CORPUS STATISTICS AND PYTHON PROGRAMMING

Komal Gujarathi

NATURAL LANGUAGE PROCESSING

LU XIAO

SYRACUSE UNIVERSITY

Table of Contents

Corpus Statistics and Python Programming Homework

Use a modified version of the State of the Union Addresses dataset. It contains three files, which are 1) the addresses delivered between 1790 and 1860, 2) the address delivered between 194 6 and 2016, and 3) a policy description of the Project Gutenberg Ebook ('state_union_policy').

Task 1. Analysis of State of the Union Addresses dataset: Description

First, review the dataset and describe the characteristics of this corpus briefly such as its history, the naming convention of its files, number of documents it contains, the related policy, etc. (no more than 450 words)

Document 1 - State of the Union Addresses dataset: Part1
Title of this corpus - 'Complete State of the Union Address from 1790 to 1860'.
Description - Contains the collection of around 70 documents, each document is the address by the President of United States to its Fellow Citizens of the Senate and House of Representatives from George Washington to James Buchanan. That is one speech is delivered per year.
Policy - There are certain policies that author of this text is abiding to as per the Gutenberg Policies. Some of these policies include, header cannot be modified, there are certain rights and restrictions on how this file can be used, etc.
Naming Convention for each file - name of the president, followed by the text 'State of the Union Address' followed by the date (Month Day, Year format).

Document 2 - State of the Union Addresses dataset: Part2
Title of this corpus - 'Complete State of the Union Address from 1946 to 2016'.
Description - Contains the collection of around 70 documents, each document is the address by the President of United States to its Fellow Citizens of the Senate and House of Representatives from Harry Truman to Barack Obama. That is one speech is delivered per year.
Policy - There are certain policies that author of this text is abiding to as per the Gutenberg Policies. Some of these policies include, header cannot be modified, there are certain rights and restrictions on how this file can be used, etc.
Naming Convention for each file - name of the president, followed by the text 'State of the Union Address' followed by the date (Month Day, Year format).

Task 2. Analysis of State of the Union Addresses dataset: Part1

Note: You will decide how to process the words, i.e. decide on tokenization and whether to use all lower case, use or modify the stop word list, or lemmatization. Briefly state why you chose the processing options that you did.

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

In order to **normalize the document**, I have tokenized the document, and then removed all the non-alpha characters from the document and then converted all the tokens in the lower case. Doing this, there is a uniformity in both the documents, to calculate the various frequency measures among both the texts.

I have chosen these measures in order to normalize mainly because –

1. Lowercase – I wanted both the words – “FIVE” and “five” to be treated the same way while making comparisons.
2. Alphabetic measure – I wanted to ignore all the non-alphabetic characters like comma, hyphen, comma, numbers from the text in order to make uniform comparison.
3. Modify the stop word list – I did not modify the stop words list in order to make find out the bigram frequencies and Mutual Information scores. This is because, if we remove the stop words before performing this operation, our sentences in the corpus will be changed and our corpus will make no sense then. I have explained this with example in the below sections.
4. Lemmatization – I didn’t perform lemmatization on the corpus, but it could be done if we are interested in analyzing the data by having the canonical form of each of the words described in the corpus. But I don’t think that was necessary for the analysis of this document.

Later, I used the FreqDist package from the nltk tool kit to find out most frequent words.

Top 50 words by frequency

```
('states', 2725)
('government', 2220)
('united', 1864)
('may', 1562)
('congress', 1500)
('upon', 1455)
('would', 1381)
('public', 1375)
('country', 1163)
('great', 1073)
('made', 1061)
('state', 1045)
('last', 911)
('war', 834)
('present', 812)
('time', 808)
('people', 786)
('year', 785)
('power', 744)
('citizens', 723)
('subject', 711)
('shall', 694)
('without', 663)
('union', 643)
('act', 627)
('treaty', 624)
('one', 620)
('part', 618)
('mexico', 605)
```

```
('general', 601)
('every', 590)
('treasury', 590)
('necessary', 575)
('constitution', 557)
('new', 548)
('duty', 529)
('foreign', 519)
('two', 510)
('commerce', 506)
('nations', 502)
('peace', 501)
('system', 494)
('laws', 492)
('duties', 488)
('within', 479)
('law', 477)
('us', 463)
('interests', 451)
('interest', 444)
('amount', 443)
```

2.2. List the top 50 bigrams by frequencies

Another important characteristic of corpus is to look at the pairs of words that are frequently collocated, that is, they occur in sequence called a bigram.
In order to find out the bigram frequency distribution, general nltk bigram function is used. Nltk bigram function returns the generator, which we need to convert into list to see the desired output. Also, I am only using the original set of tokens (only normalized for case, and alphabetical characters). I am not using the stopwords filter for calculating the bigram frequency, because this might not give us the correct results. (e.g. for the sentence in the given corpus "I feel much satisfaction", if after applying the stopwords filter, suppose 'much' is removed, them our bigrams will have 'feel satisfaction', which is not a bigram in the original data set).
Once I get the result of the bigram frequency distribution, I then apply the stop word frequency filter on it, in order to make the output more meaningful. Also, I have applied the filter such that the words less than equal to length 4 are removed.

Top 50 Bigrams by frequencies

```
(('great', 'britain'), 0.0005366645709327583)
(('last', 'session'), 0.00047398841666323907)
(('public', 'debt'), 0.0003505947379451231)
(('fiscal', 'year'), 0.00028204269421283645)
(('union', 'address'), 0.00028204269421283645)
(('public', 'lands'), 0.0002546218767199218)
(('present', 'year'), 0.0002076147610177824)
(('fellow', 'citizens'), 0.00018998709262948012)
(('general', 'government'), 0.0001821525733457902)
(('year', 'ending'), 0.0001821525733457902)
(('british', 'government'), 0.00018019394352486773)
```

```
(('federal', 'government'), 0.00016648353477841042)
(('annual', 'message'), 0.00015864901549472052)
(('public', 'service'), 0.00015669038567379804)
(('last', 'annual'), 0.00014689723656918565)
(('public', 'money'), 0.00013514545764365082)
(('indian', 'tribes'), 0.0001292695681808834)
(('mexican', 'government'), 0.0001273109383599609)
(('treasury', 'notes'), 0.0001273109383599609)
(('commercial', 'intercourse'), 0.00012339367871811596)
(('address', 'december'), 0.00010968326997165862)
(('favorable', 'consideration'), 0.00010772464015073614)
(('naval', 'force'), 0.00010772464015073614)
(('central', 'america'), 9.989012086704625e-05)
(('present', 'session'), 9.989012086704625e-05)
(('french', 'government'), 9.793149104612377e-05)
(('friendly', 'relations'), 9.401423140427882e-05)
(('ending', 'june'), 9.009697176243387e-05)
(('existing', 'laws'), 8.813834194151139e-05)
(('good', 'faith'), 8.813834194151139e-05)
(('american', 'citizens'), 8.422108229966645e-05)
(('foreign', 'nations'), 8.422108229966645e-05)
(('taken', 'place'), 8.422108229966645e-05)
(('last', 'year'), 8.226245247874397e-05)
(('past', 'year'), 8.226245247874397e-05)
(('september', 'last'), 8.226245247874397e-05)
(('american', 'people'), 8.03038226578215e-05)
(('military', 'force'), 8.03038226578215e-05)
(('ensuing', 'year'), 7.834519283689902e-05)
(('minister', 'plenipotentiary'), 7.834519283689902e-05)
(('slave', 'trade'), 7.638656301597654e-05)
(('spanish', 'government'), 7.638656301597654e-05)
(('present', 'fiscal'), 7.246930337413159e-05)
(('great', 'extent'), 6.855204373228664e-05)
(('public', 'moneys'), 6.855204373228664e-05)
(('government', 'would'), 6.659341391136417e-05)
(('foreign', 'powers'), 6.46347840904417e-05)
(('present', 'condition'), 6.46347840904417e-05)
(('public', 'interest'), 6.46347840904417e-05)
(('would', 'seem'), 6.46347840904417e-05)
```

2.3. List the top 50 bigrams by their Mutual Information Scores (using min freq 5)

Mutual information score is nothing but the association ratio (Number of times the words occur together as opposed to number of times they occur individually). Technically, the original information theoretic definition allows the two words to be in either order, but the association definition defined by Church and Hanks require the words to be in order from left to right wherever they appear in the window.

In nltk, the mutual information score is given by a function for Pointwise Mutual Information, where there is a version without the window.

Top 50 Bigrams by their Mutual Information Score after applying Minimum frequency of 5

```
(('bona', 'fide'), 15.539910019165042)
(('posse', 'comitatus'), 15.539910019165042)
(('punta', 'arenas'), 15.539910019165042)
(('ballot', 'box'), 15.276875613331246)
(('del', 'norte'), 15.276875613331246)
(('millard', 'fillmore'), 15.276875613331246)
(('clayton', 'bulwer'), 14.861838114052404)
(('guadalupe', 'hidalgo'), 14.691913112610091)
(('porto', 'rico'), 14.691913112610091)
(('writ', 'mandamus'), 14.598803708218608)
(('franklin', 'pierce'), 14.539910019165042)
(('la', 'plata'), 14.402406495415105)
(('vera', 'cruz'), 14.276875613331246)
(('entangling', 'alliances'), 14.206486285439848)
(('seminaries', 'learning'), 14.013841207497453)
(('gun', 'boats'), 13.884558190552486)
(('nucleus', 'around'), 13.861838114052404)
(('ruler', 'universe'), 13.861838114052404)
(('costa', 'rica'), 13.8618381140524)
(('santa', 'anna'), 13.774375272802065)
(('santa', 'fe'), 13.774375272802065)
(('van', 'buren'), 13.774375272802065)
(('project', 'gutenberg'), 13.774375272802063)
(('sublime', 'porte'), 13.732555097107436)
(('tea', 'coffee'), 13.613910600608818)
(('martin', 'van'), 13.604450271359752)
(('ad', 'valorem'), 13.53991001916504)
(('beacons', 'buoys'), 13.402406495415105)
(('water', 'witch'), 13.402406495415105)
(('quincy', 'adams'), 13.402406495415104)
(('statute', 'book'), 13.338276157995391)
(('buenos', 'ayres'), 13.276875613331244)
(('indiana', 'illinois'), 13.139372089581311)
(('de', 'facto'), 13.128483773438575)
(('franking', 'privilege'), 13.106950611888934)
(('rocky', 'mountains'), 13.054483191994798)
(('andrew', 'jackson'), 12.972021031802825)
(('retired', 'list'), 12.916979668244863)
(('sooner', 'later'), 12.876945006442611)
(('circulating', 'medium'), 12.81744399469395)
(('intent', 'meaning'), 12.798828316526604)
(('th', 'jefferson'), 12.774375272802065)
(('john', 'quincy'), 12.774375272802063)
(('precious', 'metals'), 12.715481583748494)
(('thomas', 'jefferson'), 12.686912431551724)
(('lake', 'erie'), 12.633019423556524)
(('almighty', 'god'), 12.604450271359752)
(('john', 'tyler'), 12.604450271359752)
(('san', 'jacinto'), 12.576435895190155)
(('san', 'juan'), 12.576435895190155)
```

Task 3. Analysis of State of the Union Addresses dataset: Part2

In this problem, you will analyze the "state_union_part2.txt" that is available for download in the Assignment folder in the course web site.

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

In order to **normalize the document**, I have tokenized the document, and then removed all the non-alpha characters from the document and then converted all the tokens in the lower case. Doing this, there is a uniformity in both the documents, to calculate the various frequency measures among both the texts.

I have chosen these measures inorder to normalize mainly because –

1. lowercase – I wanted both the words – "FIVE" and "five" to be treated the same way while making comparisions.
2. alphabetic measure – I wanted to ignore all the non-alphabetic characters like comma, hyphen, comma, numbers from the text inorder to make uniform comparision.
3. Modify the stop word list – I did not modify the stop words list in order to make find out the bigram frequencies and Mutual Information scores. This is because, if we remove the stop words before performing this operation, our sentences in the corpus will be changed and our corpus will make no sense then. I have explained this with example in the below sections.
4. Lemmatization – I didn't perform lemmatization on the corpus, but it could be done if we are interested in analyzing the data by having the canonical form of each of the words described in the corpus. But I don't think that was necessary for the analysis of this document.

Later, I used the FreqDist package from the nltk tool kit to find out most frequent words.

Top 50 words by frequency

```
('must', 1628)
('people', 1506)
('world', 1490)
('new', 1441)
('america', 1271)
('year', 1265)
('congress', 1230)
('us', 1216)
('government', 1111)
('years', 1111)
('american', 950)
('nation', 861)
('one', 804)
('every', 780)
('make', 778)
('work', 754)
('federal', 744)
```

```
('time', 741)
('states', 711)
('americans', 688)
('help', 686)
('security', 685)
('war', 674)
('economic', 671)
('peace', 668)
('united', 651)
('nations', 645)
('also', 639)
('program', 638)
('country', 630)
('national', 609)
('economy', 588)
('great', 583)
('last', 572)
('many', 563)
('free', 558)
('need', 554)
('first', 553)
('let', 549)
('would', 548)
('state', 520)
('tax', 514)
('know', 507)
('million', 507)
('freedom', 503)
('budget', 501)
('health', 489)
('future', 475)
('system', 463)
('programs', 462)
```

3.2. List the top 50 bigrams by frequencies

Another important characteristic of corpus is to look at the pairs of words that are frequently collocated, that is, they occur in sequence called a bigram.
In order to find out the bigram frequency distribution, general nltk bigram function is used. Nltk bigram function returns the generator, which we need to convert into list to see the desired output. Also, I am only using the original set of tokens (only normalized for case, and alphabetical characters). I am not using the stopwords filter for calculating the bigram frequency, because this might not give us the correct results. (e.g. for the sentence in the given corpus "I feel much satisfaction", if after applying the stopwords filter, suppose 'much' is removed, them our bigrams will have 'feel satisfaction', which is not a bigram in the original data set).
Once I get the result of the bigram frequency distribution, I then apply the stopword frequency filter on it, in order to make the output more meaningful. Also, I have applied the filter such that the words less than equal to length 4 are removed.

Top 50 Bigrams by frequencies

```
(('american', 'people'), 0.000564390057289133)
(('last', 'year'), 0.000531329551841234)
(('fiscal', 'year'), 0.0004392324295220868)
(('federal', 'government'), 0.00043450950017238693)
(('social', 'security'), 0.0004274251061478371 5)
(('health', 'care'), 0.00042034071212328737)
(('years', 'ago'), 0.00038255727732568846)
(('union', 'address'), 0.0003258821251292901 7)
(('billion', 'dollars'), 0.00030699040773049074)
(('million', 'dollars'), 0.0002999060137059409 6)
(('soviet', 'union'), 0.0002951830843562411)
(('free', 'world'), 0.00025503818488 37923)
(('every', 'american'), 0.00023378500281014296)
(('economic', 'growth'), 0.0002219776794358933)
(('middle', 'east'), 0.00021489328541134353)
(('make', 'sure'), 0.00020780889138679375)
(('free', 'nations'), 0.00019836303268739404)
(('first', 'time'), 0.00019127863866284423)
(('four', 'years'), 0.00019127863866284423)
(('armed', 'forces'), 0.00017710985061374467)
(('must', 'continue'), 0.00017474838593889474)
(('world', 'war'), 0.00017474838593889474)
(('work', 'together'), 0.0001700254565891949)
(('foreign', 'policy'), 0.00016530252723949502)
(('vice', 'president'), 0.00015821813321494524)
(('next', 'years'), 0.0001558566685400953)
(('national', 'security'), 0.0001464108098406956)
(('must', 'also'), 0.00014404934516584566)
(('address', 'january'), 0.00014168788049099575)
(('human', 'rights'), 0.0001393264158161458)
(('health', 'insurance'), 0.00013696495114129588)
(('fellow', 'americans'), 0.00013224202179159603)
(('fellow', 'citizens'), 0.00013224202179159603)
(('past', 'year'), 0.00013224202179159603)
(('civil', 'rights'), 0.00012751909244189616)
(('young', 'people'), 0.00012751909244189616)
(('past', 'years'), 0.0001227961630921963)
(('private', 'sector'), 0.0001227961630921963)
(('local', 'governments'), 0.00012043469841734637)
(('nuclear', 'weapons'), 0.00012043469841734637)
(('interest', 'rates'), 0.00011571176906764651)
(('next', 'year'), 0.00011571176906764651)
(('balanced', 'budget'), 0.00011335030439279659)
(('must', 'make'), 0.00011335030439279659)
(('high', 'school'), 0.00011098883971794665)
(('minimum', 'wage'), 0.00011098883971794665)
(('white', 'house'), 0.00010862737504309673)
(('cold', 'war'), 0.00010154298101854694)
(('middle', 'class'), 0.00010154298101854694)
(('last', 'years'), 9.918151634369702e-05)
```

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

Mutual information scores is nothing but the association ratio. Technically, the original information theoretic definition allows the two words to be in either order, but the association definition defined by Church and Hanks require the words to be in order from left to right wherever they appear in the window.

In nltk, the mutual information score is given by a function for Pointwise Mutual Information, where there is a version without the window.

Top 50 Bigrams by their Mutual Information Score after applying Minimum frequency of 5

```
(('el', 'salvador'), 15.164265341881517)
(('ladies', 'gentlemen'), 15.164265341881517)
(('bin', 'laden'), 14.94187292054507)
(('saudi', 'arabia'), 14.94187292054507)
(('sam', 'rayburn'), 14.749227842602672)
(('jimmy', 'carter'), 14.42729974771531)
(('endowed', 'creator'), 14.316268435326567)
(('northern', 'ireland'), 14.164265341881517)
(('gerald', 'ford'), 14.097151146022979)
(('floor', 'appears'), 14.012262248436468)
(('iron', 'curtain'), 13.94187292054507)
(('grass', 'roots'), 13.901230936047725)
(('thomas', 'jefferson'), 13.785753718627788)
(('sons', 'daughters'), 13.749227842602675)
(('red', 'tape'), 13.749227842602673)
(('jill', 'biden'), 13.678838514711277)
(('lyndon', 'johnson'), 13.661765001352334)
(('barack', 'obama'), 13.66176500135233)
(('teen', 'pregnancy'), 13.57930284116036)
(('abraham', 'lincoln'), 13.49183999991002)
(('mom', 'dad'), 13.456446093374828)
(('empowerment', 'zones'), 13.356910419823912)
(('william', 'clinton'), 13.327764074164396)
(('ronald', 'reagan'), 13.289796223965373)
(('synthetic', 'fuels'), 13.275296654270264)
(('greece', 'turkey'), 13.204907326378866)
(('elementary', 'secondary'), 13.122788705905355)
(('intercontinental', 'ballistic'), 13.003273465209212)
(('feeding', 'hungry'), 12.967868129078013)
(('river', 'basins'), 12.8912468474751)
(('status', 'quo'), 12.891246847475099)
(('commander', 'chief'), 12.856143046519184)
(('prime', 'minister'), 12.842337246994152)
(('nationwide', 'radio'), 12.801695262496807)
(('reported', 'floor'), 12.764334734992882)
(('radio', 'television'), 12.749227842602675)
(('introduced', 'thomas'), 12.670276501207852)
(('project', 'gutenberg'), 12.661765001352334)
```

```
(('dwight', 'eisenhower'), 12.643433178580077)
(('al', 'qaeda'), 12.619944825657706)
(('al', 'qaida'), 12.619944825657704)
(('richard', 'nixon'), 12.602871312298765)
(('saddam', 'hussein'), 12.57930284116036)
(('harry', 'truman'), 12.539774476973722)
(('supreme', 'court'), 12.525226168404565)
(('carbon', 'pollution'), 12.469119923409936)
(('baby', 'boom'), 12.463825623740426)
(('persian', 'gulf'), 12.437169991739372)
(('capitol', 'introduced'), 12.396711427881888)
(('sides', 'aisle'), 12.356910419823912)
```

Task 4. Comparison

Please compare the analysis results from question 2 and question 3.

4.1. How are state_union_part1 and state_union_part2 similar or different in the use of the language, based on your results? Why?

From the bigram frequency, it seems like the speakers in part1 were more interested in talking about its relations with other governments. For example, the bigrams like ('british', 'government') ('federal', 'government') ('Mexican',' government'), ('Indian', 'tribe'), ('French', 'government'), ('friendly', 'relations'), all these imply that the speakers intended to talk more about its relations with other Nations. While in the second part, we can notice that the bigrams that are frequently noticed include ('american', 'people'), ('social', 'security'), ('health', 'care'), ('billion', 'dollars'), ('million', 'dollars'), ('economic', 'growth'), ('national', 'security'),('health', 'insurance'), which indicate that speakers later in part2 were more interested in the welfare of the American people, and most of their talks involved taking about the health, money, social security, economic growth.

Also, as we can see, that there are around 5.5 lac tokens whereas in the second corpus, there are 4.8 lac tokens despite the fact that both these documents have around 70 speeches each. Also, if we carefully see, there is a lot of repetition (looking at the frequency count) of few words like 'states', 'government', 'united' in the first document, while the amount of repetition is relatively less in the second part. Thus, we can conclude that leaders in part2 started giving the speeches with the reduced number of words and with increased variety(vocabulary) in the words used rather than using the same words again and again.

4.2. Are there any problems with the word or bigram lists that you found? Could you get a better list of bigrams?

In the word frequency of part1, ('interests', 451), ('interest', 444) occur, these are the two different frequent wor ds which boil down to the same stem word. So, I think, additional filter criteria like 'Stemming', 'Lemmatizat ion' can be enforced while normalization to avoid such redundancies in word frequencies.

Also, words like 'let', 'know', 'may', 'also', etc, which are seen in the freq words in part1 and part2, could be ig nored as stopwords. I have tried to append few words to the stop words list but these words are very common i n normal speech, so can be appended to stopwords list.

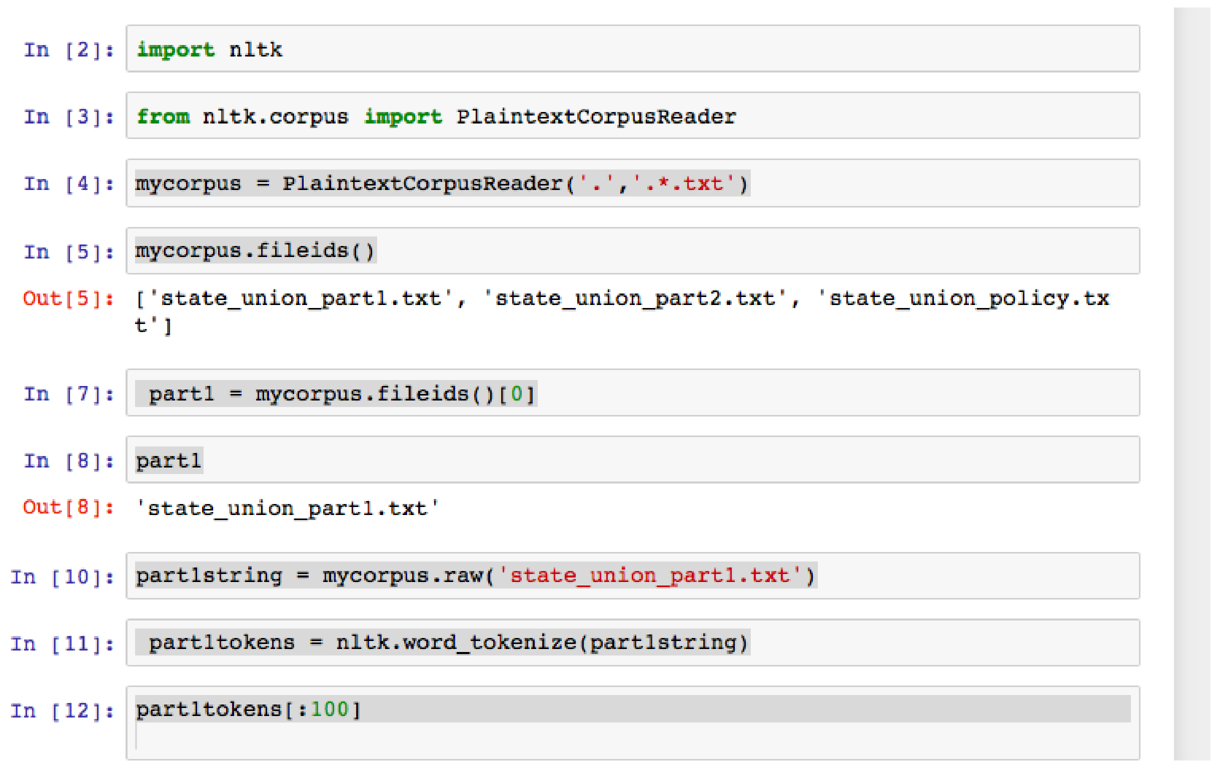
4.3. How are the top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Information?

With Mutual Information Score, we are generally getting tokens like ('almighty', 'god'),('john','tyler'), ('barack','obama'),('th','jeferreson'),('andrew','jackson'). These tokens indicate that PIM gives tokens which generally involve the proper nouns, like famous personalities, or words like ('ladies','gentlemen') which has the high frequency to occur together than individually. While with Bigram Frequency, we see that we see the topic specific words, like which mostly talks about the different governments, relations with other people, war and welfare. Bigram frequency also is that indication of the most frequently used words in the speeches by leaders as opposed to their identity individually.

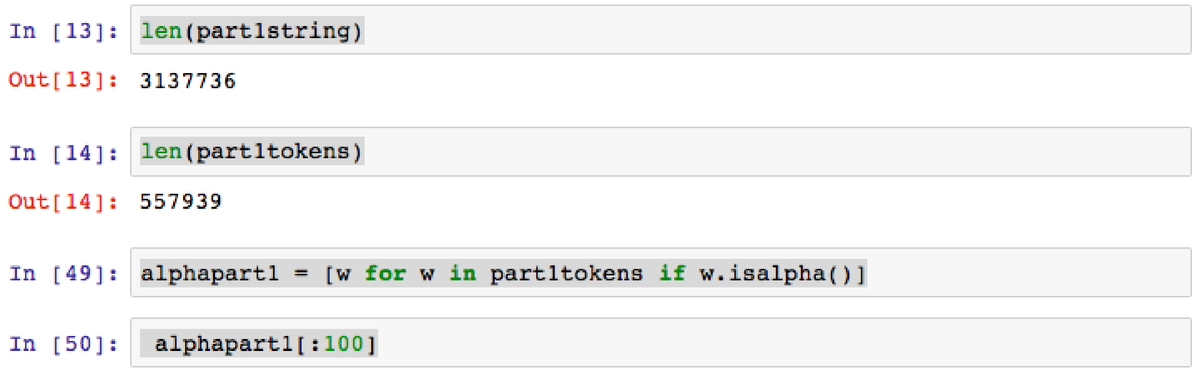
Similarly, with the part2 corpus, ('bin', 'laden'), 14.94187292054507), ('saudi', 'arabia'), 14.94187292054507), ('sam', 'rayburn'), 14.749227842602672), ('jimmy', 'carter'), 14.42729974771531), ('richard', 'nixon'), 12.602871312298765), ('saddam', 'hussein'), 12.57930284116036), ('harry', 'truman') all these show that PMI mostly gives us the words proper nouns especially the names of the well-known personalities.
While bigram frequency in part2 is mostly focused on the welfare of the public like health insurance, social security, American people.

Appendix

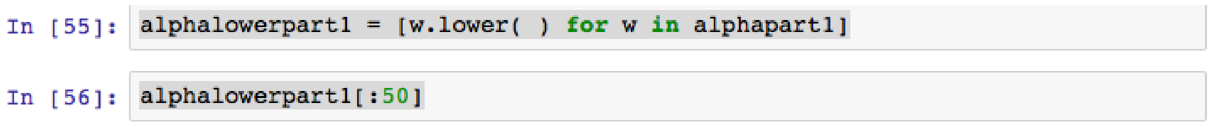
Output & Python Processing Corpus 1



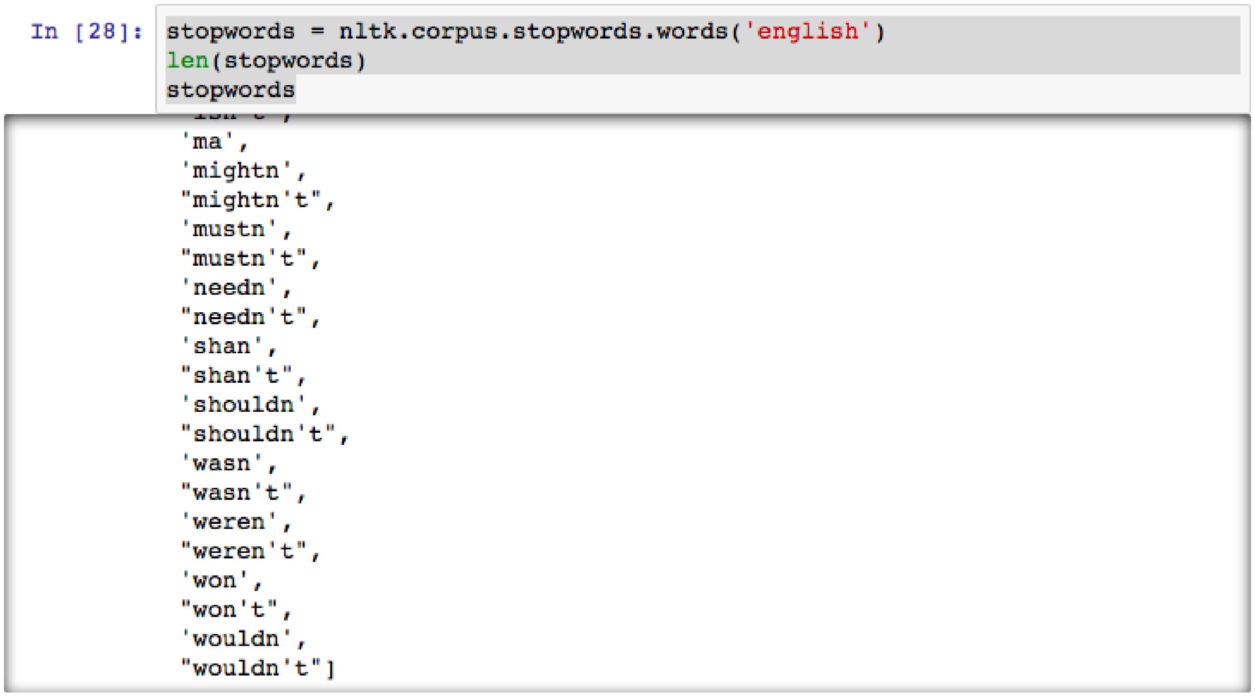
```
In [2]: import nltk

In [3]: from nltk.corpus import PlaintextCorpusReader

In [4]: mycorpus = PlaintextCorpusReader('.','.*.txt')

In [5]: mycorpus.fileids()

Out[5]: ['state_union_part1.txt', 'state_union_part2.txt', 'state_union_policy.tx
        t']

In [7]:  part1 = mycorpus.fileids()[0]

In [8]: part1

Out[8]: 'state_union_part1.txt'

In [10]: part1string = mycorpus.raw('state_union_part1.txt')

In [11]:  part1tokens = nltk.word_tokenize(part1string)

In [12]: part1tokens[:100]
```



```
In [13]: len(part1string)

Out[13]: 3137736

In [14]: len(part1tokens)

Out[14]: 557939

In [49]: alphapart1 = [w for w in part1tokens if w.isalpha()]

In [50]:  alphapart1[:100]
```

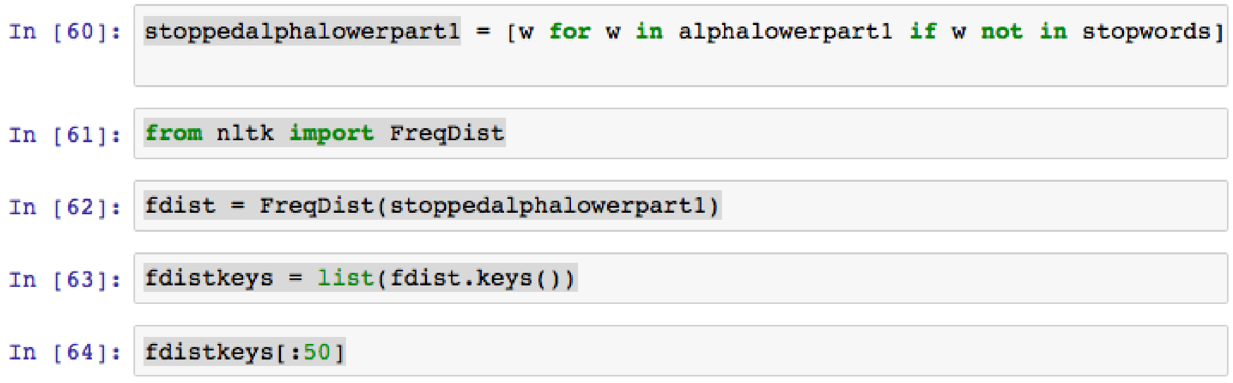


```
In [55]: alphalowerpart1 = [w.lower( ) for w in alphapart1]

In [56]: alphalowerpart1[:50]
```

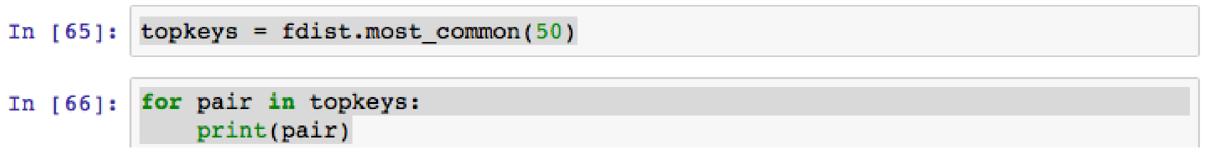


```
In [28]: stopwords = nltk.corpus.stopwords.words('english')
         len(stopwords)
         stopwords
```

```
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```



```
In [60]: stoppedalphalowerpart1 = [w for w in alphalowerpart1 if w not in stopwords]

In [61]: from nltk import FreqDist

In [62]: fdist = FreqDist(stoppedalphalowerpart1)

In [63]: fdistkeys = list(fdist.keys())

In [64]: fdistkeys[:50]
```



```
In [65]: topkeys = fdist.most_common(50)

In [66]: for pair in topkeys:
             print(pair)
```

```
('states', 2725)
('government', 2220)
('united', 1864)
('may', 1562)
('congress', 1500)
('upon', 1455)
('would', 1381)
('public', 1375)
```

```
('country', 1163)
('great', 1073)
('made', 1061)
('state', 1045)
('last', 911)
('war', 834)
('present', 812)
('time', 808)
('people', 786)
('year', 785)
('power', 744)
('citizens', 723)
('subject', 711)
('shall', 694)
('without', 663)
('union', 643)
('act', 627)
('treaty', 624)
('one', 620)
('part', 618)
('mexico', 605)
('general', 601)
('every', 590)
('treasury', 590)
('necessary', 575)
('constitution', 557)
('new', 548)
('duty', 529)
('foreign', 519)
('two', 510)
('commerce', 506)
('nations', 502)
('peace', 501)
('system', 494)
('laws', 492)
('duties', 488)
('within', 479)
('law', 477)
('us', 463)
('interests', 451)
('interest', 444)
('amount', 443)
```

Bigram Freq Dist

Another way to look for interesting characterizations of a corpus is to look at pairs of words that are frequently collocated, that is, they occur in a sequence called a bigram.

In [67]:
```
from nltk.collocations import *
```

In [68]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
```

We start by making an object called a BigramCollocationFinder. The finder then allows us to call other functions to filter the bigrams that it collected and to give scores to the bigrams. As a first step, we can score the bigrams by frequency. Be sure to use the original tokens from the document and not the filtered tokens that you might create to look at word frequencies. For example, the bigrams of the sentence "run after the fox!" include "run after", "after the", and "the fox". If you use the original tokens you will obtain these bigrams. If you filter the sentence with the stop words list, then you may not have "the" after the filtering. Then your bigrams will have "after fox" – which is not a bigram in the original dataset.

In [76]:
```
finder = BigramCollocationFinder.from_words(alphalowerpart1)
```

In [77]:
```
scored = finder.score_ngrams(bigram_measures.raw_freq)
```

In [78]:
```
type(scored)
first = scored[0]
type(first)
first
```

Out[78]:
```
(('of', 'the'), 0.023139252704378124)
```

In [79]:
```
for bscore in scored[:30]:
    print (bscore)
```

```
(('of', 'the'), 0.023139252704378124)
```

Applying filters now

Before that, adding the 'united', 'states' in the list of stop words, because these words will definitely occur frequently in this document.

In [90]:
```
stopwords.append("united")
stopwords.append("states")
```

In [91]:
```
finder.apply_word_filter(lambda w: w in stopwords)
scored = finder.score_ngrams(bigram_measures.raw_freq)
for bscore in scored[:50]:
    print (bscore)
```

def apply_ngram_filter(self, fn):

```
    # "Removes candidate ngrams (w1, w2, ...) where fn(w1, w2, ...)
evaluates to True.
```

In [94]:
```
finder.apply_ngram_filter(lambda w1, w2: len(w1) < 4)
scored = finder.score_ngrams(bigram_measures.raw_freq)
for bscore in scored[:50]:
    print (bscore)
```

```
(('great', 'britain'), 0.0005366645709327583)
(('last', 'session'), 0.00047398841666323907)
(('public', 'debt'), 0.0003505947379451231)
(('fiscal', 'year'), 0.00028204269421283645)
(('union', 'address'), 0.00028204269421283645)
(('public', 'lands'), 0.0002546218767199218)
(('present', 'year'), 0.0002076147610177824)
(('fellow', 'citizens'), 0.00018998709262948012)
(('general', 'government'), 0.0001821525733457902)
(('year', 'ending'), 0.0001821525733457902)
(('british', 'government'), 0.00018019394352486773)
(('federal', 'government'), 0.00016648353477841042)
(('annual', 'message'), 0.00015864901549472052)
(('public', 'service'), 0.00015669038567379804)
(('last', 'annual'), 0.00014689723656918565)
(('public', 'money'), 0.00013514545764365082)
(('indian', 'tribes'), 0.0001292695681808834)
(('mexican', 'government'), 0.0001273109383599609)
(('treasury', 'notes'), 0.0001273109383599609)
(('commercial', 'intercourse'), 0.00012339367871811596)
(('address', 'december'), 0.00010968326997165862)
(('favorable', 'consideration'), 0.00010772464015073614)
(('naval', 'force'), 0.00010772464015073614)
(('central', 'america'), 9.989012086704625e-05)
```

```
(('present', 'session'), 9.989012086704625e-05)
(('french', 'government'), 9.793149104612377e-05)
(('friendly', 'relations'), 9.401423140427882e-05)
(('ending', 'june'), 9.009697176243387e-05)
(('existing', 'laws'), 8.813834194151139e-05)
(('good', 'faith'), 8.813834194151139e-05)
(('american', 'citizens'), 8.422108229966645e-05)
(('foreign', 'nations'), 8.422108229966645e-05)
(('taken', 'place'), 8.422108229966645e-05)
(('last', 'year'), 8.226245247874397e-05)
(('past', 'year'), 8.226245247874397e-05)
(('september', 'last'), 8.226245247874397e-05)
(('american', 'people'), 8.03038226578215e-05)
(('military', 'force'), 8.03038226578215e-05)
(('ensuing', 'year'), 7.834519283689902e-05)
(('minister', 'plenipotentiary'), 7.834519283689902e-05)
(('slave', 'trade'), 7.638656301597654e-05)
(('spanish', 'government'), 7.638656301597654e-05)
(('present', 'fiscal'), 7.246930337413159e-05)
(('great', 'extent'), 6.855204373228664e-05)
(('public', 'moneys'), 6.855204373228664e-05)
(('government', 'would'), 6.659341391136417e-05)
(('foreign', 'powers'), 6.46347840904417e-05)
(('present', 'condition'), 6.46347840904417e-05)
(('public', 'interest'), 6.46347840904417e-05)
(('would', 'seem'), 6.46347840904417e-05)
```

```
### Mutual Information and other scores
Recall that Mutual Information is a score introduced in the paper by
Church and Hanks, where they defined it as an Association Ratio. Note that
technically the original information theoretic definition of mutual
information allows the two words to be in either order, but that the
association ratio defined by Church and Hanks requires the words to be in
order from left to right wherever they appear in the window

In NLTK, the mutual information score is given by a function for Pointwise
Mutual Information, where this is the version without the window.
```

```
In [99]: finder2 = BigramCollocationFinder.from_words(stoppedalphalowerpart1)
```

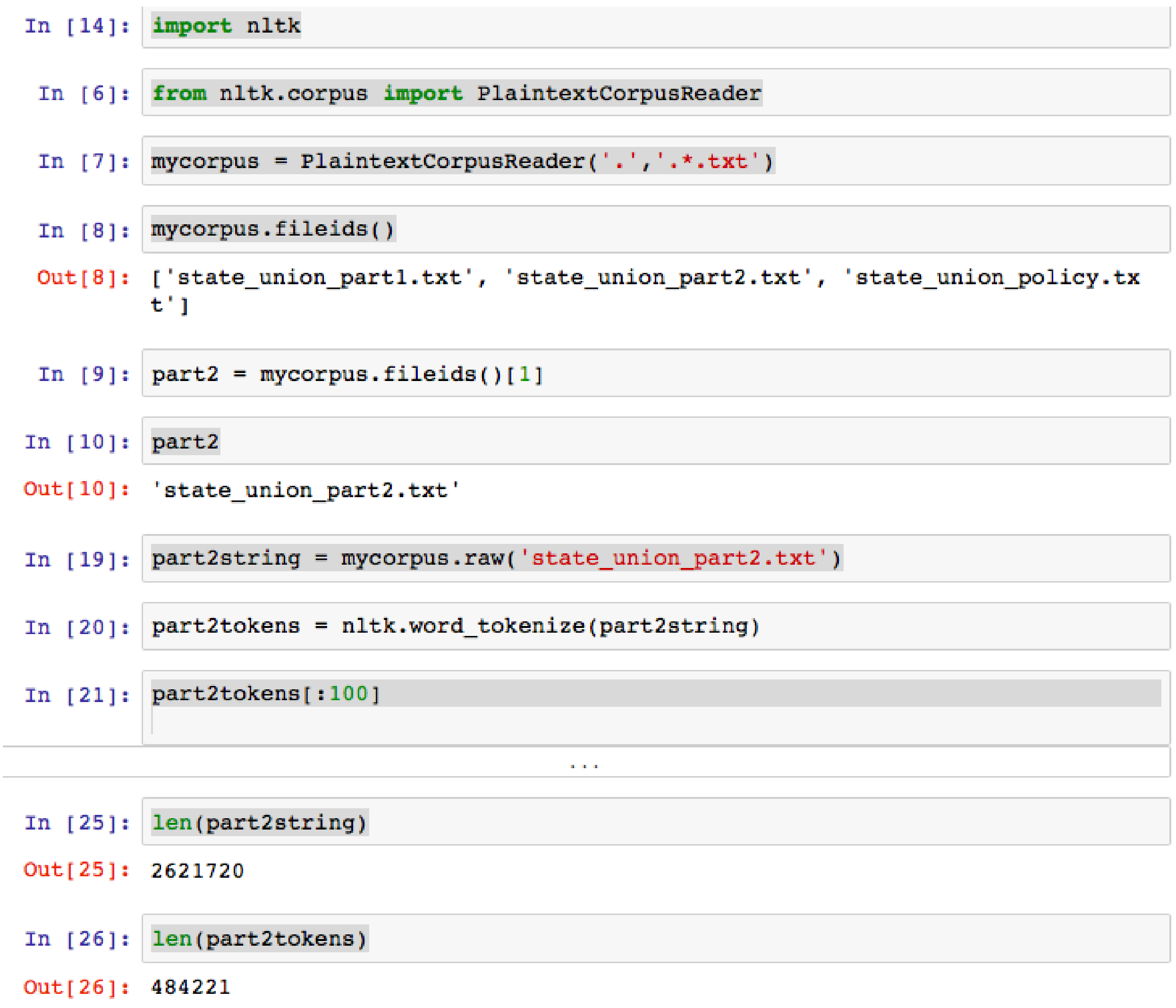
```
In [100]: finder2.apply_freq_filter(5)
```

```
In [101]: scored = finder2.score_ngrams(bigram_measures.pmi)
          for bscore in scored[:50]:
              print (bscore)
```

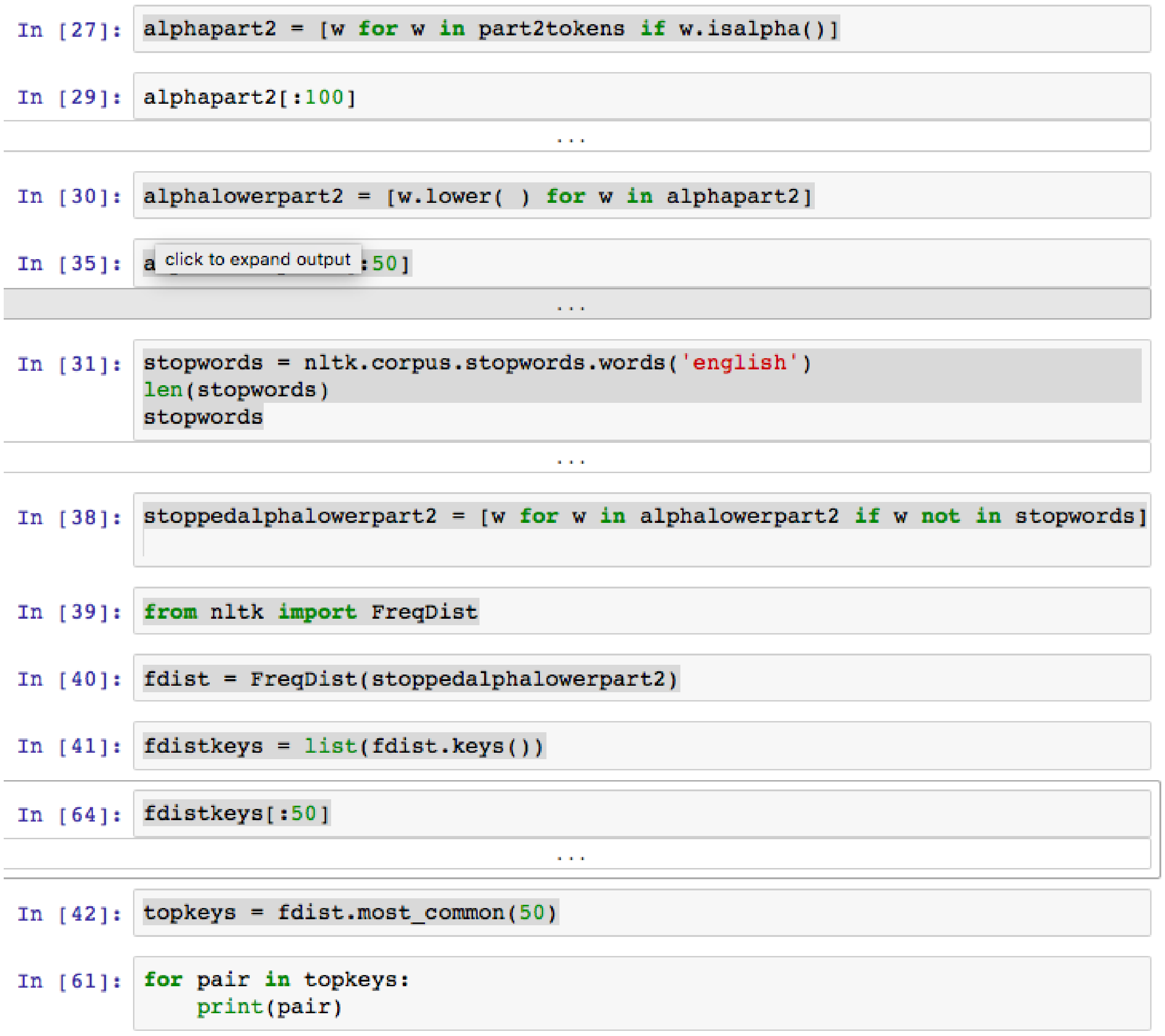
```
(('bona', 'fide'), 15.539910019165042)
```

```
(('posse', 'comitatus'), 15.539910019165042)
(('punta', 'arenas'), 15.539910019165042)
(('ballot', 'box'), 15.276875613331246)
(('del', 'norte'), 15.276875613331246)
(('millard', 'fillmore'), 15.276875613331246)
(('clayton', 'bulwer'), 14.861838114052404)
(('guadalupe', 'hidalgo'), 14.691913112610091)
(('porto', 'rico'), 14.691913112610091)
(('writ', 'mandamus'), 14.598803708218608)
(('franklin', 'pierce'), 14.539910019165042)
(('la', 'plata'), 14.402406495415105)
(('vera', 'cruz'), 14.276875613331246)
(('entangling', 'alliances'), 14.206486285439848)
(('seminaries', 'learning'), 14.013841207497453)
(('gun', 'boats'), 13.884558190552486)
(('nucleus', 'around'), 13.861838114052404)
(('ruler', 'universe'), 13.861838114052404)
(('costa', 'rica'), 13.8618381140524)
(('santa', 'anna'), 13.774375272802065)
(('santa', 'fe'), 13.774375272802065)
(('van', 'buren'), 13.774375272802065)
(('project', 'gutenberg'), 13.774375272802063)
(('sublime', 'porte'), 13.732555097107436)
(('tea', 'coffee'), 13.613910600608818)
(('martin', 'van'), 13.604450271359752)
(('ad', 'valorem'), 13.53991001916504)
(('beacons', 'buoys'), 13.402406495415105)
(('water', 'witch'), 13.402406495415105)
(('quincy', 'adams'), 13.402406495415104)
(('statute', 'book'), 13.338276157995391)
(('buenos', 'ayres'), 13.276875613331244)
(('indiana', 'illinois'), 13.139372089581311)
(('de', 'facto'), 13.128483773438575)
(('franking', 'privilege'), 13.106950611888934)
(('rocky', 'mountains'), 13.054483191994798)
(('andrew', 'jackson'), 12.972021031802825)
(('retired', 'list'), 12.916979668244863)
(('sooner', 'later'), 12.876945006442611)
(('circulating', 'medium'), 12.81744399469395)
(('intent', 'meaning'), 12.798828316526604)
(('th', 'jefferson'), 12.774375272802065)
(('john', 'quincy'), 12.774375272802063)
(('precious', 'metals'), 12.715481583748494)
(('thomas', 'jefferson'), 12.686912431551724)
(('lake', 'erie'), 12.633019423556524)
(('almighty', 'god'), 12.604450271359752)
(('john', 'tyler'), 12.604450271359752)
(('san', 'jacinto'), 12.576435895190155)
(('san', 'juan'), 12.576435895190155)
```

Output & Python Processing Corpus 2



```
In [14]: import nltk

In [6]: from nltk.corpus import PlaintextCorpusReader

In [7]: mycorpus = PlaintextCorpusReader('.','.*.txt')

In [8]: mycorpus.fileids()

Out[8]: ['state_union_part1.txt', 'state_union_part2.txt', 'state_union_policy.tx
        t']

In [9]: part2 = mycorpus.fileids()[1]

In [10]: part2

Out[10]: 'state_union_part2.txt'

In [19]: part2string = mycorpus.raw('state_union_part2.txt')

In [20]: part2tokens = nltk.word_tokenize(part2string)

In [21]: part2tokens[:100]

...

In [25]: len(part2string)

Out[25]: 2621720

In [26]: len(part2tokens)

Out[26]: 484221
```



```
In [27]: alphapart2 = [w for w in part2tokens if w.isalpha()]

In [29]: alphapart2[:100]
...

In [30]: alphalowerpart2 = [w.lower( ) for w in alphapart2]

In [35]: a                       :50]
...

In [31]: stopwords = nltk.corpus.stopwords.words('english')
         len(stopwords)
         stopwords
...

In [38]: stoppedalphalowerpart2 = [w for w in alphalowerpart2 if w not in stopwords]

In [39]: from nltk import FreqDist

In [40]: fdist = FreqDist(stoppedalphalowerpart2)

In [41]: fdistkeys = list(fdist.keys())

In [64]: fdistkeys[:50]
...

In [42]: topkeys = fdist.most_common(50)

In [61]: for pair in topkeys:
             print(pair)
```

```
('must', 1628)
('people', 1506)
('world', 1490)
('new', 1441)
('america', 1271)
('year', 1265)
('congress', 1230)
('us', 1216)
('government', 1111)
('years', 1111)
('american', 950)
```

```
('nation', 861)
('one', 804)
('every', 780)
('make', 778)
('work', 754)
('federal', 744)
('time', 741)
('states', 711)
('americans', 688)
('help', 686)
('security', 685)
('war', 674)
('economic', 671)
('peace', 668)
('united', 651)
('nations', 645)
('also', 639)
('program', 638)
('country', 630)
('national', 609)
('economy', 588)
('great', 583)
('last', 572)
('many', 563)
('free', 558)
('need', 554)
('first', 553)
('let', 549)
('would', 548)
('state', 520)
('tax', 514)
('know', 507)
('million', 507)
('freedom', 503)
('budget', 501)
('health', 489)
('future', 475)
('system', 463)
('programs', 462)
```

Bigram Freq Dist

Another way to look for interesting characterizations of a corpus is to look at pairs of words that are frequently collocated, that is, they occur in a sequence called a bigram.

```
In [44]: from nltk.collocations import *
```

```
In [45]: bigram_measures = nltk.collocations.BigramAssocMeasures()
```

We start by making an object called a BigramCollocationFinder. The finder then allows us to call other functions to filter the bigrams that it collected and to give scores to the bigrams. As a first step, we can score the bigrams by frequency. Be sure to use the original tokens from the document and not the filtered tokens that you might create to look at word frequencies. For example, the bigrams of the sentence "run after the fox!" include "run after", "after the", and "the fox". If you use the original tokens you will obtain these bigrams. If you filter the sentence with the stop words list, then you may not have "the" after the filtering. Then your bigrams will have "after fox" – which is not a bigram in the original dataset.

```
In [46]: finder = BigramCollocationFinder.from_words(alphalowerpart2)
```

```
In [47]: scored = finder.score_ngrams(bigram_measures.raw_freq)
```

```
In [48]: type(scored)
         first = scored[0]
         type(first)
         first

Out[48]: (('of', 'the'), 0.007686567516636518)
```

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

Applying filters now

Before that, adding the 'united', 'states' in the list of stop words, because these words will definitely occur frequently in this document.

```
In [51]: stopwords.append("united")
         stopwords.append("states")
```

```
In [52]: finder.apply_word_filter(lambda w: w in stopwords)
         scored = finder.score_ngrams(bigram_measures.raw_freq)
         for bscore in scored[:50]:
             print (bscore)
```

...

def apply_ngram_filter(self, fn):

```
    # "Removes candidate ngrams (w1, w2, ...) where fn(w1, w2, ...)
evaluates to True.
```

```
In [53]: finder.apply_ngram_filter(lambda w1, w2: len(w1) < 4)
         scored = finder.score_ngrams(bigram_measures.raw_freq)
         for bscore in scored[:50]:
             print (bscore)
```

```
(('american', 'people'), 0.000564390057289133)
(('last', 'year'), 0.000531329551841234)
(('fiscal', 'year'), 0.0004392324295220868)
(('federal', 'government'), 0.00043450950017238693)
(('social', 'security'), 0.00042742510614783715)
(('health', 'care'), 0.00042034071212328737)
(('years', 'ago'), 0.00038255727732568846)
(('union', 'address'), 0.00032588212512929017)
(('billion', 'dollars'), 0.00030699040773049074)
(('million', 'dollars'), 0.00029990601370594096)
(('soviet', 'union'), 0.0002951830843562411)
(('free', 'world'), 0.0002550381848837923)
(('every', 'american'), 0.00023378500281014296)
(('economic', 'growth'), 0.0002219776794358933)
(('middle', 'east'), 0.00021489328541134353)
(('make', 'sure'), 0.00020780889138679375)
(('free', 'nations'), 0.00019836303268739404)
(('first', 'time'), 0.00019127863866284423)
(('four', 'years'), 0.00019127863866284423)
(('armed', 'forces'), 0.00017710985061374467)
```

```
(('must', 'continue'), 0.00017474838593889474)
(('world', 'war'), 0.00017474838593889474)
(('work', 'together'), 0.0001700254565891949)
(('foreign', 'policy'), 0.00016530252723949502)
(('vice', 'president'), 0.00015821813321494524)
(('next', 'years'), 0.0001558566685400953)
(('national', 'security'), 0.0001464108098406956)
(('must', 'also'), 0.00014404934516584566)
(('address', 'january'), 0.00014168788049099575)
(('human', 'rights'), 0.0001393264158161458)
(('health', 'insurance'), 0.00013696495114129588)
(('fellow', 'americans'), 0.00013224202179159603)
(('fellow', 'citizens'), 0.00013224202179159603)
(('past', 'year'), 0.00013224202179159603)
(('civil', 'rights'), 0.00012751909244189616)
(('young', 'people'), 0.00012751909244189616)
(('past', 'years'), 0.0001227961630921963)
(('private', 'sector'), 0.0001227961630921963)
(('local', 'governments'), 0.00012043469841734637)
(('nuclear', 'weapons'), 0.00012043469841734637)
(('interest', 'rates'), 0.00011571176906764651)
(('next', 'year'), 0.00011571176906764651)
(('balanced', 'budget'), 0.00011335030439279659)
(('must', 'make'), 0.00011335030439279659)
(('high', 'school'), 0.00011098883971794665)
(('minimum', 'wage'), 0.00011098883971794665)
(('white', 'house'), 0.00010862737504309673)
(('cold', 'war'), 0.00010154298101854694)
(('middle', 'class'), 0.00010154298101854694)
(('last', 'years'), 9.918151634369702e-05)
```

Mutual Information and other scores

Recall that Mutual Information is a score introduced in the paper by Church and Hanks, where they defined it as an Association Ratio. Note that technically the original information theoretic definition of mutual information allows the two words to be in either order, but that the association ratio defined by Church and Hanks requires the words to be in order from left to right wherever they appear in the window

In NLTK, the mutual information score is given by a function for Pointwise Mutual Information, where this is the version without the window.

```
In [58]: finder2 = BigramCollocationFinder.from_words(stoppedalphalowerpart2)
```

```
In [59]: finder2.apply_freq_filter(5)
```

```
In [62]: scored = finder2.score_ngrams(bigram_measures.pmi)
         for bscore in scored[:50]:
             print (bscore)
```

```
(('el', 'salvador'), 15.164265341881517)
(('ladies', 'gentlemen'), 15.164265341881517)
(('bin', 'laden'), 14.94187292054507)
(('saudi', 'arabia'), 14.94187292054507)
(('sam', 'rayburn'), 14.749227842602672)
(('jimmy', 'carter'), 14.42729974771531)
(('endowed', 'creator'), 14.316268435326567)
(('northern', 'ireland'), 14.164265341881517)
(('gerald', 'ford'), 14.097151146022979)
(('floor', 'appears'), 14.012262248436468)
(('iron', 'curtain'), 13.94187292054507)
(('grass', 'roots'), 13.901230936047725)
(('thomas', 'jefferson'), 13.785753718627788)
(('sons', 'daughters'), 13.749227842602675)
(('red', 'tape'), 13.749227842602673)
(('jill', 'biden'), 13.678838514711277)
(('lyndon', 'johnson'), 13.661765001352334)
(('barack', 'obama'), 13.66176500135233)
(('teen', 'pregnancy'), 13.57930284116036)
(('abraham', 'lincoln'), 13.49183999991002)
(('mom', 'dad'), 13.456446093374828)
(('empowerment', 'zones'), 13.356910419823912)
(('william', 'clinton'), 13.327764074164396)
(('ronald', 'reagan'), 13.289796223965373)
(('synthetic', 'fuels'), 13.275296654270264)
(('greece', 'turkey'), 13.204907326378866)
(('elementary', 'secondary'), 13.122788705905355)
(('intercontinental', 'ballistic'), 13.003273465209212)
(('feeding', 'hungry'), 12.967868129078013)
(('river', 'basins'), 12.8912468474751)
```

```
(('status', 'quo'), 12.891246847475099)
(('commander', 'chief'), 12.856143046519184)
(('prime', 'minister'), 12.842337246994152)
(('nationwide', 'radio'), 12.801695262496807)
(('reported', 'floor'), 12.764334734992882)
(('radio', 'television'), 12.749227842602675)
(('introduced', 'thomas'), 12.670276501207852)
(('project', 'gutenberg'), 12.661765001352334)
(('dwight', 'eisenhower'), 12.643433178580077)
(('al', 'qaeda'), 12.619944825657706)
(('al', 'qaida'), 12.619944825657704)
(('richard', 'nixon'), 12.602871312298765)
(('saddam', 'hussein'), 12.57930284116036)
(('harry', 'truman'), 12.539774476973722)
(('supreme', 'court'), 12.525226168404565)
(('carbon', 'pollution'), 12.469119923409936)
(('baby', 'boom'), 12.463825623740426)
(('persian', 'gulf'), 12.437169991739372)
(('capitol', 'introduced'), 12.396711427881888)
(('sides', 'aisle'), 12.356910419823912)
```

Python Code

Please find 2 files attached with this pdf, solution1.py and solution2.py. Each file demonstrating the steps to get the desired results obtained for questions 2 and 3.