



FEBRUARY 17, 2019


NATURAL LANGUAGE PROCESSING

HOMEWORK ASSIGNMENT 1

RAHUL NARPATRAJ RATHOD

MS INFORMATION MANAGEMENT

rn Rathod@syr.edu



Analysis of Wikipedia Discussion forum

- Text Cleaning: Used NLTK package to pre-process the given text.
- I have made use of regular expressions to extract text between "<>" and "ahref". Html parser was not that effective to extract the words from inside this tag.
- I have tokenized all the cleaned text using regexp_tokenize.
- Remove stop words
- Convert all the words to lower case to treat each word equally and avoid any word count error.

```
In [2]: import nltk
import re
from nltk.corpus import PlaintextCorpusReader
from nltk.corpus import nps_chat
from nltk import FreqDist
from nltk.collocations import *
```

```
In [3]: mycorpus=PlaintextCorpusReader('.', '.*\.txt')
base = mycorpus.raw('HW_WikipediaDiscussions.txt')
```

```
In [7]: 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]
```

```
In [8]: dist = FreqDist(words)
freq50 = dist.most_common(50)

for x in freq50:
    print(x)
```

Out:

```
('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)
```

```
In [9]: bigmes = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(words)
freqscore = finder.score_ngrams(bigmes.raw_freq)
```

```
In [11]: for bscore in freqscore[:50]:
print(bscore)
```

Out:

```
((('please', 'add'), 0.005772381742226588)
(('add', 'new'), 0.00576027981552867)
(('new', 'comments'), 0.0057567204253234005)
(('comments', 'notice'), 0.005755771254601995)
(('notice', 'thanks'), 0.005754347498519887)
(('reliable', 'sources'), 0.003436947182208744)
(('non', 'notable'), 0.0026835429220932534)
(('significant', 'coverage'), 0.0019756988566052196)
(('non', 'admin'), 0.0014550787159143895)
(('admin', 'closure'), 0.0014346715454041747)
(('thanks', 'please'), 0.0013919588629409342)
(('wikipedia', 'articles'), 0.0010459861349886872)
(('coverage', 'reliable'), 0.0010044599159272033)
(('per', 'nom'), 0.0009664930870709897)
(('articles', 'deletion'), 0.0009522555262499096)
(('establish', 'notability'), 0.0008143884789657836)
(('find', 'sources'), 0.0007130645044557633)
(('independent', 'sources'), 0.0007097424069308446)
(('notability', 'guidelines'), 0.0006826910413707923)
```

```
(('evidence', 'notability'), 0.0006793689438458737)
(('third', 'party'), 0.0006451987978752813)
(('independent', 'reliable'), 0.0006366562613826333)
(('reliable', 'source'), 0.0006340460418987686)
(('books', 'scholar'), 0.0006236051639633098)
(('newspapers', 'books'), 0.0006200457737580398)
(('scholar', 'highbeam'), 0.0006155372128313644)
(('secondary', 'sources'), 0.0006100794811832837)
(('closure', 'non'), 0.000574248286450232)
(('notable', 'enough'), 0.0005676040914003946)
(('talk', 'page'), 0.0005654684572772326)
(('comment', 'added'), 0.0005533665305793144)
(('unsigned', 'comment'), 0.0005483833842919364)
(('preceding', 'unsigned'), 0.0005460104574884231)
(('looks', 'like'), 0.0004949925312128859)
(('google', 'search'), 0.00045916133647983426)
(('thanks', 'per'), 0.0004560765316352669)
(('free', 'images'), 0.00045109338534788886)
(('wikipedia', 'library'), 0.00044966962926578085)
(('news', 'newspapers'), 0.0004475339951426188)
(('images', 'wikipedia'), 0.0004472967024622675)
(('highbeam', 'free'), 0.0004468221171015648)
(('talk', 'contribs'), 0.00043377101968224136)
(('general', 'notability'), 0.0004266522392717013)
(('original', 'research'), 0.00041953345886116124)
(('sources', 'article'), 0.0004192961661808099)
(('could', 'find'), 0.0004181097027790532)
(('wikipedia', 'article'), 0.00039485502010462234)
(('claim', 'notability'), 0.00038441414216916356)
(('meet', 'notability'), 0.0003806174592835422)
(('reliable', 'independent'), 0.0003799055812424882)
```

```
In [10]: finder.apply_freq_filter(5)
          pmiscored = finder.score_ngrams(bigmes.pmi)
          for pscore in pmiscored[:50]:
              print(pscore)
```

Out:

```
(('bergstr', 'isacson'), 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

The NPS Chat Corpus was once created by Eric Forsyth, Jane Lin, and their thesis marketing consultant Craig Martell at the Department of Computer Science, Naval Postgraduate School in California in 2007. It is dispensed as phase of the Natural Language Toolkit (NLTK) for lookup and academic purposes. It includes extra than 10,000 posts from many online chat services. The posts are all geared up into 15 files. To hold the users of these chant rooms anonymous, all the usernames have been modified in the corpus to the form "UserN". The corpus is in addition anonymized by human beings who went through the posts to manually cast off any identifiable information. The documents of the corpus incorporate posts from a precise day and are divided by using age precise chatrooms. The naming conventions for the documents are as follows: date, age of the chat room, range of posts in that chatroom. For example, for the file, 8-26-30s_500posts.xml, there are 500 posts from the 30 12 months old chat room for 26th august 2006. All posts are from between October 19th, 2006 to November 9th, 2006.

```
In [12]: 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

In [13]: npswords = nps_chat.words()
nptokens = [x.lower() for x in npswords]
nps_tokens = [ x for x in npswords if x not in stopwords]

In [14]: tokensnps = [w for w in nps_tokens if not alpha_filter(w)]
npsDist = FreqDist(tokensnps)
Npsfreq50 = npsDist.most_common(50)

In [15]: npsraw = nps_chat.raw()
cleannps = clean(npsraw)
cleanNps = re.sub(r'\w+User', ' ', cleannps)

In [16]: npswords = nltk.regexp_tokenize(cleanNps, r'\w+|\w+|\w+|\b\w{3,}\b')
nptokens = [x.lower() for x in npswords]
nps_tokens = [ x for x in npswords if x not in stopwords]

In [22]: tokensnps = [w for w in nps_tokens if not alpha_filter(w)]
npsDist = FreqDist(tokensnps)
Npsfreq50 = npsDist.most_common(50)
for x in Npsfreq50:
    print(x)
```

```
Out:
('lol', 705)
('hey', 264)
('like', 156)
('chat', 142)
('good', 128)
('wanna', 107)
('lmao', 107)
('know', 103)
('get', 102)
```

```
( 'room', 98)
( 'one', 86)
( 'well', 80)
( 'back', 79)
( 'hiya', 78)
( 'dont', 75)
( 'yeah', 75)
( 'see', 75)
( 'hello', 71)
( 'yes', 70)
( 'want', 70)
( 'got', 68)
( 'everyone', 64)
( 'love', 62)
( 'guys', 59)
( 'talk', 56)
( 'right', 54)
( 'think', 54)
( 'nice', 52)
( 'would', 51)
( 'thanks', 50)
( 'anyone', 50)
( 'girls', 49)
( 'time', 49)
( 'thats', 45)
( 'never', 45)
( 'haha', 44)
( 'bye', 44)
( 'need', 43)
( 'make', 42)
( 'You', 42)
( 'whats', 41)
( 'really', 41)
( 'people', 41)
( 'girl', 40)
( 'much', 40)
( 'night', 40)
( 'work', 38)
( 'take', 37)
( 'hot', 37)
( 'gonna', 37)
```

```
In [18]: npsfinder = BigramCollocationFinder.from_words(tokensnps)
npsfreqscore = npsfinder.score_ngrams(bigmes.raw_freq)
for x in npsfreqscore[:50]:
    print (x)
```

Out:

```
(( '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)
```

```
In [19]: npsfinder.apply_freq_filter(5)
npspmiscored = npsfinder.score_ngrams(bigmes.pmi)
for x in npspmiscored[:50]:
    print (x)
```

Out:

```
(('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

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

Wikipedia language is substantially more nuanced and organized while NPS visit's language is filled with linguistic structure and spelling mistakes. In any case, this is justifiable as the sources are altogether different from one another. The NPS visit corpus is an accumulation of online messages from 2006 from different informing administrations. These stages don't have standard tenets and rules for posts though the Wikipedia talks page, as the site itself, has entirely set down rules for posting this limit the variety of language used on the site.

Apart from the rules and regulations, I feel the content itself performs a function in the language here. The NPS corpus has specific chats from a positive time of day. There is no connection between the chats. Wikipedia's dialogue page on the different hand is a better collective as all the users on that web page are discussing one topic: "To delete a Wikipedia web page or not" and the language used here will be headquartered around that subject as well.

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

I did use similar type of filters for both the texts. Removing the HTML tags and so on. Stop words have been removed from texts and the both had been transformed to lowercase. Punctuation was also removed, which drastically reduced the measurement of the NPS corpus due to the fact internet comments come with indescribable punctuation. Some comments were completely made up of punctuations.

c) 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 happy with the bigram for both the corpus. Since the language utilized in them two is unique, the bigram also replicates that distinction. Truly, the bigram records can be improved. There is dependably opportunity to get better. By modifying the separating and tokenization techniques another arrangement of bigrams can be acquired. It will rely upon what we further need to do with the bigram for our examination.

One issue I confronted was with the bigrams of the Wikipedia discussion pages. While the bigrams by occurring again and again were quite happening sets of words, the bigrams by PMI were somewhat unclear. I accept it is a direct result of the usernames in the record. I invested a ton of energy to tokenize better to increase important PMI bigrams. In the first place the bigrams included hexadecimal numbers, however by removing all non-alphabetic character this problem was solved. However, the bigram sets

look bad in English. Then again, the bigrams by recurrence and by PMI for the NPS chat corpus appeared well and good. I am additionally guessing that since interestingly happening sets of words get PMI high scores, the more impossible sets will be on the highest priority on the rundown.

Word and Name Puzzle

```
In [23]: letters = nltk.FreqDist('EGBDAFKJLMORCNST'.lower())
greenCell= 'M'
words = nltk.corpus.words.words()
answers = [x for x in words if len(x) >= 6 and greenCell in words and nltk.FreqDist(x) <= letters]
```

```
In [25]: len(answers)
```

```
Out[25]: 1440
```

1440 words can be made with the given criteria.