Comparing Writing Pattern of 2 Authors

I chose 2 documents from Gutenberg collection on the web. Namely,
1**: Leviathan, by Thomas Hobbes**- Leviathan or The Matter, Forme and Power of a Common Wealth Ecclesiastical and Civil. The work concerns the structure of society and legitimate government. It belongs to Political Philosophy Genre
2: **Peter Pan, by James M. Barrie** – The story of Peter Pan, the Boy Who Wouldn't Grow Up, mischievous yet innocent little boy who can fly, and has many adventures on the island of Neverland that is inhabited by mermaids, fairies, Native Americans and pirates. It belongs Fantasy Genre
Both the documents are sufficiently big, belongs to different genres, written by different authors. I'm interested in Topical words, observing authors writing pattern.

Doc 1: Leviathan, by Thomas Hobbes, Political Philosophy Genre.
1) To list top 50 words by Frequency

```
from nltk import FreqDist
from nltk.collocations import *
#list the top 50 words by frequency (normalized by the length of the document)
ABdist = FreqDist(filtered_words)
ABitems = ABdist.most_common(50)
print("top 50 words by frequency")
for item in ABitems:
    print(item[0], '\t', item[1])
```

2)**to list the top 50 bigrams by frequencies**
a)Imported the collocation finder module- from nltk.collocations import*
b)For convenience, defined a variable for the bigram measures- bigram_measures = nltk.collocations.BigramAssocMeasures()
c) Applied filters to remove updated stopwords, words with low frequency on Tokenized, lowercased alphanumeric tokens
(snippet in the end of question 2)

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

```
finder3.apply_freq_filter(5)
scored = finder3.score_ngrams(bigram_measures.pmi)
print ("pmi data with minimum frequency")
for bscore in scored[:50]:
    print (bscore)
```

Doc2- Peter Pan, by James M. Barrie
1)list of top 50 words frequency

```
from nltk import FreqDist
from nltk.collocations import *
#list the top 50 words by frequency (normalized by the length of the document)
ABdist = FreqDist(filtered_words)
ABitems = ABdist.most_common(50)
```

```
print("top 50 words by frequency")
for item in ABitems:
    print(item[0], '\t', item[1])
```

2)**Normalization.**
a. Start with getting the text from Gutenberg collection on the web.
b) Separated all the works into tokens with the word_tokenizer.
c) Converted all the characters to lower case.
d) Removed common words that appears with great frequency ie. Stopwords.
e) Updated the stopwords list, added '.' ',' ';' etc to the list using wordpunc_tokenize.
f) I did not remove the numbers because it follows a specific pattern which can helpful.
Thus extracted the normalized data. (filtered_words[])

I'm interested in the topical words which describes the gist of the Document.
From the normalized words with frequency we can get some important keys in **the Doc1**

soveraign	497
common-wealth	432
authority	319

whereas Doc2

peter	403,
wendy	356,
michael	110

The bigrams **in the doc1** are meaningful and related
```
(('civill', 'soveraign'), 0.00018667890450596206)
(('soveraign', 'power'), 0.0004161383912945404)
(('civill', 'law'), 0.000167233185286591)
(('high', 'priest'), 0.000167233185286591)
```
Whereas **in the doc2** they are not useful
```
(('said', 'peter'), 0.0003186387751525483)
(('wendy', 'said'), 0.0003186387751525483)
(('electronic', 'works'), 0.00025491102012203866)
```

In Doc1, top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual In formation. The bigrams are more interesting than the bigrams scored by Mutual information
In doc2, top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Inf ormation. The bigrams scored by mutual information is more interesting.
(by interesting, I mean useful tokens)

I modified the stopwords list. Snippet-

```
from nltk.corpus import stopwords
from nltk.tokenize import wordpunct_tokenize

#to get the stopwords list
stopwords =nltk.corpus.stopwords.words('english')  + list(string.punctuation)
print("List of stopwords")
print(stopwords)
```

```
#to remove the stopwords
stoppedABwords = [w for w in ABwords if not w in stopwords]
filtered_words = []
for w in ABwords:
    if w not in stopwords:
        filtered_words.append(w)
#tokenized, lowercase list without stopwords
print("tokenized, lowercase list without stopwords")
print(filtered_words[:200])
```

Snippet- to get meaningful bigrams I applied the following filters

```
ABbigrams = list(nltk.bigrams(ABwords))
print("Sample Bigrams")
print(ABbigrams[:50])

bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(ABwords)
scored = finder.score_ngrams(bigram_measures.raw_freq)
#print(type(scored))
#first = scored[0]
#print(type(first), first)
print ("without any filter")
for bscore in scored[:30]:
    print (bscore)

finder2 = BigramCollocationFinder.from_words(ABwords)
finder2.apply_freq_filter(2)
scored = finder2.score_ngrams(bigram_measures.raw_freq)
print ("removed low frequency words")
for bscore in scored[:30]:
    print (bscore)

finder.apply_word_filter(lambda w: w in stopwords)
scored = finder.score_ngrams(bigram_measures.raw_freq)
print ("Bigrams after removing stopwords")
for bscore in scored[:50]:
    print (bscore)
```

There are many differences between the two documents.
Firstly, both belong to different genres. By observing the bigrams and trigrams we can tell which genre the document belongs to
For example by observing these biagrams we can tell Doc1 belongs to Political Philosophy genre

```
(('civill', 'soveraign'), 0.18667890450596206)
(('soveraign', 'power'), 0.4161383912945404)
(('civill', 'law'), 0.167233185286591)
(('paris', 'aprill', '15/25'), 35.94423194809294)
(('educational', 'corporation', 'organized'), 34.35926944737179)
```

And by observing bigrams and trigrams of doc2 we can infer that it belongs to Fantasy or adv enture genre. We can also observe the frequency of words the author used and observe what words he uses to describe something.

```
(('pirate', 'ship'), 8.13036371327355)
(('flew', 'away'), 8.302130061540032)
(('genuine', 'queen', 'mab'), 31.87543727066231)
(('lamp', 'shone', 'dimly'), 29.87543727066231)
(('deliciously', 'creepy', 'song'), 28.87543727066231)
```

Some cases trigrams are meaningless

In doc1,

```
(('sustineo', 'tres', 'personas'), 35.94423194809294)
(('tantem', 'tenditque', 'sub'), 35.94423194809294)
```

In doc2

No such meaningless trigrams are found

Trigram Frequency distribution snippet-

```
print ("trigrams pmi")
trigram_measures = nltk.collocations.TrigramAssocMeasures()
finder.apply_freq_filter(2)
finder = TrigramCollocationFinder.from_words(ABwords)
scored = finder.score_ngrams(trigram_measures.pmi)
for bscore in scored[:50]:
    print (bscore)
```

Thus we can conclude writing pattern of both the authors are different, they use different diff erent words to express their thoughts. For example, document **Leviathan, by Thomas Hobb es,** the author uses words like civil, sovereign, power to express his thoughts where as in docu ment **Peter Pan, by James M. Barrie** the author uses words like fly, deliciously, angelically to express his thoughts. We can also know how frequently they use certain words.

After observing the bigrams and trigrams of the documents, we can observe the topical words the author uses to express his thoughts.

Sometimes the bigrams and trigrams yield meaningless output too.(it depends on what filters you choose to apply and stopwords list)