Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

1. **Analysis of Wikipedia discussion forum (30%)**

First, we import our nltk package, corpus reader and read the Wikipedia text file which we need to analyze.

```
import nltk
```

```
from nltk.corpus import PlaintextCorpusReader
```

```
mycorpus = PlaintextCorpusReader('.', '.*\.txt')
```

```
mytext= mycorpus.raw('wiki.txt')
```

The next step used here is removing html tags that are present in our text file, because they are tags and not texts. So, we use Beautiful soup library in Python to get rid of those html markups.

```
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(mytext, 'lxml')
```

```
braw = soup.get_text()
```

Now, that we have only texts we tokenize our text.

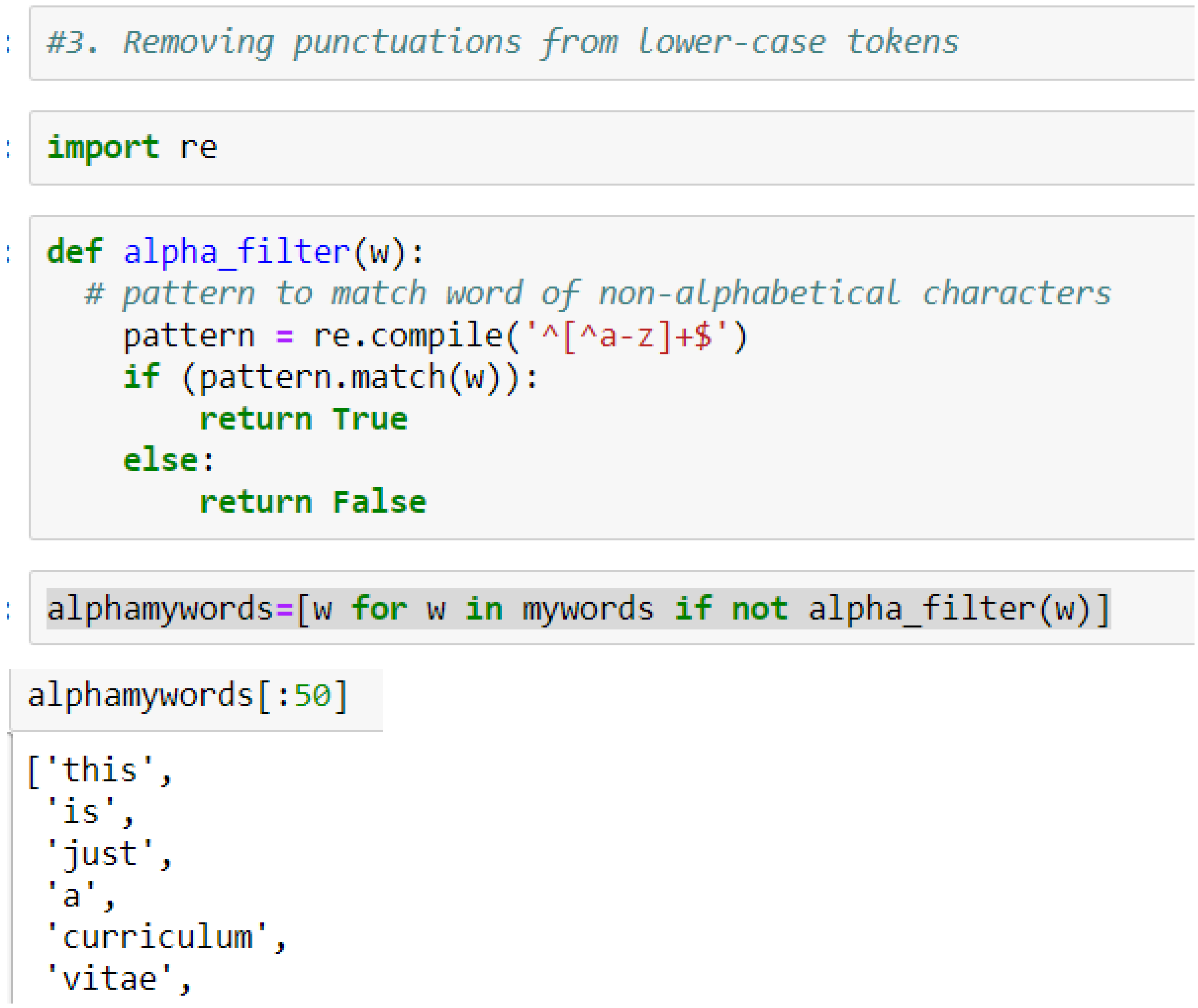
```
mytokens = nltk.word_tokenize(braw)
```

Here we convert all text to lower alphabets because we do not words starting with verbs at the beginning of the sentence and verbs in the middle of the sentences to be treated differently. E.g.: Pushing and pushing at different position in the sentence means the same thing.

```
mywords = [w.lower( ) for w in mytokens]
```

In the next step I have used Alpha filter to filter out tokens from the tokens assigned. Because we are analyzing texts and not punctuations.

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018



```
#3. Removing punctuations from lower-case tokens
```

```
import re
```

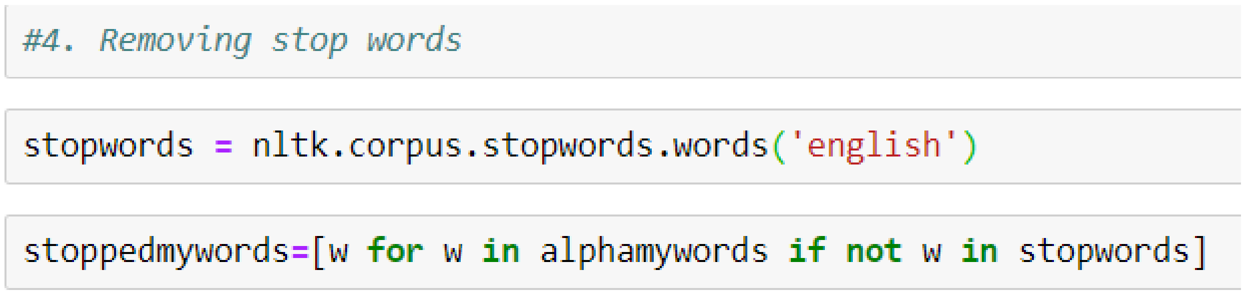
```
def alpha_filter(w):
  # pattern to match word of non-alphabetical characters
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False
```

```
alphamywords=[w for w in mywords if not alpha_filter(w)]
```

```
alphamywords[:50]
```

```
['this',
 'is',
 'just',
 'a',
 'curriculum',
 'vitae',
```

When, I analyzed my alphawords, I could see lot of common stop words so I used default stop words from nltk library to get rid of those stop words.



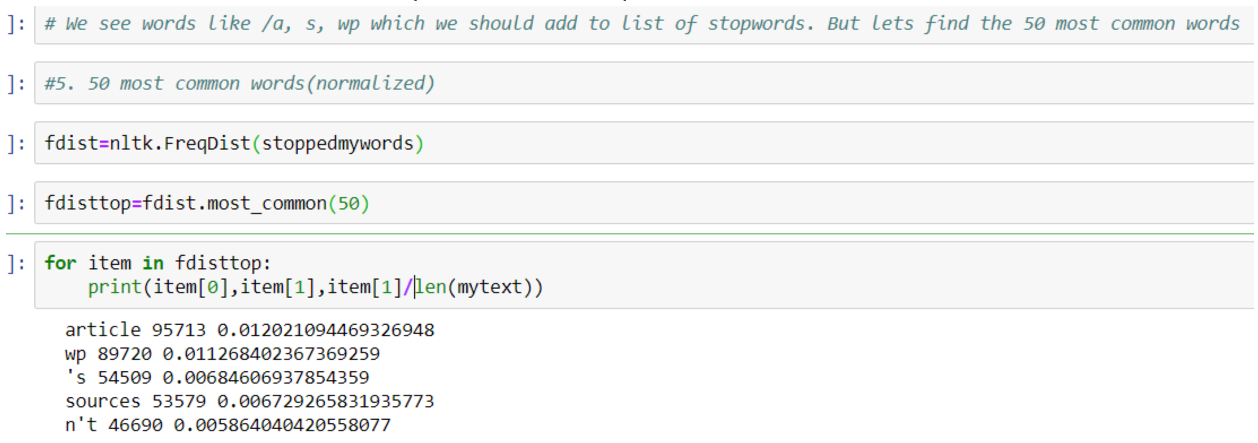
```
#4. Removing stop words
```

```
stopwords = nltk.corpus.stopwords.words('english')
```

```
stoppedmywords=[w for w in alphamywords if not w in stopwords]
```

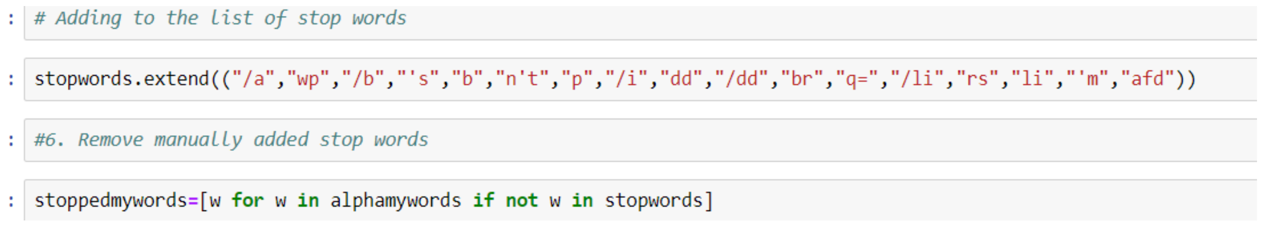
Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

But, when I analyzed the words after removing stop words, I could still find alphabets/ words which did not seemed relevant, so I manually added those stop words to the list.



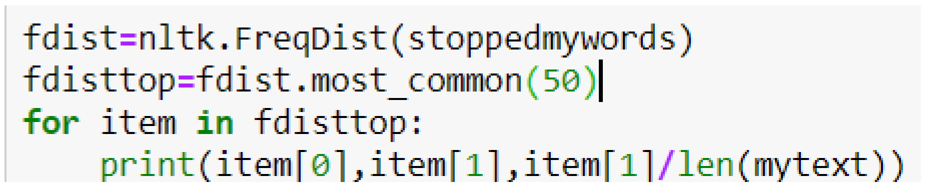
```
]: # We see words like /a, s, wp which we should add to list of stopwords. But lets find the 50 most common words

]: #5. 50 most common words(normalized)

]: fdist=nltk.FreqDist(stoppedmywords)

]: fdisttop=fdist.most_common(50)

]: for item in fdisttop:
       print(item[0],item[1],item[1]/len(mytext))

    article 95713 0.012021094469326948
    wp 89720 0.011268402367369259
    's 54509 0.00684606937854359
    sources 53579 0.006729265831935773
    n't 46690 0.005864040420558077
```

So, now my words are free of manually added stop words as well stop words that are defined by the nltk package.



```
: # Adding to the list of stop words

: stopwords.extend(("/a","wp","/b","'s","b","n't","p","/i","dd","/dd","br","q=","/li","rs","li","'m","afd"))

: #6. Remove manually added stop words

: stoppedmywords=[w for w in alphamywords if not w in stopwords]
```

- List the top 50 words by frequency (normalized by the length of the document)



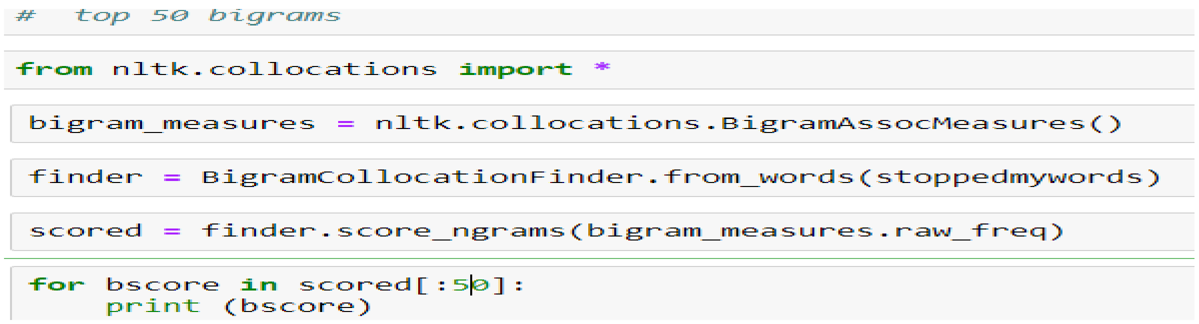
```
fdist=nltk.FreqDist(stoppedmywords)
fdisttop=fdist.most_common(50)
for item in fdisttop:
    print(item[0],item[1],item[1]/len(mytext))
```

```
article 95713 0.001090561074433901
sources 53579 0.0006104831298475023
notability 43464 0.0004952320639745393
notable 37705 0.00042961358761641 82
coverage 31552 0.00035950584581549 47
new 31264 0.00035622435229385 23
please 27492 0.000313245902420 1186
per 26943 0.0003069905553944 8766
add 26506 0.0003020113447383 8436
one 26173 0.000298217117853 9853
comments 26048 0.000296792858512 9946
thanks 25611 0.000291813647856 8913
notice 25134 0.000286378674211 67103
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
reliable 23584 0.000268717858383387
wikipedia 23015 0.0002622346298631976
articles 22389 0.0002551019390835164
would 21531 0.000245325822966956
gng 21041 0.0002397427263502733
fails 19121 0.0002178661028726570
subject 17533 0.00019977231220471187
also 17155 0.0001954653519575561
page 16548 0.0001885491485977055
find 15997 0.0001822710134226188
see 14935 0.00017017050606156
significant 14814 0.00016879182301948335
list 14708 0.0001675840510983233
like 13826 0.0001575344771882933
independent 13148 0.00014980929452276004
enough 13111 0.0001493877137578268
even 13107 0.0001493421374589151
could 12349 0.0001407054288151478
source 12134 0.0001382557027486439
seems 11644 0.0001326726061319606
meet 11155 0.0001271009035900051
deletion 11072 0.0001261551953875874
think 10946 0.0001247195419718688
references 10793 0.0001229762485384962
delete 10044 0.0001144420865672803
may 9640 0.0001098388803771985
news 9484 0.0001080614047196422
found 9443 0.0001075942476557973
non-notable 8903 0.0001014414473027177
nothing 8556 9.748770337212768e-05
google 8500 9.684963518736388e-05
two 8490 9.673569444008463e-05
search 8160 9.297564977986933e-05
information 8092 9.220085269837041e-05
keep 8048 9.169951341034171e-05
books 7791 8.877123620526494e-05
evidence 7744 8.823571469305246e-05
```

- list the top 50 bigrams by frequencies



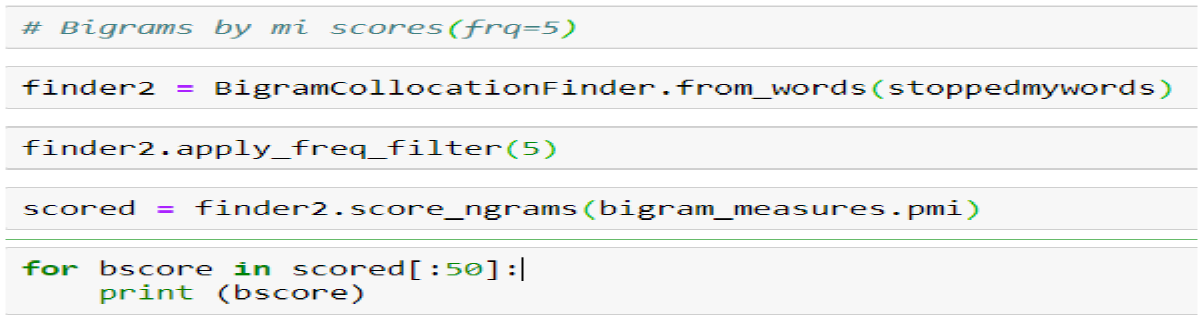
```
#   top 50 bigrams

from nltk.collocations import *

bigram_measures = nltk.collocations.BigramAssocMeasures()

finder = BigramCollocationFinder.from_words(stoppedmywords)

scored = finder.score_ngrams(bigram_measures.raw_freq)

for bscore in scored[:50]:
    print (bscore)
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
(('please', 'add'), 0.005764611157844224)
(('add', 'new'), 0.005755127549786572)
(('new', 'comments'), 0.005751808286966393)
(('comments', 'notice'), 0.005750859926160628)
(('notice', 'thanks'), 0.00574943738495198)
(('reliable', 'sources'), 0.0030127051897148372)
(('significant', 'coverage'), 0.0018196672960621215)
(('non-admin', 'closure'), 0.0014099754279715225)
(('thanks', 'please'), 0.0013829471450072122)
(('fails', 'gng'), 0.0013125313551791407)
(('wikipedia', 'articles'), 0.001002654461895337)
(('coverage', 'reliable'), 0.0009595040452330017)
(('per', 'nom'), 0.0007288152792306138)
(('establish', 'notability'), 0.0006930146588129747)
(('find', 'sources'), 0.0006925404784100921)
(('books', 'scholar'), 0.0006244955905964336)
(('newspapers', 'books'), 0.0006211763277762551)
(('scholar', 'highbeam'), 0.0006171457943517527)
(('highbeam', 'jstor'), 0.0006166716139488701)
(('independent', 'reliable'), 0.0006078992764955413)
(('could', 'find'), 0.0006052912842796868)
(('notability', 'guidelines'), 0.0005661714010418692)
(('independent', 'sources'), 0.0005647488598332212)
(('reliable', 'source'), 0.0005621408676173668)
(('comment', 'added'), 0.0005519459889553901)
(('unsigned', 'comment'), 0.0005476783653294463)
(('secondary', 'sources'), 0.0005457816437179157)
(('preceding', 'unsigned'), 0.0005453074633150331)
(('evidence', 'notability'), 0.0005329787728400846)
(('ca', 'find'), 0.0005178049999478402)
(('notable', 'enough'), 0.0005166195489406336)
(('meet', 'gng'), 0.0005144857371276617)
(('talk', 'page'), 0.0004810560187244357 4)
(('looks', 'like'), 0.0004665935164365153)
(('jstor', 'free'), 0.0004495230219327403 6)
(('free', 'images'), 0.0004492859317312990 5)
(('images', 'wikipedia'), 0.0004492859317312990 5)
(('news', 'newspapers'), 0.0004485746611269751)
(('wikipedia', 'library'), 0.0004469150297168858 4)
(('thanks', 'per'), 0.0004421732256880594 6)
(('google', 'search'), 0.0004258140017886085)
(('closure', 'non-admin'), 0.0004236801899756366 3)
(('talk', 'contribs'), 0.0004151449427237491 6)
(('general', 'notability'), 0.0004073209660761856)
(('non', 'notable'), 0.0003992598992271808)
(('reliable', 'independent'), 0.0003599029257879218 5)
(('original', 'research'), 0.0003537385805504476)
(('sources', 'article'), 0.0003527902197446823)
(('played', 'fully'), 0.0003523160393417996 5)
(('meet', 'notability'), 0.0003506564079317104)
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

- list the top 50 bigrams by their Mutual Information scores (using min f requency 5)



```
# Bigrams by mi scores(frq=5)

finder2 = BigramCollocationFinder.from_words(stoppedmywords)

finder2.apply_freq_filter(5)

scored = finder2.score_ngrams(bigram_measures.pmi)

for bscore in scored[:50]:
    print (bscore)
```

```
(('burr', 'steers'), 19.68613252954993)
(('helsingin', 'sanomat'), 19.68613252954993)
(('hemorrhagic', 'conjunctivitis'), 19.68613252954993)
(('inã©s', 'rodena'), 19.68613252954993)
(('khyber', 'pakhtunkhwa'), 19.68613252954993)
(('manadel', 'al-jamadi'), 19.68613252954993)
(('mys', '721tx'), 19.68613252954993)
(('pell', 'mell'), 19.68613252954993)
(('phnom', 'penh'), 19.68613252954993)
(('putroe', 'neng'), 19.68613252954993)
(('rot-weiãÿ', 'oberhausen'), 19.68613252954993)
(('schwã¤bisch', 'gmã¼nd'), 19.68613252954993)
(('sunanda', 'pushkar'), 19.68613252954993)
(('super-god', 'masterforce'), 19.68613252954993)
(('vis-ã', '-vis'), 19.68613252954993)
(('ashleigh', 'lollie'), 19.423098123716134)
(('beent', 'agged'), 19.423098123716134)
(('deletion/anshei', 'sfard'), 19.423098123716134)
(('deletion/beth', 'hamedrosh'), 19.423098123716134)
(('dudel250', 'chatprod'), 19.423098123716134)
(('energy-safety', 'energy-economy'), 19.423098123716134)
(('giro', "d'italia"), 19.423098123716134)
(('hamedrosh', 'hagodol-beth'), 19.423098123716134)
(('lorem', 'ipsum'), 19.423098123716134)
(('m.j.', 'ramanan'), 19.423098123716134)
(('margarita', 'martirena'), 19.423098123716134)
(('mong', 'kok'), 19.423098123716134)
(('movers', 'shakers'), 19.423098123716134)
(('rls=org.mozilla', 'en-us'), 19.423098123716134)
(('suhas', 'gopinath'), 19.423098123716134)
(('ulrike', 'ottinger'), 19.423098123716134)
(('vitalik', 'buterin'), 19.423098123716134)
(('xhulio', 'joka'), 19.423098123716134)
(('abdulhadi', 'najjar'), 19.200705702379686)
(('aqueduct', 'racetrack'), 19.200705702379686)
(('chal', 'jhoothey'), 19.200705702379686)
(('charles_manson', 'tate_murders'), 19.200705702379686)
(('deletion/tessa', 'campanelli'), 19.200705702379686)
(('diante', 'trono'), 19.200705702379686)
(('guo', 'dongli'), 19.200705702379686)
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
(('hidy', 'ochiai'), 19.200705702379686)
(('marlene', 'dietrich'), 19.200705702379686)
(('mushtaq', 'pahalgami'), 19.200705702379686)
(('officeâ€\x9dwp', 'polition'), 19.200705702379686)
(('option=com_content', 'view=article'), 19.200705702379686)
(('polition', 'states-'), 19.200705702379686)
(('rowman', 'littlefield'), 19.200705702379686)
(('sadman', 'sakibzz'), 19.200705702379686)
(('satish', 'rajwade'), 19.200705702379686)
(('sebalu', 'lule'), 19.200705702379686)
```

2. Analysis of NPS Chat corpus (25%)

NPS chat corpus is part of nltk library. NLTK library has collected 50000 posts from different chatting platforms. For the first release, which is 1.0 the NPS chat corpus has 10,567 posts. The main characteristics of NPS chat corpus are:

- Hand privacy masked: Privacy is very important. So, posts are two way masked and names of the user are changed to generic names. Also, personal information of user like address and device being used is removed.
- Part-of-speech tagged: The filename consist of all the information along with no of posts contain in the file. E.g. 12-11-50s_567posts.xml contains 567 posts gathere from 50s chat room and 12/11/2006 released. But the user becomes 12-11-50sUserN
- Dialogue-act tagged: They include greet, other, reject, accept, bye, clarify, emotion.

Here, we read the text from nps chat corpus and remove html tags using Beautiful soup library.

```
from nltk.corpus import nps_chat
```

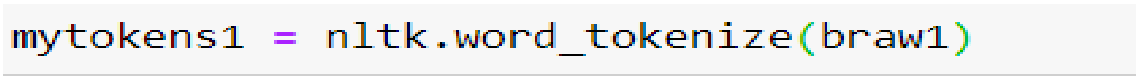
```
chattext=nps_chat.raw()
```

```
soup1 = BeautifulSoup(chattext, 'lxml')
```

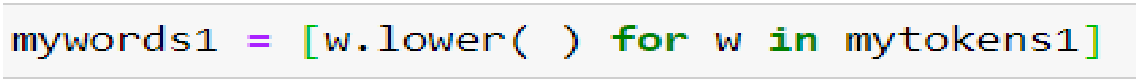
```
braw1 = soup1.get_text()
```

Now, we use tokenization and convert the tokens to lower as we don't want them to treat tokens starting with capital letter differently than words with small letter.

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

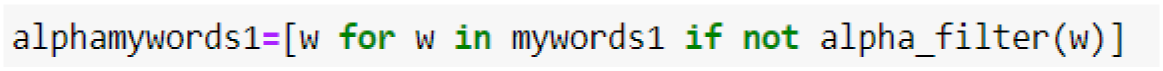


```
mytokens1 = nltk.word_tokenize(braw1)
```



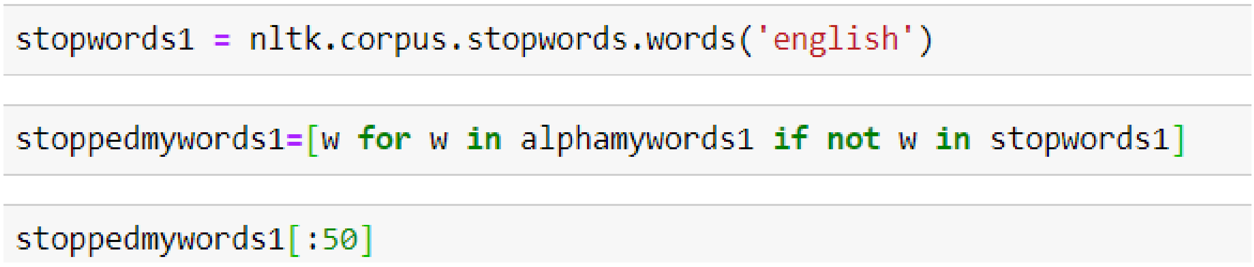
```
mywords1 = [w.lower( ) for w in mytokens1]
```

We use alpha filter to get rid of the punctuations.



```
alphamywords1=[w for w in mywords1 if not alpha_filter(w)]
```

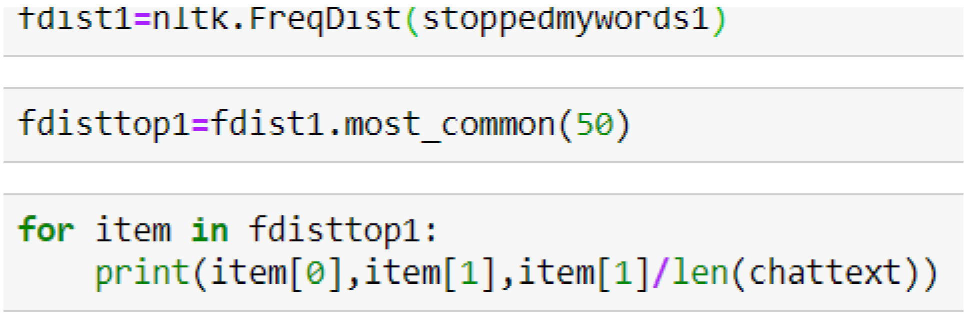
We use pre-defined stop words from nltk library to get rid of stop words.



```
stopwords1 = nltk.corpus.stopwords.words('english')
```

```
stoppedmywords1=[w for w in alphamywords1 if not w in stopwords1]
```

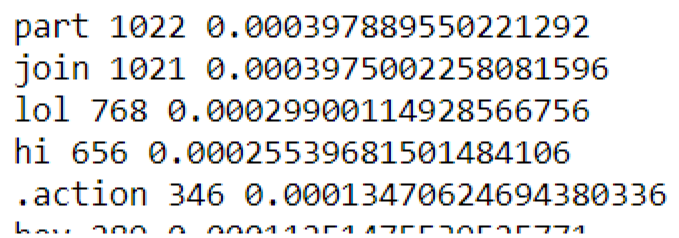
```
stoppedmywords1[:50]
```



```
fdist1=nltk.FreqDist(stoppedmywords1)
```

```
fdisttop1=fdist1.most_common(50)
```

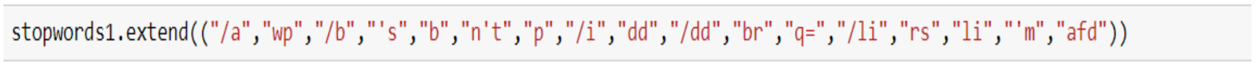
```
for item in fdisttop1:
    print(item[0],item[1],item[1]/len(chattext))
```



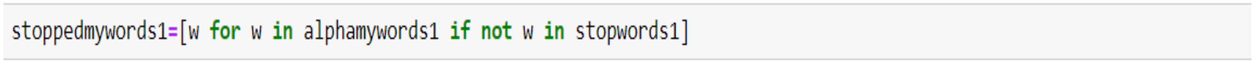
```
part 1022 0.000397889550221292
join 1021 0.0003975002258081596
lol 768 0.00029900114928566756
hi 656 0.00025539681501484106
.action 346 0.00013470624694380336
```

When we display the top 50 word, we see some words that does not make sense. So, we remove those words by adding them to the list of stop words.

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

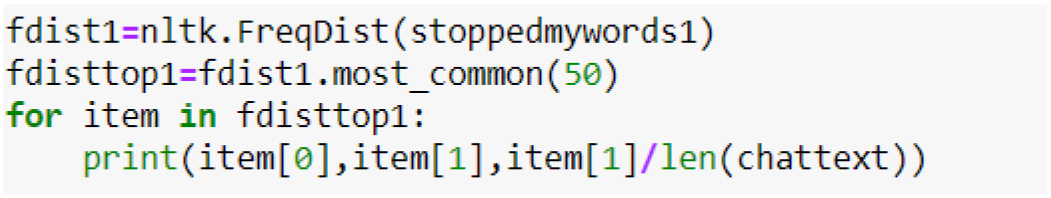


```
stopwords1.extend(("/a","wp","/b","'s","b","n't","p","/i","dd","/dd","br","q=","/li","rs","li","'m","afd"))
```



```
stoppedmywords1=[w for w in alphamywords1 if not w in stopwords1]
```

- list the top 50 words by frequency (normalized by the length of the document)



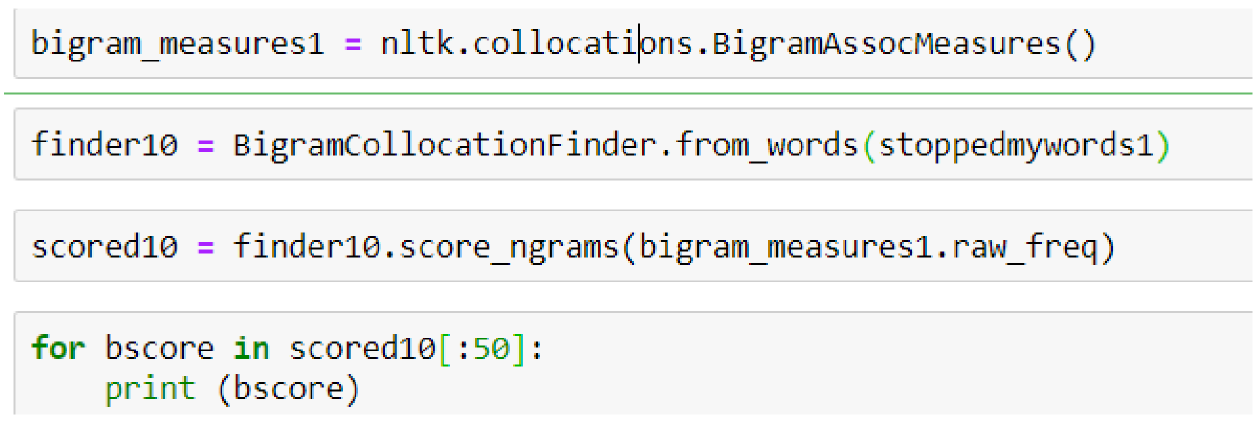
```
fdist1=nltk.FreqDist(stoppedmywords1)
fdisttop1=fdist1.most_common(50)
for item in fdisttop1:
    print(item[0],item[1],item[1]/len(chattext))
```

```
part 1022 0.0003978895502212 92
join 1021 0.0003975002258081596
lol 768 0.00029900114928566756
hi 656 0.00025539681501484106
.action 346 0.00013470624694380336
hey 289 0.00011251475539525771
u 203 7.903285586587307e-05
like 158 6.151325727491599e-05
pm 149 5.800933755672457e-05
na 147 5.723068873045981e-05
chat 145 5.645203990419505e-05
im 143 5.567339107793029e-05
good 129 5.0222849294076975e-05
lmao 122 4.7497578402150315e-05
wan 110 4.2825685444561764e-05
know 103 4.0100414552635105e-05
get 103 4.0100414552635105e-05
room 102 3.971109013950272e-05
ok 101 3.9321765726370344e-05
ya 98 3.8153792486973206e-05
wb 96 3.737514366070 8445e-05
hello 91 3.5428521595046546e-05
one 90 3.503919718191417e-05
oh 89 3.4649872768781786e-05
well 86 3.348189952938465e-05
hiya 84 3.270325070311989e-05
yes 83 3.231392628998751e-05
yeah 83 3.231392628998751e-05
back 79 3.0756628637457995e-05
got 79 3.0756628637457995e-05
go 77 2.9977979811193234e-05
dont 76 2.9588655398060854e-05
see 76 2.9588655398060854e-05
want 70 2.7252708919266575e-05
ty 70 2.7252708919266575e-05
everyone 66 2.5695411266737056e-05
love 64 2.4916762440472296e-05
anyone 60 2.3359464787942777e-05
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
guys 59 2.29701403748104e-05
talk 57 2.219149154854564e-05
would 55 2.1412842722280882e-05
right 55 2.1412842722280882e-05
think 54 2.10235183091485e-05
nice 53 2.063419389601612e-05
thanks 53 2.063419389601612e-05
girls 51 1.985554506975136e-05
time 50 1.9466220656618983e-05
11-09-40suser18 48 1.8687571830354223e-05
bye 46 1.7908923004089465e-05
haha 45 1.7519598590957085e-05
```

- list the top 50 bigrams by frequencies,



```
bigram_measures1 = nltk.collocations.BigramAssocMeasures()
```

```
finder10 = BigramCollocationFinder.from_words(stoppedmywords1)
```

```
scored10 = finder10.score_ngrams(bigram_measures1.raw_freq)
```

```
for bscore in scored10[:50]:
    print (bscore)
```

```
(('part', 'join'), 0.007656065959952885)
(('join', 'part'), 0.005378877110325873)
(('part', 'part'), 0.004789489595602674)
(('wan', 'na'), 0.0043188064389477815)
(('join', 'join'), 0.004279544562230075)
(('na', 'chat'), 0.002355712603062426)
(('join', 'hi'), 0.0021201413427561835)
(('lol', 'lol'), 0.0016882606988614056)
(('lol', 'join'), 0.0016097369454259915)
(('part', 'hi'), 0.0016097369454259915)
(('part', 'lol'), 0.0016097369454259915)
(('lol', 'hi'), 0.001531213191990577)
(('lol', 'part'), 0.0015312131919905777)
(('gon', 'na'), 0.0014526894385551163)
(('join', 'lol'), 0.0013741656851197488)
(('join', '.action'), 0.0013349038084020416)
(('mode', '14-19teens'), 0.0010993325480957799)
(('part', '.action'), 0.0010208087946603848)
(('part', 'hey'), 0.0009422850412249705)
(('join', 'hey'), 0.0009030231645072635)
(('pm', 'u'), 0.0009030231645072635)
(('14-19teens', '+o'), 0.0007852375343541421)
(('.action', 'watches'), 0.0007459756576364351)
(('18/m', 'pm'), 0.0007459756576364351)
(('chat', 'pm'), 0.0007067137809187279)
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
(('want', 'chat'), 0.0007067137809187279)
(('guys', 'wan'), 0.0006674519042010208)
(('lol', 'hey'), 0.0006281900274833137)
(('tryin', 'chat'), 0.0006281900274833137)
(('u', 'tryin'), 0.0006281900274833137)
(('anyone', 'wan'), 0.0005889281507656066)
(('r', 'u'), 0.0005889281507656066)
(('dont', 'know'), 0.0005496662740478995)
(('hi', 'everyone'), 0.0005496662740478995)
(('join', 'wb'), 0.0005104043973301924)
(('lol', '.action'), 0.0005104043973301924)
(('.action', 'sits'), 0.0004711425206124853)
(('got', 'ta'), 0.0004711425206124853)
(('join', 'lmao'), 0.0004711425206124853)
(('join', 'well'), 0.0004711425206124853)
(('la', 'la'), 0.0004711425206124853)
(('10-26-teensuser54', '10-26-teensuser54'), 0.00043188064389477815)
(('girls', 'wan'), 0.00043188064389477815)
(('hi', '10-19-40suser30'), 0.00043188064389477815)
(('na', 'talk'), 0.00043188064389477815)
(('see', 'ya'), 0.00043188064389477815)
(('.13cute.-ass', 'mp3'), 0.0003926187671770710 7)
(('.action', '.liam'), 0.0003926187671770710 7)
(('.action', 'listening'), 0.0003926187671770710 7)
(('.action', 'song'), 0.0003926187671770710 7)
```

- list the top 50 bigrams by their Mutual Information scores (using min frequency 5)

```
finder20 = BigramCollocationFinder.from_words(stoppedmywords1)
```

```
finder20.apply_freq_filter(5)
```

```
scored20 = finder20.score_ngrams(bigram_measures1.pmi)
```

```
for bscore in scored20[:50]:
    print (bscore)
```

```
(('lez', 'gurls'), 11.636511339161558)
(('gently', 'kisses'), 11.466586337719246)
(('bi', 'lez'), 11.314583244274196)
(('.13cute.-ass', 'mp3'), 11.314583244274194)
(('.liam', '.13cute.-ass'), 11.314583244274194)
(('times..', '.this'), 11.314583244274194)
(('fingers', 'thru'), 11.20355193188545)
(('neck', 'compliments'), 11.20355193188545)
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

```
(('bit', 'large'), 11.0515488384404)
(('played', 'times..'), 11.0515488384404)
(('ice', 'cream'), 10.914045314690465)
(('runs', 'fingers'), 10.788514432606608)
(('eyes', 'gently'), 10.636511339161558)
(('.9lime', 'player'), 10.549048497911217)
(('mp3', 'player'), 10.549048497911217)
(('player', '.song'), 10.549048497911217)
(('closes', 'eyes'), 10.373476933327764)
(('minutes', 'ago'), 10.314583244274193)
(('hair', 'closes'), 10.286014092077425)
(('minutes/seconds', 'music'), 10.112949383104544)
(('n', 'e'), 10.021801495046347)
((':10-26-teensuser122', '10-26-teensuser122'), 9.992655149386833)
(('talkin', 'bout'), 9.788514432606608)
(('la', 'la'), 9.725618812995542)
(('leave', 'alone'), 9.701051591356268)
((':10-26-teensuser54', '10-26-teensuser54'), 9.592117219803104)
(('.this', 'song'), 9.466586337719244)
(('player', 'listening'), 9.347414636741567)
(('14-19teens', '+o'), 9.296661336276932)
(('mode', '40splus'), 9.278959334543472)
(('busy', 'busy'), 9.221473839882712)
(('song', 'played'), 9.20355193188545)
(('mode', '14-19teens'), 9.179423660992562)
(('around', 'bit'), 8.849914977270751)
(('10-26-teensuser122', '10-26-teensuser122'), 8.518723961054421)
(('welcome', 'talkcity_adults'), 8.292708744651433)
(('14-19teens', '-o'), 8.085764553778315)
(('thru', 'back'), 8.06969618515066)
(('+o', '10-26-teensuser54'), 8.007154719081948)
(('10-26-teensuser54', '10-26-teensuser54'), 8.007154719081948)
(('mode', 'talkcity_adults'), 7.998851415350741)
(('main', 'room'), 7.964085997190061)
(('got', 'ta'), 7.830230250455269)
(('10-26-teensuser122', 'mode'), 7.805028146211061)
(('last', 'night'), 7.507228322216591)
(('females', 'want'), 7.507228322216589)
(('last', 'seen'), 7.466586337719246)
(('long', 'time'), 7.456602249146622)
(('gon', 'na'), 7.4368389943251945)
(('wan', 'na'), 7.436838994325193)
```

3. **Comparison (30%)**

a) How are Wikipedia discussions and NPS chats similar or different in the use of the language, based on your results?

Wikipedia discussions and NPS chat are quite different in use of language. Wikipedia discussion being a professiona l forum uses words with correct spellings and vocabulary is quite high compared to NPS chat. While after analyzing NPS chat we find that formal language is not used. Some interesting discoveries are, for Wikipedia discussions Tha nk you is used while in NPS chat ty is used but both imply thank you. NPS chat also has short forms like lmao. Sourc

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

es, articles are among most commonly used words for Wikipedia discussion and join, haha, part are words frequen
tly used by NPS chat.

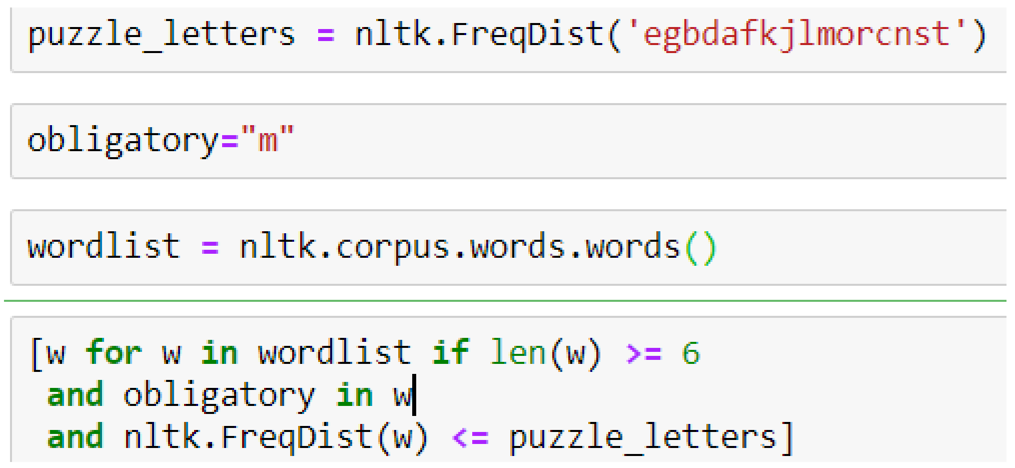
b) How are the processing options similar or different for the two analysis tasks?

Processing options are quite similar for the two analysis task. However, we cannot do anything about the sho
rt forms used for NPS chats. Also, manually added stop words added to the nltk stop words list where more w
hen analyzing the Wikipedia discussion. But for analyzing NPS chat the list of stop words added manually was
small. Because being a Human I could not discard word like lmao and haha because I know their meaning But
if it's auto run, computer will remove these words because they might not be in the vocabulary of nltk corpus

c) Are there any problems with the word or bigram lists that you found? Could you get a better list of bigram
s? How are the top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Informati
on?

Yes, e.g. words like non-admin, dudel250 appear in the bigrams. Where I have used Alpha filter so it should g
et ride of them. I do get a better list of bigrams once I remove my stop words for Wikipedia discussion. Howe
ver, same is not the case with NPS chat. Top 50 bigrams by frequency and Top 50 bigrams with mutual inform
ation are very different. The list is completely different. Hardly any Bigrams by frequency are not present in bi
grams by mutual information. Also, for Wikipedia discussion Bigrams by frequency are very clean and contain
only alphabets but Bigrams by mutual information has words containing numbers as well as punctuation.

4. Word and Name Puzzle (15%)

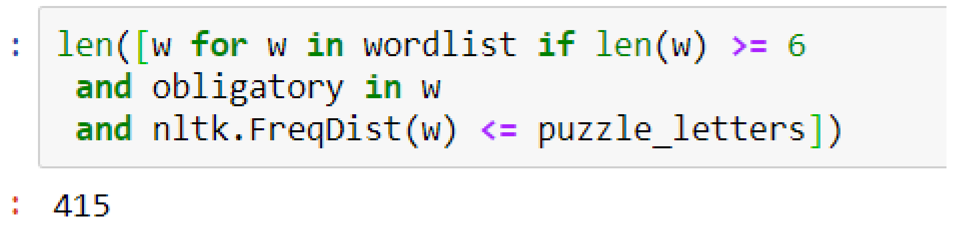


```
puzzle_letters = nltk.FreqDist('egbdafkjlmorcnst')
```

```
obligatory="m"
```

```
wordlist = nltk.corpus.words.words()
```

```
[w for w in wordlist if len(w) >= 6
 and obligatory in w
 and nltk.FreqDist(w) <= puzzle_letters]
```



```
: len([w for w in wordlist if len(w) >= 6
   and obligatory in w
   and nltk.FreqDist(w) <= puzzle_letters])

: 415
```

Name: Harsh Darji
NLP ASSIGNMENT 1
02/17/2018

References:

[1] Eric N. Forsyth and Craig H. Martell, "Lexical and Discourse Analysis of Online Chat Dialog," Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), pp. 19-26, September 2007.

[2] T. Wu, F. M. Khan, T. A. Fisher, L. A. Shuler and W. M. Pottenger, "Posting act tagging using transformation-based learning," Proceedings of the Workshop on Foundations of Data Mining and Discovery, IEEE International Conference on Data Mining, December 2002.

[3] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema and M. Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," Computational Linguistics, vol. 26, no. 3, pp. 339-373, 2000.

http://www.nltk.org/book/ch02.html

http://faculty.nps.edu/cmartell/NPSChat.htm