  #ttiv', '[#MLUC09]
Customer Innovation Award Winner: Booz Allen Hamilton -- http://ping.fm/c2hPP', '@SoChi2 I current use the Nikon D90
and love it, but not as much as the Canon 40D/50D. I chose the D90 for the  video feature. My mistake.', "@phyreman9
Google is always a good place to look. Should've mentioned I worked on the Mustang w/ my Dad, @KimbleT.", 'RT @jessve
rr I love the nerdy Stanford human biology videos - makes me miss school. http://bit.ly/13t7NR', '@spinuzzi: Has been
```

screenshot shows the positive set of sentences

```
In [51]: print(neg_sents)
```

```
['Fuck this economy. I hate aig and their non loan given asses.', "@Karoli I firmly believe that Obama/Pelosi have ZE
RO desire to be civil.  It's a charade and a slogan, but they want to destroy conservatism", 'dear nike, stop with th
e flywire. that shit is a waste of science. and ugly. love, @vincentx24x', 'I was talking to this guy last night and
he was telling me that he is a die hard Spurs fan.  He also told me that he hates LeBron James.', "@ludajuice Lebron
is a Beast, but I'm still cheering 4 the A..til the end.", 'Played with an android google phone. The slide out screen
scares me I would break that fucker so fast. Still prefer my iPhone.', 'US planning to resume the military tribunals
at Guantanamo Bay... only this time those on trial will be AIG execs and Chrysler debt holders', 'omg so bored &amp;amp;
my tattoooos are so itchy!!  help! aha =)', "I'm itchy and miserable!", "@sekseemess no. I'm not itchy for now. Maybe
later, lol.", 'cant sleep... my tooth is aching.', 'Blah, blah, blah same old same old. No plans today, going back to
sleep I guess.', "glad i didnt do Bay to Breakers today, it's 1000 freaking degrees in San Francisco wtf", '?Obama Ad
ministration Must Stop Bonuses to AIG Ponzi Schemers ... http://bit.ly/2CUIg', 'started to think that Citi is in real
ly deep s&amp;amp;^t. Are they gonna survive the turmoil or are they gonna be the next AIG?', "ShaunWoo hate'n on AiG",
'annoying new trend on the internets:  people picking apart michael lewis and malcolm gladwell.  nobody wants to read
that.', 'omg. The commercials alone on ESPN are going to drive me nuts.', "@morind45 Because the twitter api is slow
and most client's aren't good.", 'yahoo answers can be a butt sometimes', 'RT @SmartChickPDX: Was just told that Nike
layoffs started today :-(', 'needs someone to explain lambda calculus to him! :(', 'Took the Graduate Field Exam for
Computer Science today.  Nothing makes you feel like more of an idiot than lambda calculus.', "@legalgeekery Yeahhhhh
hhhh, I wouldn't really have lived in East Palo Alto if I could have avoided it.  I guess it's only for the summer.",
'Damn you North Korea. http://bit.ly/KtMeQ', 'Can we just go ahead and blow North Korea off the map already?', "North
```

screenshot shows the negative set of sentences

```
In [79]: print(neu_sents)
```

```
["Check this video out -- President Obama at the White House Correspondents' Dinner http://bit.ly/IMXUM", 'need sugge
stions for a good IR filter for my canon 40D ... got some? pls DM', '@surfit: I just checked my google for my busines
s- blip shows up as the second entry! Huh. Is that a good or ba... ? http://blip.fm/~6emhv', 'is in San Francisco at
Bay to Breakers.', 'just landed at San Francisco', 'San Francisco today.  Any suggestions?', 'On my way to see Star T
rek @ The Esquire.', 'Going to see star trek soon with my dad.', 'Bill Simmons in conversation with Malcolm Gladwell
http://bit.ly/j9o50', 'playing with cURL and the Twitter API', 'playing with Java and the Twitter API', 'Nike owns NB
A Playoffs ads w/ LeBron, Kobe, Carmelo? http://ow.ly/7Uiy  #Adidas #Billups #Howard  #Marketing #Branding', "'Next t
ime, I'll call myself Nike'", "New blog post: Nike SB Dunk Low Premium 'White Gum' http://tr.im/lOtT", 'giving weka a
n app engine interface, using the bird strike data for the tests, the logo is a given.', 'Brand New Canon EOS 50D 15M
P DSLR Camera Canon 17-85mm IS Lens ...: Web Technology Thread, Brand New Canon EOS 5.. http://u.mavrev.com/5a3t', "N
VIDIA Names Stanford's Bill Dally Chief Scientist, VP Of Research http://bit.ly/Fvvg9", 'New blog post: Harvard Versu
s Stanford - Who Wins? http://bit.ly/MCoCo', 'jQuery UI 1.6 Book Review - http://cfbloggers.org/?c=30631', 'At GWT fi
reside chat @googleio', 'Hi there, does anyone have a great source for advice on viral marketing?... http://link.gs/Y
tZ8', "Here's A case study on how to use viral marketing to add over 10,000 people to your list http://snipr.com/i50o
z", 'going to see the new night at the museum  movie with my family oh boy a three year old in the movies fuin', 'Jus
t saw the new Night at the Museum movie...it was...okay...lol 7\\10', 'Going to see night at the museum 2 with tall b
oy', 'I saw Night at the Museum: Battle of the Swithsonian today. It was okay. Your typical [kids] Ben Stiller movi
e.', 'Taking Katie to see Night at the Museum.  (she picked it)', 'GM says expects announcment on sale of Hummer soon
- Reuters: WDSUGM says expects announcment on sale of Hummer .. http://bit.ly/4E1Fv', "Time Warner Cable Pulls the Pl
```

screenshot shows the neutral set of sentences.

I have created 3 lists of positive, negative and neutral sentences for further cleaning of the sentences before classifying. To clean the data even more, I have used Regular expression to remove the username, www, http etc.

```
In [80]: df1 = pd.DataFrame(columns=['positive'])
         df2 = pd.DataFrame(columns=['negative'])
         df3 = pd.DataFrame(columns=['neutral'])

In [81]: df1['positive'] = pos_sents
         df2['negative'] = neg_sents
         df3['neutral'] = neu_sents

In [117]: df1['positive'] = df1['positive'].replace(r'http\S+', '', regex=True).replace(r'www\S+', '', regex=True).replace('@[^\s
          df2['negative'] = df2['negative'].replace(r'http\S+', '', regex=True).replace(r'www\S+', '', regex=True).replace('@[^\s
          df3['neutral'] = df3['neutral'].replace(r'http\S+', '', regex=True).replace(r'www\S+', '', regex=True).replace('@[^\s]+

In [118]: print(df1['positive'])
          print(df2['negative'])
          print(df3['neutral'])
```

The list of positive, negative and neutral sentences are tokenized using the word_tokanize()

```
In [125]: from nltk.tokenize import sent_tokenize, word_tokenize
          tokenizedpos = df1.apply(lambda row: nltk.word_tokenize(row['positive']), axis=1)
          print(tokenizedpos[:20])
          tokenizedneg = df2.apply(lambda row: nltk.word_tokenize(row['negative']), axis=1)
          print(tokenizedneg[:20])
          tokenizedneu = df3.apply(lambda row: nltk.word_tokenize(row['neutral']), axis=1)
          print(tokenizedneu[:20])
```

The cleaning process includes converting each letter of sentences to **lowercase** letters, removing words that contain **numbers**, removing **stopwords,** removing **empty tokens**, **lemmatizing** each sentence by using **WordNetLemmatizer** according to their wordnet pos tags, and removing words with only **one** letter. In order to do sentiment analysis more efficiently, data cleaning helped us to get rid of unnecessary words and marks that took place in our sentence. Because these words wouldn't help us determine sentiments in our dataset and removing these will result in more accurate analysis.

```
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer

def clean_text(text):
    # is_alpha
    text = [w for w in text if w.isalpha()]
    # lower text
    text = [w.lower() for w in text]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # join all
    #text = " ".join(text)
    return(text)
```

Code to clean the data sets

```
In [127]: # clean text data for positive, negative and neutral sentences
          cleaned_possents = []
          cleaned_negsents = []
          cleaned_neusents = []
          for sent in tokenizedpos:
              sent = clean_text(sent)
              cleaned_possents.append(sent)
          for sent in tokenizedneg:
              sent = clean_text(sent)
              cleaned_negsents.append(sent)
          for sent in tokenizedneu:
              sent = clean_text(sent)
              cleaned_neusents.append(sent)
```

Here, we can see that, unnecessary stopwords such as ‘the’, ‘is’, ‘to’, ‘be’, ‘and’,’that’,a’,’than’,’or’ are removed. Furthermore, words that contain digits\ and letters consist of one or less letters(mostly stop signs including “”, ’.’, ’,’) are also eliminated from sentence structure. Finally some of the words are modified back into their nominal form by lemmatization. (destined -> destine, going -> go, greater -> great). Finally our sentence simplified into its version. The screen shot is attached below

```
[['looooooooovvvvvveee', 'dx', 'cool', 'fantastic', 'right'], ['reading', 'love', 'lee', 'childs', 'good', 'read'],
['ok', 'first', 'assesment', 'fucking', 'rocks'], ['love', 'mine', 'months', 'never', 'looked', 'back', 'new', 'big',
'one', 'huge', 'need', 'remorse'], ['fair', 'enough', 'think', 'perfect'], ['big', 'quite', 'happy'], ['jquery', 'ne
w', 'best', 'friend'], ['loves', 'twitter'], ['love', 'obama', 'makes', 'jokes'], ['house', 'correspondents', 'dinne
r', 'last', 'night', 'whoopi', 'barbara', 'amp', 'sherri', 'went', 'obama', 'got', 'standing', 'ovation'], ['watchi
n', 'espn', 'jus', 'seen', 'new', 'nike', 'commerical', 'puppet', 'lebron', 'sh', 'hilarious', 'lmao'], ['lebron', 'b
est', 'athlete', 'generation', 'time', 'basketball', 'related', 'want', 'get', 'debates'], ['love', 'lebron'], ['lebr
on', 'boss'], ['lebron', 'hometown', 'hero', 'lol', 'love', 'lakers', 'let', 'go', 'cavs', 'lol'], ['lebron', 'zydrun
as', 'awesome', 'duo'], ['lebron', 'beast', 'nobody', 'nba', 'comes', 'even', 'close'], ['downloading', 'apps', 'ipho
ne', 'much', 'fun', 'literally', 'app', 'anything'], ['good', 'news', 'call', 'visa', 'office', 'saying', 'everythin
g', 'fine', 'relief', 'sick', 'scams', 'stealing'], ['awesome', 'come', 'back', 'via'], ['montreal', 'long', 'weeken
d', 'amp', 'much', 'needed'], ['booz', 'allen', 'hamilton', 'bad', 'ass', 'homegrown', 'social', 'collaboration', 'pl
atform', 'way', 'cool', 'ttiv'], ['customer', 'innovation', 'award', 'winner', 'booz', 'allen', 'hamilton'], ['curren
t', 'use', 'nikon', 'love', 'much', 'canon', 'chose', 'video', 'feature', 'mistake'], ['google', 'always', 'good', 'p
lace', 'look', 'mentioned', 'worked', 'mustang', 'dad'], ['rt', 'love', 'nerdy', 'stanford', 'human', 'biology', 'vid
eos', 'makes', 'miss', 'school'], ['bit', 'crazy', 'steep', 'learning', 'curve', 'lyx', 'really', 'good', 'long', 'do
cs', 'anything', 'shorter', 'would', 'insane'], ['listening', 'danny', 'gokey', 'lt', 'lt', 'lt', 'aww', 'amazing',
'lt', 'much'], ['going', 'sleep', 'bike', 'ride'], ['regret', 'going', 'see', 'star', 'trek', 'awesome'], ['highly',
'recommend', 'malcolm', 'gladwell'], ['blink', 'malcolm', 'gladwell', 'amazing', 'book', 'tipping', 'point'], ['malco
lm', 'gladwell', 'might', 'new', 'man', 'crush'], ['playing', 'twitter', 'api', 'sounds', 'fun', 'may', 'need', 'tak
e', 'class', 'find', 'new', 'friend', 'like', 'generate', 'results', 'api', 'code'], ['hello', 'twitter', 'api'], ['s
crapbooking', 'nic'], ['rt', 'five', 'things', 'wolfram', 'alpha', 'better', 'vastly', 'different', 'google'], ['chan
ged', 'default', 'pic', 'nike', 'basketball', 'cause', 'bball', 'awesome'], ['back', 'worked', 'nike', 'one', 'fav',
'word'], ['way', 'totally', 'inspired', 'freaky', 'nike', 'commercial'], ['class', 'supposed', 'come', 'today'], ['sh
out', 'outs', 'east', 'palo', 'alto', 'buildin', 'karizmakaze', 'gta', 'also', 'thanks', 'profits', 'doom', 'universa
l', 'hempz', 'cracka'], ['great', 'stanford', 'course', 'thanks', 'making', 'available', 'public', 'really', 'helpfu
l', 'informative', 'starting'], ['work', 'til', 'lets', 'go', 'lakers'], ['omgg', 'ohhdee', 'want', 'mcdonalds', 'dam
n', 'wonder', 'open', 'lol'], ['really', 'liked', 'learning', 'jquery', 'book', 'worth', 'look'], ['interesting', 'a
d', 'adobe', 'goodby', 'silverstein', 'amp', 'partners', 'youtube', 'adobe', 'le', 'sens', 'propre'], ['goodby', 'sil
verstein', 'agency', 'new', 'site', 'great'], ['rt', 'goodby', 'silverstein', 'new', 'site', 'enjoy', 'nice', 'fin
d'], ['ever', 'amazing', 'psyop', 'goodby', 'silverstein', 'amp', 'partners', 'hp', 'go', 'play', 'effects'], ['top',
'ten', 'watched', 'chart', 'love', 'nike', 'mostvaluablepuppets', 'campaign', 'wieden', 'amp', 'kennedy'], ['zomg'],
['ok', 'lots', 'buzz', 'lucky', 'free'], ['got', 'free', 'android', 'google'], ['guess', 'retiring', 'start', 'usin
g', 'developer', 'woot', 'googleio'], ['happy', 'philip', 'googleio', 'today'], ['lakers', 'played', 'great', 'wait',
'thursday', 'night', 'lakers', 'vs'], ['judd', 'apatow', 'creates', 'fake', 'sitcom', 'market', 'new', 'movie', 'vira
l', 'marketing', 'best'], ['watching', 'night', 'museum', 'lmao'], ['loved', 'night', 'museum'], ['got', 'back', 'mov
```

2. Classification Tasks on Selected Features

For processing classification tasks, initially we created tuples for each cleaned sentences. The first element of tuple is list of cleaned words in the sentence and the second element is the label for polarity. **'positive'** for positive sentences and **'negative'** for negative sentences and **'neutral'** for neutral sentences.

Then, we merge lists of positive, negative and neutral documents into a single document which contains all of the positive, negative and neutral sentences with their polarity values. By that way, we categorized and prepared our **bag of words** for usage in classification tasks.

In [ ]:
```
#Classification
```

In [131]:
```
pos_documents = [(sent, 'pos') for sent in cleaned_possents]
print(pos_documents[0])
print(len(pos_documents))
```

```
(['looooooooovvvvvveee', 'dx', 'cool', 'fantastic', 'right'], 'pos')
182
```

In [132]:
```
neg_documents = [(sent, 'neg') for sent in cleaned_negsents]
print(neg_documents[0])
print(len(neg_documents))
```

```
(['fuck', 'economy', 'hate', 'aig', 'non', 'loan', 'give', 'ass'], 'neg')
177
```

In [133]:
```
neu_documents = [(sent, 'neu') for sent in cleaned_neusents]
print(neu_documents[0])
print(len(neu_documents))
```

```
(['check', 'video', 'president', 'obama', 'white', 'house', 'correspondent', 'dinner'], 'neu')
139
```

In [143]:
```
all_documents = pos_documents + neg_documents + neu_documents
print(all_documents[0])
len(all_documents)
```

```
(['looooooooovvvvvveee', 'dx', 'cool', 'fantastic', 'right'], 'pos')
```

Out[143]:
```
498
```

Generate the bag of words for the classifier

After that, we shuffle the positive and negative sentences in the list by using random library of python.

In [144]:
```
import random
random.shuffle(all_documents)
print(all_documents[0])
```

```
(['time', 'warner', 'devil', 'bad', 'possible', 'time', 'internet', 'go'], 'neg')
```

Next, we implemented a frequency distribution of all words in **all_documents** by using **FreqDist()** method of nltk library. Furthermore, we extracted top 2000 most frequently appearing words from all words. At the end, all words became ready to be used as final version of bag of words.

```
In [145]: from nltk import FreqDist
          all_words_list = [word for (sent,cat) in all_documents for word in sent]
          all_words = FreqDist(all_words_list)
          #print(len(all_words))
          word_items = all_words.most_common(2000)
          #word_items = all_words
          word_features = [word for (word,count) in word_items]
          word_items[:10]
          #word_features[0]
```

```
Out[145]: [('time', 48),
           ('get', 46),
           ('go', 42),
           ('love', 38),
           ('new', 36),
           ('night', 35),
           ('warner', 34),
           ('good', 34),
           ('amp', 33),
           ('see', 31)]
```

code for implementing frequency distribution of all of words in the nltk corpus with Top 10 most frequent words from all of words in the nltk corpus

After finalizing bag of words, we created a method called **document_features.** In that method, we created a tuple for each review sentence. The first element of tuple is a dictionary in which keys are each of the 2000 most frequent words from bag of words and value for each key is either **True** if word appears in the review sentence or **False** if it doesn't. The second element is the label for polarity. **'pos'** for positive sentences and **'neg'** for negative sentences and **'neu'** for neutral sentences.

```
In [148]: def document_features(document, word_features):
              document_words = set(document)
              features = {}
              for word in word_features:
                  features['contains({})'.format(word)] = (word in document_words)
              return features
```

code for document_features method which inputs a sentence and frequent words.

```
In [154]: featuresets = [(document_features(d, word_features), c) for (d, c) in all_documents]
          featuresets[0]
          print(len(featuresets))
```

```
498
```

code to extract **featuresets** from document_features method.

After retrieving featuresets, we split them into **train_set** and **test_set**. We used approximately **80%** of it for training set and remaining **20%** for the testing set from our training data set. Then, we implement our classifier by using **nltk.NaiveBayesClassier** function from nltk module to train our training set. We calculated the accuracy of created classifier by using **nltk.classify.accuracy** method. Finally, we displayed 50 most informative features of the classifier by using **classifier.show_most_informative_features(50)** method.

```
In [189]: import nltk
          train_set, test_set = featuresets[:350], featuresets[351:]
          classifier = nltk.NaiveBayesClassifier.train(train_set)
          print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, test_set))*100)
          classifier.show_most_informative_features(50)
```

code to implement nltk naïve bayes classifier from the previously extracted featureset.

```
Classifier accuracy percent: 63.94557823129252
Most Informative Features
        contains(jquery) = True              neu : neg    =     11.5 : 1.0
           contains(eat) = True              neu : pos    =      8.7 : 1.0
          contains(good) = True              pos : neu    =      8.0 : 1.0
         contains(great) = True              pos : neu    =      7.0 : 1.0
          contains(hate) = True              neg : pos    =      6.3 : 1.0
          contains(love) = True              pos : neg    =      5.9 : 1.0
        contains(warner) = True              neg : neu    =      5.6 : 1.0
         contains(still) = True              neg : pos    =      5.6 : 1.0
          contains(nike) = True              neu : neg    =      5.3 : 1.0
         contains(watch) = True              neu : neg    =      5.1 : 1.0
          contains(time) = True              neg : pos    =      5.0 : 1.0
       contains(safeway) = True              neu : neg    =      4.7 : 1.0
       contains(awesome) = True              pos : neg    =      4.4 : 1.0
           contains(one) = True              pos : neg    =      4.4 : 1.0
         contains(could) = True              neg : pos    =      4.3 : 1.0
       contains(dentist) = True              neg : pos    =      4.3 : 1.0
       contains(twitter) = True              neg : pos    =      4.3 : 1.0
          contains(play) = True              neu : neg    =      4.1 : 1.0
           contains(amp) = True              neg : neu    =      3.9 : 1.0
          contains(look) = True              pos : neg    =      3.7 : 1.0
          contains(fail) = True              neg : pos    =      3.6 : 1.0
        contains(iphone) = True              neg : pos    =      3.6 : 1.0
          contains(best) = True              pos : neg    =      3.4 : 1.0
        contains(lebron) = True              pos : neu    =      3.4 : 1.0
          contains(high) = True              neu : neg    =      3.2 : 1.0
    contains(marketing) = True              neu : neg    =      3.2 : 1.0
        contains(movie) = True              neu : neg    =      3.2 : 1.0
```

Besides classifier accuracy, we also observed other useful metrics by using **nltk.metrics.scores** module such as **precision, recall** and **f-measure**. These metrics provide much greater insight into the performance characteristics of Naïve Bayes classifier. Precision measures the exactness of a classifier. A higher precision means less false positives whereas a lower precision means more false positives. On the other hand, recall measures the sensitivity of a classifier. A higher recall means less false negatives, while a lower recall means more false negatives. Finally, f-measure is found by calculated weighted harmonic mean of precision and recall. In this assignment, we collected reference and observed values for each label(pos,neg,neu) and store them in different sets to calculate precision, recall and f-measures separately of the Naïve Bayes classifier.

```
import collections
import nltk.metrics
from nltk.metrics.scores import precision
from nltk.metrics.scores import recall
from nltk.metrics.scores import f_measure
refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (feats, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)

print('pos precision:', precision(refsets['pos'], testsets['pos']))
print('pos recall:', recall(refsets['pos'], testsets['pos']))
print('pos F-measure:', f_measure(refsets['pos'], testsets['pos']))

print('neg precision:', precision(refsets['neg'], testsets['neg']))
print('neg recall:', recall(refsets['neg'], testsets['neg']))
print('neg F-measure:', f_measure(refsets['neg'], testsets['neg']))

print('neu precision:', precision(refsets['neu'], testsets['neu']))
print('neu recall:', recall(refsets['neu'], testsets['neu']))
print('neu F-measure:', f_measure(refsets['neu'], testsets['neu']))
```

```
pos precision: 0.6440677966101694
pos recall: 0.7169811320754716
pos F-measure: 0.6785714285714285
neg precision: 0.6744186046511628
neg recall: 0.6041666666666666
neg F-measure: 0.6373626373626373
neu precision: 0.6
neu recall: 0.5869565217391305
neu F-measure: 0.5934065934065934
```

• **Subjectivity Features**

After analyzing unigram features, we also did experiment on subjectivity features. We started by reading subjectivity file which contains various words with their corresponding **types, pos tags**, **stem statuses** and **polarities**.

In [176]:
```
##CREATING SUBJECTIVITY CLASSIFIER
SLpath = 'subjclueslen1-HLTEMNLP05.tff'
def readSubjectivity(path):
    flexicon = open(path, 'r')
    # initialize an empty dictionary
    sldict = { }
    for line in flexicon:
        fields = line.split()   # default is to split on whitespace
        # split each field on the '=' and keep the second part as the value
        strength = fields[0].split("=")[1]
        word = fields[2].split("=")[1]
        posTag = fields[3].split("=")[1]
        stemmed = fields[4].split("=")[1]
        polarity = fields[5].split("=")[1]
        if (stemmed == 'y'):
            isStemmed = True
        else:
            isStemmed = False
        # put a dictionary entry with the word as the keyword
        #     and a list of the other values
        sldict[word] = [strength, posTag, isStemmed, polarity]
    return sldict
SL = readSubjectivity(SLpath)
```

Then, we extract **sl_featureset** by counting variables which mutually exist in both **all_documents** and **SL** file for the four classes of subjectivity which are **weak positive**, **strong positive**, **weak negative** and **strong negative**.

In [208]:
```
def SL_features(document, word_features, SL):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1

            features['positivecount'] = weakPos + (2 * strongPos)
            features['negativecount'] = weakNeg + (2 * strongNeg)

    return features
```

SL_featureset contains a list of tuples, similar to unigram features, which consists of set of high frequency words and polarity labels. Similarly, we split sl_featuresets into train and test sets by 80-20 ratio, and created Naïve Bayes classifier from the training set. Finally, we calculated the accuracy, precision, recall and f-measures of created classifier.

In [190]:
```
sl_train_set, sl_test_set = SL_featuresets[:350], SL_featuresets[351:]
sl_classifier = nltk.NaiveBayesClassifier.train(sl_train_set)
print("Classifier accuracy percent:",(nltk.classify.accuracy(sl_classifier, sl_test_set))*100)
sl_classifier.show_most_informative_features(50)

sl_refsets = collections.defaultdict(set)
sl_testsets = collections.defaultdict(set)

for i, (feats, label) in enumerate(sl_test_set):
    sl_refsets[label].add(i)
    observed = sl_classifier.classify(feats)
    sl_testsets[observed].add(i)

print('pos precision:', precision(sl_refsets['pos'], sl_testsets['pos']))
print('pos recall:', recall(sl_refsets['pos'], sl_testsets['pos']))
print('pos F-measure:', f_measure(sl_refsets['pos'], sl_testsets['pos']))

print('neg precision:', precision(refsets['neg'], sl_testsets['neg']))
print('neg recall:', recall(sl_refsets['neg'], sl_testsets['neg']))
print('neg F-measure:', f_measure(sl_refsets['neg'], sl_testsets['neg']))

print('neu precision:', precision(sl_refsets['neu'], sl_testsets['neu']))
print('neu recall:', recall(sl_refsets['neu'], sl_testsets['neu']))
print('neu F-measure:', f_measure(sl_refsets['neu'], sl_testsets['neu']))
```

```
Classifier accuracy percent: 68.70748299319727
Most Informative Features
        contains(jquery) = True              neu : neg    =     11.5 : 1.0
           contains(eat) = True              neu : pos    =      8.7 : 1.0
          contains(good) = True              pos : neu    =      8.0 : 1.0
           positivecount = 3                 pos : neu    =      7.6 : 1.0
          contains(great) = True             pos : neu    =      7.0 : 1.0
           negativecount = None              neu : neg    =      6.3 : 1.0
           positivecount = None              neu : neg    =      6.3 : 1.0
           negativecount = 2                 neg : neu    =      6.3 : 1.0
          contains(hate) = True              neg : pos    =      6.3 : 1.0
          contains(love) = True              pos : neg    =      5.9 : 1.0
        contains(warner) = True              neg : neu    =      5.6 : 1.0
         contains(still) = True              neg : pos    =      5.6 : 1.0
          contains(nike) = True              neu : neg    =      5.3 : 1.0
           positivecount = 4                 pos : neu    =      5.2 : 1.0
         contains(watch) = True              neu : neg    =      5.1 : 1.0
          contains(time) = True              neg : pos    =      5.0 : 1.0
       contains(safeway) = True              neu : neg    =      4.7 : 1.0
       contains(awesome) = True              pos : neg    =      4.4 : 1.0
           contains(one) = True              pos : neg    =      4.4 : 1.0
         contains(could) = True              neg : pos    =      4.3 : 1.0
       contains(dentist) = True              neg : pos    =      4.3 : 1.0
       contains(twitter) = True              neg : pos    =      4.3 : 1.0
          contains(play) = True              neu : neg    =      4.1 : 1.0
           positivecount = 0                 neg : pos    =      4.0 : 1.0
           contains(amp) = True              neg : neu    =      3.9 : 1.0
          contains(look) = True              pos : neg    =      3.7 : 1.0
           positivecount = 5                 pos : neu    =      3.7 : 1.0
          contains(fail) = True              neg : pos    =      3.6 : 1.0
```

```
          contains(cable) = True                neg : ne
pos precision: 0.6515151515151515
pos recall: 0.8113207547169812
pos F-measure: 0.7226890756302521
neg precision: 0.6923076923076923
neg recall: 0.75
neg F-measure: 0.72
neu precision: 0.7586206896551724
neu recall: 0.4782608695652174
neu F-measure: 0.5866666666666667
```

• **Negation Features**

Determining negative words are essential to conduct sentiment analysis. In order to handle negation words in sentences, we extracted negation features from our corpus.

```
In [191]: negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing', 'noone', 'rather', 'hardly', 'scarcely', 'rarely',

In [192]: def NOT_features(document, word_features, negationwords):
              features = {}
              for word in word_features:
                  features['contains({})'.format(word)] = False
                  features['contains(NOT{})'.format(word)] = False
              # go through document words in order
              for i in range(0, len(document)):
                  word = document[i]
                  if ((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
                      i += 1
                      features['contains(NOT{})'.format(document[i])] = (document[i] in word_features)
                  else:
                      features['contains({})'.format(word)] = (word in word_features)
              return features
```

code for NOT_features method which inputs a sentence, frequent words and negation words.

Similar to unigram and SL_features, we use Naïve Bayes classifier to train not_featuresets by splitting train and test sets by 80-20. The implementation process is similar to previous features.

In [196]:
```
not_train_set, not_test_set = NOT_featuresets[:350], NOT_featuresets[351:]
not_classifier = nltk.NaiveBayesClassifier.train(not_train_set)
print("Classifier accuracy percent:",(nltk.classify.accuracy(not_classifier, not_test_set))*100)
not_classifier.show_most_informative_features(50)

not_refsets = collections.defaultdict(set)
not_testsets = collections.defaultdict(set)

for i, (feats, label) in enumerate(not_test_set):
    not_refsets[label].add(i)
    observed = not_classifier.classify(feats)
    not_testsets[observed].add(i)

print('pos precision:', precision(not_refsets['pos'], not_testsets['pos']))
print('pos recall:', recall(not_refsets['pos'], not_testsets['pos']))
print('pos F-measure:', f_measure(not_refsets['pos'], not_testsets['pos']))

print('neg precision:', precision(not_refsets['neg'], not_testsets['neg']))
print('neg recall:', recall(not_refsets['neg'], not_testsets['neg']))
print('neg F-measure:', f_measure(not_refsets['neg'], not_testsets['neg']))

print('neu precision:', precision(not_refsets['neu'], not_testsets['neu']))
print('neu recall:', recall(not_refsets['neu'], not_testsets['neu']))
print('neu F-measure:', f_measure(not_refsets['neu'], not_testsets['neu']))
```

```
Classifier accuracy percent: 63.94557823129252
Most Informative Features
        contains(jquery) = True              neu : neg    =     11.5 : 1.0
           contains(eat) = True              neu : pos    =      8.7 : 1.0
          contains(good) = True              pos : neu    =      8.0 : 1.0
         contains(great) = True              pos : neu    =      7.0 : 1.0
          contains(hate) = True              neg : pos    =      6.3 : 1.0
          contains(love) = True              pos : neg    =      5.9 : 1.0
        contains(warner) = True              neg : neu    =      5.6 : 1.0
         contains(still) = True              neg : pos    =      5.6 : 1.0
          contains(nike) = True              neu : neg    =      5.3 : 1.0
         contains(watch) = True              neu : neg    =      5.1 : 1.0
          contains(time) = True              neg : pos    =      5.0 : 1.0
       contains(safeway) = True              neu : neg    =      4.7 : 1.0
       contains(awesome) = True              pos : neg    =      4.4 : 1.0
           contains(one) = True              pos : neg    =      4.4 : 1.0
         contains(could) = True              neg : pos    =      4.3 : 1.0
       contains(dentist) = True              neg : pos    =      4.3 : 1.0
       contains(twitter) = True              neg : pos    =      4.3 : 1.0
          contains(play) = True              neu : neg    =      4.1 : 1.0
           contains(amp) = True              neg : neu    =      3.9 : 1.0
          contains(look) = True              pos : neg    =      3.7 : 1.0
          contains(fail) = True              neg : pos    =      3.6 : 1.0
        contains(iphone) = True              neg : pos    =      3.6 : 1.0
          contains(best) = True              pos : neg    =      3.4 : 1.0
        contains(lebron) = True              pos : neu    =      3.4 : 1.0
```

```
        contains(people) = True                 neg
pos precision: 0.6440677966101694
pos recall: 0.7169811320754716
pos F-measure: 0.6785714285714285
neg precision: 0.6744186046511628
neg recall: 0.6041666666666666
neg F-measure: 0.6373626373626373
neu precision: 0.6
neu recall: 0.5869565217391305
neu F-measure: 0.5934065934065934
```

NLP – Homework3 SUID: 387781563
Name: Chaithanya Chikkannaswamy NetID: cchikkan

COVID19.csv
Data Pre-Processing:
First step in Data Pre-processing is **data extraction.** I have used the data extracted into "covid19.csv" in my previous assignment. Here we are analyzing the "text" column in the dataset to conduct sentiment analysis to gain the understanding of the sentiments reflected om the texts.

First, I have used the python data frames to read the csv files as dataframes using .read_csv()

```
: get_text = open("covid19.csv", 'r')
  df = pd.read_csv("covid19.csv")
  columns = df.text
  print(len(columns))
```

```
106017
```

Python code to read the data from csv file

After loading the dataset I tokenized the text by sentences and have appended into the list **tweetsentences()**. In this task I have to perform sentimental analysis on the retrieved dataset and differentiate the positive, negative and neutral sentences.

```
texttweets = []
tweetsentence = []
for line in columns:
    texttweets.append(line)
print(texttweets[:5])

for tweet in texttweets:
    tweet = nltk.sent_tokenize(tweet)
    for sent in tweet:
        tweetsentence.append(sent)
print(tweetsentence[:20])
```

Python code to tokenize the text column in covid19.csv file

```
['Bengaluru: Isolation wards in hospitals across Karnataka and helpline to take calls on coronavirus-related queries
are ready to prevent any further spread of the virus after the first case in India was reported from Kerala yesterda
y.', 'The Chief Secretary of the state government on Thursday held a meeting with the Additional Chief Secretary (Hea
lth), Health Commissioner, Mission Director of the National Health Mission and other health department officials and
reviewed the state preparedness to tackle any cases of coronavirus whenever reported.', 'Rajiv Gandhi Institute of Ch
est Diseases (RGICD) with 15 beds and Wenlock Hospital at Mangaluru with 10 beds have been selected for the treatment
of the virus.', 'All district hospitals will have five beds isolated for patients carrying the virus.', 'Along with t
his, at least ten private hospitals in Bengaluru will be setting up similar isolation wards.', 'The National Institut
e of Virology and the Viral Research and Diagnostic Laboratory at the Bangalore Medical College & Research Institute
have been roped in for the testing of the virus for the next few days.', '104 Arogya Sahayavani helpline run by the H
ealth Department will take all calls related to the virus.', 'Udayavani English', 'The government making sure that th
e new coronavirus does not make its way to the country.', 'Aside from monitoring suspected cases, authorities are als
o preparing plans to repatriate Filipinos in the Chinese province of Hubei.', "Hubei's capital city is Wuhan, ground
zero of the new coronavirus.", "Apart from more people falling sick (as bad as that is), is there a more fundamental
concern that if it runs wild in a less developed country, it'll mutate into something more dangerous?", 'As infection
s climb, the coronavirus (based on less stable RNA) gets more chances to mutate into something even more virulent or
deadly?', 'Asian stock markets are mostly higher on Friday following the positive cues overnight from Wall Street aft
er the World Health Organization declared the coronavirus outbreak a global health emergency, but did not recommend m
easures restricting international trade or travel.', 'The international agency noted that China was doing everything
it could to contain the outbreak.', "Investors also digested China's official manufacturing data that met expectation
s.", 'The Australian market is rising following the positive cues overnight from Wall Street.', 'The benchmark S&P/AS
X 200 Index is adding 27.80 points or 0.40 percent to 7,036.20, after touching a high of 7,042.50 earlier.', 'The bro
ader All Ordinaries Index is up 28.80 points or 0.41 percent to 7,137.40.', 'Australian shares closed lower on Thursd
ay.']
```

NLP – Homework3 SUID: 387781563
Name: Chaithanya Chikkannaswamy NetID: cchikkan

Sentence tokenized texts

reviewcorpus[] is the list that contains each tweets as a sentence and each of those sentences are work tokenized using python's nltk.word_tokenize() and then we call the clean_text() function to process and clean the data. The screenshot of the clean_text() function is attached below:

```
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer

def clean_text(text):
    # is_alpha
    text = [w for w in text if w.isalpha()]
    # lower text
    text = [w.lower() for w in text]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # join all
    #text = " ".join(text)
    return(text)
```

In this we function, we are removing the alpha numeric characters using the isalpha(), convert all the characters to lower case letters using lower(), we remove all the digits in the text using stopwords(), we remove the stop words and the empty tokens. We also generate the pos_tags for the text and user WordNetLemmatizer() and remove the words with only 1 letter. The purpose of defining the clean_text() is to process the data so that the analysis will be more accurate when we have the proper texts.

To analyze the sentiments in the text we use the training data set defined previously for the prediction purpose. Here, for each tweets we predict the sentiment based on our training data set and calculate the probability. The screenshot is attached below:

```
# TESTING CLASSIFIER
texttweets = texttweets[:1000]
cleaned_input = []
for tweet in texttweets:
    tweet = nltk.word_tokenize(tweet)
    tweet = clean_text(tweet)
    cleaned_input.append(tweet)

print("Cleaned input: ", cleaned_input[0])
test_featuresets = [document_features(d, word_features) for d in cleaned_input]
print(cleaned_input)

#classifier percentage
#print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, test_featuresets[0]))*100)
#Predictions
print("\nPredictions:")
for i in range(0,len(texttweets)):
    print("\nTweet:", texttweets[i])
    probdist = classifier.prob_classify(test_featuresets[i])
    pred_sentiment = probdist.max()
    print("Predicted sentiment:", pred_sentiment)
    print("Probability:", round(probdist.prob(pred_sentiment), 2))
```

```
Tweet: Related stories:
WHO’s global emergency declaration: What does it mean for Malaysia?
Expert: M’sia shows progress in handling virus outbreak
Predicted sentiment: pos
Probability: 0.72
```

Output of the sentiment analysis – positive

```
Tweet: The outbreak of the novel and deadly Coronavirus in China has become a serious danger for Pakistan. The fatal
virus also seems to have spread to other countries like the US, France and the UK. However, given the substantial Chi
nese immigrant population in Pakistan and the frequent travel between China and Pakistan, the danger of the virus spr
eading to our shores looms large.
It is good to know that the government has taken swift and commendable steps by installing virus detection scanners a
t airports. In addition, the media has a crucial role to play in spreading awareness of this virus and its preventive
measures.
Ammar Aslam Muhammadi
Sukkur
The outbreak of the novel and deadly Coronavirus in China has become a serious danger for Pakistan. The fatal virus a
lso seems to have spread to other countries like the US, France and the UK. However, given the substantial Chinese im
migrant population in Pakistan and the frequent travel between China and Pakistan, the danger of the virus spreading
to our shores looms large.
It is good to know that the government has taken swift and commendable steps by installing virus detection scanners a
t airports. In addition, the media has a crucial role to play in spreading awareness of this virus and its preventive
measures.
Ammar Aslam Muhammadi
Predicted sentiment: neg
```

Output of sentiment analysis – negative

```
One shopper who lived in Beijing during the SARS epidemic hopes the current outbreak will be similarly short with lit
tle long-lasting effects.
?"2003 was really devastating, but we recovered," said the man, 36, who gave only his last name, Tang.?
The Chinese government will have to inspire confidence to get people to spend again, analysts say. To that end, China
has done a better job than during SARS, they say.
"The crisis response of the Chinese government often follows a predictable pattern from delay and denial initially, t
o full mobilization, overreaction and then correction to a more sustainable course," wrote Gavekal Dragonomics analys
ts Andrew Batson and Ernan Cui in a research note. "The delay-and-denial phase of the response to this coronavirus ou
tbreak has clearly been much shorter than it was during the SARS outbreak of 2003, when officials hid the spread of t
he virus for months."
(Pierson reported from Singapore and Chang from Beijing.)
(c)2020 Los Angeles Times
Visit the Los Angeles Times at www.latimes.com
GRAPHIC (for help with images, contact 312-222-4194): 20200130 China virus
Share Comment
Predicted sentiment: neu
Probability: 1.0
```

Output of sentiment analysis – neutral

To calculate the top 50 adjective phrases, adverb phrases, and verb phrases we define two function where we define the grammar, word tokenize the sentences and pass it to the RegexParser(). These functions returns the adjective, adverb and verb phrases.

```
#calculate the adjective and verb ohrases
def cal_Adj_Phrases(sentence):
    adjective_phrases = []
    grammar = "ADJP: {<RB. ?>+<JJ.?>}"
    words = nltk.word_tokenize(sentence)
    parser = nltk.RegexpParser(grammar)
    t = parser.parse(nltk.pos_tag(words))
    flag = False
    for s in t.subtrees():
        if s.label() == "ADJP":
            adjective_phrases.append(s)
    return adjective_phrases

def cal_Verb_Pharses(sentence):
    verb_phrases = []
    grammar = "VP: {<VB.><DT><NN.>}"
    words = nltk.word_tokenize(sentence)
    parser = nltk.RegexpParser(grammar)
    t = parser.parse(nltk.pos_tag(words))
    flag = False
    for s in t.subtrees():
        if s.label() == "VP":
            verb_phrases.append(s)
    return verb_phrases
```

Python code to calculate the top 50 adjective, adverb and verb phrases

To create a new CSV file with positive negative and neutral sentences count:

```
In [ ]: #get the classified file for positive, negative and neutral sentiments in data
list_of_sentences = []
neg_adjective_phrases = []
pos_adjective_phrases = []
neu_adjective_phrases = []
neg_verb_phrases = []
pos_verb_phrases = []
neu_verb_phrases = []

result_outputfile = open("covid_summary.csv", "w", encoding="utf8")
summary_writer = csv.writer(result_outputfile)
with open("covid19.csv", "r") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    count = 0
    for row in csv_reader:
        pos_count = 0
        neg_count = 0
        neu_count = 0
        if row:
            if count == 0:
                count+=1
                header = ["title", "author", "country", "number of pos sentences", "number of neg sentences", "number c
                summary_writer.writerow(header)
            else:
                tokens = nltk.sent_tokenize(row[7])
                for sent in tokens:
                    p = document_features(sent.lower(), word_features)
                    res = classifier.classify(p)
                    if(res == "0"):
                        neg_count = neg_count+1
                        neg_adjective_phrases.append(call_Adj_Phrases(sent))
                        neg_verb_phrases.append(call_Verb_Phrases(sent))
                    if(res == "4"):
                        pos_count = pos_count+1
                        pos_adjective_phrases.append(call_Adj_Phrases(sent))
                        pos_verb_phrases.append(call_Verb_Phrases(sent))
```

title	author	country	the number of positive sentences	the number of negative sentences
Karnataka: Helplines, isolation wards set up for coronaviru	Udayavani	IN	2	1
Health dept. monitoring 24 people for possible infection		US	2	1
	jmccorm	US	1	1
Asian Markets Mostly Higher	rttnews.com	US	9	0
Tesla soars as bearish analysts left with little to highlight - E	Joe Easton	CA	8	1
Last flights from Beijing & Shanghai - Video - CityNews Montreal		CA	10	6
	Accujack	US	2	8
CSL postponed due to coronavirus	Loop Pacific	DE	4	1
Global markets hit hard as indices catch Wuhan flu; Sensex	Jash Kriplani	IN	14	1
DOH-Cordillera monitors man from China amid novel coronavirus scare		US	6	0

CSV file

Result interpretation:
At the end, when we compare three featuresets, we can see that Naïve Bayes classifier which trains **not_featureset** has the most accuracy with **63.94** percent. Classifier with sl_featureset has the second most accuracy with **68.70** percent and classifier which uses regular unigram featureset is the third with **63.94** percent accuracy.