[ ]:
```
import nltk
nltk.download('punkt')
```

[21]:
```
from nltk.tokenize import sent_tokenize
```

[22]:
```
text="""Hello Mr. Smith, how are you doing today? The weather is
great, and city is awesome. The sky is pinkish-blue. You shouldn't
eat cardboard"""
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is\ngreat, and city
is awesome.', 'The sky is pinkish-blue.', "You shouldn't\neat cardboard"]
```

[23]:
```
from nltk.tokenize import word_tokenize
tokenize_word=word_tokenize(text)
print(tokenize_word)
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?',
'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.',
'The', 'sky', 'is', 'pinkish-blue', '.', 'You', "shouldn't", 'eat', 'cardboard']
```

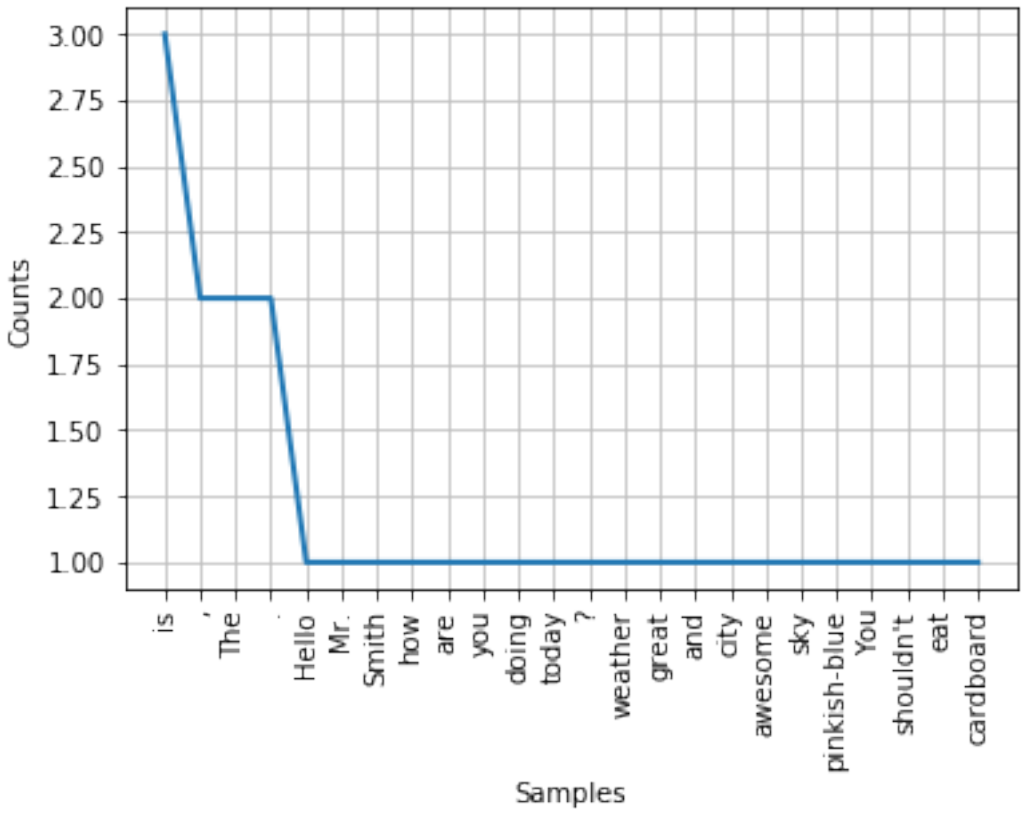
[24]:
```
from nltk.probability import FreqDist
fdist=FreqDist(tokenize_word)
print(fdist)
```

```
<FreqDist with 24 samples and 29 outcomes>
```

[25]:
```
fdist.most_common(2)
```

[25]:
```
[('is', 3), (',', 2)]
```

[26]:
```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



[27]:
```
sent = "Albert Einstein was born in Ulm, Germany in 1879."
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879',
'.']
```

[ ]:
```
nltk.download('averaged_perceptron_tagger')
```

[29]:
```
nltk.pos_tag(tokens)
```

[29]:
```
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
```

```
 ('.', '.')]
```

[ ]:
```
import nltk
nltk.download('stopwords')
```

[34]:
```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'aren', 'was', 'needn', 'why', 'weren', 'yours', 'so', 'both', "hadn't",
'they', "weren't", 'whom', 'is', 'do', 'until', 'again', 'should', 'doesn',
'wouldn', 'm', 'into', 'on', 'because', 'being', 've', 'up', 't', 'his',
'herself', 'y', 'when', 'which', 'no', "didn't", 'myself', "shan't",
"shouldn't", 'some', 'that', 'once', 'before', 'what', 'most', 'mustn', 'of',
'ma', 'any', 'your', "doesn't", 'each', 'shouldn', "you'd", 'down', 'such',
'didn', 'them', 'isn', 'from', 'its', 'while', 'than', 'won', 'there', 'or',
'we', 'above', 'hasn', "she's", 'to', 'having', 'here', 'own', 'were',
'themselves', 'himself', 'will', 'had', 'same', "it's", 'd', 'doing', 'too',
"you'll", 're', 'are', 'very', 'haven', 'for', 's', "wasn't", 'all', "mustn't",
"wouldn't", 'below', 'after', "couldn't", 'yourselves', 'shan', 'through', 'in',
'other', 'between', "you've", 'these', 'by', "won't", 'have', 'hadn', 'further',
'only', 'at', 'itself', 'couldn', 'those', 'then', 'this', 'she', 'during',
'don', 'me', 'more', 'mightn', 'been', 'does', 'few', 'now', 'o', 'but',
'yourself', 'theirs', 'the', 'their', 'be', 'if', 'not', 'where', "hasn't",
'an', "aren't", "haven't", 'about', 'can', 'ain', 'under', 'it', 'hers', 'i',
'wasn', 'nor', 'how', 'ourselves', 'our', 'him', 'my', 'her', 'against',
"mightn't", 'a', 'off', 'ours', 'll', "should've", 'as', 'has', "you're", 'you',
"don't", 'and', 'did', 'am', 'who', "that'll", 'out', 'with', "needn't", 'over',
'just', "isn't", 'he'}
```

[38]:
```
filtered_sent=[]
for w in tokenize_word:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenize_word)
print('\n')
print("Filterd Sentence:",filtered_sent)
```

```
Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing',
'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is',
'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', "shouldn't",
'eat', 'cardboard']


Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather',
'great', ',', 'city', 'awesome', '.', 'The', 'sky', 'pinkish-blue', '.', 'You',
'eat', 'cardboard']
```

[40]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
stemmed_words=[]
for w in filtered_sent:
 stemmed_words.append(ps.stem(w))
print("Filtered Sentence:",filtered_sent)
print('\n')
print("Stemmed Sentence:",stemmed_words)
```

```
Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The',
'weather', 'great', ',', 'city', 'awesome', '.', 'The', 'sky', 'pinkish-blue',
'.', 'You', 'eat', 'cardboard']


Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?', 'the', 'weather',
'great', ',', 'citi', 'awesom', '.', 'the', 'sky', 'pinkish-blu', '.', 'you',
'eat', 'cardboard']
```

[ ]:
```
import nltk
nltk.download('wordnet')
```

[44]:
```
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print('\n')
print("Stemmed Word:",stem.stem(word))
```

```
Lemmatized Word: fly


Stemmed Word: fli
```

[45]:
```
import pandas as pd
import sklearn as sk
import math
first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"
first_sentence= first_sentence.split(" ")
second_sentence = second_sentence.split(" ")
total=set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'21st', 'for', 'is', 'of', 'sexiest', 'data', 'learning', 'Data', 'key',
'century', 'machine', 'job', 'Science', 'the', 'science'}
```

[48]:
```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
 wordDictA[word]+=1

for word in second_sentence:
 wordDictB[word]+=1
pd.DataFrame([wordDictA, wordDictB])
```

[48]:

	21st	for	is	of	sexiest	data	learning	Data	key	century	machine
0	1	0	1	1	1	0	0	1	0	1	0
1	0	1	1	0	0	1	1	0	1	0	1

	job	Science	the	science
0	1	1	2	0
1	0	0	1	1

[57]:
```
def computeTF(wordDict, doc):
 tfDict = {}
 corpusCount = len(doc)
 for word, count in wordDict.items():
     tfDict[word] = count/float(corpusCount)
 return(tfDict)                                          #running our sentences␣
 ↪through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)                    ␣
 ↪#Converting to dataframe for visualization
tf=pd.DataFrame([tfFirst,tfSecond])
print(tf)
```

	21st	for	is	of	sexiest	data	learning	Data	key	century
0	0.1	0.000	0.100	0.1	0.1	0.000	0.000	0.1	0.000	0.1
1	0.0	0.125	0.125	0.0	0.0	0.125	0.125	0.0	0.125	0.0

	machine	job	Science	the	science
0	0.000	0.1	0.1	0.200	0.000
1	0.125	0.0	0.0	0.125	0.125

[58]:
```
from nltk.corpus import stopwords
stop_words =set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['21st', 'sexiest', 'data', 'learning', 'Data', 'key', 'century', 'machine',
'job', 'Science', 'science']
```

[62]:
```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)
    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))
    return(idfDict)

#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
print(idfs)
```

```
{'21st': 0.3010299956639812, 'for': 0.3010299956639812, 'is':
0.3010299956639812, 'of': 0.3010299956639812, 'sexiest': 0.3010299956639812,
'data': 0.3010299956639812, 'learning': 0.3010299956639812, 'Data':
0.3010299956639812, 'key': 0.3010299956639812, 'century': 0.3010299956639812,
'machine': 0.3010299956639812, 'job': 0.3010299956639812, 'Science':
0.3010299956639812, 'the': 0.3010299956639812, 'science': 0.3010299956639812}
```

[64]:
```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)

#running our two sentences through the IDF:
idfFirst =computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond])
print(idf)
```

```
       21st       for        is        of   sexiest      data  learning  \
0  0.030103  0.000000  0.030103  0.030103  0.030103  0.000000  0.000000
1  0.000000  0.037629  0.037629  0.000000  0.000000  0.037629  0.037629

       Data       key   century   machine       job   Science       the  \
0  0.030103  0.000000  0.030103  0.000000  0.030103  0.030103  0.060206
1  0.000000  0.037629  0.000000  0.037629  0.000000  0.000000  0.037629

    science
0  0.000000
1  0.037629
```

[67]:
```
#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
#for the sentence, make sure all words are lowercase or you will run
#into error. for simplicity, I just made the same sentence all
#lowercase

firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"

#calling the TfidfVectorizer
vectorize= TfidfVectorizer()

#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])
print(response)
```

```
  (0, 2)        0.24342026926924518
  (0, 10)       0.24342026926924518
  (0, 4)        0.24342026926924518
  (0, 12)       0.48684053853849035
  (0, 11)       0.34211869506421816
  (0, 5)        0.34211869506421816
  (0, 9)        0.34211869506421816
  (0, 0)        0.34211869506421816
  (0, 1)        0.34211869506421816
  (1, 2)        0.28986933576883284
  (1, 10)       0.28986933576883284
  (1, 4)        0.28986933576883284
  (1, 12)       0.28986933576883284
  (1, 8)        0.40740123733358447
  (1, 7)        0.40740123733358447
  (1, 6)        0.40740123733358447
  (1, 3)        0.40740123733358447
```