Basic NLP Project

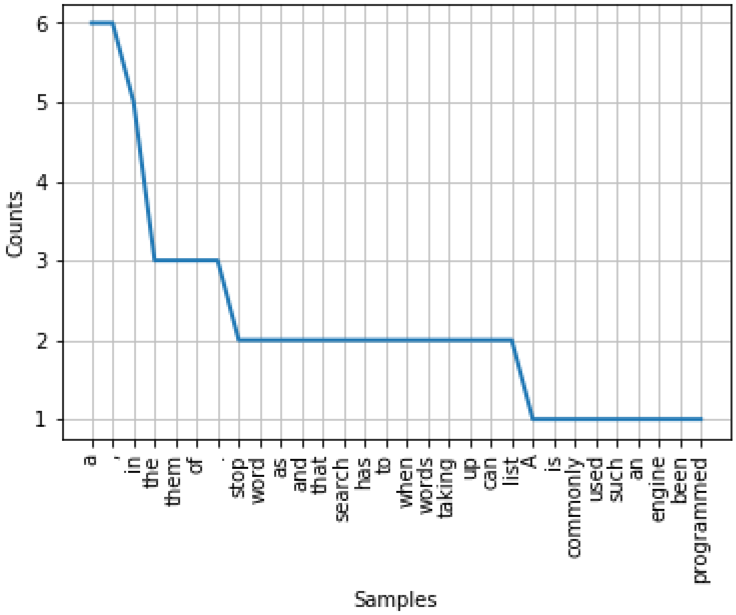
Parse a Text File to show the count of Words in the File.

Also, illustrate the count of words in a graph.

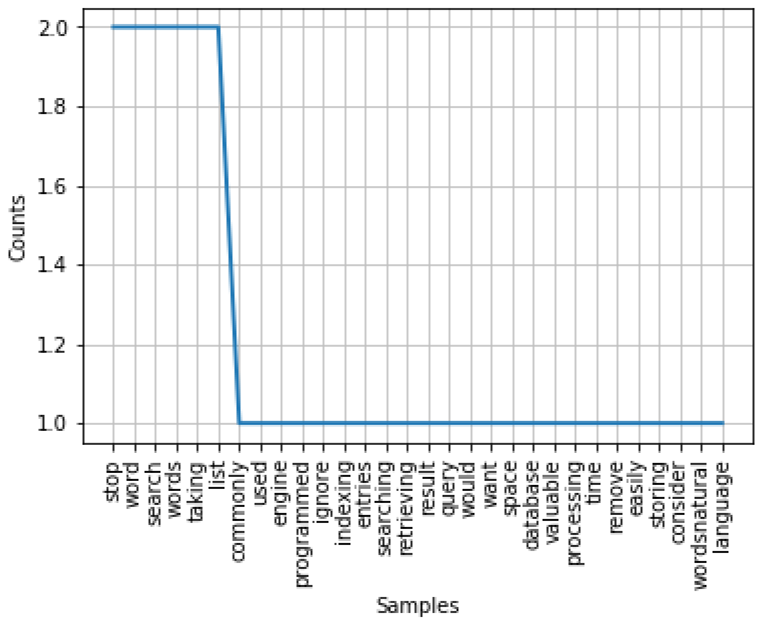
Compare, the results obtained before and after cleaning the texts.

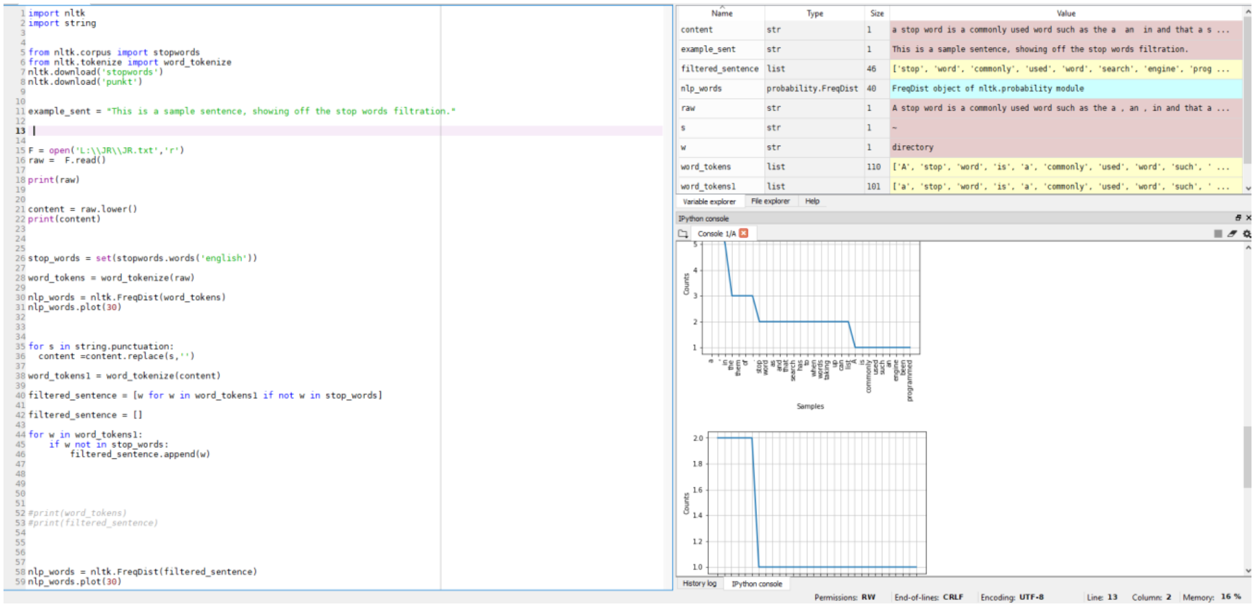
Solution:

Plot before cleaning the text



Plot after cleaning text

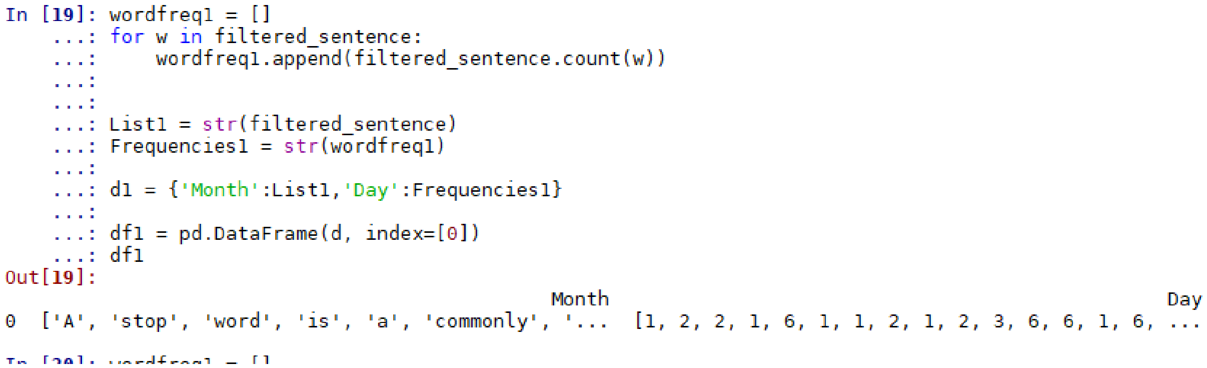




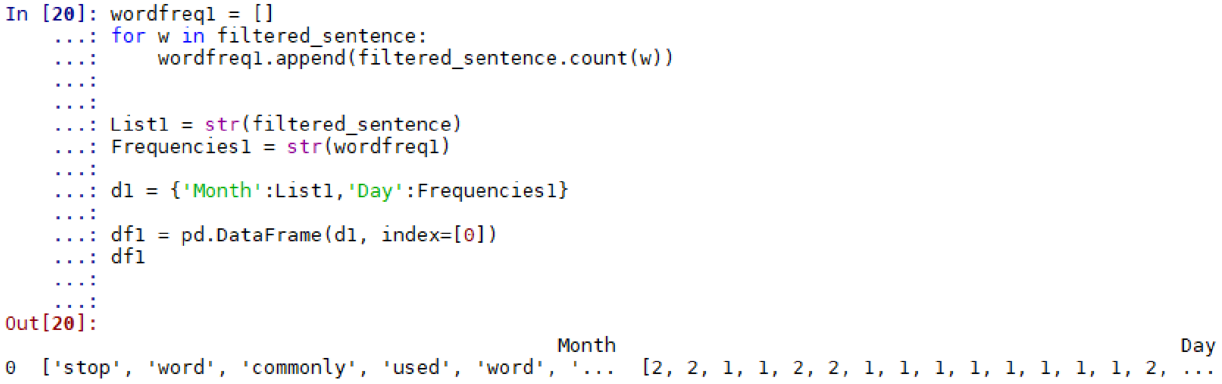
```
import nltk
import string


from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
nltk.download('punkt')


example_sent = "This is a sample sentence, showing off the stop words filtration."

 

F = open('L:\\JR\\JR.txt','r')
raw =  F.read()

print(raw)


content = raw.lower()
print(content)



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

word_tokens = word_tokenize(raw)

nlp_words = nltk.FreqDist(word_tokens)
nlp_words.plot(30)



for s in string.punctuation:
  content =content.replace(s,'')

word_tokens1 = word_tokenize(content)

filtered_sentence = [w for w in word_tokens1 if not w in stop_words]

filtered_sentence = []

for w in word_tokens1:
    if w not in stop_words:
        filtered_sentence.append(w)





#print(word_tokens)
#print(filtered_sentence)




nlp_words = nltk.FreqDist(filtered_sentence)
nlp_words.plot(30)
```

Name	Type	Size	Value
content	str	1	a stop word is a commonly used word such as the a an in and that a s ...
example_sent	str	1	This is a sample sentence, showing off the stop words filtration.
filtered_sentence	list	46	['stop', 'word', 'commonly', 'used', 'word', 'search', 'engine', 'prog ...
nlp_words	probability.FreqDist	40	FreqDist object of nltk.probability module
raw	str	1	A stop word is a commonly used word such as the a , an , in and that a ...
s	str	1	~
w	str	1	directory
word_tokens	list	110	['A', 'stop', 'word', 'is', 'a', 'commonly', 'used', 'word', 'such', ' ...
word_tokens1	list	101	['a', 'stop', 'word', 'is', 'a', 'commonly', 'used', 'word', 'such', ' ...

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 13   Column: 2   Memory: 16 %



```
In [19]: wordfreq1 = []
    ...: for w in filtered_sentence:
    ...:     wordfreq1.append(filtered_sentence.count(w))
    ...:
    ...:
    ...: List1 = str(filtered_sentence)
    ...: Frequencies1 = str(wordfreq1)
    ...:
    ...: d1 = {'Month':List1,'Day':Frequencies1}
    ...:
    ...: df1 = pd.DataFrame(d, index=[0])
    ...: df1
Out[19]:
                                               Month                                                 Day
0  ['A', 'stop', 'word', 'is', 'a', 'commonly', '...  [1, 2, 2, 1, 6, 1, 1, 2, 1, 2, 3, 6, 6, 1, 6, ...

In [20]: wordfreq1 = []
```



```
In [20]: wordfreq1 = []
    ...: for w in filtered_sentence:
    ...:     wordfreq1.append(filtered_sentence.count(w))
    ...:
    ...:
    ...: List1 = str(filtered_sentence)
    ...: Frequencies1 = str(wordfreq1)
    ...:
    ...: d1 = {'Month':List1,'Day':Frequencies1}
    ...:
    ...: df1 = pd.DataFrame(d1, index=[0])
    ...: df1
    ...:
    ...:
Out[20]:
                                               Month                                                 Day
0  ['stop', 'word', 'commonly', 'used', 'word', '...  [2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, ...
```

Looking at the two graphs the first thing I noticed that in the first graph before cleaning the data the words in with smaller and upper caps were counted as a different word.

The maximum of any word found in the first graph was 6 but in the second it reduced to 2. Along with that it can be seen that the most occurring words of the first graph were removed from the data after data cleaning which shows how many meaning less words we have in a normal text.

Further, stop words and punctuations were also present in the first graph which were removed in the second graph after cleaning the data.

For our ease we only plotted 30 words in a graph so the words on x-axis are visible and readable.