Name: Shubham Nagesh Gaikwad

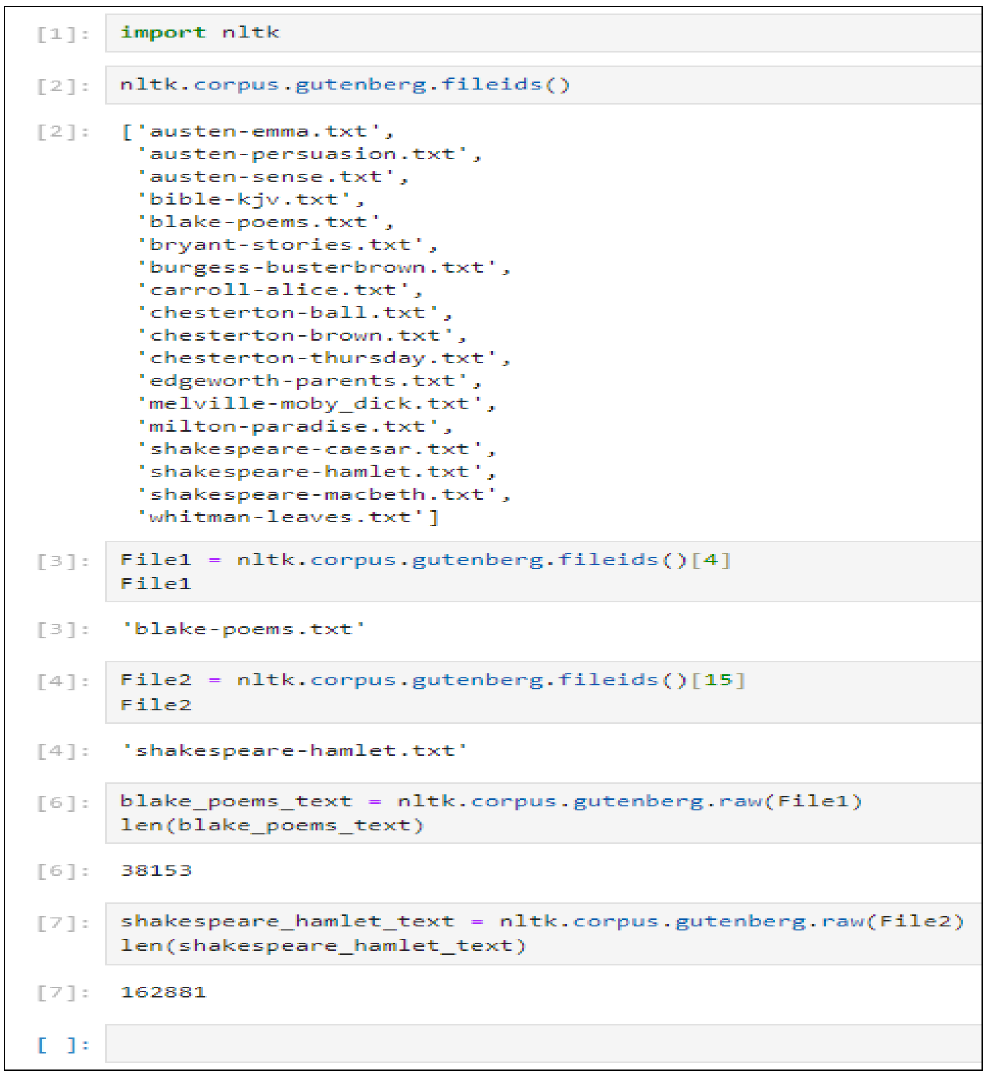
SUID: 216327540

Assignment 1: Comparing Corpora with Corpus Statistics

1.

Phase 1: Choosing documents from the Gutenberg collection on the web.

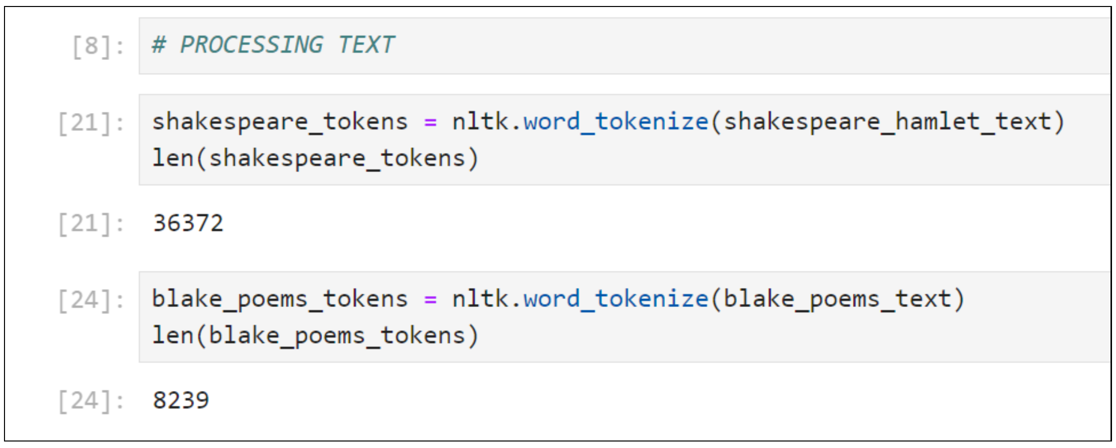
I chose 2 documents, which are different in text wise, 'blake-poems.txt' and ''shakespeare-hamlet.txt'. These two documents are different in character in topic style, genre aspect.



```
[1]: import nltk

[2]: nltk.corpus.gutenberg.fileids()

[2]: ['austen-emma.txt',
      'austen-persuasion.txt',
      'austen-sense.txt',
      'bible-kjv.txt',
      'blake-poems.txt',
      'bryant-stories.txt',
      'burgess-busterbrown.txt',
      'carroll-alice.txt',
      'chesterton-ball.txt',
      'chesterton-brown.txt',
      'chesterton-thursday.txt',
      'edgeworth-parents.txt',
      'melville-moby_dick.txt',
      'milton-paradise.txt',
      'shakespeare-caesar.txt',
      'shakespeare-hamlet.txt',
      'shakespeare-macbeth.txt',
      'whitman-leaves.txt']

[3]: File1 = nltk.corpus.gutenberg.fileids()[4]
     File1

[3]: 'blake-poems.txt'

[4]: File2 = nltk.corpus.gutenberg.fileids()[15]
     File2

[4]: 'shakespeare-hamlet.txt'

[6]: blake_poems_text = nltk.corpus.gutenberg.raw(File1)
     len(blake_poems_text)

[6]: 38153

[7]: shakespeare_hamlet_text = nltk.corpus.gutenberg.raw(File2)
     len(shakespeare_hamlet_text)

[7]: 162881

[ ]:
```

2.

Phase 2: Examining the text in both documents and deciding how to process the words. [Tokenization]. Deciding whether to use all lower case, stopwords, lemmatization.

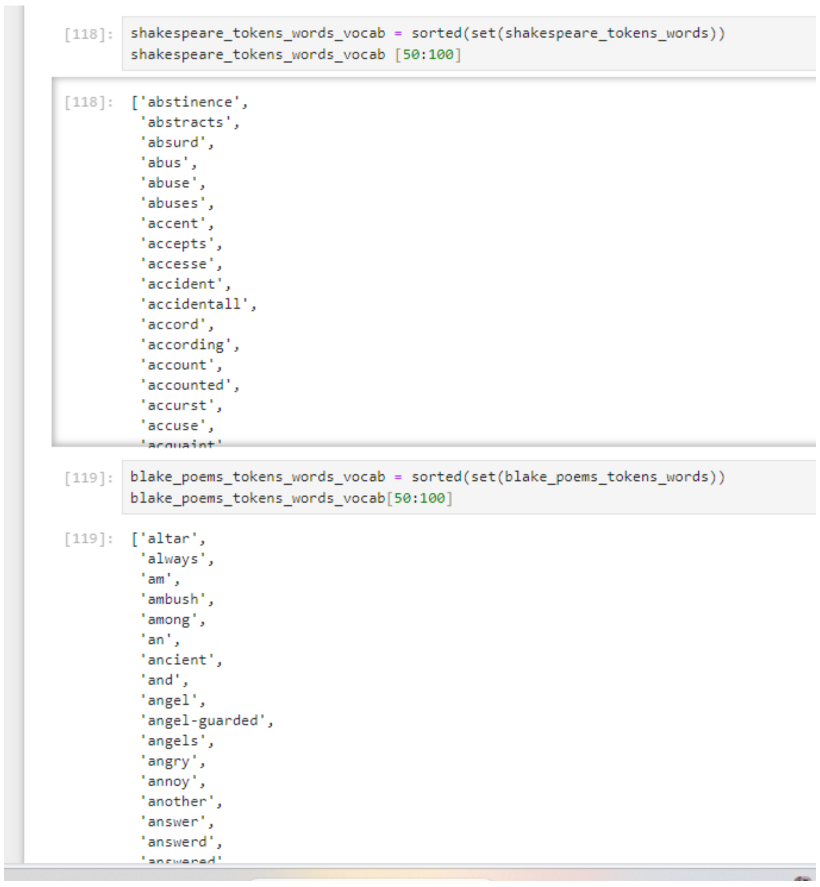


```
[8]:  # PROCESSING TEXT

[21]: shakespeare_tokens = nltk.word_tokenize(shakespeare_hamlet_text)
      len(shakespeare_tokens)

[21]: 36372

[24]: blake_poems_tokens = nltk.word_tokenize(blake_poems_text)
      len(blake_poems_tokens)

[24]: 8239
```

Total tokens of 'shakespeare-hamlet.txt' = 36372

Total tokens of ‘blake-poems.txt’ = 8239

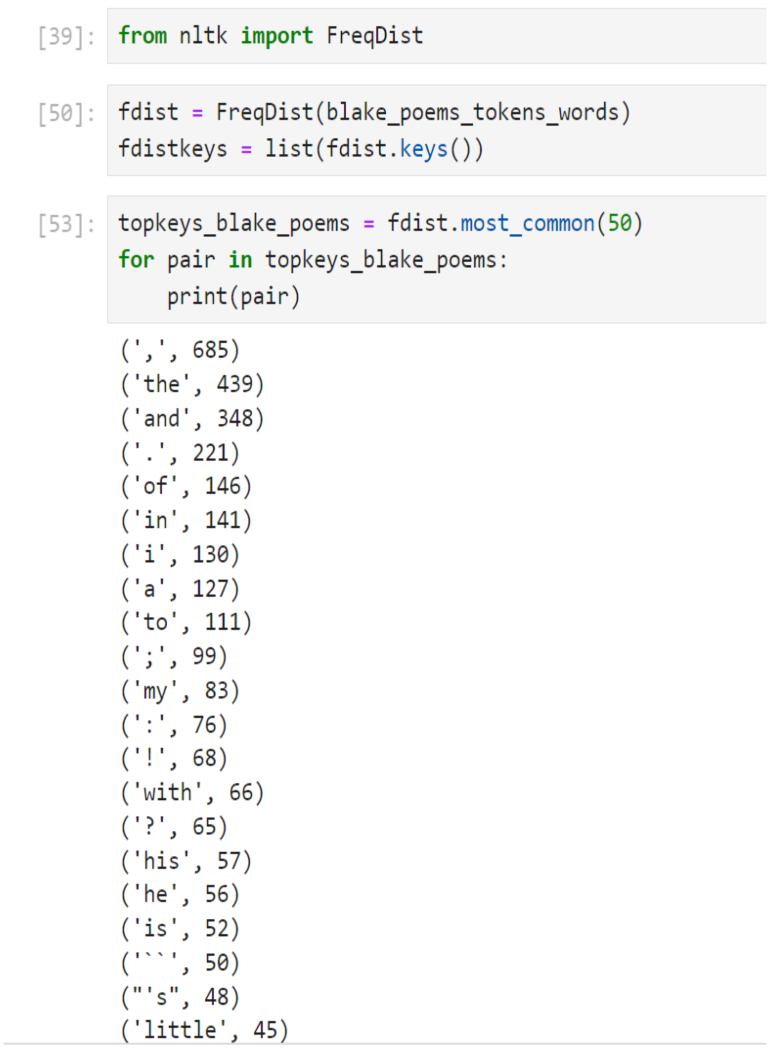
Case lowered but it was still not helping to understand the text hence used sorting.



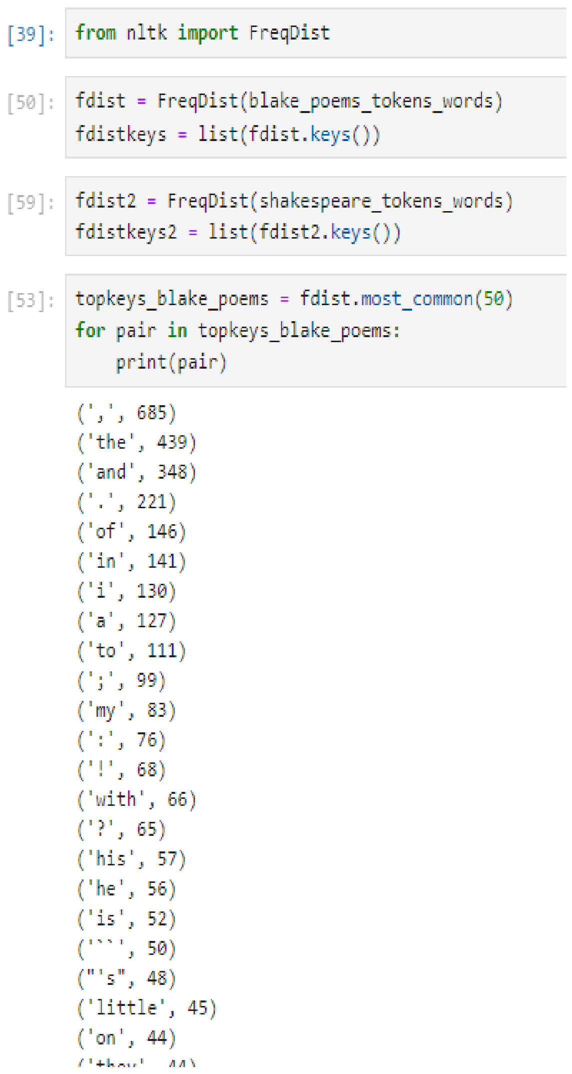
```
[118]: shakespeare_tokens_words_vocab = sorted(set(shakespeare_tokens_words))
       shakespeare_tokens_words_vocab [50:100]

[118]: ['abstinence',
        'abstracts',
        'absurd',
        'abus',
        'abuse',
        'abuses',
        'accent',
        'accepts',
        'accesse',
        'accident',
        'accidentall',
        'accord',
        'according',
        'account',
        'accounted',
        'accurst',
        'accuse',

[119]: blake_poems_tokens_words_vocab = sorted(set(blake_poems_tokens_words))
       blake_poems_tokens_words_vocab[50:100]

[119]: ['altar',
        'always',
        'am',
        'ambush',
        'among',
        'an',
        'ancient',
        'and',
        'angel',
        'angel-guarded',
        'angels',
        'angry',
        'annoy',
        'another',
        'answer',
        'answerd',
```

We are not getting enough idea from just sorted list of words from both texts hence we need to find bigram distribution to know more about text.

Phase 3:

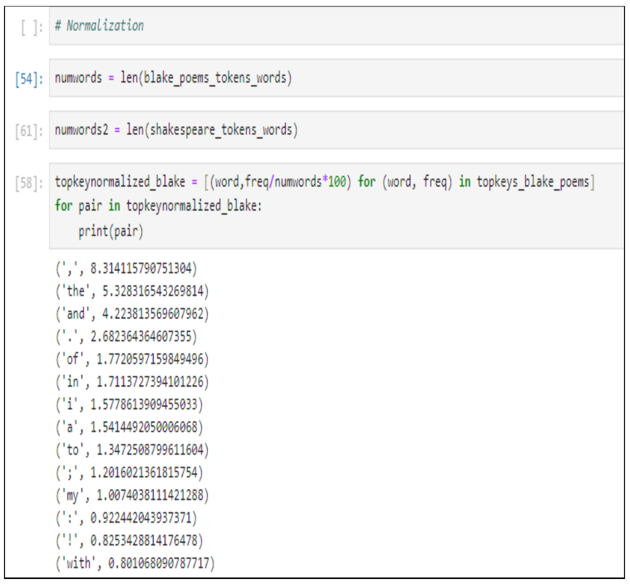


```
[39]: from nltk import FreqDist

[50]: fdist = FreqDist(blake_poems_tokens_words)
      fdistkeys = list(fdist.keys())

[53]: topkeys_blake_poems = fdist.most_common(50)
      for pair in topkeys_blake_poems:
          print(pair)

      (',', 685)
      ('the', 439)
      ('and', 348)
      ('.', 221)
      ('of', 146)
      ('in', 141)
      ('i', 130)
      ('a', 127)
      ('to', 111)
      (';', 99)
      ('my', 83)
      (':', 76)
      ('!', 68)
      ('with', 66)
      ('?', 65)
      ('his', 57)
      ('he', 56)
      ('is', 52)
      ('``', 50)
      ("'s", 48)
      ('little', 45)
```

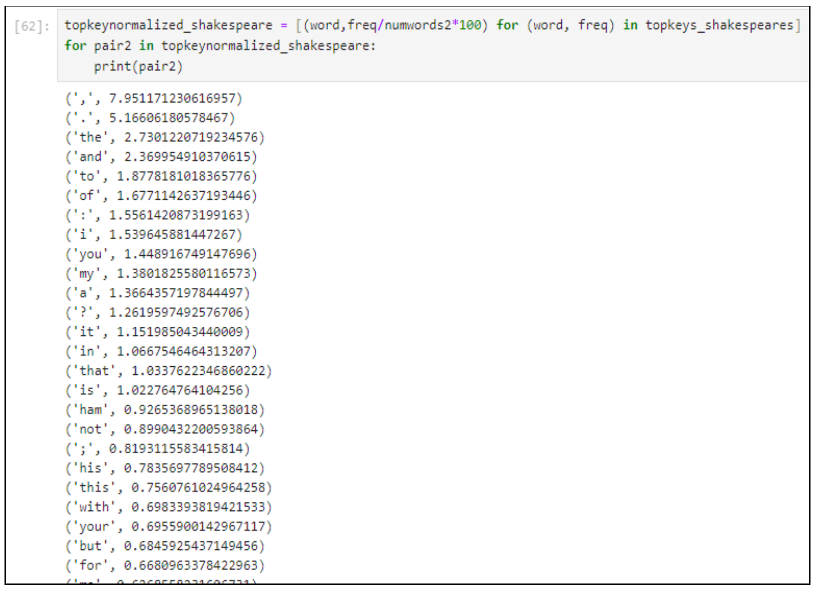


```
[39]: from nltk import FreqDist

[50]: fdist = FreqDist(blake_poems_tokens_words)
      fdistkeys = list(fdist.keys())

[59]: fdist2 = FreqDist(shakespeare_tokens_words)
      fdistkeys2 = list(fdist2.keys())

[53]: topkeys_blake_poems = fdist.most_common(50)
      for pair in topkeys_blake_poems:
          print(pair)

      (',', 685)
      ('the', 439)
      ('and', 348)
      ('.', 221)
      ('of', 146)
      ('in', 141)
      ('i', 130)
      ('a', 127)
      ('to', 111)
      (';', 99)
      ('my', 83)
      (':', 76)
      ('!', 68)
      ('with', 66)
      ('?', 65)
      ('his', 57)
      ('he', 56)
      ('is', 52)
      ('``', 50)
      ("'s", 48)
      ('little', 45)
      ('on', 44)
```

We need to normalize the text for both documents to have proper comparison between them.



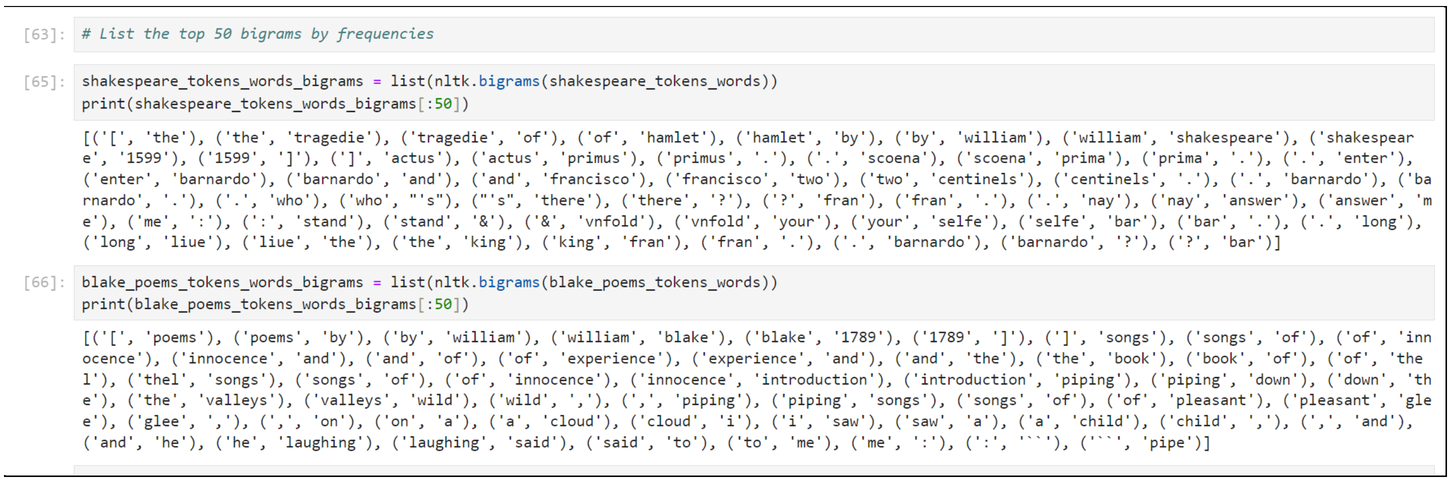
```
[ ]: # Normalization

[54]: numwords = len(blake_poems_tokens_words)

[61]: numwords2 = len(shakespeare_tokens_words)

[58]: topkeynormalized_blake = [(word,freq/numwords*100) for (word, freq) in topkeys_blake_poems]
      for pair in topkeynormalized_blake:
          print(pair)

      (',', 8.314115790751304)
      ('the', 5.328316543269814)
      ('and', 4.223813569607962)
      ('.', 2.682364364607355)
      ('of', 1.7720597159849496)
      ('in', 1.7113727394101226)
      ('i', 1.5778613909455033)
      ('a', 1.5414492050006068)
      ('to', 1.3472508799611604)
      (';', 1.2016021361815754)
      ('my', 1.0074038111421288)
      (':', 0.922442043937371)
      ('!', 0.8253428814176478)
      ('with', 0.801068090787717)
```



```
[62]: topkeynormalized_shakespeare = [(word,freq/numwords2*100) for (word, freq) in topkeys_shakespeares]
      for pair2 in topkeynormalized_shakespeare:
          print(pair2)

      (',', 7.951171230616957)
      ('.', 5.16606180578467)
      ('the', 2.7301220719234576)
      ('and', 2.369954910370615)
      ('to', 1.8778181018365776)
      ('of', 1.6771142637193446)
      (':', 1.5561420873199163)
      ('i', 1.539645881447267)
      ('you', 1.448916749147696)
      ('my', 1.3801825580116573)
      ('a', 1.3664357197844497)
      ('?', 1.2619597492576706)
      ('it', 1.151985043440009)
      ('in', 1.0667546464313207)
      ('that', 1.0337622346860222)
      ('is', 1.022764764104256)
      ('ham', 0.9265368965138018)
      ('not', 0.8990432200593864)
      (';', 0.8193115583415814)
      ('his', 0.7835697789508412)
      ('this', 0.7560761024964258)
      ('with', 0.6983393819421533)
      ('your', 0.6955900142967117)
      ('but', 0.6845925437149456)
      ('for', 0.6680963378422963)
```

Phase 4:

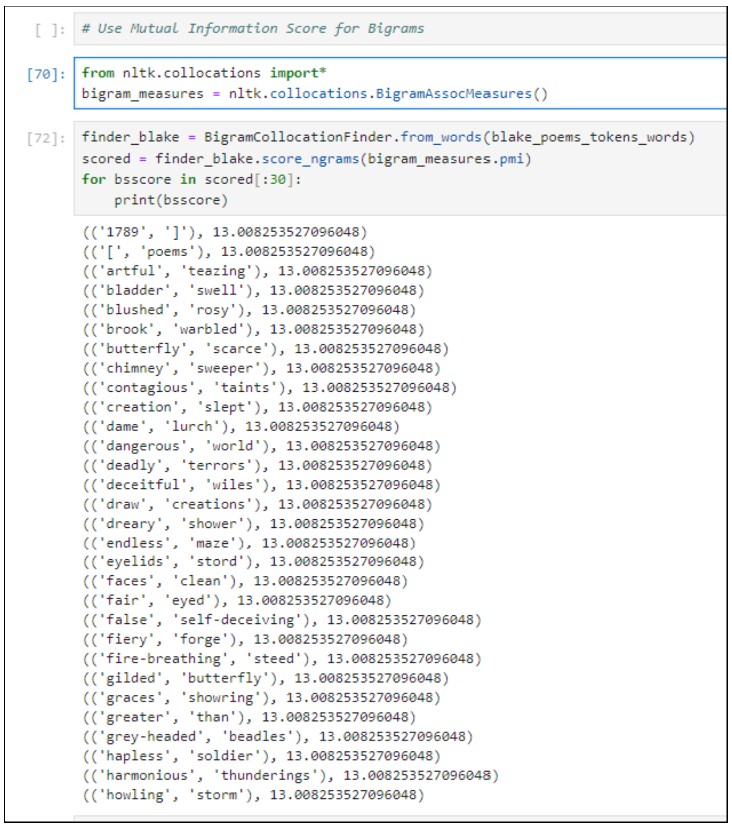
Bigram by frequencies for both document



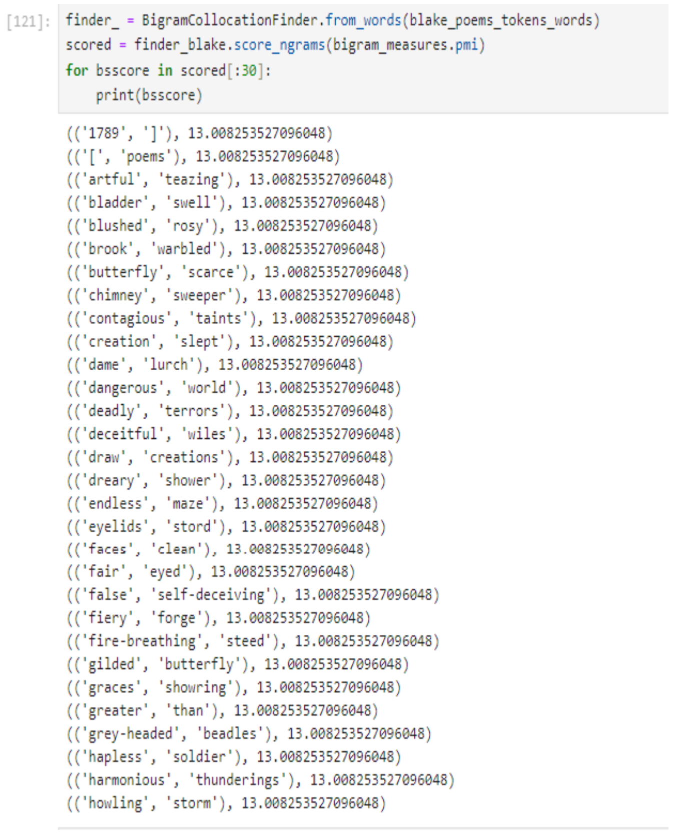
```
[63]: # List the top 50 bigrams by frequencies

[65]: shakespeare_tokens_words_bigrams = list(nltk.bigrams(shakespeare_tokens_words))
      print(shakespeare_tokens_words_bigrams[:50])

      [('[', 'the'), ('the', 'tragedie'), ('tragedie', 'of'), ('of', 'hamlet'), ('hamlet', 'by'), ('by', 'william'), ('william', 'shakespeare'), ('shakespear
      e', '1599'), ('1599', ']'), (']', 'actus'), ('actus', 'primus'), ('primus', '.'), ('.', 'scoena'), ('scoena', 'prima'), ('prima', '.'), ('.', 'enter'),
      ('enter', 'barnardo'), ('barnardo', 'and'), ('and', 'francisco'), ('francisco', 'two'), ('two', 'centinels'), ('centinels', '.'), ('.', 'barnardo'), ('ba
      rnardo', '.'), ('.', 'who'), ('who', "'s"), ("'s", 'there'), ('there', '?'), ('?', 'fran'), ('fran', '.'), ('.', 'nay'), ('nay', 'answer'), ('answer', 'm
      e'), ('me', ':'), (':', 'stand'), ('stand', '&'), ('&', 'vnfold'), ('vnfold', 'your'), ('your', 'selfe'), ('selfe', 'bar'), ('bar', '.'), ('.', 'long'),
      ('long', 'liue'), ('liue', 'the'), ('the', 'king'), ('king', 'fran'), ('fran', '.'), ('.', 'barnardo'), ('barnardo', '?'), ('?', 'bar')]

[66]: blake_poems_tokens_words_bigrams = list(nltk.bigrams(blake_poems_tokens_words))
      print(blake_poems_tokens_words_bigrams[:50])

      [('[', 'poems'), ('poems', 'by'), ('by', 'william'), ('william', 'blake'), ('blake', '1789'), ('1789', ']'), (']', 'songs'), ('songs', 'of'), ('of', 'inn
      ocence'), ('innocence', 'and'), ('and', 'of'), ('of', 'experience'), ('experience', 'and'), ('and', 'the'), ('the', 'book'), ('book', 'of'), ('of', 'the
      l'), ('thel', 'songs'), ('songs', 'of'), ('of', 'innocence'), ('innocence', 'introduction'), ('introduction', 'piping'), ('piping', 'down'), ('down', 'th
      e'), ('the', 'valleys'), ('valleys', 'wild'), ('wild', ','), (',', 'piping'), ('piping', 'songs'), ('songs', 'of'), ('of', 'pleasant'), ('pleasant', 'gle
      e'), ('glee', ','), (',', 'on'), ('on', 'a'), ('a', 'cloud'), ('cloud', 'i'), ('i', 'saw'), ('saw', 'a'), ('a', 'child'), ('child', ','), (',', 'and'),
      ('and', 'he'), ('he', 'laughing'), ('laughing', 'said'), ('said', 'to'), ('to', 'me'), ('me', ':'), (':', '``'), ('``', 'pipe')]
```

We received a list of bigrams from both texts but it's not helping to understand important bigrams from text hence we will use mutual information score for bigrams to achieve it.

Mutual Information Score:

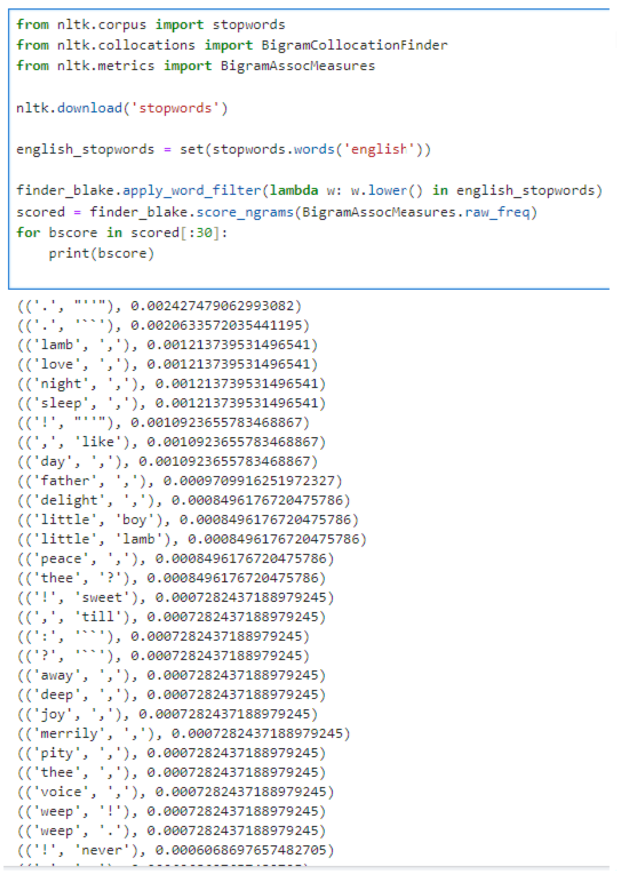


```
[ ]: # Use Mutual Information Score for Bigrams

[70]: from nltk.collocations import*
      bigram_measures = nltk.collocations.BigramAssocMeasures()

[72]: finder_blake = BigramCollocationFinder.from_words(blake_poems_tokens_words)
      scored = finder_blake.score_ngrams(bigram_measures.pmi)
      for bsscore in scored[:30]:
          print(bsscore)

      (('1789', ']'), 13.008253527096048)
      (('[', 'poems'), 13.008253527096048)
      (('artful', 'teazing'), 13.008253527096048)
      (('bladder', 'swell'), 13.008253527096048)
      (('blushed', 'rosy'), 13.008253527096048)
      (('brook', 'warbled'), 13.008253527096048)
      (('butterfly', 'scarce'), 13.008253527096048)
      (('chimney', 'sweeper'), 13.008253527096048)
      (('contagious', 'taints'), 13.008253527096048)
      (('creation', 'slept'), 13.008253527096048)
      (('dame', 'lurch'), 13.008253527096048)
      (('dangerous', 'world'), 13.008253527096048)
      (('deadly', 'terrors'), 13.008253527096048)
      (('deceitful', 'wiles'), 13.008253527096048)
      (('draw', 'creations'), 13.008253527096048)
      (('dreary', 'shower'), 13.008253527096048)
      (('endless', 'maze'), 13.008253527096048)
      (('eyelids', 'stord'), 13.008253527096048)
      (('faces', 'clean'), 13.008253527096048)
      (('fair', 'eyed'), 13.008253527096048)
      (('false', 'self-deceiving'), 13.008253527096048)
      (('fiery', 'forge'), 13.008253527096048)
      (('fire-breathing', 'steed'), 13.008253527096048)
      (('gilded', 'butterfly'), 13.008253527096048)
      (('graces', 'showring'), 13.008253527096048)
      (('greater', 'than'), 13.008253527096048)
      (('grey-headed', 'beadles'), 13.008253527096048)
      (('hapless', 'soldier'), 13.008253527096048)
      (('harmonious', 'thunderings'), 13.008253527096048)
      (('howling', 'storm'), 13.008253527096048)
```



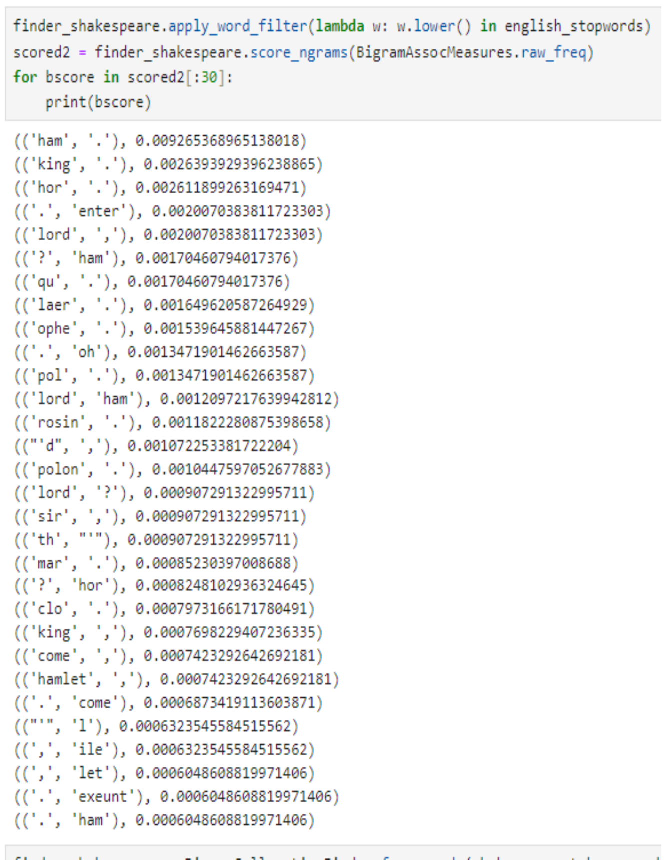
```
[121]: finder_ = BigramCollocationFinder.from_words(blake_poems_tokens_words)
       scored = finder_blake.score_ngrams(bigram_measures.pmi)
       for bsscore in scored[:30]:
           print(bsscore)

       (('1789', ']'), 13.008253527096048)
       (('[', 'poems'), 13.008253527096048)
       (('artful', 'teazing'), 13.008253527096048)
       (('bladder', 'swell'), 13.008253527096048)
       (('blushed', 'rosy'), 13.008253527096048)
       (('brook', 'warbled'), 13.008253527096048)
       (('butterfly', 'scarce'), 13.008253527096048)
       (('chimney', 'sweeper'), 13.008253527096048)
       (('contagious', 'taints'), 13.008253527096048)
       (('creation', 'slept'), 13.008253527096048)
       (('dame', 'lurch'), 13.008253527096048)
       (('dangerous', 'world'), 13.008253527096048)
       (('deadly', 'terrors'), 13.008253527096048)
       (('deceitful', 'wiles'), 13.008253527096048)
       (('draw', 'creations'), 13.008253527096048)
       (('dreary', 'shower'), 13.008253527096048)
       (('endless', 'maze'), 13.008253527096048)
       (('eyelids', 'stord'), 13.008253527096048)
       (('faces', 'clean'), 13.008253527096048)
       (('fair', 'eyed'), 13.008253527096048)
       (('false', 'self-deceiving'), 13.008253527096048)
       (('fiery', 'forge'), 13.008253527096048)
       (('fire-breathing', 'steed'), 13.008253527096048)
       (('gilded', 'butterfly'), 13.008253527096048)
       (('graces', 'showring'), 13.008253527096048)
       (('greater', 'than'), 13.008253527096048)
       (('grey-headed', 'beadles'), 13.008253527096048)
       (('hapless', 'soldier'), 13.008253527096048)
       (('harmonious', 'thunderings'), 13.008253527096048)
       (('howling', 'storm'), 13.008253527096048)
```

We need to remove stop words using available 'stopwords' library in nltk for both the documents to avoid unnecessary content which is not helpful to understand the document.



```python
from nltk.corpus import stopwords
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures

nltk.download('stopwords')

english_stopwords = set(stopwords.words('english'))

finder_blake.apply_word_filter(lambda w: w.lower() in english_stopwords)
scored = finder_blake.score_ngrams(BigramAssocMeasures.raw_freq)
for bscore in scored[:30]:
    print(bscore)
```

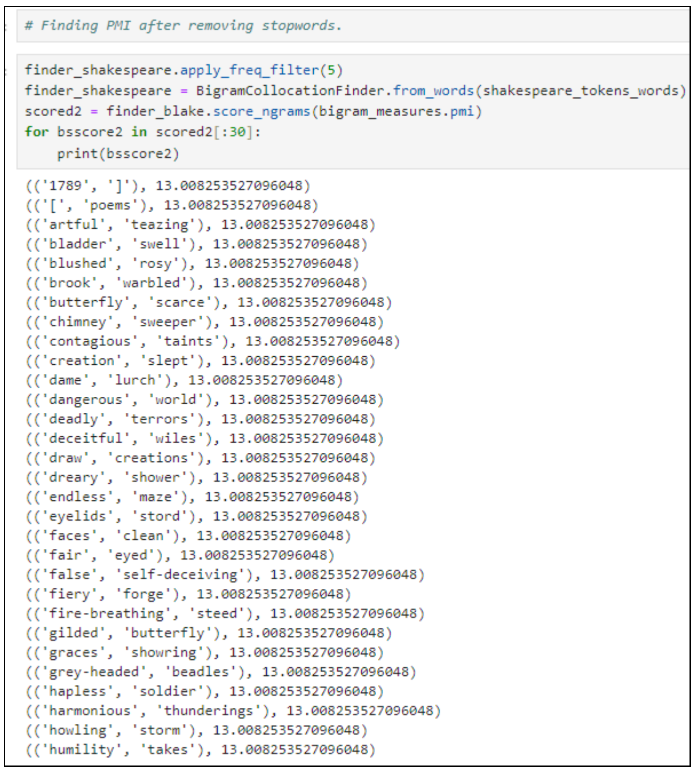
```
(('.', "''"), 0.002427479062993082)
(('.', '``'), 0.0020633572035441195)
(('lamb', ','), 0.001213739531496541)
(('love', ','), 0.001213739531496541)
(('night', ','), 0.001213739531496541)
(('sleep', ','), 0.001213739531496541)
(('!', "''"), 0.0010923655783468867)
((',', 'like'), 0.0010923655783468867)
(('day', ','), 0.0010923655783468867)
(('father', ','), 0.0009709916251972327)
(('delight', ','), 0.0008496176720475786)
(('little', 'boy'), 0.0008496176720475786)
(('little', 'lamb'), 0.0008496176720475786)
(('peace', ','), 0.0008496176720475786)
(('thee', '?'), 0.0008496176720475786)
(('!', 'sweet'), 0.0007282437188979245)
((',', 'till'), 0.0007282437188979245)
((':', '``'), 0.0007282437188979245)
(('?', '``'), 0.0007282437188979245)
(('away', ','), 0.0007282437188979245)
(('deep', ','), 0.0007282437188979245)
(('joy', ','), 0.0007282437188979245)
(('merrily', ','), 0.0007282437188979245)
(('pity', ','), 0.0007282437188979245)
(('thee', ','), 0.0007282437188979245)
(('voice', ','), 0.0007282437188979245)
(('weep', '!'), 0.0007282437188979245)
(('weep', '.'), 0.0007282437188979245)
(('!', 'never'), 0.0006068697657482705)
```



```python
finder_shakespeare.apply_word_filter(lambda w: w.lower() in english_stopwords)
scored2 = finder_shakespeare.score_ngrams(BigramAssocMeasures.raw_freq)
for bscore in scored2[:30]:
    print(bscore)
```

```
(('ham', '.'), 0.009265368965138018)
(('king', '.'), 0.0026393929396238865)
(('hor', '.'), 0.002611899263169471)
(('.', 'enter'), 0.0020070383811723303)
(('lord', ','), 0.0020070383811723303)
(('?', 'ham'), 0.00170460794017376)
(('qu', '.'), 0.00170460794017376)
(('laer', '.'), 0.001649620587264929)
(('ophe', '.'), 0.001539645881447267)
(('.', 'oh'), 0.0013471901462663587)
(('pol', '.'), 0.0013471901462663587)
(('lord', 'ham'), 0.0012097217639942812)
(('rosin', '.'), 0.0011822280875398658)
(("'d", ','), 0.001072253381722204)
(('polon', '.'), 0.0010447597052677883)
(('lord', '?'), 0.000907291322995711)
(('sir', ','), 0.000907291322995711)
(('th', "'"), 0.000907291322995711)
(('mar', '.'), 0.00085230397008688)
(('?', 'hor'), 0.0008248102936324645)
(('clo', '.'), 0.0007973166171780491)
(('king', ','), 0.0007698229407236335)
(('come', ','), 0.0007423292642692181)
(('hamlet', ','), 0.0007423292642692181)
(('.', 'come'), 0.0006873419113603871)
(("'", 'l'), 0.0006323545584515562)
((',', 'ile'), 0.0006323545584515562)
((',', 'let'), 0.0006048608819971406)
(('.', 'exeunt'), 0.0006048608819971406)
(('.', 'ham'), 0.0006048608819971406)
```

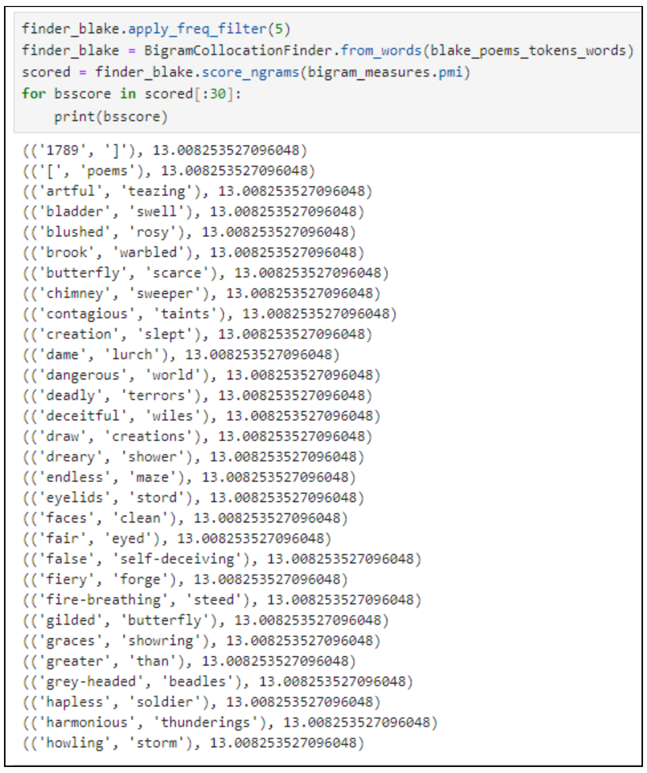
Now we can find PMI after removing stopwords.



```python
# Finding PMI after removing stopwords.
```

```python
finder_shakespeare.apply_freq_filter(5)
finder_shakespeare = BigramCollocationFinder.from_words(shakespeare_tokens_words)
scored2 = finder_blake.score_ngrams(bigram_measures.pmi)
for bsscore2 in scored2[:30]:
    print(bsscore2)
```

```
(('1789', ']'), 13.008253527096048)
(('[', 'poems'), 13.008253527096048)
(('artful', 'teazing'), 13.008253527096048)
(('bladder', 'swell'), 13.008253527096048)
(('blushed', 'rosy'), 13.008253527096048)
(('brook', 'warbled'), 13.008253527096048)
(('butterfly', 'scarce'), 13.008253527096048)
(('chimney', 'sweeper'), 13.008253527096048)
(('contagious', 'taints'), 13.008253527096048)
(('creation', 'slept'), 13.008253527096048)
(('dame', 'lurch'), 13.008253527096048)
(('dangerous', 'world'), 13.008253527096048)
(('deadly', 'terrors'), 13.008253527096048)
(('deceitful', 'wiles'), 13.008253527096048)
(('draw', 'creations'), 13.008253527096048)
(('dreary', 'shower'), 13.008253527096048)
(('endless', 'maze'), 13.008253527096048)
(('eyelids', 'stord'), 13.008253527096048)
(('faces', 'clean'), 13.008253527096048)
(('fair', 'eyed'), 13.008253527096048)
(('false', 'self-deceiving'), 13.008253527096048)
(('fiery', 'forge'), 13.008253527096048)
(('fire-breathing', 'steed'), 13.008253527096048)
(('gilded', 'butterfly'), 13.008253527096048)
(('graces', 'showring'), 13.008253527096048)
(('grey-headed', 'beadles'), 13.008253527096048)
(('hapless', 'soldier'), 13.008253527096048)
(('harmonious', 'thunderings'), 13.008253527096048)
(('howling', 'storm'), 13.008253527096048)
(('humility', 'takes'), 13.008253527096048)
```



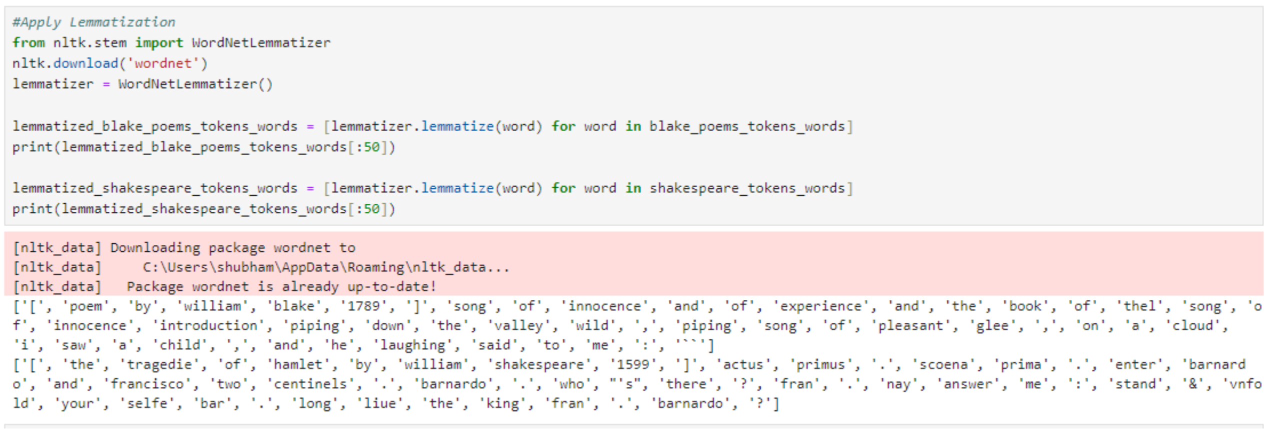
```python
finder_blake.apply_freq_filter(5)
finder_blake = BigramCollocationFinder.from_words(blake_poems_tokens_words)
scored = finder_blake.score_ngrams(bigram_measures.pmi)
for bsscore in scored[:30]:
    print(bsscore)
```

```
(('1789', ']'), 13.008253527096048)
(('[', 'poems'), 13.008253527096048)
(('artful', 'teazing'), 13.008253527096048)
(('bladder', 'swell'), 13.008253527096048)
(('blushed', 'rosy'), 13.008253527096048)
(('brook', 'warbled'), 13.008253527096048)
(('butterfly', 'scarce'), 13.008253527096048)
(('chimney', 'sweeper'), 13.008253527096048)
(('contagious', 'taints'), 13.008253527096048)
(('creation', 'slept'), 13.008253527096048)
(('dame', 'lurch'), 13.008253527096048)
(('dangerous', 'world'), 13.008253527096048)
(('deadly', 'terrors'), 13.008253527096048)
(('deceitful', 'wiles'), 13.008253527096048)
(('draw', 'creations'), 13.008253527096048)
(('dreary', 'shower'), 13.008253527096048)
(('endless', 'maze'), 13.008253527096048)
(('eyelids', 'stord'), 13.008253527096048)
(('faces', 'clean'), 13.008253527096048)
(('fair', 'eyed'), 13.008253527096048)
(('false', 'self-deceiving'), 13.008253527096048)
(('fiery', 'forge'), 13.008253527096048)
(('fire-breathing', 'steed'), 13.008253527096048)
(('gilded', 'butterfly'), 13.008253527096048)
(('graces', 'showring'), 13.008253527096048)
(('greater', 'than'), 13.008253527096048)
(('grey-headed', 'beadles'), 13.008253527096048)
(('hapless', 'soldier'), 13.008253527096048)
(('harmonious', 'thunderings'), 13.008253527096048)
(('howling', 'storm'), 13.008253527096048)
```

Now we have list of bigrams for both the documents which gives enough idea about texts.

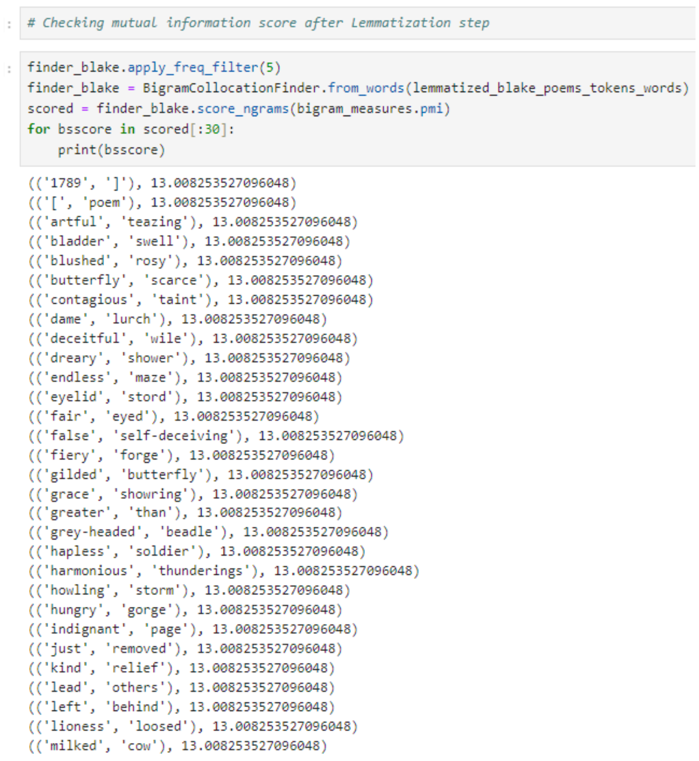
Example. ‘Shakespeare’ text have more negative words compare to ‘blake poems’ text. ‘dangerous world’ vs ‘artful teazing’

We can apply lemmatization as well here to use words as their original form to avoid redundancy.



```
#Apply Lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()

lemmatized_blake_poems_tokens_words = [lemmatizer.lemmatize(word) for word in blake_poems_tokens_words]
print(lemmatized_blake_poems_tokens_words[:50])

lemmatized_shakespeare_tokens_words = [lemmatizer.lemmatize(word) for word in shakespeare_tokens_words]
print(lemmatized_shakespeare_tokens_words[:50])
```

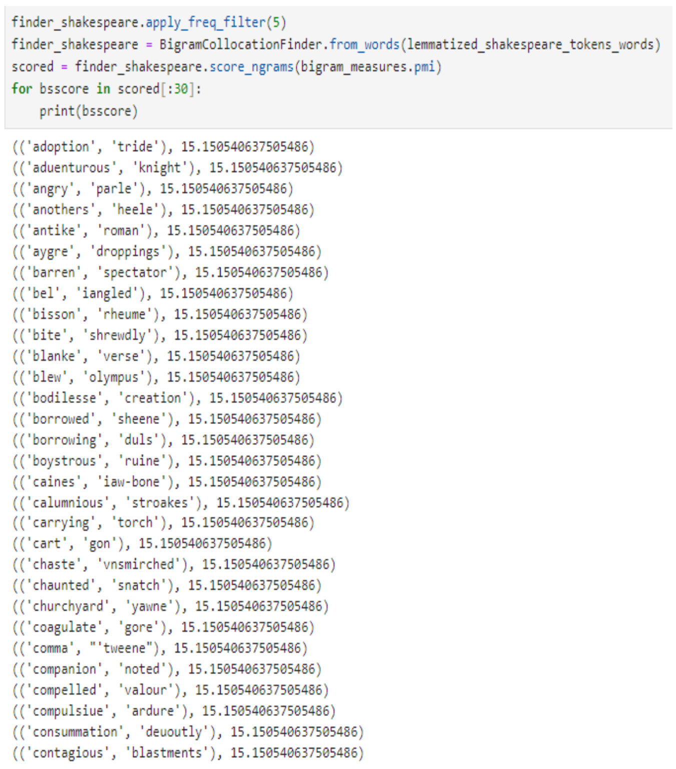
```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\shubham\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
['[', 'poem', 'by', 'william', 'blake', '1789', ']', 'song', 'of', 'innocence', 'and', 'of', 'experience', 'and', 'the', 'book', 'of', 'thel', 'song', 'o
f', 'innocence', 'introduction', 'piping', 'down', 'the', 'valley', 'wild', ',', 'piping', 'song', 'of', 'pleasant', 'glee', ',', 'on', 'a', 'cloud',
'i', 'saw', 'a', 'child', ',', 'and', 'he', 'laughing', 'said', 'to', 'me', ':', '``']
['[', 'the', 'tragedie', 'of', 'hamlet', 'by', 'william', 'shakespeare', '1599', ']', 'actus', 'primus', '.', 'scoena', 'prima', '.', 'enter', 'barnard
o', 'and', 'francisco', 'two', 'centinels', '.', 'barnardo', '.', 'who', "'s", 'there', '?', 'fran', '.', 'nay', 'answer', 'me', ':', 'stand', '&', 'vnfo
ld', 'your', 'selfe', 'bar', '.', 'long', 'liue', 'the', 'king', 'fran', '.', 'barnardo', '?']
```



```
# Checking mutual information score after Lemmatization step
```

```
finder_blake.apply_freq_filter(5)
finder_blake = BigramCollocationFinder.from_words(lemmatized_blake_poems_tokens_words)
scored = finder_blake.score_ngrams(bigram_measures.pmi)
for bsscore in scored[:30]:
    print(bsscore)
```

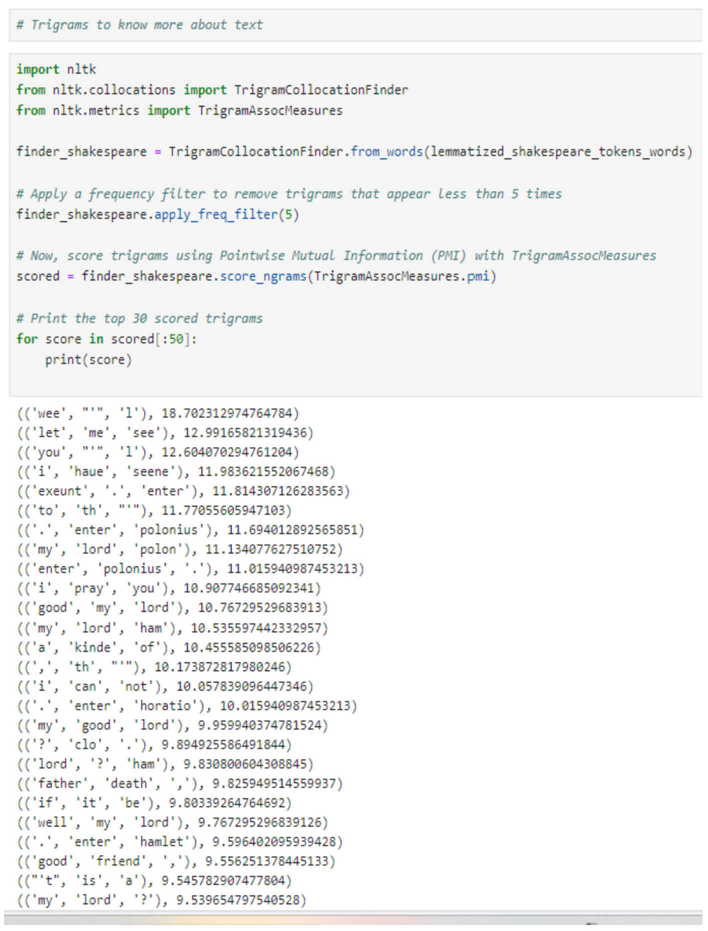
```
(('1789', ']'), 13.008253527096048)
(('[', 'poem'), 13.008253527096048)
(('artful', 'teazing'), 13.008253527096048)
(('bladder', 'swell'), 13.008253527096048)
(('blushed', 'rosy'), 13.008253527096048)
(('butterfly', 'scarce'), 13.008253527096048)
(('contagious', 'taint'), 13.008253527096048)
(('dame', 'lurch'), 13.008253527096048)
(('deceitful', 'wile'), 13.008253527096048)
(('dreary', 'shower'), 13.008253527096048)
(('endless', 'maze'), 13.008253527096048)
(('eyelid', 'stord'), 13.008253527096048)
(('fair', 'eyed'), 13.008253527096048)
(('false', 'self-deceiving'), 13.008253527096048)
(('fiery', 'forge'), 13.008253527096048)
(('gilded', 'butterfly'), 13.008253527096048)
(('grace', 'showring'), 13.008253527096048)
(('greater', 'than'), 13.008253527096048)
(('grey-headed', 'beadle'), 13.008253527096048)
(('hapless', 'soldier'), 13.008253527096048)
(('harmonious', 'thunderings'), 13.008253527096048)
(('howling', 'storm'), 13.008253527096048)
(('hungry', 'gorge'), 13.008253527096048)
(('indignant', 'page'), 13.008253527096048)
(('just', 'removed'), 13.008253527096048)
(('kind', 'relief'), 13.008253527096048)
(('lead', 'others'), 13.008253527096048)
(('left', 'behind'), 13.008253527096048)
(('lioness', 'loosed'), 13.008253527096048)
(('milked', 'cow'), 13.008253527096048)
```



```
finder_shakespeare.apply_freq_filter(5)
finder_shakespeare = BigramCollocationFinder.from_words(lemmatized_shakespeare_tokens_words)
scored = finder_shakespeare.score_ngrams(bigram_measures.pmi)
for bsscore in scored[:30]:
    print(bsscore)
```

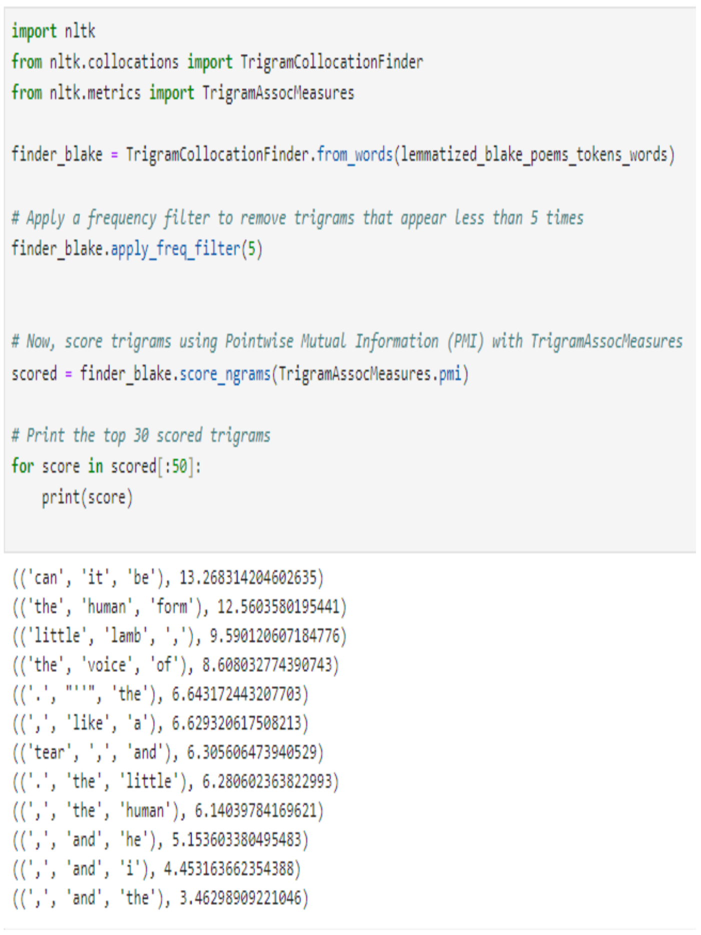
```
(('adoption', 'tride'), 15.150540637505486)
(('aduenturous', 'knight'), 15.150540637505486)
(('angry', 'parle'), 15.150540637505486)
(('anothers', 'heele'), 15.150540637505486)
(('antike', 'roman'), 15.150540637505486)
(('aygre', 'droppings'), 15.150540637505486)
(('barren', 'spectator'), 15.150540637505486)
(('bel', 'iangled'), 15.150540637505486)
(('bisson', 'rheume'), 15.150540637505486)
(('bite', 'shrewdly'), 15.150540637505486)
(('blanke', 'verse'), 15.150540637505486)
(('blew', 'olympus'), 15.150540637505486)
(('bodilesse', 'creation'), 15.150540637505486)
(('borrowed', 'sheene'), 15.150540637505486)
(('borrowing', 'duls'), 15.150540637505486)
(('boystrous', 'ruine'), 15.150540637505486)
(('caines', 'iaw-bone'), 15.150540637505486)
(('calumnious', 'stroakes'), 15.150540637505486)
(('carrying', 'torch'), 15.150540637505486)
(('cart', 'gon'), 15.150540637505486)
(('chaste', 'vnsmirched'), 15.150540637505486)
(('chaunted', 'snatch'), 15.150540637505486)
(('churchyard', 'yawne'), 15.150540637505486)
(('coagulate', 'gore'), 15.150540637505486)
(('comma', "'tweene"), 15.150540637505486)
(('companion', 'noted'), 15.150540637505486)
(('compelled', 'valour'), 15.150540637505486)
(('compulsiue', 'ardure'), 15.150540637505486)
(('consummation', 'deuoutly'), 15.150540637505486)
(('contagious', 'blastments'), 15.150540637505486)
```

We can use trigram to know more about text,



```
# Trigrams to know more about text

import nltk
from nltk.collocations import TrigramCollocationFinder
from nltk.metrics import TrigramAssocMeasures

finder_shakespeare = TrigramCollocationFinder.from_words(lemmatized_shakespeare_tokens_words)

# Apply a frequency filter to remove trigrams that appear less than 5 times
finder_shakespeare.apply_freq_filter(5)

# Now, score trigrams using Pointwise Mutual Information (PMI) with TrigramAssocMeasures
scored = finder_shakespeare.score_ngrams(TrigramAssocMeasures.pmi)

# Print the top 30 scored trigrams
for score in scored[:50]:
    print(score)
```

```
(('wee', "'", 'l'), 18.702312974764784)
(('let', 'me', 'see'), 12.99165821319436)
(('you', "'", 'l'), 12.604070294761204)
(('i', 'haue', 'seene'), 11.983621552067468)
(('exeunt', '.', 'enter'), 11.814307126283563)
(('to', 'th', "'"), 11.77055605947103)
(('.', 'enter', 'polonius'), 11.694012892565851)
(('my', 'lord', 'polon'), 11.134077627510752)
(('enter', 'polonius', '.'), 11.015940987453213)
(('i', 'pray', 'you'), 10.907746685092341)
(('good', 'my', 'lord'), 10.76729529683913)
(('my', 'lord', 'ham'), 10.535597442332957)
(('a', 'kinde', 'of'), 10.455585098506226)
((',', 'th', "'"), 10.173872817980246)
(('i', 'can', 'not'), 10.057839096447346)
(('.', 'enter', 'horatio'), 10.015940987453213)
(('my', 'good', 'lord'), 9.959940374781524)
(('?', 'clo', '.'), 9.894925586491844)
(('lord', '?', 'ham'), 9.830800604308845)
(('father', 'death', ','), 9.825949514559937)
(('if', 'it', 'be'), 9.80339264764692)
(('well', 'my', 'lord'), 9.767295296839126)
(('.', 'enter', 'hamlet'), 9.596402095939428)
(('good', 'friend', ','), 9.556251378445133)
(("'t", 'is', 'a'), 9.545782907477804)
(('my', 'lord', '?'), 9.539654797540528)
```



```
import nltk
from nltk.collocations import TrigramCollocationFinder
from nltk.metrics import TrigramAssocMeasures

finder_blake = TrigramCollocationFinder.from_words(lemmatized_blake_poems_tokens_words)

# Apply a frequency filter to remove trigrams that appear less than 5 times
finder_blake.apply_freq_filter(5)


# Now, score trigrams using Pointwise Mutual Information (PMI) with TrigramAssocMeasures
scored = finder_blake.score_ngrams(TrigramAssocMeasures.pmi)

# Print the top 30 scored trigrams
for score in scored[:50]:
    print(score)
```

```
(('can', 'it', 'be'), 13.268314204602635)
(('the', 'human', 'form'), 12.5603580195441)
(('little', 'lamb', ','), 9.590120607184776)
(('the', 'voice', 'of'), 8.608032774390743)
(('.', "''", 'the'), 6.643172443207703)
((',', 'like', 'a'), 6.629320617508213)
(('tear', ',', 'and'), 6.305606473940529)
(('.', 'the', 'little'), 6.280602363822993)
((',', 'the', 'human'), 6.14039784169621)
((',', 'and', 'he'), 5.153603380495483)
((',', 'and', 'i'), 4.453163662354388)
((',', 'and', 'the'), 3.46298909221046)
```

Trigrams are not that enough helpful here as ‘blake poems’ text has not enough trigrams.

3.

Phase 5:

Sentiment Analysis: "What is the overall sentiment (positive, negative, or neutral) conveyed in 'Hamlet' compared to Blake's poems?

- After observing list of bigrams and trigrams, we can predict that content of ‘Hamlet’ has more negative words compared to ‘Blake’s poem’.
- Please find below list of negative bigrams from ‘Hamlet’ text
  a. Hapless Soldier
  b. Bladder Swell
  c. Left Behind
  d. Endless maze
  <u>vs</u>
  e. Adventurous Knight
  f. Adoption tride
