NLP Homework 1

Contents:

1. Wikipedia Discussions

The file was first read and split into tokens. Then, all the non-alphabetical tokens were removed. HTML tags were removed using Beautiful Soup. Then stop words were removed to generate the top 50 words by frequency. Then top 50 bigrams were generated according to frequency with stop words and without stop words. After applying the frequency filter of 5 the top 50 bigrams were generated using pointwise mutual information.

Top 50 words by frequency:

```
fd=FreqDist(new_words)
top50=fd.most_common(50)
for w in top50:
... print(w[0],w[1])
```

article 95713

wp 89720

sources 53579

notability 43464

notable 37705

coverage 31552

new 31264

please 27492

per 26943

add 26509

one 26173

comments 26048

thanks 25611

notice 25134

reliable 23584

wikipedia 23015

articles 22389

would 21531

gng 21041

fails 19121

subject 17533

also 17155

page 16548

find 15997

see 14935

significant 14814

list 14708

like 13826

independent 13148

enough 13111

even 13107

could 12349

source 12134

seems 11644

afd 11202

meet 11155

deletion 11072

think 10946

references 10793

delete 10044

may 9640

news 9484

found 9443

nothing 8556

google 8500

two 8490

search 8160

information 8092

keep 8048

books 7791

Top 50 words normalized by the length of document:

for w in top50:

... print(w[0],w[1]/len(tokens))

article 0.00939575718928122

wp 0.008807448674916795

sources 0.005259633220612649

notability 0.004266684676845559

notable 0.0037013469938446024

coverage 0.0030973319281205384

new 0.0030690601356731908

please 0.0026987781873697336

per 0.002644885083016977

add 0.00260228106245396

one 0.002569297304598721

comments 0.002557026561349004
thanks 0.002514128042947994
notice 0.0024673028867070743
reliable 0.002315145670410585
wikipedia 0.002259289247137874
articles 0.0021978373649432915
would 0.002113610983277235
gng 0.0020655096697383447
fails 0.001877031053422693
subject 0.0017211435311782898
also 0.001684036803591146
page 0.0016244500743705207
find 0.0015703606381257686
see 0.001466108403476174
significant 0.001454230324010448
list 0.001443824733734688
like 0.0013572423693646858
independent 0.0012906858579782212
enough 0.0012870537179763052
even 0.0012866610541923142
could 0.0012122512671260308
source 0.0011911455887365179
seems 0.0011430442751976277
afd 0.0010996549270666288
meet 0.0010950411276047353
deletion 0.0010868933540869232
think 0.0010745244448912086
references 0.0010595050551535552
delete 0.0009859787616012515
may 0.0009463197194181665

news 0.0009310058318425198

found 0.0009269810280566127

nothing 0.0008399078339566216

google 0.0008344105409807484

two 0.0008334288815207711

search 0.0008010341193415185

information 0.0007943588350136726

keep 0.0007900395333897722

books 0.0007648108852683543

Top 50 bigrams with stop words:

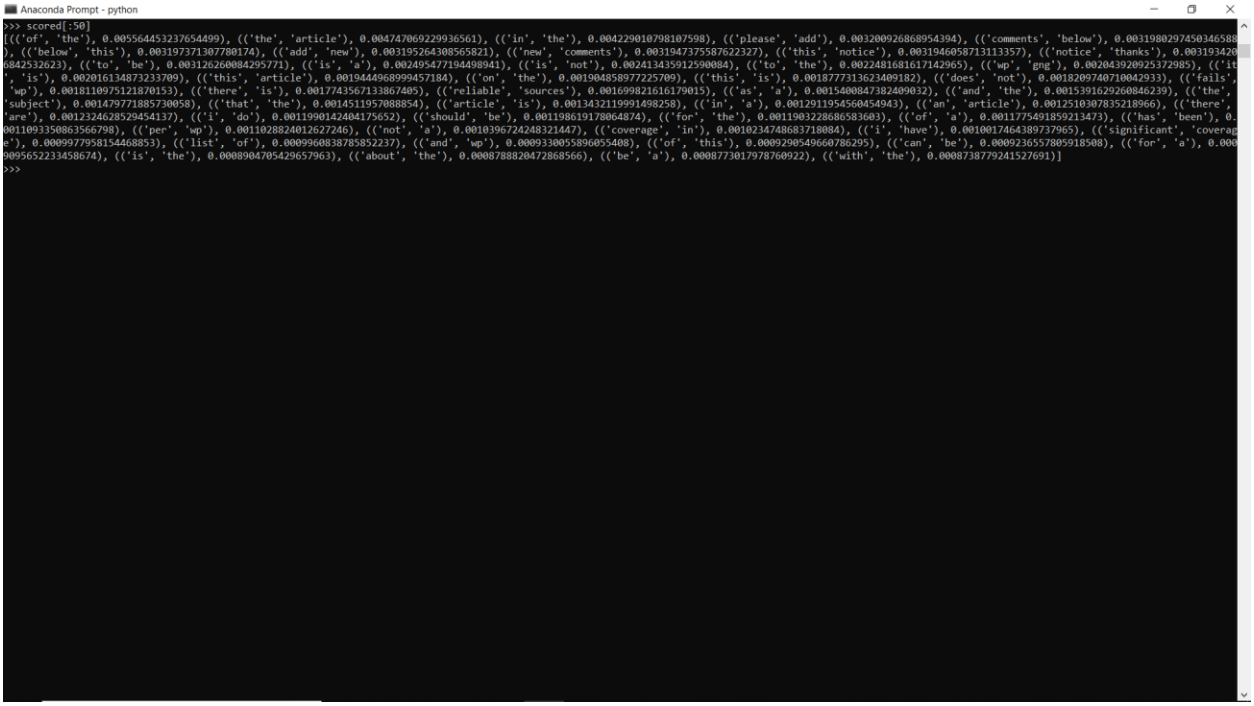
bigram_measures=nltk.collocations.BigramAssocMeasures()

finder=BigramCollocationFinder.from_words(words)

scored=finder.score_ngrams(bigram_measures.raw_freq)

```
scored[:50]
```

```
[(('of', 'the'), 0.005564453237654499), (('the', 'article'), 0.004747069229936561),
(('in', 'the'), 0.004229010798107598), (('please', 'add'), 0.003200926868954394),
 (('comments', 'below'), 0.0031980297450346588), (('below', 'this'), 0.003197371307780174),
(('add', 'new'), 0.003195264308565821), (('new', 'comments'), 0.0031947375587622327),
 (('this', 'notice'), 0.0031946058713113357), (('notice', 'thanks'), 0.0031934206842532623),
 (('to', 'be'), 0.003126260084295771), (('is', 'a'), 0.002495477194498941),
(('is', 'not'), 0.002413435912590084), (('to', 'the'), 0.0022481681617142965),
(('wp', 'gng'), 0.002043920925372985), (('it', 'is'), 0.002016134873233709),
(('this', 'article'), 0.0019444968999457184), (('on', 'the'), 0.001904858977225709),
 (('this', 'is'), 0.0018777313623409182), (('does', 'not'), 0.0018209740710042933),
(('fails', 'wp'), 0.0018110975121870153), (('there', 'is'), 0.0017743567133867405),
(('reliable', 'sources'), 0.001699821616179015), (('as', 'a'), 0.0015400847382409032),
 (('and', 'the'), 0.0015391629260846239), (('the', 'subject'), 0.001479771885730058),
(('that', 'the'), 0.0014511957088854), (('article', 'is'), 0.0013432119991498258),
(('in', 'a'), 0.0012911954560454943), (('an', 'article'), 0.0012510307835218966),
 (('there', 'are'), 0.0012324628529454137), (('i', 'do'), 0.0011990142404175652),
 (('should', 'be'), 0.001198619178064874), (('for', 'the'), 0.0011903228686583603),
(('of', 'a'), 0.0011775491859213473), (('has', 'been'), 0.0011093350863566798),
(('per', 'wp'), 0.0011028824012627246), (('not', 'a'), 0.0010396724248321447),
 (('coverage', 'in'), 0.0010234748683718084), (('i', 'have'), 0.0010017464389737965),
 (('significant', 'coverage'), 0.0009977958154468853), (('list', 'of'), 0.0009960838785852237),
 (('and', 'wp'), 0.0009330055896055408), (('of', 'this'), 0.0009290549660786295),
(('can', 'be'), 0.0009236557805918508), (('for', 'a'), 0.0009095652233458674),
(('is', 'the'), 0.0008904705429657963), (('about', 'the'), 0.0008788820472868566),
(('be', 'a'), 0.0008773017978760922), (('with', 'the'), 0.0008738779241527691)]
```



Top 50 bigrams without stop words:

finder.apply_word_filter(lambda w: w in stopwords)

scored=finder.score_ngrams(bigram_measures.raw_freq)

scored[:50]

[(('please', 'add'), 0.003200926868954394), (('add', 'new'), 0.003195264308565821),

(('new', 'comments'), 0.0031947375587622327), (('notice', 'thanks'), 0.0031934206842532623),

(('wp', 'gng'), 0.002043920925372985), (('fails', 'wp'), 0.0018110975121870153),

(('reliable', 'sources'), 0.001699821616179015), (('per', 'wp'), 0.0011028824012627246),

(('significant', 'coverage'), 0.0009977958154468853),

(('thanks', 'please'), 0.0007724785869620468), (('meet', 'wp'), 0.0007495649705059616),

(('wikipedia', 'articles'), 0.0005490049827897671), (('per', 'nom'), 0.0004007249130796980 4),

(('meets', 'wp'), 0.0003961158522983016), (('books', 'scholar'), 0.000343967621743073),

(('newspapers', 'books'), 0.000343704246841279),

(('scholar', 'highbeam'), 0.0003426507472341026),

(('highbeam', 'jstor'), 0.00034251905978320557),

(('establish', 'notability'), 0.00033896349860898544),

(('independent', 'sources'), 0.0003243461915594138),

(('reliable', 'source'), 0.00031499638254572384),

(('notability', 'guidelines'), 0.00031275769588047416),

(('find', 'sources'), 0.00031078238411701853),

(('independent', 'reliable'), 0.0003106506966661215),

(('comment', 'added'), 0.00030551488608113685),

(('unsigned', 'comment'), 0.00030419801157216645),

(('secondary', 'sources'), 0.0003038029492194753),

(('preceding', 'unsigned'), 0.00030288113706319604), (('ca', 'find'), 0.0002873420178573451),

(('notable', 'enough'), 0.00027970414570531666),(('talk', 'page'), 0.0002671938378700977),

(('closure', 'closure'), 0.00026100452767793673), (('could', 'find'), 0.0002604777778743486),

(('passes', 'wp'), 0.00024994278180258525),(('jstor', 'free'), 0.00024967940690079116),

(('free', 'images'), 0.0002495477194498941),

(('images', 'wikipedia'), 0.0002492843445481),

(('news', 'newspapers'), 0.00024862590729361484),

(('wikipedia', 'library'), 0.0002484942198427178),

(('looks', 'like'), 0.0002432267218068361),

(('google', 'search'), 0.00023479872494942543),

(('talk', 'contribs'), 0.0002305847265207201),

(('pass', 'wp'), 0.00023018966416802898),

(('general', 'notability'), 0.0002258439782884266),

(('thanks', 'per'), 0.00022386866652497096),

(('non', 'notable'), 0.00022162997985972126),

(('wp', 'rs'), 0.00022031310535075083),

(('wp', 'bio'), 0.00021925960574357452),

(('original', 'research'), 0.00019647767673838627),

(('third', 'party'), 0.00018409905635406436)]

Top 50 bigrams using PMI:

finder1=BigramCollocationFinder.from_words(words)

finder1.apply_freq_filter(5)

scored1=finder1.score_ngrams(bigram_measures.pmi)

scored1[:50]

[(('burr', 'steers'), 20.534450698226408), (('helsingin', 'sanomat'), 20.534450698226408),

(('hemorrhagic', 'conjunctivitis'), 20.534450698226408),

(('khyber', 'pakhtunkhwa'), 20.534450698226408),

(('pell', 'mell'), 20.534450698226408), (('phnom', 'penh'), 20.534450698226408),

(('putroe', 'neng'), 20.534450698226408), (('sunanda', 'pushkar'), 20.534450698226408),

(('xhulio', 'joka'), 20.271416292392615), (('ashleigh', 'lollie'), 20.271416292392612),

(('beent', 'agged'), 20.271416292392612), (('lorem', 'ipsum'), 20.271416292392612),

(('margarita', 'martirena'), 20.271416292392612), (('mong', 'kok'), 20.271416292392612),

(('suhas', 'gopinath'), 20.271416292392612), (('ulrike', 'ottinger'), 20.271416292392612),

(('vitalik', 'buterin'), 20.271416292392612), (('aqueduct', 'racetrack'), 20.049023871056168),

(('chal', 'jhoothey'), 20.049023871056168), (('mushtaq', 'pahalgami'), 20.049023871056168),

(('sebalu', 'lule'), 20.049023871056168), (('virtuti', 'militari'), 20.049023871056168),

(('xanthine', 'oxidase'), 20.049023871056168), (('abdulhadi', 'najjar'), 20.049023871056164),

(('guo', 'dongli'), 20.049023871056164), (('hidy', 'ochiai'), 20.049023871056164),

(('marlene', 'dietrich'), 20.049023871056164), (('sadman', 'sakibzz'), 20.049023871056164),

(('satish', 'rajwade'), 20.049023871056164), (('axl', 'hazarika'), 20.00838188655882),

(('lata', 'mangeshkar'), 20.00838188655882), (('akihiko', 'saito'), 19.85637879311377),

(('akira', 'toriyama'), 19.85637879311377), (('arugas', 'habosem'), 19.85637879311377),

(('chhota', 'bheem'), 19.85637879311377), (('copulatory', 'vocalizations'), 19.85637879311377),

(('jahaniyan', 'jahangasht'), 19.85637879311377), (('kylie', 'minogue'), 19.85637879311377),

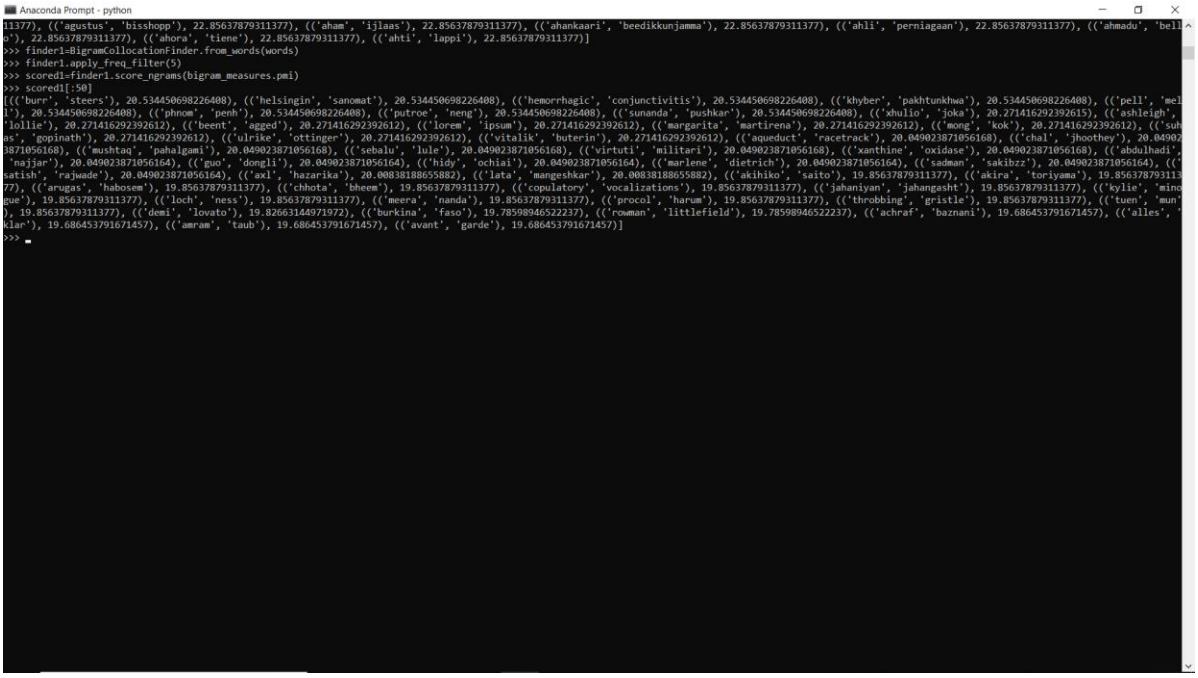
(('loch', 'ness'), 19.85637879311377), (('meera', 'nanda'), 19.85637879311377),

(('procol', 'harum'), 19.85637879311377), (('throbbing', 'gristle'), 19.85637879311377),

(('tuen', 'mun'), 19.85637879311377), (('demi', 'lovato'), 19.82663144971972),

(('burkina', 'faso'), 19.78598946522237), (('rowman', 'littlefield'), 19.78598946522237), (('achraf', 'baznani'), 19.686453791671457), (('alles', 'klar'), 19.686453791671457),

(('amram', 'taub'), 19.686453791671457), (('avant', 'garde'), 19.686453791671457)]



2. NPS Chat Corpus

The chat was first loaded from nltk and then split into tokens and all the non-alphabetical characters were remove. HTML tags are removed using Beautiful Soup. All the stop words are removed to generate top 50 words by frequency. Then top 50 bigrams are generated with stop words and without stop words using raw frequency. Then after applying a count filter of 5 top 500 bigrams are generated using pointwise mutual information.

Top 50 words by frequency:

fd_chat=FreqDist(chat_new_words)

top50_chat=fd_chat.most_common(50)

for w in top50_chat:

... print(w[0],w[1])

join 659

part 654

lol 369

hi 313

hey 170

u 137

chat 120
pm 116
na 109
like 96
wan 90
im 81
good 71
lmao 62
room 59
get 55
know 54
one 53
ok 53
ya 51
talk 50
yes 49
dont 48
want 48
hiya 48
oh 47
see 47
well 46
wb 46
guys 45
girls 44
back 42
go 40
yeah 40
anyone 39
got 37

hello 37

love 35

haha 34

hot 34

mode 34

everyone 33

time 33

right 31

would 30

people 30

song 30

take 29

really 29

need 29

Top 50 words normalized by the length of document:

for w in top50_chat:

... print(w[0],w[1]/len(chat_tokens))

join 0.023723810209518324
part 0.023543811649506804
lol 0.01328389372885017
hi 0.011267909856721147
hey 0.006119951040391677
u 0.0049319605443156455
chat 0.004319965440276478
pm 0.004175966592267262
na 0.003923968608251134
like 0.003455972352221182
wan 0.003239974080207358
im 0.0029159766721866226
good 0.0025559795521635825
lmao 0.002231982144142847
room 0.002123983008135935
get 0.001979984160126719
know 0.001943984448124415
one 0.0019079847361221111
ok 0.0019079847361221111
ya 0.0018359853121175031
talk 0.0017999856001151991
yes 0.001763985888112895
dont 0.001727986176110591
want 0.001727986176110591
hiya 0.001727986176110591
oh 0.001691986464108287
see 0.001691986464108287
well 0.001655986752105983
wb 0.001655986752105983
guys 0.001619987040103679

girls 0.001583987328101375

back 0.0015119879040967673

go 0.0014399884800921593

yeah 0.0014399884800921593

anyone 0.0014039887680898553

got 0.0013319893440852473

hello 0.0013319893440852473

love 0.0012599899200806393

haha 0.0012239902080783353

hot 0.0012239902080783353

mode 0.0012239902080783353

everyone 0.0011879904960760315

time 0.0011879904960760315

right 0.0011159910720714235

would 0.0010799913600691195

people 0.0010799913600691195

song 0.0010799913600691195

take 0.0010439916480668155

really 0.0010439916480668155

need 0.0010439916480668155

Top 50 bigrams:

finder_chat=BigramCollocationFinder.from_words(chat_new_words)

scored_chat=finder_chat.score_ngrams(bigram_measures.raw_freq)

for w in scored_chat[:50]:

... print(w)

(('part', 'join'), 0.011572160687716979)

(('part', 'part'), 0.007439246156389486)

(('wan', 'na'), 0.007439246156389486)

(('join', 'part'), 0.007356587865762936)

(('join', 'join'), 0.007025954703256737)

(('na', 'chat'), 0.004380889403207142)

(('hi', 'hi'), 0.003223673334435444)

(('join', 'hi'), 0.002231773846916846)
(('lol', 'join'), 0.002231773846916846)
(('part', 'hi'), 0.002066457265663746)
(('pm', 'u'), 0.0019011406844106464)
(('join', 'lol'), 0.0018184823937840966)
(('join', 'mode'), 0.0017358241031575467)
(('lol', 'hi'), 0.0017358241031575467)
(('lol', 'lol'), 0.0017358241031575467)
(('hi', 'part'), 0.0016531658125309968)
(('part', 'lol'), 0.0016531658125309968)
(('gon', 'na'), 0.001570507521904447)
(('chat', 'pm'), 0.0014787849231277897)
(('hi', 'lol'), 0.0014051909406513474)
(('lol', 'part'), 0.0014051909406513474)
(('hey', 'part'), 0.0013225326500247974)
(('hi', 'hey'), 0.0013225326500247974)
(('hi', 'join'), 0.0013225326500247974)
(('tryin', 'chat'), 0.0013225326500247974)
(('u', 'tryin'), 0.0013225326500247974)
(('guys', 'wan'), 0.0012398743593982477)
(('hey', 'hi'), 0.0011572160687716977)
(('part', 'hey'), 0.0011572160687716977)
(('want', 'chat'), 0.001074557778145148)
(('anyone', 'wan'), 0.000991899487518598)
(('join', 'hey'), 0.000991899487518598)
(('lol', 'hey'), 0.000991899487518598)
(('hi', 'hiya'), 0.0009092411968920483)
(('girls', 'wan'), 0.0008265829062654984)
(('la', 'la'), 0.0008265829062654984)
(('played', 'song'), 0.0008265829062654984)

(('player', 'listening'), 0.0008265829062654984)

(('song', 'music'), 0.0008265829062654984)

(('song', 'played'), 0.0008265829062654984)

(('chat', 'join'), 0.0007439246156389485)

(('dont', 'know'), 0.0007439246156389485)

(('join', 'wb'), 0.0007439246156389485)

(('na', 'talk'), 0.0007439246156389485)

(('pm', 'join'), 0.0007439246156389485)

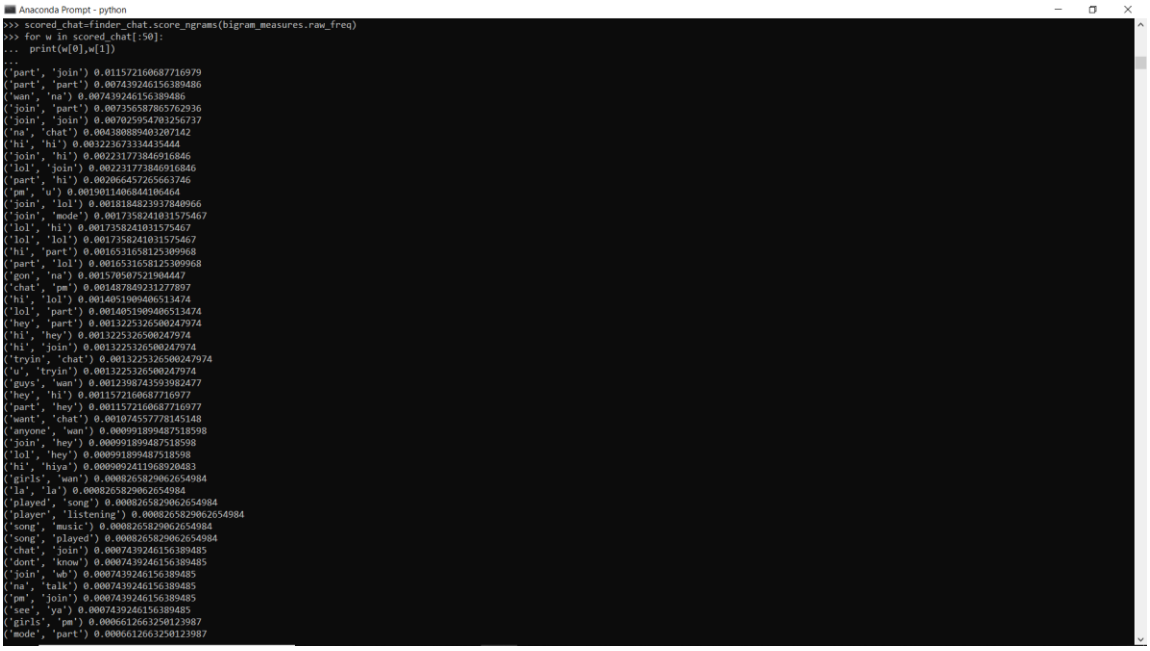
(('see', 'ya'), 0.0007439246156389485)

(('girls', 'pm'), 0.0006612663250123987)

(('mode', 'part'), 0.0006612663250123987)

(('part', 'u'), 0.0006612663250123987)

(('r', 'u'), 0.0006612663250123987)



finder_chat.apply_freq_filter(5)

scored_chat=finder_chat.score_ngrams(bigram_measures.pmi)

for w in scored_chat[:50]:

... print(w)

(('lez', 'gurls'), 10.562480945349314)

(('bi', 'lez'), 10.240552850461953)
(('bit', 'large'), 10.103049326712018)
(('n', 'e'), 9.299446539515522)
(('slaps', 'around'), 9.299446539515522)
(('la', 'la'), 9.269699196121472)
(('hug', 'watches'), 9.12952153807321)
(('player', 'listening'), 8.796946198986339)
(('played', 'song'), 8.518086825990864)
(('song', 'played'), 8.518086825990864)
(('around', 'bit'), 8.42497742159938)
(('song', 'music'), 7.807593443185848)
(('last', 'seen'), 7.599006821374429)
(('long', 'time'), 6.933124325269706)
(('gon', 'na'), 6.794296620572387)
(('wan', 'na'), 6.794296620572387)
(('tryin', 'chat'), 6.655590349740795)
(('u', 'tryin'), 6.464448862388787)
(('wants', 'talk'), 6.433197928404349)
(('wana', 'chat'), 6.1701635225705544)
(('gurls', 'pm'), 6.026428045109105)
(('im', 'bored'), 5.844119319210961)
(('welcome', 'room'), 5.679837895987474)
(('na', 'chat'), 5.615326479527068)
(('r', 'u'), 5.557558266780269)
(('see', 'ya'), 5.505391753142492)
(('guys', 'wan'), 5.485665348298484)
(('dont', 'know'), 5.392555943907002)
(('anyone', 'wan'), 5.370188130878548)
(('hot', 'guys'), 5.305093102656663)
(('wb', 'wb'), 5.100319533956446)

(('anyone', 'want'), 5.014044320653273)

(('girls', 'wan'), 4.9331243252697075)

(('want', 'talk'), 4.918624755574591)

(('oh', 'well'), 4.806258232502028)

(('hello', 'room'), 4.792312625245886)

(('want', 'chat'), 4.771067567160733)

(('na', 'talk'), 4.320365432239974)

(('girls', 'pm'), 4.245068331584445)

(('pm', 'u'), 4.130029823318226)

(('chat', 'pm'), 3.9675343560555394)

(('ok', 'im'), 3.816638582788853)

(('guys', 'pm'), 3.5345749487794293)

(('join', 'mode'), 3.503200871869687)

(('hey', 'everyone'), 3.4306239847405227)

(('hi', 'hiya'), 3.1469312163328347)

(('room', 'hey'), 2.855409460570929)

(('hi', 'everyone'), 2.5499960739456053)

(('right', 'lol'), 2.402735723789405)

(('ppl', 'part'), 2.3613372132962827)

3. Comparison

i) Both Wikipedia discussions and chat corpus are different in a manner where in chat corpus more short form of words and slangs are used while in Wikipedia discussions, it is more about meaningful content and use of words is also different. Both have many short forms which make no sense and would need a different approach to understand them. For example- gng, wp in Wikipedia discussions and wb in chat corpus.

ii) The processing options are somewhat the same. Removing the HTML tags, filtering out the words and then removing the stop words. This is the most general approach that we use when we are at a very early stage of language processing. A more specific approach can be to automatically recognize the slang words when we analyze chats. In this case it might be possible that some slangs might be removed because they are not considered as alphabets or words.

iii) There are a set of bigrams in both the cases that do not make sense. For example- (wp,rs), (closure, closure) in Wikipedia discussions and (lol,lol), (wan,na) in chat corpus. The list of bigrams could be more precise if we have detailed corpuses. The pointwise mutual information determines the correlation between two events x and y where x and y are considered as specific events. Also, it is the ratio of two events occurring together under a joint distribution to the two events occurring together assuming that they are independent. Raw frequency is just the number of times the events x and y occur together.

4. Word Puzzle

['abdomen', 'abelmosk', 'acneform', 'almond', 'almoner', 'almost', 'ambler', 'ambrose', 'amelcorn', 'amends', 'amlong', 'amober', 'amongst', 'amoret', 'angeldom', 'angstrom', 'antdom', 'armbone', 'armlet', 'asmoke', 'asmolder', 'atomerg', 'backmost', 'bakerdom', 'bankerdom', 'barksome', 'barmote', 'barsom', 'beardom', 'beastdom', 'becalm', 'beclamor', 'becram', 'bedamn', 'bedlam', 'bedman', 'befoam', 'beldam', 'beloam', 'beltman', 'bemask', 'bemoan', 'bemoat', 'bemock', 'bemolt', 'benmost', 'bergamot', 'bestorm', 'blamed', 'blamer', 'blastoderm', 'blockman', 'bogman', 'boltmaker', 'bregma', 'bromal', 'bromate', 'calmer', 'camber', 'cambrel', 'camlet', 'camstone', 'carmot', 'catdom', 'cembalo', 'ceroma', 'clamber', 'clamer', 'clamor', 'clerkdom', 'cloamen', 'cloamer', 'clogmaker', 'clomben', 'coagment', 'cobleman', 'codman', 'cogman', 'cokeman', 'colmar', 'comaker', 'comart', 'comate', 'combat', 'combater', 'combed', 'comber', 'comble', 'comrade', 'conamed', 'cormel', 'cotman', 'crambe', 'cramble', 'crambo', 'daemon', 'dambose', 'damner', 'damsel', 'damson', 'darksome', 'deform', 'demark', 'demast', 'democrat', 'dermal', 'dermoblast', 'dermol', 'desman', 'desmon', 'dockman', 'dockmaster', 'dogman', 'dolesman', 'dolman', 'dolmen', 'doment', 'dormant', 'dreamt', 'earldom', 'embank', 'embargo', 'embark', 'enamor', 'endmost', 'engram', 'enjamb', 'enmask', 'entomb', 'escambron', 'estmark', 'fabledom', 'fadmonger', 'famble', 'fandom', 'farmost', 'femora', 'femoral', 'flamberg', 'flamed', 'flamen', 'flamenco', 'flamer', 'flatdom', 'flockman', 'flockmaster', 'flogmaster', 'flotsam', 'foamer', 'foeman', 'fogman', 'fogram', 'foment', 'foramen', 'foreman', 'foremast', 'forgeman', 'forkman', 'formable', 'formagen', 'formal', 'formant', 'format', 'formate', 'formed', 'formel', 'fotmal', 'fragment', 'framed', 'freakdom', 'frogman', 'gambeson', 'gambet', 'gamble', 'gambler', 'gambol', 'gambrel', 'gamont', 'garment', 'gemsbok', 'geomant', 'germal', 'german', 'germon', 'gladsome', 'gnomed', 'godmaker', 'gomart', 'gomeral', 'gormed', 'graftdom', 'jambone', 'jambstone', 'jarldom', 'jasmone', 'jermonal', 'jetsam', 'jobman', 'jobmaster', 'katmon', 'kerslam', 'lambent', 'lamber', 'lambert', 'lament', 'landstorm', 'larksome', 'leafdom', 'legman', 'lemnad', 'lockerman', 'lockman', 'lockram', 'locksman', 'lodesman', 'lodgeman', 'lodgment', 'loftman', 'loftsman', 'logman', 'lombard', 'loment', 'mackle', 'macled', 'macron', 'madstone', 'maestro', 'magnes', 'magnet', 'magneto', 'magnetod', 'malfed', 'malter', 'maltose', 'manbot', 'mandore', 'mandrel', 'mangel', 'manger', 'mangle', 'mangler', 'manlet', 'manred', 'mantel', 'manter', 'mantes', 'mantle', 'mantled', 'marble', 'marbled', 'marbles', 'marcel', 'marengo', 'margent', 'marked', 'market', 'marled', 'marlock', 'martel', 'marten', 'mascled', 'mascot', 'masked', 'masker', 'masoned', 'masoner', 'masted', 'master', 'matfelon', 'matron', 'medlar', 'megadont', 'megaron', 'megaton', 'megotalc', 'melano', 'melosa', 'melton', 'menald', 'menfolk', 'mensal', 'mental', 'mentor', 'mercal', 'merfold', 'merfolk', 'merlon', 'mescal', 'mesobar', 'mobster', 'mockable', 'mocker', 'modena', 'moderant', 'modern', 'modest', 'molder', 'molecast', 'molest', 'molten', 'molter', 'monase', 'monaster', 'moneral', 'monger', 'mongler', 'mongrel', 'mongst', 'monkcraft', 'monster', 'montage', 'morale', 'morals', 'morate', 'mordant', 'mordent', 'morgan', 'morgen', 'morned', 'morsal', 'morsel', 'mortal', 'mosker', 'nearmost', 'neckmold', 'nemoral', 'normal', 'normated', 'omental', 'oreman', 'orgasm', 'osmate', 'ostmark', 'radome', 'ramble', 'rambong', 'rament', 'ramose', 'ramson', 'randem', 'random', 'ransom', 'recomb', 'remand', 'remask', 'remast', 'remock', 'remold', 'retomb', 'rockman', 'rodman', 'rodsman', 'romance', 'salmon',

'salmonet', 'samlet', 'sarment', 'scamble', 'scambler', 'scamler', 'scleroma', 'scramble', 'scream', 'seamrog', 'seldom', 'selfdom', 'semblant', 'semola', 'serfdom', 'sermon', 'sjambok', 'smacker', 'smalter', 'smarten', 'smocker', 'smoked', 'smoker', 'smolder', 'socman', 'sokeman', 'solemn', 'somber', 'sombre', 'sorema', 'stagedom', 'stamen', 'stardom', 'stockman', 'storeman', 'stormable', 'stream', 'stroam', 'stroma', 'stromal', 'stromb', 'strome', 'tamber', 'tambor', 'tandem', 'tarsome', 'telamon', 'temblor', 'termon', 'tombac', 'tombal', 'tormen', 'transmold', 'transom', 'transomed', 'tromba', 'trombe', 'tsardom', 'almost', 'market', 'normal']