NLP Homework 1

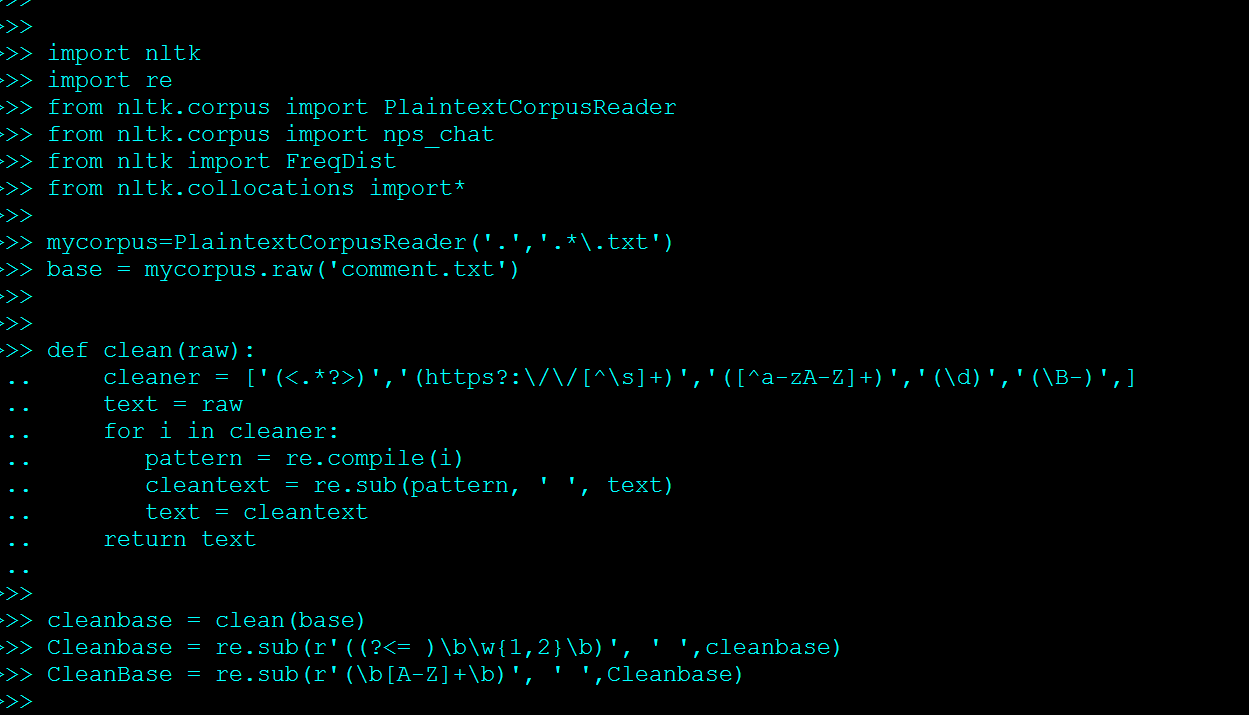
**Analysis of Wikipedia discussion forum**

* Cleaning the text:
* Remove html tags: Did not use HTML parser because text contains only formatting and ‘a href’ tags that exist with in the triangle brackets (<>) thus RegEx to remove all text inside <>
* Convert all to lowercase: since this is a comment section many users will use uppercase text to emphasize a point etc. It is best to reduce all text to lowercase so as to treat them equally without context.
* Remove stop words
* No salutations

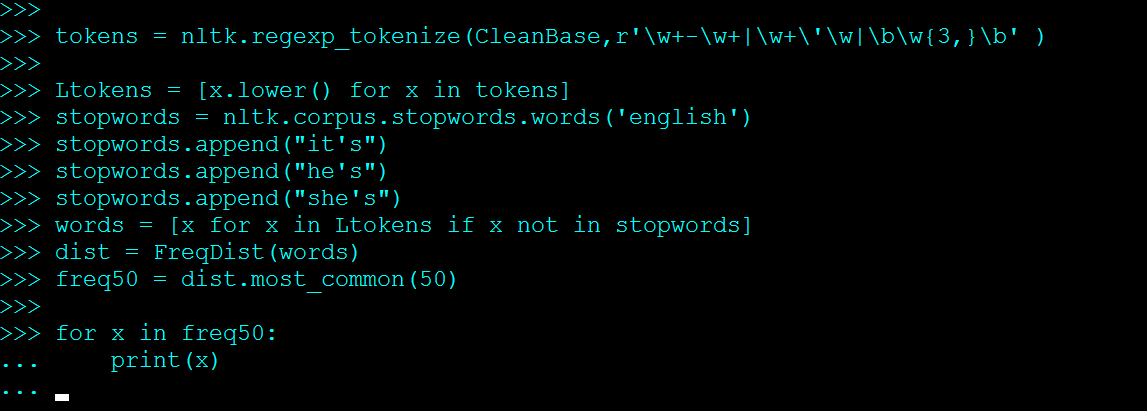
Filter:

(<.\*?>)|(https?:\/\/[^\s]+)|([^a-zA-Z]+)|(\d)|(\B-)| ((?<= )\b\w{1,2}\b)| (\b[A-Z]+\b)

For some reason the (\b[A-Z]+\b) and the ((?<= )\b\w{1,2}\b) didn’t run on the text. So I had to run them separately as two different filters.



Tokenized the cleaned text using ‘\w+-\w+|\w+\'\w|\b\w{3,}\b’



List the top 50 words by frequency

('article', 104515)

('sources', 58772)

('notable', 52245)

('notability', 49610)

('coverage', 34018)

('new', 31597)

('per', 28811)

('one', 28436)

('please', 27741)

('add', 26691)

('comments', 26246)

('thanks', 25904)

('notice', 25206)

('reliable', 24817)

('wikipedia', 24745)

('articles', 23999)

('non', 23532)

('would', 21205)

('fails', 20145)

('subject', 19036)

('page', 18596)

('also', 17944)

('find', 16755)

('deletion', 16600)

('list', 15885)

('see', 15679)

('significant', 15631)

('like', 14755)

('enough', 14061)

('independent', 13897)

('even', 13734)

('source', 13108)

('seems', 12177)

('references', 11808)

('meet', 11738)

('think', 11545)

('delete', 11480)

('could', 11157)

('news', 10205)

('found', 10196)

('may', 10027)

('afd', 9882)

('well', 9535)

('google', 9123)

('two', 9052)

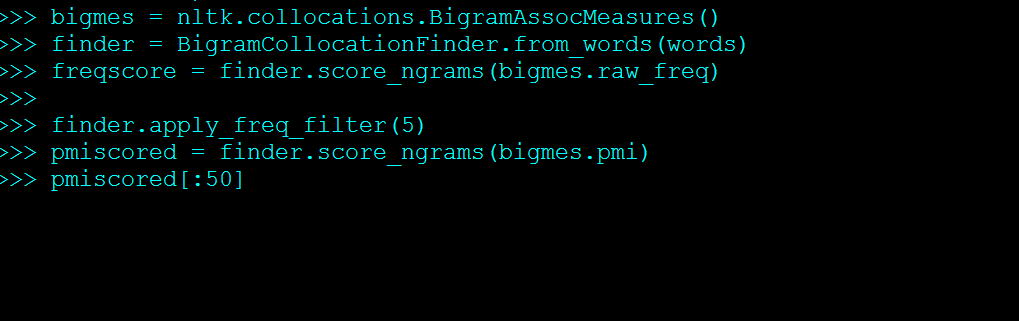
('nothing', 9039)

('keep', 8883)

('time', 8775)

('search', 8706)

('information', 8650)



List the top 50 bigrams by frequencies

(('please', 'add'), 0.005752366827245494)

(('add', 'new'), 0.005740306862261958)

(('new', 'comments'), 0.0057367598137373875)

(('comments', 'notice'), 0.005735813934130836)

(('notice', 'thanks'), 0.0057343951147210085)

(('reliable', 'sources'), 0.0034006736554557863)

(('significant', 'coverage'), 0.0019664837020214395)

(('non-admin', 'closure'), 0.0014242582175655518)

(('thanks', 'please'), 0.001387132443008389)

(('wikipedia', 'articles'), 0.0010075982508794316)

(('coverage', 'reliable'), 0.0009995582742237403)

(('per', 'nom'), 0.0009532101735026963)

(('articles', 'deletion'), 0.0009191585076668272)

(('establish', 'notability'), 0.0008113282325199084)

(('find', 'sources'), 0.000710592054422129)

(('independent', 'sources'), 0.0006935662215041945)

(('notability', 'guidelines'), 0.0006803239070124677)

(('evidence', 'notability'), 0.000675830978881346)

(('independent', 'reliable'), 0.0006335028664881477)

(('books', 'scholar'), 0.0006214429015046107)

(('reliable', 'source'), 0.0006207334917996967)

(('newspapers', 'books'), 0.000617895852980041)

(('scholar', 'highbeam'), 0.0006134029248489194)

(('secondary', 'sources'), 0.0006077276472096079)

(("don't", 'see'), 0.0005911747540949493)

(("don't", 'think'), 0.0005687101134393412)

(('talk', 'page'), 0.00055948778727546)

(('notable', 'enough'), 0.000558778377570546)

(('comment', 'added'), 0.0005514478106197687)

(('unsigned', 'comment'), 0.0005464819426853711)

(('preceding', 'unsigned'), 0.0005441172436689913)

(("can't", 'find'), 0.0005372596165214899)

(('looks', 'like'), 0.0004925668051119118)

(('closure', 'non-admin'), 0.0004793244906201849)

(('google', 'search'), 0.0004556775004563869)

(('thanks', 'per'), 0.000454495150948197)

(('free', 'images'), 0.00044811046360397155)

(('wikipedia', 'library'), 0.00044811046360397155)

(('news', 'newspapers'), 0.0004459822344892297)

(('images', 'wikipedia'), 0.00044574576458759175)

(('highbeam', 'free'), 0.0004452728247843158)

(('non', 'notable'), 0.00044456341507940187)

(('talk', 'contribs'), 0.00043226698019422694)

(('general', 'notability'), 0.00042493641324344955)

(('original', 'research'), 0.0004171329064893962)

(('could', 'find'), 0.00041571408707956835)

(('sources', 'article'), 0.00040980233953861886)

(('wikipedia', 'article'), 0.000384736529964993)

(('claim', 'notability'), 0.0003830812406535271)

(('meet', 'notability'), 0.00037858831252240554)

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

(('bergstr', 'isacsson'), 19.68490097019854)

(('buah', 'pukul'), 19.68490097019854)

(('burr', 'steers'), 19.68490097019854)

(('cristine', 'nickol'), 19.68490097019854)

(('helsingin', 'sanomat'), 19.68490097019854)

(('hemorrhagic', 'conjunctivitis'), 19.68490097019854)

(('khyber', 'pakhtunkhwa'), 19.68490097019854)

(('krav', 'maga'), 19.68490097019854)

(('manadel', 'jamadi'), 19.68490097019854)

(('putroe', 'neng'), 19.68490097019854)

(('schw', 'bisch'), 19.68490097019854)

(('sunanda', 'pushkar'), 19.68490097019854)

(('tallinna', 'linnatranspordi'), 19.68490097019854)

(('timi', 'oara'), 19.68490097019854)

(('toki', 'pona'), 19.68490097019854)

(('valentia', 'nesterova'), 19.68490097019854)

(('wishy', 'washy'), 19.68490097019854)

(('adev', 'rul'), 19.421866564364745)

(('anshei', 'sfard'), 19.421866564364745)

(('ashleigh', 'lollie'), 19.421866564364745)

(('atl', 'tico'), 19.421866564364745)

(('bech', 'bruun'), 19.421866564364745)

(('beent', 'agged'), 19.421866564364745)

(('berkin', 'elvan'), 19.421866564364745)

(('celso', 'barbosa'), 19.421866564364745)

(('folken', 'fanel'), 19.421866564364745)

(('hamedrosh', 'hagodol'), 19.421866564364745)

(('hottopics', 'lnacademic'), 19.421866564364745)

(('inglourious', 'basterds'), 19.421866564364745)

(('jurnalul', 'ional'), 19.421866564364745)

(('kuala', 'lumpur'), 19.421866564364745)

(('margarita', 'martirena'), 19.421866564364745)

(('palo', 'alto'), 19.421866564364745)

(('pell', 'mell'), 19.421866564364745)

(('phnom', 'penh'), 19.421866564364745)

(('rodr', 'guez'), 19.421866564364745)

(('stj', 'rdal'), 19.421866564364745)

(('ulrike', 'ottinger'), 19.421866564364745)

(('xhulio', 'joka'), 19.421866564364745)

(('diante', 'trono'), 19.1994741430283)

(('sadman', 'sakibzz'), 19.1994741430283)

(('virtuti', 'militari'), 19.1994741430283)

(('wyser', 'pratte'), 19.1994741430283)

(('xanthine', 'oxidase'), 19.1994741430283)

(('aqueduct', 'racetrack'), 19.199474143028297)

(('axl', 'hazarika'), 19.199474143028297)

(('chal', 'jhoothey'), 19.199474143028297)

(('hidy', 'ochiai'), 19.199474143028297)

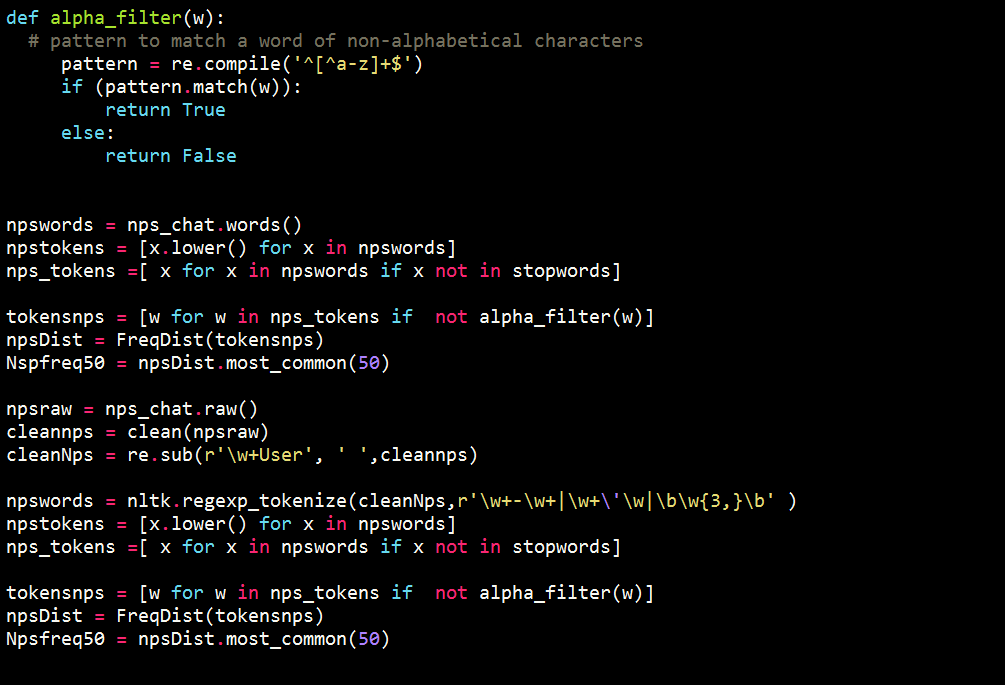
(('lorem', 'ipsum'), 19.199474143028297)

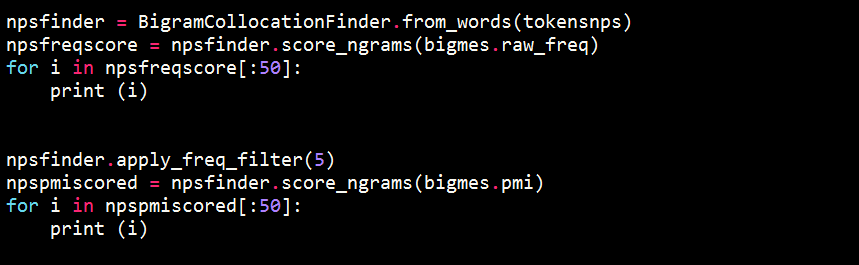
(('marlene', 'dietrich'), 19.199474143028297)

**Analysis of NPS Chat corpus**

Brief Description

The NPS Chat Corpus was created by [Eric Forsyth](http://faculty.nps.edu/cmartell/07Sep_Forsyth.pdf), [Jane Lin](http://faculty.nps.edu/cmartell/07Mar_Lin.pdf), and their thesis advisor [Craig Martell](http://faculty.nps.edu/cmartell) at the Department of Computer Science, Naval Postgraduate School in California in 2007. It is distributed as part of the Natural Language Toolkit ([NLTK](http://nltk.org/)) for research and educational purposes. It contains more than 10,000 posts from many online chat services. The posts are all organized into 15 files. To keep the users of these chant rooms anonymous, all the usernames have been changed in the corpus to the form “UserN”. The corpus is further anonymized by humans who went through the posts to manually remove any identifiable information. The files of the corpus contain posts from a specific day and are divided by age specific chatrooms. The naming conventions for the files are as follows: date, age of the chat room, number of posts in that chatroom. For example, for the file, 8-26-30s\_500posts.xml, there are 500 posts from the 30 year old chat room for 26th august 2006. All posts are from between October 19th 2006 to November 9th 2006.





List the top 50 words by frequency

('lol', 705)

('hey', 264)

('like', 156)

('chat', 142)

('good', 128)

('lmao', 107)

('wanna', 107)

('know', 103)

('get', 102)

('room', 98)

('one', 86)

('well', 80)

('back', 79)

('hiya', 78)

('see', 75)

('yeah', 75)

('dont', 75)

('hello', 71)

('yes', 70)

('want', 70)

('got', 68)

('everyone', 64)

('love', 62)

('guys', 59)

('talk', 56)

('think', 54)

('right', 54)

('nice', 52)

('would', 51)

('thanks', 50)

('anyone', 50)

('girls', 49)

('time', 49)

('never', 45)

('thats', 45)

('bye', 44)

('haha', 44)

('need', 43)

('make', 42)

('You', 42)

('people', 41)

('really', 41)

('whats', 41)

('much', 40)

('girl', 40)

('night', 40)

('work', 38)

('gonna', 37)

('take', 37)

('hot', 37)

List the top 50 bigrams by frequencies

(('wanna', 'chat'), 0.0034864150036066363)

(('lol', 'lol'), 0.003426304400096177)

(('lol', 'hey'), 0.0015027650877614811)

(('hey', 'hey'), 0.001262322673719644)

(('want', 'chat'), 0.0011421014666987256)

(('guys', 'wanna'), 0.0010819908631882664)

(('talkcity', 'adults'), 0.0010218802596778072)

(('tryin', 'chat'), 0.0009617696561673479)

(('dont', 'know'), 0.0008415484491464295)

(('anyone', 'wanna'), 0.0006612166386150517)

(('girls', 'wanna'), 0.0006612166386150517)

(('lol', 'lmao'), 0.0006612166386150517)

(('Liam', 'cute'), 0.0006011060351045924)

(('This', 'listening'), 0.0006011060351045924)

(('ass', 'player'), 0.0006011060351045924)

(('cute', 'ass'), 0.0006011060351045924)

(('hey', 'welcome'), 0.0006011060351045924)

(('lasts', 'minutes'), 0.0006011060351045924)

(('lmao', 'lol'), 0.0006011060351045924)

(('minutes', 'seconds'), 0.0006011060351045924)

(('played', 'times'), 0.0006011060351045924)

(('player', 'This'), 0.0006011060351045924)

(('seconds', 'Music'), 0.0006011060351045924)

(('song', 'lasts'), 0.0006011060351045924)

(('song', 'played'), 0.0006011060351045924)

(('times', 'song'), 0.0006011060351045924)

(('wanna', 'talk'), 0.0006011060351045924)

(('welcome', 'room'), 0.0006011060351045924)

(('last', 'night'), 0.0005409954315941332)

(('like', 'lol'), 0.0005409954315941332)

(('long', 'time'), 0.0005409954315941332)

(('hey', 'everyone'), 0.00048088482808367395)

(('hey', 'hiya'), 0.00048088482808367395)

(('hey', 'lol'), 0.00048088482808367395)

(('hug', 'watches'), 0.00048088482808367395)

(('know', 'lol'), 0.00048088482808367395)

(('lol', 'well'), 0.00048088482808367395)

(('room', 'hey'), 0.00048088482808367395)

(('teens', 'teens'), 0.00048088482808367395)

(('busy', 'busy'), 0.00042077422457321474)

(('hiya', 'hiya'), 0.00042077422457321474)

(('lol', 'hiya'), 0.00042077422457321474)

(('lol', 'love'), 0.00042077422457321474)

(('lol', 'thanks'), 0.00042077422457321474)

(('right', 'lol'), 0.00042077422457321474)

(('room', 'lol'), 0.00042077422457321474)

(('Last', 'seen'), 0.0003606636210627555)

(('anyone', 'want'), 0.0003606636210627555)

(('bye', 'bye'), 0.0003606636210627555)

(('females', 'want'), 0.0003606636210627555)

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

(('Lime', 'Player'), 11.700092874758997)

(('Player', 'Song'), 11.437058468925201)

(('lez', 'gurls'), 11.02202096964636)

(('gently', 'kisses'), 10.852095968204047)

(('neck', 'Compliments'), 10.58906156237025)

(('bit', 'large'), 10.437058468925201)

(('fingers', 'thru'), 10.437058468925201)

(('ice', 'cream'), 10.437058468925201)

(('seconds', 'Music'), 10.321581251505267)

(('Liam', 'cute'), 10.214666047588754)

(('player', 'This'), 10.174024063091409)

(('eyes', 'gently'), 10.11513037403784)

(('played', 'times'), 9.951631641754961)

(('closes', 'eyes'), 9.852095968204047)

(('lasts', 'minutes'), 9.852095968204047)

(('runs', 'fingers'), 9.852095968204047)

(('talkcity', 'adults'), 9.6916312960108)

(('hair', 'closes'), 9.671523722562226)

(('minutes', 'seconds'), 9.473584344950316)

(('This', 'listening'), 9.235424607755553)

(('ass', 'player'), 9.174024063091409)

(('slaps', 'around'), 9.174024063091409)

(('talkin', 'bout'), 9.174024063091409)

(('leave', 'alone'), 9.08656122184107)

(('cute', 'ass'), 8.951631641754961)

(('Last', 'seen'), 8.892737952701392)

(('song', 'lasts'), 8.852095968204045)

(('busy', 'busy'), 8.654450209203281)

(('Welcome', 'talkcity'), 8.63658993245284)

(('song', 'played'), 8.58906156237025)

(('minutes', 'ago'), 8.530167873316682)

(('times', 'song'), 8.366669141033803)

(('around', 'bit'), 8.174024063091409)

(('hug', 'watches'), 8.130237266428045)

(('last', 'night'), 7.477700453422544)

(('thru', 'back'), 7.455205815635461)

(('main', 'room'), 7.407311125531148)

(('females', 'want'), 7.01826883478525)

(('teens', 'teens'), 6.933232730929449)

(('tryin', 'chat'), 6.784811008891333)

(('long', 'time'), 6.769881204915855)

(('wana', 'chat'), 6.45723635086283)

(('wants', 'talk'), 6.144276719697356)

(('wanna', 'chat'), 5.988787858868099)

(('never', 'seen'), 5.72281295125908)

(('bye', 'bye'), 5.688120233092919)

(('welcome', 'room'), 5.599956203473546)

(('guys', 'wanna'), 5.567835935325679)

(('hot', 'guys'), 5.251852649542929)

(('girls', 'wanna'), 5.1252757577672945)

**Comparison**

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

Wikipedia language is much more nuanced and structured while NPS chat’s language is riddled with syntax and spelling errors. But this is understandable as the sources are very different from each other. The NPS chat corpus is a collection of online messages from 2006 from multiple messaging services. These platforms don’t have standard rules and guidelines for posts whereas the Wikipedia discussions page, as the website itself, has strictly laid down guidelines for posting and this limits he kind of language used on the site.

Apart from the rules and regulations, I feel the content itself plays a role in the language here. The NPS corpus has different chats from a certain times of day. There is no connection between the chats. Wikipedia’s discussion page on the other hand is a better collective as all the users on that site are discussing one topic: “To delete a Wikipedia page or not” and the language used here will be centered around that topic as well.

1. How are the processing options similar or different for the two analysis tasks?

I did use similar kind of filters for both the texts. Removing the HTML tags and so on. Stop words were removed from texts and the both were converted to lowercase. Punctuation was also remove, which significantly reduced the size of the NPS corpus because internet comments come with indescribable punctuation. Some comments were entirely made up of punctuations.. eg ☺ ;) etc.

And extra filter for the NPS text was used. \w+User

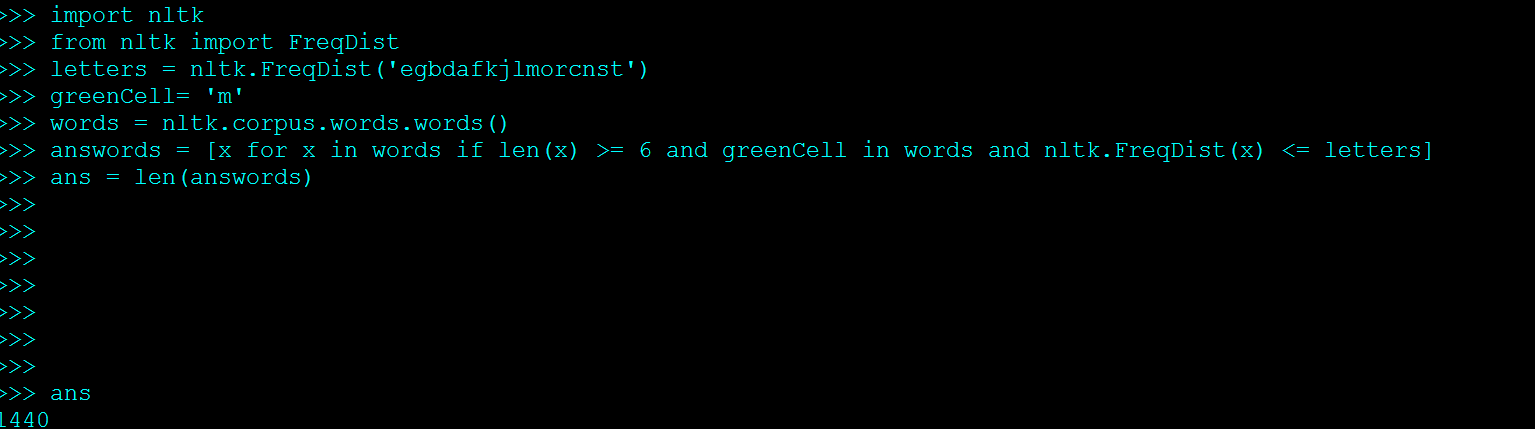
1. Are there any problems with the word or bigram lists that you found? Could you get a better list of bigrams? How are the top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Information?

I am satisfied with the bigram for both the chat corpus. Since the language used in both of them is different, the bigram also reflect that difference. Yes, the bigram lists can be made better. There is always room for improvement. By altering the filtering and tokenization methods a new set of bigrams can be obtained. It will depend on what we further need to do with the bigram for our analysis. If the bigrams are used to predict fully structured sentences then maybe we could also include punctuations. But if the bigrams are used to calculate the occurrence of possible word pairs then punctuations can be stripped from the tokens.

One problem I faced was with the bigrams of the Wikipedia discussion pages. While the bigrams by frequency were actually naturally occurring pairs of words, the bigrams by PMI were a little muddled. I assume it is because of the usernames in the file. I spent a lot of time trying to filter/ tokenize better to gain meaningful PMI bigrams. In the beginning the bigrams involved hexadecimal numbers, but after filtering all non-alphabetical characters that problem was taken care of. Yet the bigram pairs make no sense in English. On the other hand the bigrams by frequency and by PMI for the NPS chat corpus made much more sense. I am also guessing that since uniquely occurring pairs of words get PMI high scores, the more unlikely pairs will be on the top of the list.

**Word and Name Puzzle**

This took a very long time to process.



1440 words can be made with the given criteria.

**Code:**

import nltk

import re

from nltk.corpus import PlaintextCorpusReader

from nltk.corpus import nps\_chat

from nltk import FreqDist

from nltk.collocations import\*

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

base = mycorpus.raw('comment.txt')

def clean(raw):

cleaner = ['(<.\*?>)','(https?:\/\/[^\s]+)','([^a-zA-Z]+)','(\d)','(\B-)',]

text = raw

for i in cleaner:

pattern = re.compile(i)

cleantext = re.sub(pattern, ' ', text)

text = cleantext

return text

cleanbase = clean(base)

Cleanbase = re.sub(r'((?<= )\b\w{1,2}\b)', ' ',cleanbase)

CleanBase = re.sub(r'(\b[A-Z]+\b)', ' ',Cleanbase)

tokens = nltk.regexp\_tokenize(CleanBase,r'\w+-\w+|\w+\'\w|\b\w{3,}\b' )

Ltokens = [x.lower() for x in tokens]

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

stopwords.append("it's")

stopwords.append("he's")

stopwords.append("she's")

words = [x for x in Ltokens if x not in stopwords]

dist = FreqDist(words)

freq50 = dist.most\_common(50)

for x in freq50:

print(x)

bigmes = nltk.collocations.BigramAssocMeasures()

finder = BigramCollocationFinder.from\_words(words)

freqscore = finder.score\_ngrams(bigmes.raw\_freq)

finder.apply\_freq\_filter(5)

pmiscored = finder.score\_ngrams(bigmes.pmi)

pmiscored[:50]

def alpha\_filter(w):

# pattern to match a word of non-alphabetical characters

pattern = re.compile('^[^a-z]+$')

if (pattern.match(w)):

return True

else:

return False

npswords = nps\_chat.words()

npstokens = [x.lower() for x in npswords]

nps\_tokens =[ x for x in npswords if x not in stopwords]

tokensnps = [w for w in nps\_tokens if not alpha\_filter(w)]

npsDist = FreqDist(tokensnps)

Nspfreq50 = npsDist.most\_common(50)

npsraw = nps\_chat.raw()

cleannps = clean(npsraw)

cleanNps = re.sub(r'\w+User', ' ',cleannps)

npswords = nltk.regexp\_tokenize(cleanNps,r'\w+-\w+|\w+\'\w|\b\w{3,}\b' )

npstokens = [x.lower() for x in npswords]

nps\_tokens =[ x for x in npswords if x not in stopwords]

tokensnps = [w for w in nps\_tokens if not alpha\_filter(w)]

npsDist = FreqDist(tokensnps)

Npsfreq50 = npsDist.most\_common(50)

npsfinder = BigramCollocationFinder.from\_words(tokensnps)

npsfreqscore = npsfinder.score\_ngrams(bigmes.raw\_freq)

for i in npsfreqscore[:50]:

print (i)

npsfinder.apply\_freq\_filter(5)

npspmiscored = npsfinder.score\_ngrams(bigmes.pmi)

for i in npspmiscored[:50]:

print (i)

#%%

letters = nltk.FreqDist('EGBDAFKJLMORCNST'.lower())

greenCell= 'M'

words = nltk.corpus.words.words()

answords = [x for x in words if len(x) >= 6 and greenCell in words and nltk.FreqDist(x) <= letters]

ans = len(answords)