I. Analysis of State of the Union Addresses dataset:

1. State_union_part1

Title: Complete State of the Union Addresses from 1790 to 1860

History: release data is February, 2004 and the data last updated is June 24, 2007. Its edition is 12.

Language: English

Number of documents it contains: 72

The related policy: General Term of Use and Redistributing Project Gutenberg-tm electronic works

Information about the Mission of Project Gutenberg-tm

Information about the Project Gutenberg Literary Archive Foundation

Information about Donations to the Project Gutenberg Literary Archive Foundation

General Information About Project Gutenberg-tm electronic works

2. State_union_part2:

Title: Complete State of the Union Addresses from 1946 to the Present

History: 12's edition

Language: English

Number of Documents it contains: 70

The related policy: General Term of Use and Redistributing Project Gutenberg-tm electronic works

Information about the Mission of Project Gutenberg-tm

Information about the Project Gutenberg Literary Archive Foundation

Information about Donations to the Project Gutenberg Literary Archive Foundation

General Information About Project Gutenberg-tm electronic works

II. Analysis of State of the Union Addresses dataset: Part1

1. First, the necessary packages are as followings:

```
Import nltk
from nltk import FreqDist
from nltk.corpus import PlaintextCorpusReader
import re
from nltk.collocations import *
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
```

2. Second, I have several functions to help me to analyze.

alpha_filter: its work is to match a word of non-alphabetical characters

```
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
```

get_part_content: its work is to get the text file content in filename and return the content as a string

```
def get_part_content(filename):
   # get text file content and return the string
   corpus = PlaintextCorpusReader('..', '.*\.txt')
   content = corpus.raw(filename)
   return content
```

get_word_pos: get the words tags from pos_tag and then return the corresponding wordnet tags

```
def get_word_pos(word_pos_tag):
    # get the words pos tags from pos_tag and return the corresponding
wordnet tags
    if word_pos_tag.startswith('J'):
        return wordnet.ADJ
    elif word_pos_tag.startswith('V'):
        return wordnet.VERB
    elif word_pos_tag.startswith('N'):
        return wordnet.NOUN
    elif word_pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

do_word_tokenize: its work is to do the word tokenize which is separating the content into tokens with the word tokenizer

```
d
ef do_word_tokenize(content):
    # do word tokenizing process
    tokens = nltk.word_tokenize(content)
    return tokens
```

do_lower: its work is to set all words in the content as lowercase

```
def do_lower(content):
    # set all words as lowercase
    words = [w.lower() for w in content]
    return words
```

get_alphabetical_words: its work is to get only alphabetical words from the content

```
def get_alphabetical_words(content):
    # get alphabetical words
    alpha_words = [w for w in content if not alpha_filter(w)]
    return alpha_words
```

Get_stopwords: its work is to get the stop words which is in nltk package

```
def get_stopwords():
    # get stopwords
    stop_words = nltk.corpus.stopwords.words('english')
    return stop_words
```

Filter_content: its work is to filter the special words which is in filter_words (filter_words is a list of words)

```
def filter_content(content, filter_words):
    # filter special words in content
    filtered_words = [w for w in content if w not in filter_words]
    return filtered_words
```

do_print_top_50: its work is to print the top 50 frequency pair (word, frequency)

```
def do_print_top_50(content):
    # print top 50 frequency
    freq_dist = FreqDist(content)
    top_keys = freq_dist.most_common(50)
    for pair in top_keys:
        print(pair)
    print('-----------------------------------')
```

Get_bi_gram_association_measures: its work is to get the bigram measures defined by a general NLTK bigram function

```
def get_bi_gram_association_measures():
    # get bigram measures
    measures = nltk.collocations.BigramAssocMeasures()
    return measures
```

finder_filter: its work is to apply filter functions to finder by only using apply_freq_filter which can remove candidates which have frequency less than min_freq. Here I filter the words whose frequency is less than 5 since if we apply the Mutual Information score to all the bigrams, uniquely occurring pairs of words get high scores, the analysis result is not so well.

```
def finder_filter(content):
    # apply filter to finder
    finder = BigramCollocationFinder.from_words(content)
    finder.apply_freq_filter(5)
    return finder
```

Print_bigram_score: its work is to print out the top 50 bigram and its corresponding scores (score_ngrams can sort the scores as the order of decreasing frequency)

```
def print_bigram_score(bigram_measures, finder):
    # print scores which are sorted into order by decreasing frequency
    scored = finder.score_ngrams(bigram_measures.raw_freq)
    for one_score in scored[:50]:
        print(one_score)
    print('-----------------------------------')
```

print_PMI_measures: its work is to print out the top 50 PMI pair (bigram and the corresponding scores). NLTK provides the mutual information score which is given by a function Pointwise Mutual Information.

```
def print_PMI_measures(bigram_measures, finder):
    # print PMI measures scores
    scored = finder.score_ngrams(bigram_measures.pmi)
    for one_score in scored[:50]:
        print(one_score)
```

3. Third, my analysis process is as following:

Get the content from the text file as string

```
# get content in state_union_part text file
onePartContent = get_part_content('state_union_part1.txt')
```

Do word tokenize of the raw content from text file

```
```

```
# do word tokenize of the raw content
tokens_content = do_word_tokenize(onePartContent)
```

Get the pos tags for all words

```
# get the pos tags for all words
words_tags = pos_tag(tokens_content)
```

Get the lemmatization list

```
# do lemmatization
word_net_le = WordNetLemmatizer()
lemmatization_words = []
for tag in words_tags:
    word_pos = get_word_pos(tag[1]) or wordnet.NOUN
    lemmatization_words.append(word_net_le.lemmatize(tag[0], pos=word_pos))
```

Set all the words in the tokens content as lowercase

```
# set all the words in the content to be as lowercase
lower_words = do_lower(lemmatization_words)
```

Get only alphabetical words

```
# get alphabetical words
alphabetical_words = get_alphabetical_words(lower_words)
```

Get stop words from nltk package

```
# get stop words
stopwords = get_stopwords()
```

Filter the stop words

```
```

```
# filter the stop words
filter_stopwords = filter_content(alphabetical_words, stopwords)
```

Print out the top 50 words

```
# print out the top 50 words
do_print_top_50(filter_stopwords)
```

Get the bigram measures

```
# get bigram measures
bi_measures = get_bi_gram_association_measures()
```

Get the BigramCollocationFinder which is in nltk.collocations package

```
# get the finder
finder = BigramCollocationFinder.from_words(filter_stopwords)
```

Print out the top 50 bigram scores

```
# print out the top 50 bigram score
print_bigram_score(bi_measures, finder)
```

Apply filter to finder (filter the words whose frequency is less than 5)

```
# apply filter to finder
finder1 = finder_filter(filter_stopwords)
```

Print out the top 50 PMI measures scores

```
# print out the top 50 PMI measures score
print_PMI_measures(bi_measures, finder1)
```

4. Why I chose these processing options

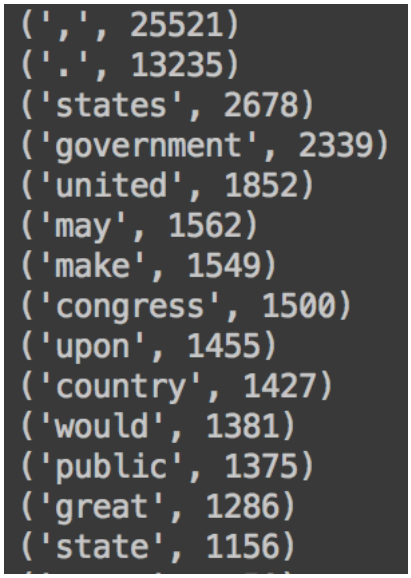
For the analysis process, I do lemmatization. If we do not do lemmatization, the result will have some words whose difference is only the form. For example, if we do not do lemmatization:

```
('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)
```

In the above list, we can see that interest and interests both appear in the result. And by using the lemmatization, we can let these two words to be one word: interest.

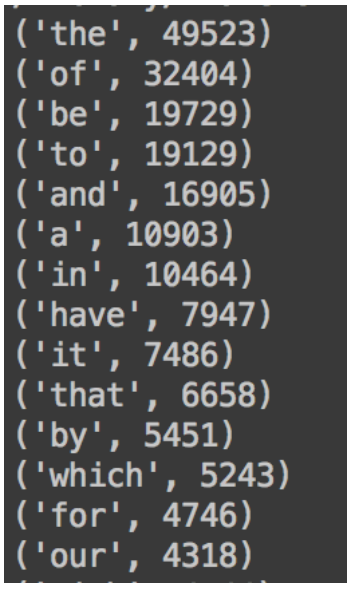
I also set all the word to be as lower case since if some words have first letter or all letter as uppercase, then these two words are different and will affect the following analysis.

I also select to filter those non-alphabetical words since punctuations will influence analysis. For example, if we do not filer non-alphabetical words:



From above list, we can get to know that the characteristics ‘,’ and ‘.’ are both punctuations and do not have any meanings but also show in top50 frequency words. For the alpha filter function, I use the alpha_filter() function instead of isalpha() since isalpha only get the words which length is bigger than zero and also have all characters are alphas. And this will delete some meaningful words.

I also select to filter stop words. Since some words do not have any meanings and will influence analysis. For example, if we do not filter stop words:



From the above list, we can see some words: the, of, be, etc. These words are not useful and also do not have any meanings. So we should remove those words which are stop words.

For the mutual Information scores, I use the min frequency 5 which is the requirement. (Since the result will not be good if we apply the mutual Information score to all the bigrams. Some uniquely occurring pairs of words will get high scores.)

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

I list top 50 words by frequency using above analysis methods:

```
('states', 2678)
('government', 2339)
('united', 1852)
('congress', 1500)
('country', 1427)
('public', 1375)
('great', 1286)
('state', 1156)
('power', 1150)
('year', 1145)
('present', 1027)
('duty', 1016)
('law', 969)
('subject', 936)
('time', 914)
('interest', 913)
('war', 897)
('act', 873)
('nation', 822)
('people', 786)
('citizen', 781)
('part', 744)
('treaty', 739)
('union', 643)
('mexico', 605)
('general', 604)
('give', 604)
('treasury', 592)
('amount', 574)
('constitution', 570)
('effect', 548)
('object', 538)
('foreign', 519)
('system', 509)
('commerce', 506)
('peace', 501)
('require', 500)
```

```
('measure', 497)
('receive', 493)
('consideration', 493)
('territory', 484)
('force', 483)
('service', 479)
('condition', 476)
('exist', 476)
('relation', 475)
('increase', 467)
('place', 461)
('revenue', 459)
('view', 458)
```

From the above top 50 frequency words, states, government, united appear frequently. Since this text is about complete state of the Union address from 1790 to 1860. These words also reflect the topic.

Then words: congress, country, public, state, power, duty, law, citizen appear in the top 50 words which is about U.S Federal Government.

Some words such as war, Mexico, peace, foreign, commerce also reflects the history in United States.

6. list the top 50 bigrams by frequencies

```
(('united', 'states'), 0.008851192067465428)
(('great', 'britain'), 0.0013318103385422996)
(('public', 'debt'), 0.0008797725229056796)
(('state', 'union'), 0.0008408875495175833)
(('house', 'representatives'), 0.0007193720076797823)
(('fiscal', 'year'), 0.0007047901426592461)
(('report', 'secretary'), 0.0007047901426592461)
(('union', 'address'), 0.0006999295209857341)
(('public', 'land'), 0.0006513233042506136)
(('act', 'congress'), 0.0006124383308625173)
(('present', 'year'), 0.0005200865190657885)
(('session', 'congress'), 0.0005152258973922765)
(('public', 'money'), 0.0005006440323717403)
(('secretary', 'treasury'), 0.00048120154567769216)
(('year', 'end'), 0.000461759058983644)
(('general', 'government'), 0.0004520378156366199)
(('british', 'government'), 0.0004471771939631079)
(('citizen', 'united'), 0.0004325953289425718)
(('annual', 'message'), 0.0004131528422485236)
(('federal', 'government'), 0.0004131528422485236)
(('secretary', 'war'), 0.00040829222057501153)
(('fellow', 'citizen'), 0.00040343159890149995)
(('consideration', 'congress'), 0.00039371035555544754)
(('public', 'service'), 0.00038884973388096335)
(('senate', 'house'), 0.00038884973388096335)
(('carry', 'effect'), 0.00036940724718691523)
(('government', 'united'), 0.00036940724718691523)
(('attention', 'congress'), 0.00035968600383989114)
(('part', 'united'), 0.0003305222737988189)
(('naval', 'force'), 0.0003208010304517948)
(('indian', 'tribe'), 0.0003159404087782827)
(('mexican', 'government'), 0.0003159404087782827)
(('treasury', 'note'), 0.0003159404087782827)
(('article', 'treaty'), 0.00030621916543125863)
(('commercial', 'intercourse'), 0.00030621916543125863)
(('secretary', 'state'), 0.0003013585437577466)
```

```
(('states', 'great'), 0.0002964979220842346)
(('claim', 'citizen'), 0.0002867766787372105)
(('public', 'interest'), 0.0002867766787372105)
(('american', 'citizen'), 0.00028191605706369844)
(('interest', 'country'), 0.00028191605706369844)
(('foreign', 'power'), 0.0002770554353901864)
(('address', 'december'), 0.00027219481371667435)
(('end', '30th'), 0.00027219481371667435)
(('favorable', 'consideration'), 0.00026733419204316234)
(('people', 'united'), 0.00026733419204316234)
(('30th', 'june'), 0.00026247357036965027)
(('bank', 'united'), 0.00026247357036965027)
(('secretary', 'navy'), 0.00026247357036965027)
(('exist', 'law'), 0.0002527523270226262)
```

From above statistics, we can get to know some words pair: united states, great britain, public debt, state union, etc. These words appear together frequently in the text file.

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

```
(('bona', 'fide'), 15.328499628108926)
(('punta', 'arenas'), 15.328499628108926)
(('del', 'norte'), 15.065465222275126)
(('millard', 'fillmore'), 15.065465222275126)
(('ballot', 'box'), 14.843072800938682)
(('clayton', 'bulwer'), 14.650427722996284)
(('guadalupe', 'hidalgo'), 14.480502721553972)
(('porto', 'rico'), 14.480502721553972)
(('writ', 'mandamus'), 14.387393317162491)
(('franklin', 'pierce'), 14.328499628108922)
(('la', 'plata'), 14.190996104358986)
(('vera', 'cruz'), 14.065465222275126)
(('posse', 'comitatus'), 13.843072800938682)
(('costa', 'rica'), 13.650427722996284)
(('santa', 'anna'), 13.562964881745945)
(('santa', 'fe'), 13.562964881745945)
(('van', 'buren'), 13.562964881745945)
(('sublime', 'porte'), 13.521144706051317)
(('tea', 'coffee'), 13.402500209552699)
(('martin', 'van'), 13.393039880303633)
(('ad', 'valorem'), 13.32849962810892)
(('quincy', 'adams'), 13.190996104358984)
(('retired', 'list'), 13.126865766939273)
(('rocky', 'mountains'), 13.12686576693927)
(('ruler', 'universe'), 13.065465222275126)
(('buenos', 'ayres'), 13.065465222275124)
(('beacon', 'buoy'), 13.053492580609053)
(('barbary', 'powers'), 13.006571533221559)
(('indiana', 'illinois'), 12.927961698525195)
(('de', 'facto'), 12.917073382382458)
(('gun', 'boat'), 12.780063003412879)
(('andrew', 'jackson'), 12.760610640746705)
(('intent', 'meaning'), 12.70289514289042)
(('project', 'gutenberg'), 12.650427722996282)
(('john', 'quincy'), 12.562964881745943)
(('thomas', 'jefferson'), 12.475502040495604)
```

```
(('precious', 'metal'), 12.405423604759463)
(('almighty', 'god'), 12.393039880303633)
(('john', 'tyler'), 12.393039880303633)
(('san', 'jacinto'), 12.365025504134039)
(('san', 'juan'), 12.365025504134035)
(('san', 'francisco'), 12.365025504134033)
(('seizure', 'confiscation'), 12.328499628108924)
(('rio', 'grande'), 12.258110300217526)
(('effusion', 'blood'), 12.177939951533544)
(('seminary', 'learn'), 12.120606776467593)
(('lake', 'erie'), 12.027330093388358)
(('corps', 'engineers'), 11.936182205330159)
(('st.', 'augustine'), 11.922507268433087)
(('st.', 'lawrence'), 11.922507268433087)
```

From above statistics, we can get to know some words pair: bona fide, punta arenas, del norte, etc. These words have higher mutual information scores.

III. Analysis of State of the Union Addresses dataset: Part2

1. First, the necessary packages are as followings:

```
Import nltk
from nltk import FreqDist
from nltk.corpus import PlaintextCorpusReader
import re
from nltk.collocations import *
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
```

2. Second, I have several functions to help me to analyze.

alpha_filter: its work is to match a word of non-alphabetical characters

```
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
```

get_part_content: its work is to get the text file content in filename and return the content as a string

```
def get_part_content(filename):
    # get text file content and return the string
    corpus = PlaintextCorpusReader('..', '.*\.txt')
    content = corpus.raw(filename)
    return content
```

get_word_pos: get the words tags from pos_tag and then return the corresponding wordnet tags

```
def get_word_pos(word_pos_tag):
    # get the words pos tags from pos_tag and return the corresponding
wordnet tags
    if word_pos_tag.startswith('J'):
        return wordnet.ADJ
    elif word_pos_tag.startswith('V'):
        return wordnet.VERB
    elif word_pos_tag.startswith('N'):
        return wordnet.NOUN
    elif word_pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

do_word_tokenize: its work is to do the word tokenize which is separating the content into tokens with the word tokenizer

```
d
ef do_word_tokenize(content):
    # do word tokenizing process
    tokens = nltk.word_tokenize(content)
    return tokens
```

do_lower: its work is to set all words in the content as lowercase

```
def do_lower(content):
    # set all words as lowercase
    words = [w.lower() for w in content]
    return words
```

get_alphabetical_words: its work is to get only alphabetical words from the content

```
def get_alphabetical_words(content):
    # get alphabetical words
    alpha_words = [w for w in content if not alpha_filter(w)]
    return alpha_words
```

Get_stopwords: its work is to get the stop words which is in nltk package

```
def get_stopwords():
    # get stopwords
    stop_words = nltk.corpus.stopwords.words('english')
    return stop_words
```

Filter_content: its work is to filter the special words which is in filter_words (filter_words is a list of words)

```
def filter_content(content, filter_words):
    # filter special words in content
    filtered_words = [w for w in content if w not in filter_words]
    return filtered_words
```

do_print_top_50: its work is to print the top 50 frequency pair (word, frequency)

```
def do_print_top_50(content):
    # print top 50 frequency
    freq_dist = FreqDist(content)
    top_keys = freq_dist.most_common(50)
    for pair in top_keys:
        print(pair)
    print('-----------------------------------')
```

Get_bi_gram_association_measures: its work is to get the bigram measures defined by a general NLTK bigram function

```
def get_bi_gram_association_measures():
    # get bigram measures
    measures = nltk.collocations.BigramAssocMeasures()
    return measures
```

finder_filter: its work is to apply filter functions to finder by only using apply_freq_filter which can remove candidates which have frequency less than

min_freq. Here I filter the words whose frequency is less than 5 since if we apply the Mutual Information score to all the bigrams, uniquely occurring pairs of words get high scores, the analysis result is not so well.

```
def finder_filter(content):
    # apply filter to finder
    finder = BigramCollocationFinder.from_words(content)
    finder.apply_freq_filter(5)
    return finder
```

Print_bigram_score: its work is to print out the top 50 bigram and its corresponding scores (score_ngrams can sort the scores as the order of decreasing frequency)

```
def print_bigram_score(bigram_measures, finder):
    # print scores which are sorted into order by decreasing frequency
    scored = finder.score_ngrams(bigram_measures.raw_freq)
    for one_score in scored[:50]:
        print(one_score)
    print('-----------------------------------')
```

print_PMI_measures: its work is to print out the top 50 PMI pair (bigram and the corresponding scores). NLTK provides the mutual information score which is given by a function Pointwise Mutual Information.

```
def print_PMI_measures(bigram_measures, finder):
    # print PMI measures scores
    scored = finder.score_ngrams(bigram_measures.pmi)
    for one_score in scored[:50]:
        print(one_score)
```

3. Third, my analysis process is as following:

Get the content from the text file as string

```
# get content in state_union_part text file
onePartContent = get_part_content('state_union_part2.txt')
```

Do word tokenize of the raw content from text file

```
# do word tokenize of the raw content
tokens_content = do_word_tokenize(onePartContent)
```

Get the pos tags for all words

```
# get the pos tags for all words
words_tags = pos_tag(tokens_content)
```

Get the lemmatization list

```
# do lemmatization
word_net_le = WordNetLemmatizer()
lemmatization_words = []
for tag in words_tags:
    word_pos = get_word_pos(tag[1]) or wordnet.NOUN
    lemmatization_words.append(word_net_le.lemmatize(tag[0], pos=word_pos))
```

Set all the words in the tokens content as lowercase

```
# set all the words in the content to be as lowercase
lower_words = do_lower(lemmatization_words)
```

Get only alphabetical words

```
# get alphabetical words
alphabetical_words = get_alphabetical_words(lower_words)
```

Get stop words from nltk package

```
# get stop words
stopwords = get_stopwords()
```

Filter the stop words

```
# filter the stop words
filter_stopwords = filter_content(alphabetical_words, stopwords)
```

Print out the top 50 words

```
# print out the top 50 words
do_print_top_50(filter_stopwords)
```

Get the bigram measures

```
# get bigram measures
bi_measures = get_bi_gram_association_measures()
```

Get the BigramCollocationFinder which is in nltk.collocations package

```
# get the finder
finder = BigramCollocationFinder.from_words(filter_stopwords)
```

Print out the top 50 bigram scores

```
# print out the top 50 bigram score
print_bigram_score(bi_measures, finder)
```

Apply filter to finder (filter the words whose frequency is less than 5)

```
# apply filter to finder
finder1 = finder_filter(filter_stopwords)
```

Print out the top 50 PMI measures scores

```
```

```
# print out the top 50 PMI measures score
print_PMI_measures(bi_measures, finder1)
```

4. Why I chose these processing options

For the analysis process, I do lemmatization. If we do not do lemmatization, the result will have some words whose difference is only the form. For example, if we do not do lemmatization:

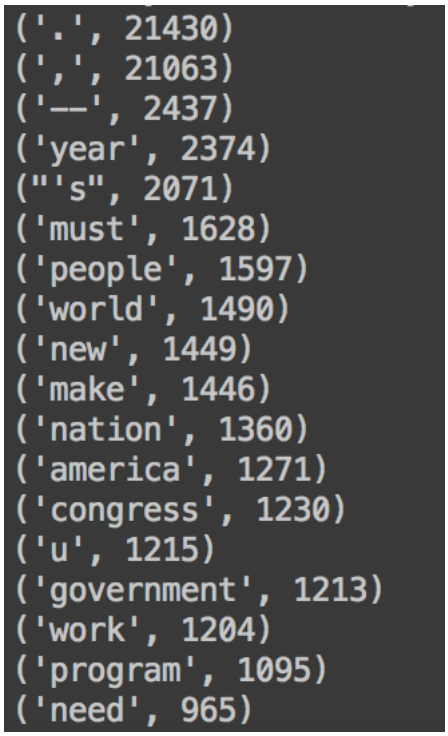
```
('nation', 861)
('one', 804)
('every', 780)
('make', 778)
('work', 754)
('federal', 744)
('time', 741)
('states', 711)
('americans', 688)
('help', 686)
('security', 685)
('war', 674)
('economic', 671)
('peace', 668)
('united', 651)
('nations', 645)
```

In the above list, we can see that nation and nations both appear in the result. And by using the lemmatization, we can let these two words to be one word: nation.

I also set all the word to be as lower case since if some words have first letter or all letter as uppercase, then these two words are different and will affect the following analysis.

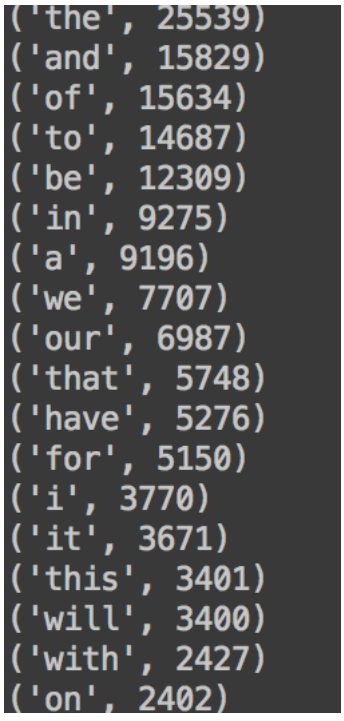
For the alpha filter function, I also use the alpha_filter() function instead of isalpha() since isalpha only get the words which length is bigger than zero and also have all characters are alphas. And this will delete some meaningful words.

I also select to filter those non-alphabetical words since punctuations will influence analysis. For example, if we do not filer non-alphabetical words:



From above list, we can get to know that the characteristics ‘,’ , ‘.’ and ‘—’ are both punctuations and do not have any meanings but also show in top50 frequency words.

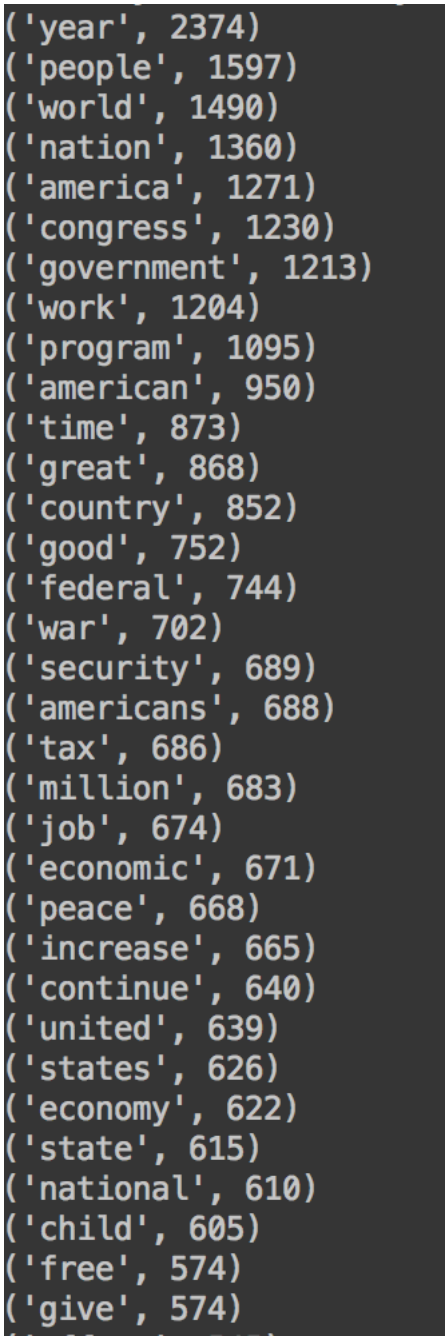
I also select to filter stop words. Since some words do not have any meanings and will influence analysis. For example, if we do not filter stop words:

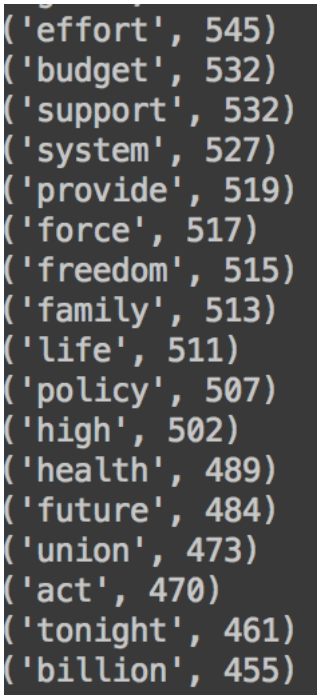


From the above list, we can see some words: the, and, of, etc. These words are not useful and also do not have any meanings. We should remove those words which are stop words.

For the mutual Information scores, I use the min frequency 5 which is the requirement. (Since the result will not be good if we apply the mutual Information score to all the bigrams. Some uniquely occurring pairs of words will get high scores.)

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





From above statistics, we can see the words: year, people, words, nation, america, congress, government, work, program, etc. These words appear very frequently in part2 which may be the part2' focus topic. Some interesting words such as war, security, job, economic, peace also reflect some part2's content.

6. list the top 50 bigrams by frequencies

```
(('united', 'states'), 0.002454874413513499)
(('state', 'union'), 0.0014187261220954639)
(('american', 'people'), 0.0012699458546097972)
(('fiscal', 'year'), 0.0010361482914180353)
(('past', 'year'), 0.0010148939674915116)
(('year', 'ago'), 0.0010148939674915116)
(('federal', 'government'), 0.0009830124816017259)
(('social', 'security'), 0.000967071738656833)
(('health', 'care'), 0.0009458174147303092)
(('billion', 'dollar'), 0.0008129778901895355)
(('union', 'address'), 0.0007332741754650712)
(('united', 'nations'), 0.0007173334325201783)
(('million', 'dollar'), 0.0007013926895752854)
(('soviet', 'union'), 0.0006641976227038688)
(('men', 'woman'), 0.0006004346509242974)
(('free', 'world'), 0.0005898074889610355)
(('economic', 'growth'), 0.0004994766122733093)
(('middle', 'east'), 0.0004835358693284165)
(('free', 'nation'), 0.0004782222834678557)
(('small', 'business'), 0.00043571364049373797)
(('world', 'war'), 0.00043571364049373797)
(('state', 'local'), 0.000430400059512107)
(('million', 'americans'), 0.0004144593165672141)
(('tax', 'cut'), 0.0004038321546039522)
(('foreign', 'policy'), 0.0003932049926406903)
(('21st', 'century'), 0.0003878914116590594)
(('mr.', 'speaker'), 0.0003719506687141665)
(('create', 'job'), 0.0003613235067509046)
(('vice', 'president'), 0.0003560099257692737)
(('year', 'year'), 0.0003506963447876427)
(('local', 'government'), 0.0003453827638060118)
(('urge', 'congress'), 0.0003453827638060118)
(('national', 'security'), 0.0003347556018427499)
(('address', 'january'), 0.0003188144858897857)
```

```
(('nation', 'world'), 0.0003188144858897857)
(('health', 'insurance'), 0.0003081876969345951)
(('tax', 'credit'), 0.0003028741159529642)
(('fellow', 'americans'), 0.00029756053497133323)
(('fellow', 'citizen'), 0.00029756053497133323)
(('states', 'america'), 0.00029756053497133323)
(('high', 'school'), 0.00028693337300807133)
(('nuclear', 'weapon'), 0.00028693337300807133)
(('private', 'sector'), 0.00028693337300807133)
(('young', 'people'), 0.00028693337300807133)
(('god', 'bless'), 0.00027630621104480943)
(('interest', 'rate'), 0.00027630621104480943)
(('minimum', 'wage'), 0.00027099263006317845)
(('members', 'congress'), 0.00026567904908154753)
(('balanced', 'budget'), 0.00026036546809991655)
(('armed', 'force'), 0.00025505188711828563)
```

From above statistics, some words pair: united states, state union, american people, fiscal year, past year, year ago have higher bigrams and appear together more frequently. These words pairs have higher correlation.

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

```
(('el', 'salvador'), 14.93692160433461)
(('bin', 'laden'), 14.714529182998163)
(('saudi', 'arabia'), 14.71452918299816)
(('australia', 'zealand'), 14.521884105055765)
(('sam', 'rayburn'), 14.521884105055765)
(('gerald', 'r.'), 14.299491683719317)
(('jimmy', 'carter'), 14.199956010168403)
(('endow', 'creator'), 14.088924697779657)
(('vol', 'p.'), 14.062452486418469)
(('northern', 'ireland'), 13.93692160433461)
(("o'neill", 'jr.'), 13.866532276443209)
(('r.', 'ford'), 13.840060065082021)
(('lyndon', 'b.'), 13.82144438691467)
(('iron', 'curtain'), 13.614993509447249)
(('william', 'j.'), 13.614993509447249)
(('thomas', 'jefferson'), 13.558409981080878)
(('red', 'tape'), 13.521884105055767)
(('200th', 'anniversary'), 13.477489985697314)
(('jill', 'biden'), 13.451494777164367)
(('b.', 'johnson'), 13.434421263805424)
(('barack', 'obama'), 13.434421263805424)
(('teen', 'pregnancy'), 13.299491683719317)
(('abraham', 'lincoln'), 13.264496262363114)
(('p.', "o'neill"), 13.214455579863518)
(('j.', 'clinton'), 13.199956010168405)
(('mom', 'dad'), 13.129566682277005)
(('ronald', 'reagan'), 13.062452486418467)
(('greece', 'turkey'), 12.977563588831956)
(('weapons', 'convention'), 12.948994436635182)
(('elementary', 'secondary'), 12.895444968358449)
(('harry', 's.'), 12.858919092333334)
(('small-business', 'owner'), 12.8520327067481)
(('dwight', 'd.'), 12.799418080584672)
```

```
('intercontinental', 'ballistic'), 12.775929727662305)
('h.w', 'bush'), 12.766996602892299)
('w.', 'bush'), 12.766996602892295)
('old-age', 'survivor'), 12.697455669639218)
('thomas', 'p.'), 12.683940863164738)
('status', 'quo'), 12.663903109928192)
('nationwide', 'radio'), 12.5743515249499)
('radio', 'television'), 12.521884105055765)
('f.', 'kennedy'), 12.434421263805426)
('prime', 'minister'), 12.3926010881108)
('al', 'qaida'), 12.392601088110798)
('al', 'qaeda'), 12.392601088110796)
('richard', 'nixon'), 12.375527574751858)
('george', 'h.w'), 12.351959103613453)
('saddam', 'hussein'), 12.351959103613453)
('george', 'w.'), 12.35195910361345)
('d.', 'eisenhower'), 12.339986461947376)
```

From above statistics, some words pair: el salvador, bin laden, Saudi arabia, Australia Zealand have higher mutual information scores.

IV. Comparison

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

For the differences and similarities, I made a analysis in the python.

1). Compare of top 50 words by frequency

The same words which appear in both part1 and part2's top 50 words by frequency are as following. And first one is the corresponding word record in part1, second one is the corresponding word record in part2:

```
Same words
('states', 2678)
('states', 626)
('government', 2339)
('government', 1213)
('united', 1852)
('united', 639)
('congress', 1500)
('congress', 1230)
('country', 1427)
('country', 852)
('great', 1286)
('great', 868)
('state', 1156)
('state', 615)
('year', 1145)
('year', 2374)
('time', 914)
('time', 873)
('war', 897)
('war', 702)
('act', 873)
('act', 470)
('nation', 822)
('nation', 1360)
('people', 786)
('people', 1597)
('union', 643)
('union', 473)
('give', 604)
('give', 574)
('system', 509)
('system', 527)
('peace', 501)
('peace', 668)
('force', 483)
('force', 517)
('increase', 467)
('increase', 665)
------------------------
```

From the above statistics, we can get to know that words: states, government and united all appear. And these words appear more frequently in part1. And the words: nation, people appear more frequently in part2. This also reflects the history of United States. Other words' (such as country, war, act union, peace, force) frequencies in

part1 and part2 have little difference. And this also the words both part1 (1790~1860) and part2 (1946~2016) focus on.

The different words which appear in only part1's top 50 words by frequency are as following.

```
Different words in part1
('public', 1375)
('power', 1150)
('present', 1027)
('duty', 1016)
('law', 969)
('subject', 936)
('interest', 913)
('citizen', 781)
('part', 744)
('treaty', 739)
('mexico', 605)
('general', 604)
('treasury', 592)
('amount', 574)
('constitution', 570)
('effect', 548)
('object', 538)
('foreign', 519)
('commerce', 506)
('require', 500)
('measure', 497)
('receive', 493)
('consideration', 493)
('territory', 484)
('service', 479)
('condition', 476)
('exist', 476)
('relation', 475)
('place', 461)
('revenue', 459)
('view', 458)
--------------------------
```

From the above statistics, we can get to know that words such as public, power, duty, citizen, foreign, commerce appear in the top 50 words in part1. This is related to United States' history, since 1790~1860 is the early history of United States. For this period, United States may focus to build law, politics and commerce with foreign countries.

The different words which appear in only part2's top 50 words by frequency are as following.

```
Different words in part2
('world', 1490)
('america', 1271)
('work', 1204)
('program', 1095)
('american', 950)
('good', 752)
('federal', 744)
('security', 689)
('americans', 688)
('tax', 686)
('million', 683)
('job', 674)
('economic', 671)
('continue', 640)
('economy', 622)
('national', 610)
('child', 605)
('free', 574)
('effort', 545)
('budget', 532)
('support', 532)
('provide', 519)
('freedom', 515)
('family', 513)
('life', 511)
('policy', 507)
('high', 502)
('health', 489)
('future', 484)
('tonight', 461)
('billion', 455)
------------------------
```

From above statistics, we can get to know that words such as world, ammerica, ammerican, federal, security, tax, job, economic, national, freedom, life appear in the part2 top 50 words. And these words may be the new focus topics.

2). Compare of top 50 bigrams by frequencies

For the following statistics, words: united states, state union, fiscal year, union address, federal, government, fellow citizen appear both in part1 and part2. And first

one is the corresponding word in part1, second one is the corresponding word in part2. They are as following:

```
same words
(('united', 'states'), 0.0088511920674654 28)
(('united', 'states'), 0.002454874413513499)
(('state', 'union'), 0.0008408875495175833)
(('state', 'union'), 0.0014187261220954639)
(('fiscal', 'year'), 0.0007047901426592461)
(('fiscal', 'year'), 0.0010361482914180353)
(('union', 'address'), 0.0006999295209857341)
(('union', 'address'), 0.0007332741754650712)
(('federal', 'government'), 0.0004131528422485236)
(('federal', 'government'), 0.0009830124816017259)
(('fellow', 'citizen'), 0.0004034315989014995)
(('fellow', 'citizen'), 0.00029756053497133323)
--------------------
```

For the Following statistics, we can get to know that words pairs which only appear in the part1's top 50 bigram by frequency. They are as following:

```
different words in part1
(('great', 'britain'), 0.0013318103385422996)
(('public', 'debt'), 0.0008797725229056796)
(('house', 'representatives'), 0.0007193720076797823)
(('report', 'secretary'), 0.0007047901426592461)
(('public', 'land'), 0.0006513233042506136)
(('act', 'congress'), 0.0006124383308625173)
(('present', 'year'), 0.0005200865190657885)
(('session', 'congress'), 0.0005152258973922765)
(('public', 'money'), 0.0005006440323717403)
(('secretary', 'treasury'), 0.00048120154567769216)
(('year', 'end'), 0.000461759058983644)
(('general', 'government'), 0.0004520378156366199)
(('british', 'government'), 0.0004471771939631079)
(('citizen', 'united'), 0.0004325953289425718)
(('annual', 'message'), 0.0004131528422485236)
(('secretary', 'war'), 0.00040829222057501153)
(('consideration', 'congress'), 0.00039371035555544754)
(('public', 'service'), 0.00038884973388096335)
(('senate', 'house'), 0.00038884973388096335)
(('carry', 'effect'), 0.00036940724718691523)
(('government', 'united'), 0.00036940724718691523)
(('attention', 'congress'), 0.00035968600383989114)
(('part', 'united'), 0.00033305222737988189)
(('naval', 'force'), 0.00032080103045179480)
(('indian', 'tribe'), 0.00031594040877827827)
(('mexican', 'government'), 0.00031594040877827827)
(('treasury', 'note'), 0.00031594040877827827)
(('article', 'treaty'), 0.00030621916543125863)
(('commercial', 'intercourse'), 0.00030621916543125863)
(('secretary', 'state'), 0.00030135854375774660)
(('states', 'great'), 0.00029264979220842346)
(('claim', 'citizen'), 0.00028677667873721050)
(('public', 'interest'), 0.00028677667873721050)
(('american', 'citizen'), 0.00028191605706369844)
(('interest', 'country'), 0.00028191605706369844)
(('foreign', 'power'), 0.00027705543539018640)
(('address', 'december'), 0.00027219481371667435)
(('end', '30th'), 0.00027219481371667435)
(('favorable', 'consideration'), 0.00026733419204316234)
(('people', 'united'), 0.00026733419204316234)
(('30th', 'june'), 0.00026247357036965027)
(('bank', 'united'), 0.00026247357036965027)
(('secretary', 'navy'), 0.00026247357036965027)
(('exist', 'law'), 0.00025275232702262620)
----------------------
```

For the Following statistics, we can get to know that words pairs which only appear in the part2's top 50 bigram by frequency. They are as following:

```
different words in part2
(('american', 'people'), 0.0012699458546097972)
(('past', 'year'), 0.0010148939674915116)
(('year', 'ago'), 0.0010148939674915116)
(('social', 'security'), 0.000967071738656833)
(('health', 'care'), 0.0009458174147303092)
(('billion', 'dollar'), 0.0008129778901895355)
(('united', 'nations'), 0.0007173334325201783)
(('million', 'dollar'), 0.0007013926895752854)
(('soviet', 'union'), 0.0006641976227038688)
(('men', 'woman'), 0.0006004346509242974)
(('free', 'world'), 0.0005898074889610355)
(('economic', 'growth'), 0.0004994766122733093)
(('middle', 'east'), 0.0004835358693284165)
(('free', 'nation'), 0.0004782222288346785 57)
(('small', 'business'), 0.00043571364049373797)
(('world', 'war'), 0.00043571364049373797)
(('state', 'local'), 0.000430400059512107)
(('million', 'americans'), 0.0004144593165672141)
(('tax', 'cut'), 0.0004038321546039522)
(('foreign', 'policy'), 0.0003932049926406903)
(('21st', 'century'), 0.0003878914116590594)
(('mr.', 'speaker'), 0.0003719506687141665)
(('create', 'job'), 0.0003613235067509046)
(('vice', 'president'), 0.0003560099257692737)
(('year', 'year'), 0.0003506963447876427)
(('local', 'government'), 0.0003453827638060118)
(('urge', 'congress'), 0.0003453827638060118)
(('national', 'security'), 0.0003347556018427499)
(('address', 'january'), 0.0003188814858897857)
(('nation', 'world'), 0.0003188814858897857)
(('health', 'insurance'), 0.0003081876969345951)
(('tax', 'credit'), 0.0003028741159529642)
(('fellow', 'americans'), 0.00029756053497133323)
(('states', 'america'), 0.0002975605349713323)
(('high', 'school'), 0.00028693337300807133)
(('nuclear', 'weapon'), 0.00028693337300807133)
(('private', 'sector'), 0.00028693337300807133)
(('young', 'people'), 0.00028693337300807133)
(('god', 'bless'), 0.00027630621104480943)
(('interest', 'rate'), 0.00027630621104480943)
(('minimum', 'wage'), 0.00027099263006317845)
(('members', 'congress'), 0.00026567904908154753)
(('balanced', 'budget'), 0.00026036546809991655)
(('armed', 'force'), 0.00025505188711828563)
```

3). Compare of the top 50 bigrams by their Mutual Information scores

Following statistics is the words pairs appear in both part1 and part2's top 50 bigrams by their Mutual Information scores. And first one is the corresponding word in part1, second one is the corresponding word in part2. They are as following:

```
same words
(('united', 'states'), 0.008851192067465428)
(('united', 'states'), 0.002454874413513499)
(('state', 'union'), 0.0008408875495175833)
(('state', 'union'), 0.0014187261220954639)
(('fiscal', 'year'), 0.0007047901426592461)
(('fiscal', 'year'), 0.0010361482914180353)
(('union', 'address'), 0.0006999295209857341)
(('union', 'address'), 0.0007332741754650712)
(('federal', 'government'), 0.0004131528422485236)
(('federal', 'government'), 0.0009830124816017259)
(('fellow', 'citizen'), 0.0004034315989014995)
(('fellow', 'citizen'), 0.0002975605349713323)
----------
```

Following statistics is the words pair which only appear in part1 top 50 bigrams by their Mutual Information scores:

```
different words in part1
(('great', 'britain'), 0.0013318103385422996)
(('public', 'debt'), 0.0008797725229056796)
(('house', 'representatives'), 0.0007193720076797823)
(('report', 'secretary'), 0.0007047901426592461)
(('public', 'land'), 0.0006513233042506136)
(('act', 'congress'), 0.0006124383308625173)
(('present', 'year'), 0.0005200865190657885)
(('session', 'congress'), 0.0005152258973922765)
(('public', 'money'), 0.0005006440323717403)
(('secretary', 'treasury'), 0.0004812015456776921 6)
(('year', 'end'), 0.000461759058983644)
(('general', 'government'), 0.0004520378156366199)
(('british', 'government'), 0.0004471771939631079)
(('citizen', 'united'), 0.0004325953289425718)
(('annual', 'message'), 0.0004131528422485236)
(('secretary', 'war'), 0.0004082922205750115 3)
(('consideration', 'congress'), 0.0003937103555544754)
(('public', 'service'), 0.0003888497338809633 5)
(('senate', 'house'), 0.000388849733880963 35)
(('carry', 'effect'), 0.0003694072471869152 3)
(('government', 'united'), 0.000369407247186915 23)
(('attention', 'congress'), 0.0003596860038398911 4)
(('part', 'united'), 0.0003305222737988189)
(('naval', 'force'), 0.0003208010304517948)
(('indian', 'tribe'), 0.0003159404087782827)
(('mexican', 'government'), 0.000315940408778282 7)
(('treasury', 'note'), 0.000315940408778282 7)
(('article', 'treaty'), 0.0003062191654312586 3)
(('commercial', 'intercourse'), 0.0003062191654312586 3)
(('secretary', 'state'), 0.0003013585437577466)
(('states', 'great'), 0.0002964979220842346)
(('claim', 'citizen'), 0.0002867766787372105)
(('public', 'interest'), 0.0002867766787372105)
(('american', 'citizen'), 0.0002819160570636984 4)
(('interest', 'country'), 0.000281916057063698 44)
(('foreign', 'power'), 0.0002770554353901864)
(('address', 'december'), 0.0002721948137166743 5)
(('end', '30th'), 0.000272194813716674 35)
(('favorable', 'consideration'), 0.0002673341920431623 4)
(('people', 'united'), 0.000267334192043162 34)
(('30th', 'june'), 0.0002624735703696502 7)
(('bank', 'united'), 0.000262473570369650 27)
(('secretary', 'navy'), 0.0002624735703696502 7)
(('exist', 'law'), 0.0002527523270226262)
----------
```

Following statistics is the words pair which only appear in part2 top 50 bigrams by their Mutual Information scores:

```
different words in part2
(('american', 'people'), 0.0012699458546097972)
(('past', 'year'), 0.0010148939674915116)
(('year', 'ago'), 0.0010148939674915116)
(('social', 'security'), 0.000967071738656833)
(('health', 'care'), 0.0009458174147303092)
(('billion', 'dollar'), 0.0008129778901895355)
(('united', 'nations'), 0.0007173334325201783)
(('million', 'dollar'), 0.0007013926895752854)
(('soviet', 'union'), 0.0006641976227038688)
(('men', 'woman'), 0.0006004346509242974)
(('free', 'world'), 0.0005898074889610355)
(('economic', 'growth'), 0.0004994766122733093)
(('middle', 'east'), 0.0004835358693284165)
(('free', 'nation'), 0.0004782222288834678557)
(('small', 'business'), 0.00043571364049373797)
(('world', 'war'), 0.00043571364049373797)
(('state', 'local'), 0.000430400059512107)
(('million', 'americans'), 0.0004144593165672141)
(('tax', 'cut'), 0.0004038321546039522)
(('foreign', 'policy'), 0.0003932049926406903)
(('21st', 'century'), 0.0003878914116590594)
(('mr.', 'speaker'), 0.0003719506687141665)
(('create', 'job'), 0.0003613235067509046)
(('vice', 'president'), 0.0003560099257692737)
(('year', 'year'), 0.0003506963447876427)
(('local', 'government'), 0.0003453827638060118)
(('urge', 'congress'), 0.0003453827638060118)
(('national', 'security'), 0.0003347556018427499)
(('address', 'january'), 0.0003188148588897857)
(('nation', 'world'), 0.0003188148588897857)
(('health', 'insurance'), 0.0003081876969345951)
(('tax', 'credit'), 0.0003028741159529642)
(('fellow', 'americans'), 0.00029756053497133323)
(('states', 'america'), 0.00029756053497133323)
(('high', 'school'), 0.00028693337300807133)
(('nuclear', 'weapon'), 0.00028693337300807133)
(('private', 'sector'), 0.00028693337300807133)
(('young', 'people'), 0.00028693337300807133)
(('god', 'bless'), 0.00027630621104480943)
(('interest', 'rate'), 0.00027630621104480943)
(('minimum', 'wage'), 0.00027099263006317845)
(('members', 'congress'), 0.00026567904908154753)
(('balanced', 'budget'), 0.00026036546809991655)
(('armed', 'force'), 0.00025505188711828563)
```

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

In both part1 and part2's top 50 bigrams scored by Mutual Information, I find some problems about the words pair. Many words pair are about city's name, people's name, place's name, etc. And we can also use some methods to filter these kinds of words to get a better and meaningful results. (If these names are also meaningful, we can do name analysis especially)

Also from part1 and part2's result, there are some words which are not like English. And we can also use some methods to translate these words to English to make it easy for us to analyze.

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

The results in top 50 bigrams by frequency and top 50 bigrams scored by Mutual Information are very different. The possible reason is that the higher pmi does not necessarily mean that the importance of its bigrams. If some words pairs appear rarely but they appear together in a higher probability. Then the pmi will be higher but its corresponding bigrams by frequency is lower. This can be seen from the results of both part1 and part2.

In part1, words such as ('punta', 'arenas'), ('san', 'jacinto'), ('john', 'tyler'). In part2, some words pairs such as ('saudi', 'arabia'), ('gerald', 'r.'), ('jimmy', 'carter'), these words pairs just appear in top 50 bigrams scored by mutual information. And they do not appear both in top 50 frequent words and top 50 bigrams by frequency.

V. Appendix

Output:

1. Part1

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

```
('states', 2678)
('government', 2339)
('united', 1852)
('congress', 1500)
('country', 1427)
('public', 1375)
('great', 1286)
('state', 1156)
('power', 1150)
('year', 1145)
('present', 1027)
('duty', 1016)
('law', 969)
('subject', 936)
('time', 914)
('interest', 913)
('war', 897)
('act', 873)
('nation', 822)
('people', 786)
('citizen', 781)
('part', 744)
('treaty', 739)
('union', 643)
('mexico', 605)
('general', 604)
('give', 604)
('treasury', 592)
('amount', 574)
('constitution', 570)
('effect', 548)
('object', 538)
('foreign', 519)
('system', 509)
('commerce', 506)
('peace', 501)
('require', 500)
```

```
('measure', 497)
('receive', 493)
('consideration', 493)
('territory', 484)
('force', 483)
('service', 479)
('condition', 476)
('exist', 476)
('relation', 475)
('increase', 467)
('place', 461)
('revenue', 459)
('view', 458)
```

list the top 50 bigrams by frequencies

```
(('united', 'states'), 0.008851192067465428)
(('great', 'britain'), 0.0013318103385422996)
(('public', 'debt'), 0.0008797725229056796)
(('state', 'union'), 0.0008408875495175833)
(('house', 'representatives'), 0.0007193720076797823)
(('fiscal', 'year'), 0.0007047901426592461)
(('report', 'secretary'), 0.0007047901426592461)
(('union', 'address'), 0.0006999295209857341)
(('public', 'land'), 0.0006513233042506136)
(('act', 'congress'), 0.0006124383308625173)
(('present', 'year'), 0.0005200865190657885)
(('session', 'congress'), 0.0005152258973922765)
(('public', 'money'), 0.0005006440323717403)
(('secretary', 'treasury'), 0.00048120154567769216)
(('year', 'end'), 0.000461759058983644)
(('general', 'government'), 0.0004520378156366199)
(('british', 'government'), 0.0004471771939631079)
(('citizen', 'united'), 0.0004325953289425718)
(('annual', 'message'), 0.0004131528422485236)
(('federal', 'government'), 0.0004131528422485236)
(('secretary', 'war'), 0.00040829222057501153)
(('fellow', 'citizen'), 0.0004034315989014995)
(('consideration', 'congress'), 0.00039371035555544754)
(('public', 'service'), 0.00038884973388096335)
(('senate', 'house'), 0.00038884973388096335)
(('carry', 'effect'), 0.00036940724718691523)
(('government', 'united'), 0.00036940724718691523)
(('attention', 'congress'), 0.00035968600383989114)
(('part', 'united'), 0.0003305222737988189)
(('naval', 'force'), 0.0003208010304517948)
(('indian', 'tribe'), 0.0003159404087782827)
(('mexican', 'government'), 0.0003159404087782827)
(('treasury', 'note'), 0.0003159404087782827)
(('article', 'treaty'), 0.00030621916543125863)
(('commercial', 'intercourse'), 0.00030621916543125863)
(('secretary', 'state'), 0.0003013585437577466)
```

```
(('states', 'great'), 0.0002964979220842346)
(('claim', 'citizen'), 0.0002867766787372105)
(('public', 'interest'), 0.0002867766787372105)
(('american', 'citizen'), 0.00028191605706369844)
(('interest', 'country'), 0.00028191605706369844)
(('foreign', 'power'), 0.0002770554353901864)
(('address', 'december'), 0.00027219481371667435)
(('end', '30th'), 0.00027219481371667435)
(('favorable', 'consideration'), 0.00026733419204316234)
(('people', 'united'), 0.00026733419204316234)
(('30th', 'june'), 0.00026247357036965027)
(('bank', 'united'), 0.00026247357036965027)
(('secretary', 'navy'), 0.00026247357036965027)
(('exist', 'law'), 0.0002527523270226262)
```

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

```
(('bona', 'fide'), 15.328499628108926)
(('punta', 'arenas'), 15.328499628108926)
(('del', 'norte'), 15.065465222275126)
(('millard', 'fillmore'), 15.065465222275126)
(('ballot', 'box'), 14.843072800938682)
(('clayton', 'bulwer'), 14.650427722996284)
(('guadalupe', 'hidalgo'), 14.480502721553972)
(('porto', 'rico'), 14.480502721553972)
(('writ', 'mandamus'), 14.387393317162491)
(('franklin', 'pierce'), 14.328499628108922)
(('la', 'plata'), 14.190996104358986)
(('vera', 'cruz'), 14.065465222275126)
(('posse', 'comitatus'), 13.843072800938682)
(('costa', 'rica'), 13.650427722996284)
(('santa', 'anna'), 13.562964881745945)
(('santa', 'fe'), 13.562964881745945)
(('van', 'buren'), 13.562964881745945)
(('sublime', 'porte'), 13.521144706051317)
(('tea', 'coffee'), 13.402500209552699)
(('martin', 'van'), 13.393039880303633)
(('ad', 'valorem'), 13.32849962810892)
(('quincy', 'adams'), 13.190996104358984)
(('retired', 'list'), 13.126865766939273)
(('rocky', 'mountains'), 13.12686576693927)
(('ruler', 'universe'), 13.065465222275126)
(('buenos', 'ayres'), 13.065465222275124)
(('beacon', 'buoy'), 13.053492580609053)
(('barbary', 'powers'), 13.006571533221559)
(('indiana', 'illinois'), 12.927961698525195)
(('de', 'facto'), 12.917073382382458)
(('gun', 'boat'), 12.780063003412879)
(('andrew', 'jackson'), 12.760610640746705)
(('intent', 'meaning'), 12.70289514289042)
(('project', 'gutenberg'), 12.650427722996282)
(('john', 'quincy'), 12.562964881745943)
(('thomas', 'jefferson'), 12.475502040495604)
```

```
(('precious', 'metal'), 12.405423604759463)
(('almighty', 'god'), 12.393039880303633)
(('john', 'tyler'), 12.393039880303633)
(('san', 'jacinto'), 12.365025504134039)
(('san', 'juan'), 12.365025504134035)
(('san', 'francisco'), 12.365025504134033)
(('seizure', 'confiscation'), 12.328499628108924)
(('rio', 'grande'), 12.258110300217526)
(('effusion', 'blood'), 12.177939951533544)
(('seminary', 'learn'), 12.120606776467593)
(('lake', 'erie'), 12.027330093388358)
(('corps', 'engineers'), 11.936182205330159)
(('st.', 'augustine'), 11.922507268433087)
(('st.', 'lawrence'), 11.922507268433087)
```

2. Part2

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

```
('year', 2374)
('people', 1597)
('world', 1490)
('nation', 1360)
('america', 1271)
('congress', 1230)
('government', 1213)
('work', 1204)
('program', 1095)
('american', 950)
('time', 873)
('great', 868)
('country', 852)
('good', 752)
('federal', 744)
('war', 702)
('security', 689)
('americans', 688)
('tax', 686)
('million', 683)
('job', 674)
('economic', 671)
('peace', 668)
('increase', 665)
('continue', 640)
('united', 639)
('states', 626)
('economy', 622)
('state', 615)
('national', 610)
('child', 605)
('free', 574)
('give', 574)
```

```
('effort', 545)
('budget', 532)
('support', 532)
('system', 527)
('provide', 519)
('force', 517)
('freedom', 515)
('family', 513)
('life', 511)
('policy', 507)
('high', 502)
('health', 489)
('future', 484)
('union', 473)
('act', 470)
('tonight', 461)
('billion', 455)
```

list the top 50 bigrams by frequencies

```
(('united', 'states'), 0.002454874413513499)
(('state', 'union'), 0.0014187261220954639)
(('american', 'people'), 0.0012699458546097972)
(('fiscal', 'year'), 0.0010361482914180353)
(('past', 'year'), 0.0010148939674915116)
(('year', 'ago'), 0.0010148939674915116)
(('federal', 'government'), 0.0009830124816017259)
(('social', 'security'), 0.000967071738656833)
(('health', 'care'), 0.0009458174147303092)
(('billion', 'dollar'), 0.0008129778901895355)
(('union', 'address'), 0.0007332741754650712)
(('united', 'nations'), 0.0007173334325201783)
(('million', 'dollar'), 0.0007013926895752854)
(('soviet', 'union'), 0.0006641976227038688)
(('men', 'woman'), 0.0006004346509242974)
(('free', 'world'), 0.0005898074889610355)
(('economic', 'growth'), 0.0004994766122733093)
(('middle', 'east'), 0.0004835358693284165)
(('free', 'nation'), 0.0004782222288346785 57)
(('small', 'business'), 0.0004357136404937 3797)
(('world', 'war'), 0.0004357136404937 3797)
(('state', 'local'), 0.0004304000595121 07)
(('million', 'americans'), 0.0004144593165672141)
(('tax', 'cut'), 0.0004038321546039522)
(('foreign', 'policy'), 0.0003932049926406903)
(('21st', 'century'), 0.0003878914116590594)
(('mr.', 'speaker'), 0.0003719506687141665)
(('create', 'job'), 0.0003613235067509046)
(('vice', 'president'), 0.0003560099257692737)
(('year', 'year'), 0.0003506963447876427)
(('local', 'government'), 0.0003453827638060118)
(('urge', 'congress'), 0.0003453827638060118)
(('national', 'security'), 0.0003347556018427499)
(('address', 'january'), 0.0003188148588978 57)
```

```
(('nation', 'world'), 0.0003188148588978 57)
(('health', 'insurance'), 0.0003081876969345951)
(('tax', 'credit'), 0.0003028741159529642)
(('fellow', 'americans'), 0.00029756053497133323)
(('fellow', 'citizen'), 0.00029756053497133323)
(('states', 'america'), 0.00029756053497133323)
(('high', 'school'), 0.00028693337300807133)
(('nuclear', 'weapon'), 0.00028693337300807133)
(('private', 'sector'), 0.00028693337300807133)
(('young', 'people'), 0.00028693337300807133)
(('god', 'bless'), 0.00027630621104480943)
(('interest', 'rate'), 0.00027630621104480943)
(('minimum', 'wage'), 0.00027099263006317845)
(('members', 'congress'), 0.00026567904908154753)
(('balanced', 'budget'), 0.00026036546809991655)
(('armed', 'force'), 0.00025505188711828563)
```

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

```
(('el', 'salvador'), 14.93692160433461)
(('bin', 'laden'), 14.714529182998163)
(('saudi', 'arabia'), 14.71452918299816)
(('australia', 'zealand'), 14.521884105055765)
(('sam', 'rayburn'), 14.521884105055765)
(('gerald', 'r.'), 14.299491683719317)
(('jimmy', 'carter'), 14.199956010168403)
(('endow', 'creator'), 14.088924697779657)
(('vol', 'p.'), 14.062452486418469)
(('northern', 'ireland'), 13.93692160433461)
(("o'neill", 'jr.'), 13.866532276443209)
(('r.', 'ford'), 13.840060065082021)
(('lyndon', 'b.'), 13.82144438691467)
(('iron', 'curtain'), 13.614993509447249)
(('william', 'j.'), 13.614993509447249)
(('thomas', 'jefferson'), 13.558409981080878)
(('red', 'tape'), 13.521884105055767)
(('200th', 'anniversary'), 13.477489985697314)
(('jill', 'biden'), 13.451494777164367)
(('b.', 'johnson'), 13.434421263805424)
(('barack', 'obama'), 13.434421263805424)
(('teen', 'pregnancy'), 13.299491683719317)
(('abraham', 'lincoln'), 13.264496262363114)
(('p.', "o'neill"), 13.214455579863518)
(('j.', 'clinton'), 13.199956010168405)
(('mom', 'dad'), 13.129566682277005)
(('ronald', 'reagan'), 13.062452486418467)
(('greece', 'turkey'), 12.977563588831956)
(('weapons', 'convention'), 12.948994436635182)
(('elementary', 'secondary'), 12.895444968358449)
(('harry', 's.'), 12.858919092333334)
(('small-business', 'owner'), 12.8520327067481)
(('dwight', 'd.'), 12.799418080584672)
```

```
('intercontinental', 'ballistic'), 12.775929727662305)
('h.w', 'bush'), 12.766996602892299)
('w.', 'bush'), 12.766996602892295)
('old-age', 'survivor'), 12.697455669639218)
('thomas', 'p.'), 12.683940863164738)
('status', 'quo'), 12.663903109928192)
('nationwide', 'radio'), 12.5743515249499)
('radio', 'television'), 12.521884105055765)
('f.', 'kennedy'), 12.434421263805426)
('prime', 'minister'), 12.3926010881108)
('al', 'qaida'), 12.392601088110798)
('al', 'qaeda'), 12.392601088110796)
('richard', 'nixon'), 12.375527574751858)
('george', 'h.w'), 12.351959103613453)
('saddam', 'hussein'), 12.351959103613453)
('george', 'w.'), 12.35195910361345)
('d.', 'eisenhower'), 12.339986461947376)
```

Python processing screenshot：

Part1 Analysis:

```
>>> import nltk
>>> from nltk import FreqDist
>>> from nltk.corpus import PlaintextCorpusReader
>>> import re
>>> from nltk.collocations import *
>>> from nltk import word_tokenize, pos_tag
>>> from nltk.corpus import wordnet
>>> from nltk.stem import WordNetLemmatizer
>>> 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
...
>>> def get_part_content(filename):
...     # get text file content and return the string
...     corpus = PlaintextCorpusReader('..', '.*\.txt')
...     content = corpus.raw(filename)
...     return content
...
```

```
>>> def get_word_pos(word_pos_tag):
...     # get the words pos tags from pos_tag and return the corresponding wordn
et tags
...     if word_pos_tag.startswith('J'):
...         return wordnet.ADJ
...     elif word_pos_tag.startswith('V'):
...         return wordnet.VERB
...     elif word_pos_tag.startswith('N'):
...         return wordnet.NOUN
...     elif word_pos_tag.startswith('R'):
...         return wordnet.ADV
...     else:
[...         return None                                                        ]
[...                                                                           ]
>>> def do_word_tokenize(content):
...     # do word tokenizing process
...     tokens = nltk.word_tokenize(content)
[...     return tokens                                                         ]
[...                                                                           ]
```

```
>>> def do_lower(content):
...     # set all words as lowercase
...     words = [w.lower() for w in content]
...     return words
...
>>> def get_alphabetical_words(content):
...     # get alphabetical words
...     alpha_words = [w for w in content if not alpha_filter(w)]
...     return alpha_words
...
>>> def get_stopwords():
...     # get stopwords
...     stop_words = nltk.corpus.stopwords.words('english')
...     stop = open('../Smart.English.stop', 'r')
...     stop_text = stop.read()
...     stop.close()
...     c_stopwords = nltk.word_tokenize(stop_text)
...     c_stopwords.extend(["'m", "n't", "'s", "make"])
...     stop_words.extend(c_stopwords)
...     stop_words = list(set(stop_words))
...     return stop_words
...
```

```
>>> def filter_content(content, filter_words):
...     # filter special words in content
...     filtered_words = [w for w in content if w not in filter_words]
...     return filtered_words
...
>>> def do_print_top_50(content):
...     # print top 50 frequency
...     freq_dist = FreqDist(content)
...     top_keys = freq_dist.most_common(50)
...     for pair in top_keys:
...         print(pair)
...     print('-----------------------------------')
...
>>> def get_bi_gram_association_measures():
...     # get bigram measures
...     measures = nltk.collocations.BigramAssocMeasures()
...     return measures


>>> def finder_filter(content):
...     # apply filter to finder
...     finder = BigramCollocationFinder.from_words(content)
...     finder.apply_freq_filter(5)
...     return finder
...
>>> def print_bigram_score(bigram_measures, finder):
...     # print scores which are sorted into order by decreasing frequency
...     scored = finder.score_ngrams(bigram_measures.raw_freq)
...     for one_score in scored[:50]:
...         print(one_score)
...     print('-----------------------------------')
...
>>> def print_PMI_measures(bigram_measures, finder):
...     # print PMI measures scores
...     scored = finder.score_ngrams(bigram_measures.pmi)
...     for one_score in scored[:50]:
...         print(one_score)
...


>>> onePartContent = get_part_content('state_union_part1.txt')
>>> tokens_content = do_word_tokenize(onePartContent)
>>> words_tags = pos_tag(tokens_content)
>>> word_net_le = WordNetLemmatizer()
>>> lemmatization_words = []
>>> for tag in words_tags:
...     word_pos = get_word_pos(tag[1]) or wordnet.NOUN
...     lemmatization_words.append(word_net_le.lemmatize(tag[0], pos=word_pos))
...

>>>
>>> lower_words = do_lower(lemmatization_words)
>>> alphabetical_words = get_alphabetical_words(lower_words)
>>> stopwords = get_stopwords()
>>> filter_stopwords = filter_content(alphabetical_words, stopwords)
>>> do_print_top_50(filter_stopwords)
```

```
('states', 2678)
('government', 2339)
('united', 1852)
('congress', 1500)
('country', 1427)
('public', 1375)
('great', 1286)
('state', 1156)
('power', 1150)
('year', 1145)
('present', 1027)
('duty', 1016)
('law', 969)
('subject', 936)
('time', 914)
('interest', 913)
('war', 897)
('act', 873)
('nation', 822)
('people', 786)
('citizen', 781)
('part', 744)
('treaty', 739)
('union', 643)
('mexico', 605)
('general', 604)
('give', 604)
('treasury', 592)
('amount', 574)
('constitution', 570)
('effect', 548)
('object', 538)
('foreign', 519)
('system', 509)
('commerce', 506)
('peace', 501)
('require', 500)
('measure', 497)
('receive', 493)
('consideration', 493)
('territory', 484)
('force', 483)
('service', 479)
('condition', 476)
('exist', 476)
('relation', 475)
('increase', 467)
('place', 461)
('revenue', 459)
('view', 458)
```

```
>>> bi_measures = get_bi_gram_association_measures()
>>> finder = BigramCollocationFinder.from_words(filter_stopwords)
>>> print_bigram_score(bi_measures, finder)
(('united', 'states'), 0.008851192067465428)
(('great', 'britain'), 0.0013318103385422996)
(('public', 'debt'), 0.0008797725229056796)
(('state', 'union'), 0.0008408875495175833)
(('house', 'representatives'), 0.0007193720076797823)
(('fiscal', 'year'), 0.0007047901426592461)
(('report', 'secretary'), 0.0007047901426592461)
(('union', 'address'), 0.0006999295209857341)
(('public', 'land'), 0.0006513233042506136)
(('act', 'congress'), 0.0006124383308625173)
(('present', 'year'), 0.0005200865190657885)
(('session', 'congress'), 0.0005152258973922765)
(('public', 'money'), 0.0005006440323717403)
(('secretary', 'treasury'), 0.0004812015456769216)
(('year', 'end'), 0.000461759058983644)
(('general', 'government'), 0.0004520378156366199)
(('british', 'government'), 0.0004471771939631079)
(('citizen', 'united'), 0.0004325953289425718)
(('annual', 'message'), 0.0004131528422485236)
(('federal', 'government'), 0.0004131528422485236)
(('secretary', 'war'), 0.00040829222057501153)
(('fellow', 'citizen'), 0.0004034315989014995)
(('consideration', 'congress'), 0.00039371035555544754)
(('public', 'service'), 0.00038884973388096335)
(('senate', 'house'), 0.00038884973388096335)
(('carry', 'effect'), 0.00036940724718691523)
(('government', 'united'), 0.00036940724718691523)
(('attention', 'congress'), 0.00035968600383989114)
(('part', 'united'), 0.0003305222737988189)
(('naval', 'force'), 0.0003208010304517948)
(('indian', 'tribe'), 0.0003159404087782827)
(('mexican', 'government'), 0.0003159404087782827)
(('treasury', 'note'), 0.0003159404087782827)
(('article', 'treaty'), 0.00030621916543125863)
(('commercial', 'intercourse'), 0.00030621916543125863)
(('secretary', 'state'), 0.0003013585437577466)
(('states', 'great'), 0.0002964979220842346)
(('claim', 'citizen'), 0.0002867766787372105)
(('public', 'interest'), 0.0002867766787372105)
(('american', 'citizen'), 0.00028191605706369844)
(('interest', 'country'), 0.00028191605706369844)
(('foreign', 'power'), 0.0002770554353901864)
(('address', 'december'), 0.00027219481371667435)
(('end', '30th'), 0.00027219481371667435)
(('favorable', 'consideration'), 0.00026733419204316234)
(('people', 'united'), 0.00026733419204316234)
(('30th', 'june'), 0.00026247357036965027)
(('bank', 'united'), 0.00026247357036965027)
(('secretary', 'navy'), 0.00026247357036965027)
(('exist', 'law'), 0.0002527523270226262)
```

```
>>> finder1 = finder_filter(filter_stopwords)
>>> print_PMI_measures(bi_measures, finder1)
(('bona', 'fide'), 15.328499628108926)
(('punta', 'arenas'), 15.328499628108926)
(('del', 'norte'), 15.065465222275126)
(('millard', 'fillmore'), 15.065465222275126)
(('ballot', 'box'), 14.843072800938682)
(('clayton', 'bulwer'), 14.650427722996284)
(('guadalupe', 'hidalgo'), 14.480502721553972)
(('porto', 'rico'), 14.480502721553972)
(('writ', 'mandamus'), 14.387393317162491)
(('franklin', 'pierce'), 14.328499628108922)
(('la', 'plata'), 14.190996104358986)
(('vera', 'cruz'), 14.065465222275126)
(('posse', 'comitatus'), 13.843072800938682)
(('costa', 'rica'), 13.650427722996284)
(('santa', 'anna'), 13.562964881745945)
(('santa', 'fe'), 13.562964881745945)
(('van', 'buren'), 13.562964881745945)
(('sublime', 'porte'), 13.521144706051317)
(('tea', 'coffee'), 13.402500209552699)
(('martin', 'van'), 13.393039880303633)
(('ad', 'valorem'), 13.32849962810892)
(('quincy', 'adams'), 13.190996104358984)
(('retired', 'list'), 13.126865766939273)
(('rocky', 'mountains'), 13.12686576693927)
(('ruler', 'universe'), 13.065465222275126)
(('buenos', 'ayres'), 13.065465222275124)
(('beacon', 'buoy'), 13.053492580609053)
(('barbary', 'powers'), 13.006571533221559)
(('indiana', 'illinois'), 12.927961698525195)
(('de', 'facto'), 12.917073382382458)
(('gun', 'boat'), 12.780063003412879)
(('andrew', 'jackson'), 12.760610640746705)
(('intent', 'meaning'), 12.70289514289042)
(('project', 'gutenberg'), 12.650427722996282)
(('john', 'quincy'), 12.562964881745943)
(('thomas', 'jefferson'), 12.475502040495604)
(('precious', 'metal'), 12.405423604759463)
(('almighty', 'god'), 12.393039880303633)
(('john', 'tyler'), 12.393039880303633)
(('san', 'jacinto'), 12.365025504134039)
(('san', 'juan'), 12.365025504134035)
(('san', 'francisco'), 12.365025504134033)
(('seizure', 'confiscation'), 12.328499628108924)
(('rio', 'grande'), 12.258110300217526)
(('effusion', 'blood'), 12.177939951533544)
(('seminary', 'learn'), 12.120606776467593)
(('lake', 'erie'), 12.027330093388358)
(('corps', 'engineers'), 11.936182205330159)
(('st.', 'augustine'), 11.922507268433087)
(('st.', 'lawrence'), 11.922507268433087)
```

Part 2 Analysis:

```
>>> import nltk
>>> from nltk import FreqDist
>>> from nltk.corpus import PlaintextCorpusReader
>>> import re
>>> from nltk.collocations import *
>>> from nltk import word_tokenize, pos_tag
>>> from nltk.corpus import wordnet
>>> from nltk.stem import WordNetLemmatizer
>>> 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
...
>>> def get_part_content(filename):
...     # get text file content and return the string
...     corpus = PlaintextCorpusReader('..', '.*\.txt')
...     content = corpus.raw(filename)
...     return content
...
```

```
>>> def get_word_pos(word_pos_tag):
...     # get the words pos tags from pos_tag and return the corresponding wordn
et tags
...     if word_pos_tag.startswith('J'):
...         return wordnet.ADJ
...     elif word_pos_tag.startswith('V'):
...         return wordnet.VERB
...     elif word_pos_tag.startswith('N'):
...         return wordnet.NOUN
...     elif word_pos_tag.startswith('R'):
...         return wordnet.ADV
...     else:
...         return None
...
>>> def do_word_tokenize(content):
...     # do word tokenizing process
...     tokens = nltk.word_tokenize(content)
...     return tokens
...
>>> def do_lower(content):
...     # set all words as lowercase
...     words = [w.lower() for w in content]
...     return words
...
>>> def get_alphabetical_words(content):
...     # get alphabetical words
...     alpha_words = [w for w in content if not alpha_filter(w)]
...     return alpha_words
...
```

```
>>> def get_stopwords():
...     # get stopwords
...     stop_words = nltk.corpus.stopwords.words('english')
...     stop = open('../Smart.English.stop', 'r')
...     stop_text = stop.read()
...     stop.close()
...     c_stopwords = nltk.word_tokenize(stop_text)
...     c_stopwords.extend(["'m", "n't", "'s", "make"])
...     stop_words.extend(c_stopwords)
...     stop_words = list(set(stop_words))
...     return stop_words
...
>>> def filter_content(content, filter_words):
...     # filter special words in content
...     filtered_words = [w for w in content if w not in filter_words]
...     return filtered_words
...
>>> def do_print_top_50(content):
...     # print top 50 frequency
...     freq_dist = FreqDist(content)
...     top_keys = freq_dist.most_common(50)
...     for pair in top_keys:
...         print(pair)
...     print('-----------------------------------')
...
>>> def get_bi_gram_association_measures():
...     # get bigram measures
...     measures = nltk.collocations.BigramAssocMeasures()
...     return measures
```

```
>>> def finder_filter(content):
...     # apply filter to finder
...     finder = BigramCollocationFinder.from_words(content)
...     finder.apply_freq_filter(5)
...     return finder
...
>>> def print_bigram_score(bigram_measures, finder):
...     # print scores which are sorted into order by decreasing frequency
...     scored = finder.score_ngrams(bigram_measures.raw_freq)
...     for one_score in scored[:50]:
...         print(one_score)
...     print('-----------------------------------')
...
>>> def print_PMI_measures(bigram_measures, finder):
...     # print PMI measures scores
...     scored = finder.score_ngrams(bigram_measures.pmi)
...     for one_score in scored[:50]:
...         print(one_score)
...
```

```
>>> onePartContent = get_part_content('state_union_part2.txt')
>>> tokens_content = do_word_tokenize(onePartContent)
>>> words_tags = pos_tag(tokens_content)
>>> word_net_le = WordNetLemmatizer()
>>> lemmatization_words = []
>>> for tag in words_tags:
...     word_pos = get_word_pos(tag[1]) or wordnet.NOUN
...     lemmatization_words.append(word_net_le.lemmatize(tag[0], pos=word_pos))
...
>>> lower_words = do_lower(lemmatization_words)
>>> alphabetical_words = get_alphabetical_words(lower_words)
>>> stopwords = get_stopwords()
>>> filter_stopwords = filter_content(alphabetical_words, stopwords)
>>> do_print_top_50(filter_stopwords)
```

```
('year', 2374)
('people', 1597)
('world', 1490)
('nation', 1360)
('america', 1271)
('congress', 1230)
('government', 1213)
('work', 1204)
('program', 1095)
('american', 950)
('time', 873)
('great', 868)
('country', 852)
('good', 752)
('federal', 744)
('war', 702)
('security', 689)
('americans', 688)
('tax', 686)
('million', 683)
('job', 674)
('economic', 671)
('peace', 668)
('increase', 665)
('continue', 640)
('united', 639)
('states', 626)
('economy', 622)
('state', 615)
('national', 610)
('child', 605)
('free', 574)
('give', 574)
('effort', 545)
('budget', 532)
('support', 532)
('system', 527)
('provide', 519)
('force', 517)
('freedom', 515)
('family', 513)
('life', 511)
('policy', 507)
('high', 502)
('health', 489)
('future', 484)
('union', 473)
('act', 470)
('tonight', 461)
('billion', 455)
----------------------------------
```

```
>>> bi_measures = get_bi_gram_association_measures()
>>> finder = BigramCollocationFinder.from_words(filter_stopwords)
>>> print_bigram_score(bi_measures, finder)
(('united', 'states'), 0.002454874413513499)
(('state', 'union'), 0.0014187261220954639)
(('american', 'people'), 0.0012699458546097972)
(('fiscal', 'year'), 0.0010361482914180353)
(('past', 'year'), 0.0010148939674915116)
(('year', 'ago'), 0.0010148939674915116)
(('federal', 'government'), 0.0009830124816017259)
(('social', 'security'), 0.000967071738656833)
(('health', 'care'), 0.0009458174147303092)
(('billion', 'dollar'), 0.0008129778901895355)
(('union', 'address'), 0.0007332741754650712)
(('united', 'nations'), 0.0007173334325201783)
(('million', 'dollar'), 0.0007013926895752854)
(('soviet', 'union'), 0.0006641976227038688)
(('men', 'woman'), 0.0006004346509242974)
(('free', 'world'), 0.0005898074889610355)
(('economic', 'growth'), 0.0004994766122733093)
(('middle', 'east'), 0.0004835358693284165)
(('free', 'nation'), 0.0004782228834678557)
(('small', 'business'), 0.0004357136404937797)
(('world', 'war'), 0.0004357136404937797)
(('state', 'local'), 0.000430400059512107)
(('million', 'americans'), 0.0004144593165672141)
(('tax', 'cut'), 0.0004038321546039522)
(('foreign', 'policy'), 0.0003932049926406903)
(('21st', 'century'), 0.0003878914116590594)
(('mr.', 'speaker'), 0.0003719506687141665)
(('create', 'job'), 0.0003613235067509046)
(('vice', 'president'), 0.0003560099257692737)
(('year', 'year'), 0.0003506963447876427)
(('local', 'government'), 0.0003453827638060118)
(('urge', 'congress'), 0.0003453827638060118)
(('national', 'security'), 0.0003347556018427499)
(('address', 'january'), 0.000318814858897857)
(('nation', 'world'), 0.000318814858897857)
(('health', 'insurance'), 0.0003081876969345951)
(('tax', 'credit'), 0.0003028741159529642)
(('fellow', 'americans'), 0.00029756053497133323)
(('fellow', 'citizen'), 0.0002975605349713323)
(('states', 'america'), 0.0002975605349713323)
(('high', 'school'), 0.00028693337300807133)
(('nuclear', 'weapon'), 0.00028693337300807133)
(('private', 'sector'), 0.00028693337300807133)
(('young', 'people'), 0.00028693337300807133)
(('god', 'bless'), 0.00027630621104480943)
(('interest', 'rate'), 0.00027630621104480943)
(('minimum', 'wage'), 0.00027099263006317845)
(('members', 'congress'), 0.00026567904908154753)
(('balanced', 'budget'), 0.00026036546809991655)
(('armed', 'force'), 0.00025505188711828563)
----------------------------------
```

```
>>> finder1 = finder_filter(filter_stopwords)
>>> print_PMI_measures(bi_measures, finder1)
(('el', 'salvador'), 14.93692160433461)
(('bin', 'laden'), 14.714529182998163)
(('saudi', 'arabia'), 14.71452918299816)
(('australia', 'zealand'), 14.521884105055765)
(('sam', 'rayburn'), 14.521884105055765)
(('gerald', 'r.'), 14.299491683719317)
(('jimmy', 'carter'), 14.199956010168403)
(('endow', 'creator'), 14.088924697779657)
(('vol', 'p.'), 14.062452486418469)
(('northern', 'ireland'), 13.93692160433461)
(("o'neill", 'jr.'), 13.866532276443209)
(('r.', 'ford'), 13.840060065082021)
(('lyndon', 'b.'), 13.82144438691467)
(('iron', 'curtain'), 13.614993509447249)
(('william', 'j.'), 13.614993509447249)
(('thomas', 'jefferson'), 13.558409981080878)
(('red', 'tape'), 13.521884105055767)
(('200th', 'anniversary'), 13.477489985697314)
(('jill', 'biden'), 13.451494777164367)
(('b.', 'johnson'), 13.434421263805424)
(('barack', 'obama'), 13.434421263805424)
(('teen', 'pregnancy'), 13.299491683719317)
(('abraham', 'lincoln'), 13.264496262363114)
(('p.', "o'neill"), 13.214455579863518)
(('j.', 'clinton'), 13.199956010168405)
(('mom', 'dad'), 13.129566682277005)
(('ronald', 'reagan'), 13.062452486418467)
(('greece', 'turkey'), 12.977563588831956)
(('weapons', 'convention'), 12.948994436635182)
(('elementary', 'secondary'), 12.895444968358449)
(('harry', 's.'), 12.858919092333334)
(('small-business', 'owner'), 12.8520327067481)
(('dwight', 'd.'), 12.799418080584672)
(('intercontinental', 'ballistic'), 12.775929727662305)
(('h.w', 'bush'), 12.766996602892299)
(('w.', 'bush'), 12.766996602892295)
(('old-age', 'survivor'), 12.697455669639218)
(('thomas', 'p.'), 12.683940863164738)
(('status', 'quo'), 12.663903109928192)
(('nationwide', 'radio'), 12.5743515249499)
(('radio', 'television'), 12.521884105055765)
(('f.', 'kennedy'), 12.434421263805426)
(('prime', 'minister'), 12.3926010881108)
(('al', 'qaida'), 12.392601088110798)
(('al', 'qaeda'), 12.392601088110796)
(('richard', 'nixon'), 12.375527574751858)
(('george', 'h.w'), 12.351959103613453)
(('saddam', 'hussein'), 12.351959103613453)
(('george', 'w.'), 12.35195910361345)
(('d.', 'eisenhower'), 12.339986461947376)
```