IN4080_2022_Ex_1_solutions

August 24, 2022

[1]:
```
import numpy as np
import nltk
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, …, text9 and sent1, …, sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

0.1 NLTK book Ch. 1, sec. 3 Simple statistics

[2]:
```
fdist1 = nltk.FreqDist(text1)
fdist1
```

[2]:
```
FreqDist({',': 18713, 'the': 13721, '.': 6862, 'of': 6536, 'and': 6024, 'a':
4569, 'to': 4542, ';': 4072, 'in': 3916, 'that': 2982, …})
```

[3]:
```
fdist1.max()
```

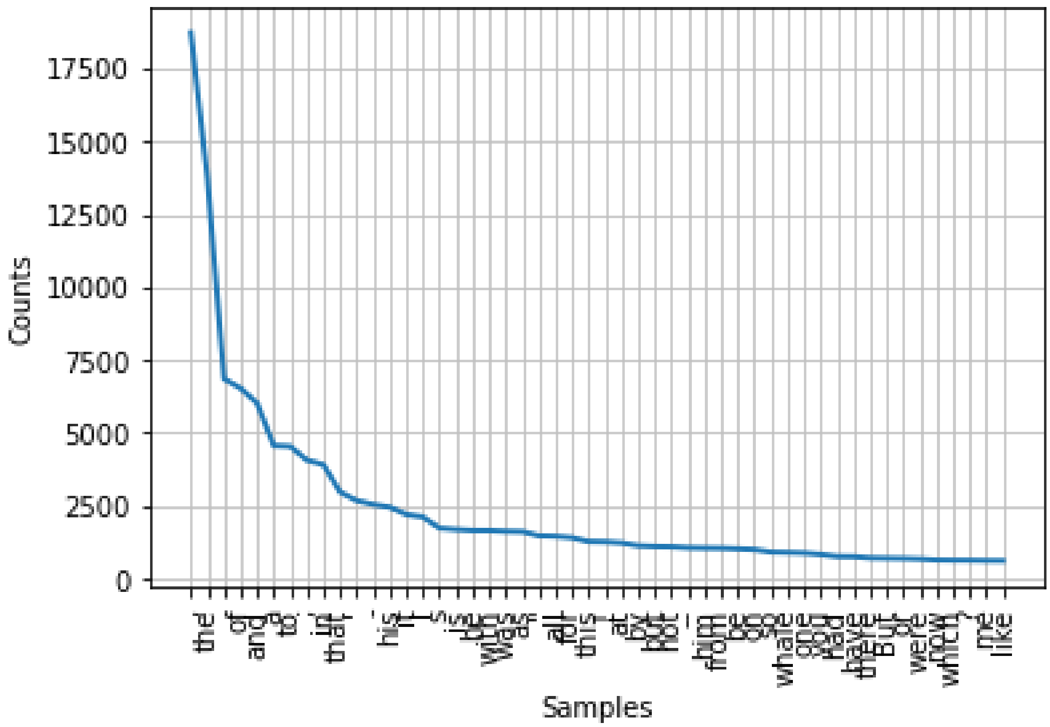
[3]:
```
','
```

[4]:
```
fdist1.most_common(50)
```

[4]:
```
[(',', 18713),
 ('the', 13721),
 ('.', 6862),
 ('of', 6536),
 ('and', 6024),
 ('a', 4569),
 ('to', 4542),
 (';', 4072),
 ('in', 3916),
 ('that', 2982),
 ("'", 2684),
 ('-', 2552),
 ('his', 2459),
 ('it', 2209),
 ('I', 2124),
 ('s', 1739),
 ('is', 1695),
 ('he', 1661),
 ('with', 1659),
 ('was', 1632),
 ('as', 1620),
 ('"', 1478),
 ('all', 1462),
 ('for', 1414),
 ('this', 1280),
 ('!', 1269),
 ('at', 1231),
 ('by', 1137),
 ('but', 1113),
 ('not', 1103),
 ('--', 1070),
 ('him', 1058),
 ('from', 1052),
 ('be', 1030),
 ('on', 1005),
 ('so', 918),
 ('whale', 906),
 ('one', 889),
 ('you', 841),
 ('had', 767),
 ('have', 760),
 ('there', 715),
 ('But', 705),
 ('or', 697),
 ('were', 680),
 ('now', 646),
 ('which', 640),
 ('?', 637),
 ('me', 627),
 ('like', 624)]
```

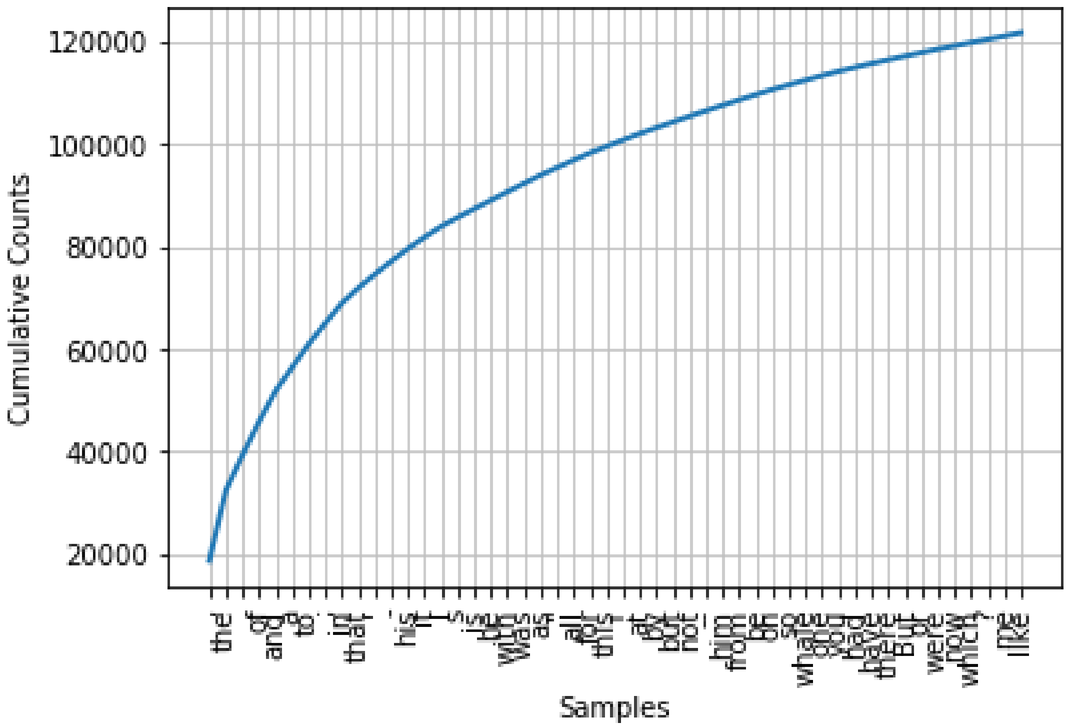
[5]:

```
fdist1.plot(50)
```



```
[5]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

```
[6]: fdist1.plot(50, cumulative=True)
```



[6]: <AxesSubplot:xlabel='Samples', ylabel='Cumulative Counts'>

[7]:
```python
fdist1["whale"]
```

[7]: 906

[8]:
```python
fdist1.freq("whale")
```

[8]: 0.003473673313677301

[9]:
```python
fdist1.tabulate(5)
```

```
    ,   the     .    of   and
18713 13721  6862  6536  6024
```

0.1.1 Exercises

[10]:
```python
# Exercise 1
def my_frequency(j):
    d={}
    for it in j:
        if it not in d:
            d[it]=1
        else:
            d[it]+=1
    return d
```

[11]:

```
a = """this is a stupid sentence
this is also stupid
this not sentence""".split()
a
```

[11]:

```
['this',
 'is',
 'a',
 'stupid',
 'sentence',
 'this',
 'is',
 'also',
 'stupid',
 'this',
 'not',
 'sentence']
```

[12]:

```
my = my_frequency(a)
```

[13]:

```
my.items()
```

[13]:

```
dict_items([('this', 3), ('is', 2), ('a', 1), ('stupid', 2), ('sentence', 2),
('also', 1), ('not', 1)])
```

[14]:

```
my.keys()
```

[14]:

```
dict_keys(['this', 'is', 'a', 'stupid', 'sentence', 'also', 'not'])
```

[15]:

```
my.values()
```

[15]:

```
dict_values([3, 2, 1, 2, 2, 1, 1])
```

[16]:

```
my.values()
```

[16]:

```
dict_values([3, 2, 1, 2, 2, 1, 1])
```

Comparing to Counter

[17]:

```
from collections import Counter
```

[18]:

```
myc = Counter(a)
```

[19]:

```
myc.keys()
```

```
[19]: dict_keys(['this', 'is', 'a', 'stupid', 'sentence', 'also', 'not'])
```

```
[20]: myc.most_common(3)
```

```
[20]: [('this', 3), ('is', 2), ('stupid', 2)]
```

0.2 Exercise 2

```
[21]: # NLTK Ch. 1 Exercise 27
      def vocab_size(text):
          # text is list of tokens
          fd = nltk.FreqDist(text)
          return len(fd.keys())
```

```
[22]: vocab_size(text1)
```

```
[22]: 19317
```

```
[23]: vocab_size(text9)
```

```
[23]: 6807
```

```
[24]: # NLTK Ch. 1 Exercise 28
      def percent(word, text):
          return 100*text.count(word)/len(text)
```

```
[25]: percent('the', text1)
```

```
[25]: 5.260736372733581
```

0.3 NLTK book, Ch. 2, Sec. 1.0-1.5 Accessing Text Copora

```
[26]: emma = nltk.corpus.gutenberg.words('austen-emma.txt')
      len(emma)
```

```
[26]: 192427
```

```
[27]: from nltk.corpus import gutenberg
      for fileid in gutenberg.fileids():
          num_chars = len(gutenberg.raw(fileid))
          num_words = len(gutenberg.words(fileid))
          num_sents = len(gutenberg.sents(fileid))
          num_vocab = len(set(w.lower() for w in gutenberg.words(fileid)))
          print(round(num_chars/num_words, 2), round(num_words/num_sents),␣
       ↪round(num_words/num_vocab), fileid)
```

```
4.61 25 26 austen-emma.txt
4.75 26 17 austen-persuasion.txt
4.75 28 22 austen-sense.txt
4.29 34 79 bible-kjv.txt
4.57 19 5 blake-poems.txt
4.49 19 14 bryant-stories.txt
4.46 18 12 burgess-busterbrown.txt
4.23 20 13 carroll-alice.txt
4.72 20 12 chesterton-ball.txt
4.72 23 11 chesterton-brown.txt
4.63 19 11 chesterton-thursday.txt
4.44 21 25 edgeworth-parents.txt
4.77 26 15 melville-moby_dick.txt
4.84 52 11 milton-paradise.txt
4.35 12 9 shakespeare-caesar.txt
4.36 12 8 shakespeare-hamlet.txt
4.34 12 7 shakespeare-macbeth.txt
4.59 36 12 whitman-leaves.txt
```

[28]:

```python
from nltk.corpus import webtext
for fileid in webtext.fileids():
    print(fileid, webtext.raw(fileid)[:65], '...')
```

```
firefox.txt Cookie Manager: "Don't allow sites that set removed cookies to se
…
grail.txt SCENE 1: [wind] [clop clop clop]
KING ARTHUR: Whoa there!  [clop …
overheard.txt White guy: So, do you have any plans for this evening?
Asian girl …
pirates.txt PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terr
…
singles.txt 25 SEXY MALE, seeks attrac older single lady, for discreet encoun
…
wine.txt Lovely delicate, fragrant Rhone wine. Polished leather and strawb …
```

[29]:

```python
from nltk.corpus import brown
brown.categories()
```

[29]:

```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

[30]:

```
news_text = brown.words(categories='news')
fdist = nltk.FreqDist(w.lower() for w in news_text)
modals = ['can', 'could', 'may', 'might', 'must', 'will']
for m in modals:
    print(m + ':', fdist[m], end=' ')
```

```
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

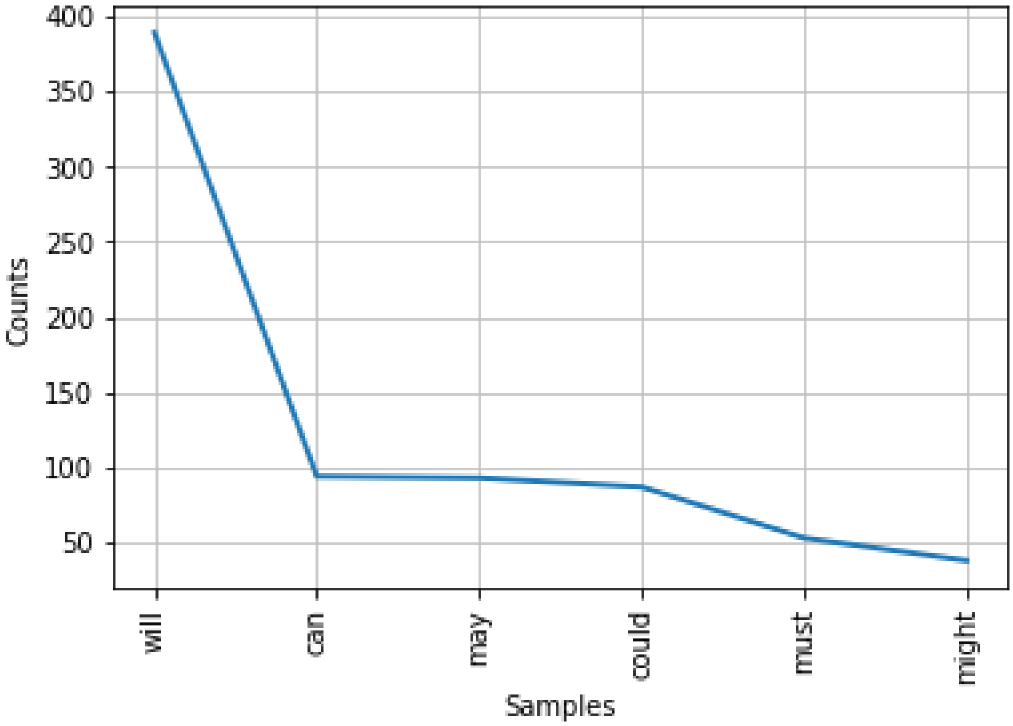
[31]:

```
fmdist = nltk.FreqDist(w.lower() for w in news_text if w.lower() in modals)
for m in modals:
    print(m + ':', fmdist[m], end=' ')
```

```
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

[32]:

```
fmdist.plot()
```

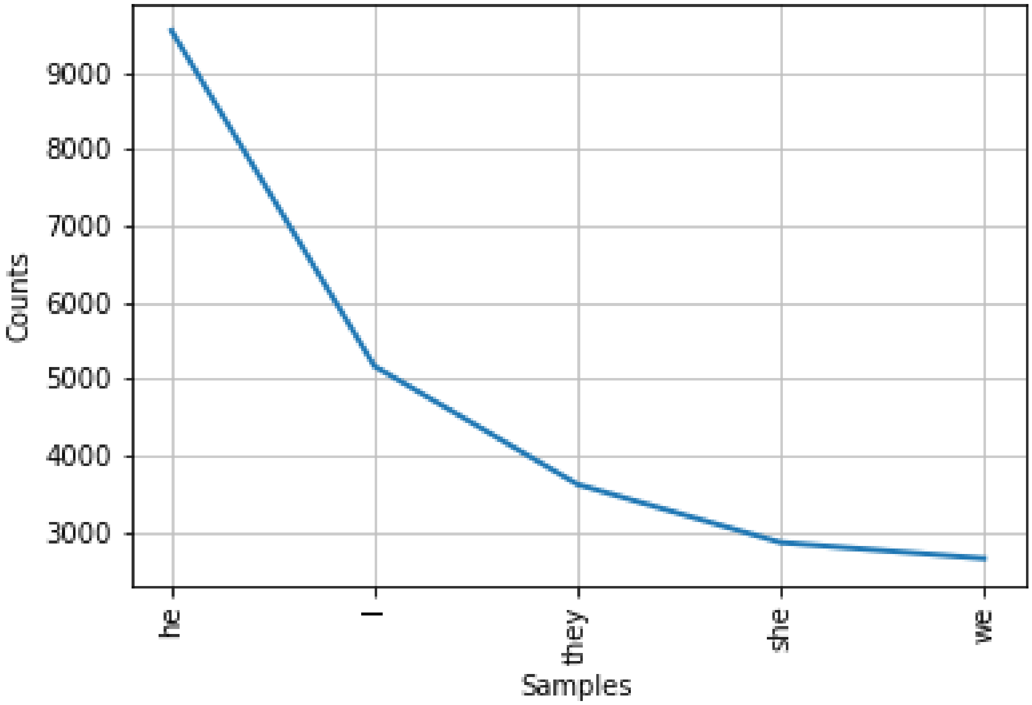


```
[32]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

0.4 Exercise 3

```
[33]: # Exercise 3
      # How to handle lower case?
      labels = ['i', 'we', 'he', 'she', 'they']

      brown_freq=nltk.FreqDist(
          [w.lower() for w in brown.words() if
           w.lower() in set(labels)])

      # To replace 'i' with 'I' as key:
      v=brown_freq.pop('i')
      brown_freq['I']=v

      brown_freq.tabulate()
      brown_freq.plot()
```

```
  he    I they  she   we
9548 5164 3620 2860 2652
```



```
[33]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

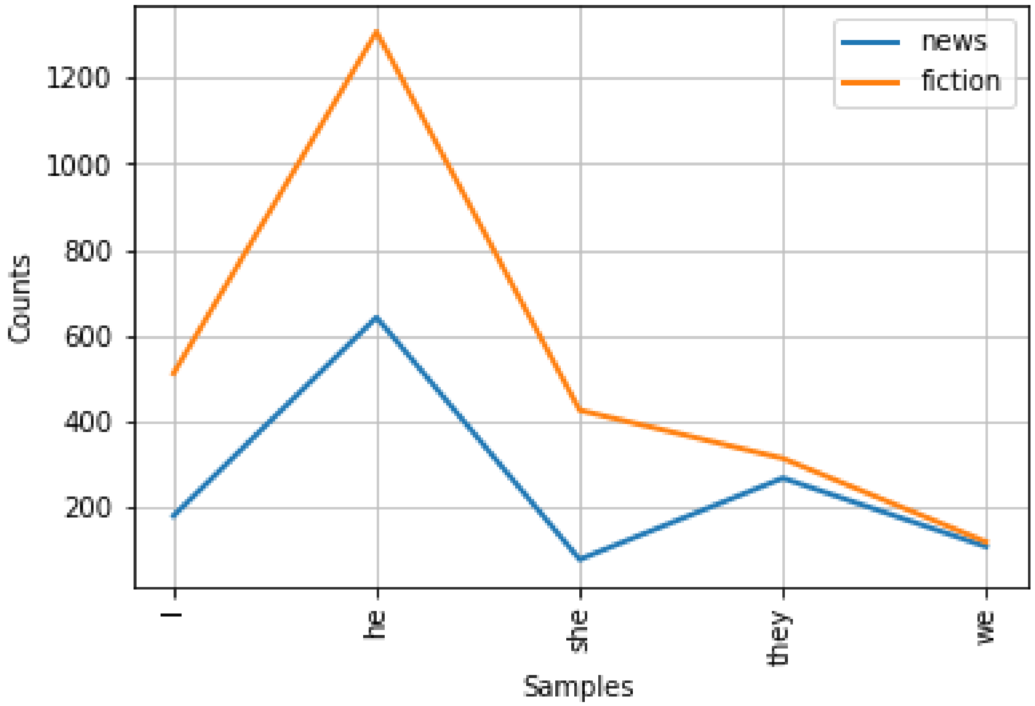
[34]:

```
# Exercise 4
# the following does not do lower case correctly
cond = nltk.ConditionalFreqDist([(genre,word)
for genre in ['news','fiction']
for word in brown.words(categories=genre)
if word in set(labels)])

# to get lower case:
cf_low = nltk.ConditionalFreqDist([(genre,word.lower())
for genre in ['news','fiction']
for word in brown.words(categories=genre)
if word.lower() in set(labels)])

# to decorate the result:
cf_low['news']['I']=cf_low['news'].pop('i')
cf_low['fiction']['I']=cf_low['fiction'].pop('i')

print("\nTable without case folding:")
cond.tabulate()

print("\nTable with case folding:")
cf_low.tabulate()
```

```
Table without case folding:
          he  she they   we
fiction  813  280  230   85
   news  451   42  205   77

Table with case folding:
           I   he  she they   we
fiction  511 1308  425  313  118
   news  179  642   77  267  107
```

[35]:

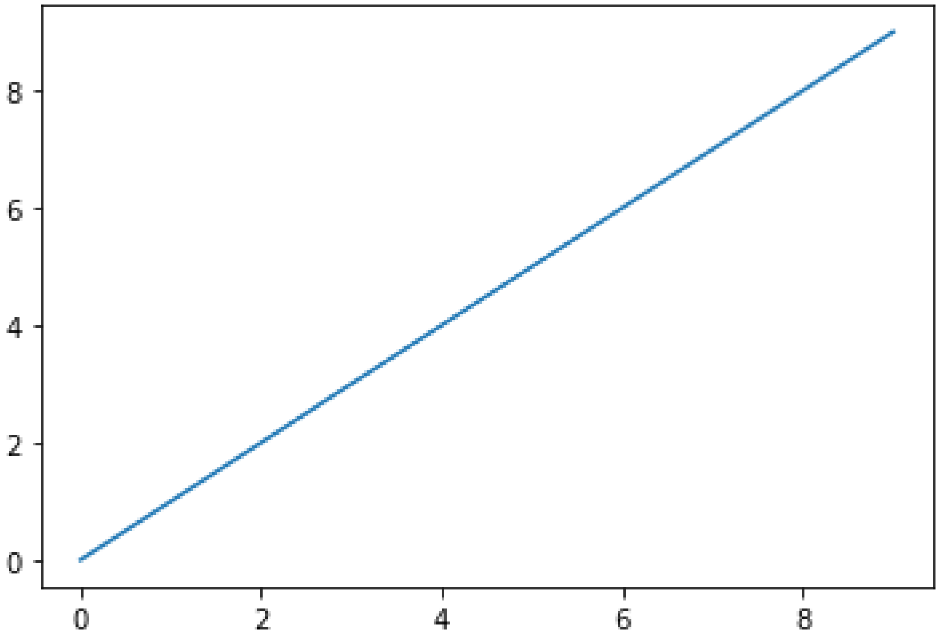
```
cf_low.plot()
```



```
[35]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

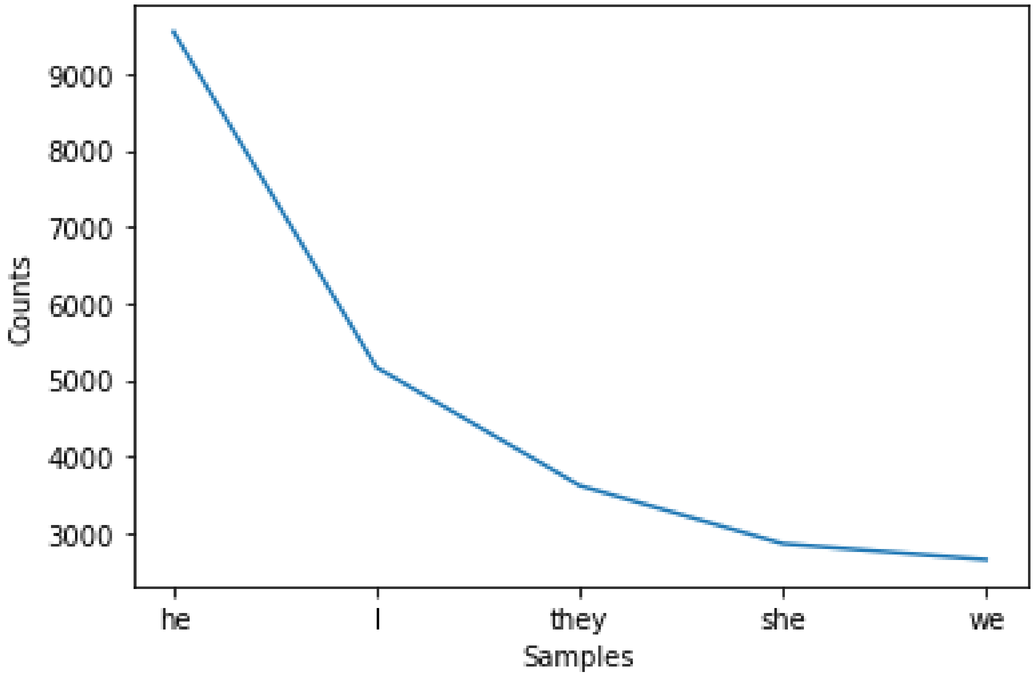
0.5 matplotlib

```
[36]: import matplotlib.pyplot as plt
plt.plot(np.arange(10))
plt.show()
```



0.5.1 Exercise 5

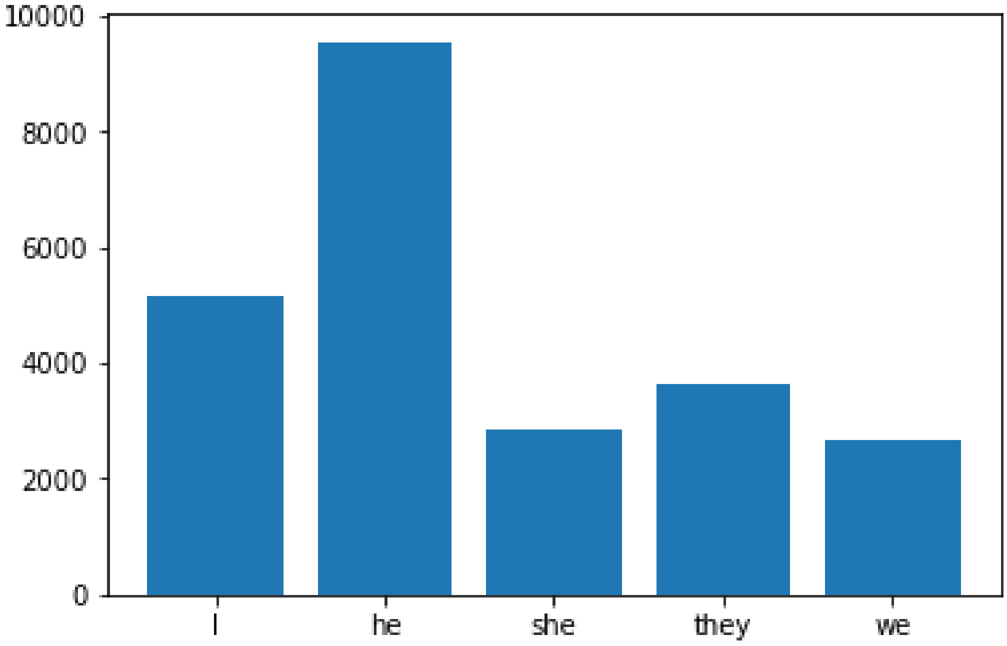
[37]:
```
plt.plot(np.arange(len(brown_freq)), [f for w,f in brown_freq.most_common()])
plt.xticks(np.arange(len(brown_freq)),[w for w,f in brown_freq.most_common()])
plt.xlabel('Samples')
plt.ylabel('Counts')
plt.show()
```



0.5.2 Exercise 6

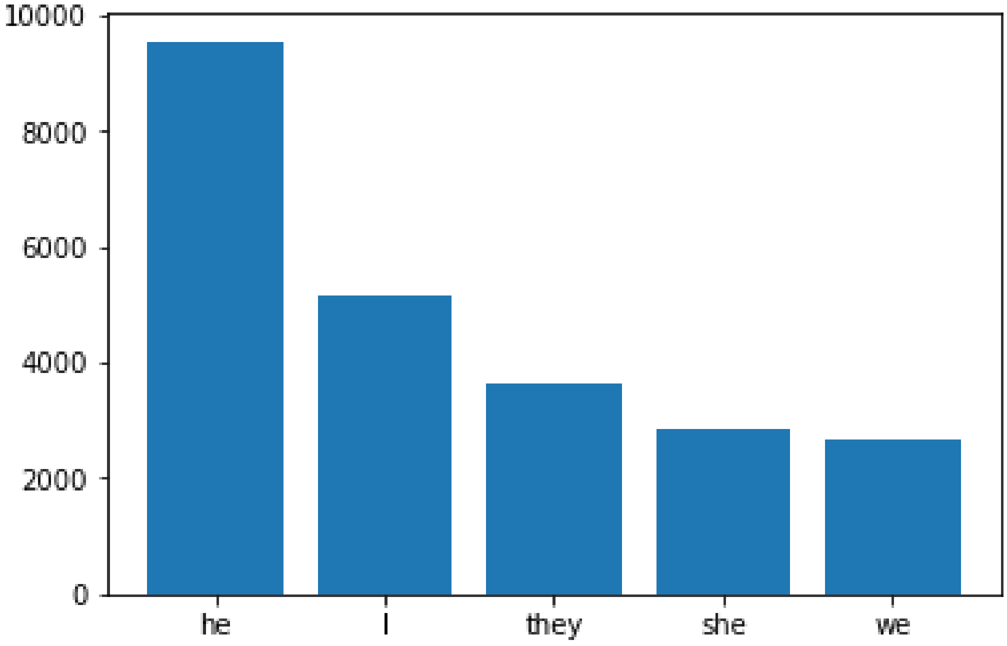
[38]:
```
xs = sorted([k for k in brown_freq.keys()])
```

[39]:
```
plt.bar(np.arange(len(brown_freq)),
        [brown_freq[k] for k in xs],
        tick_label=xs)
plt.show()
```



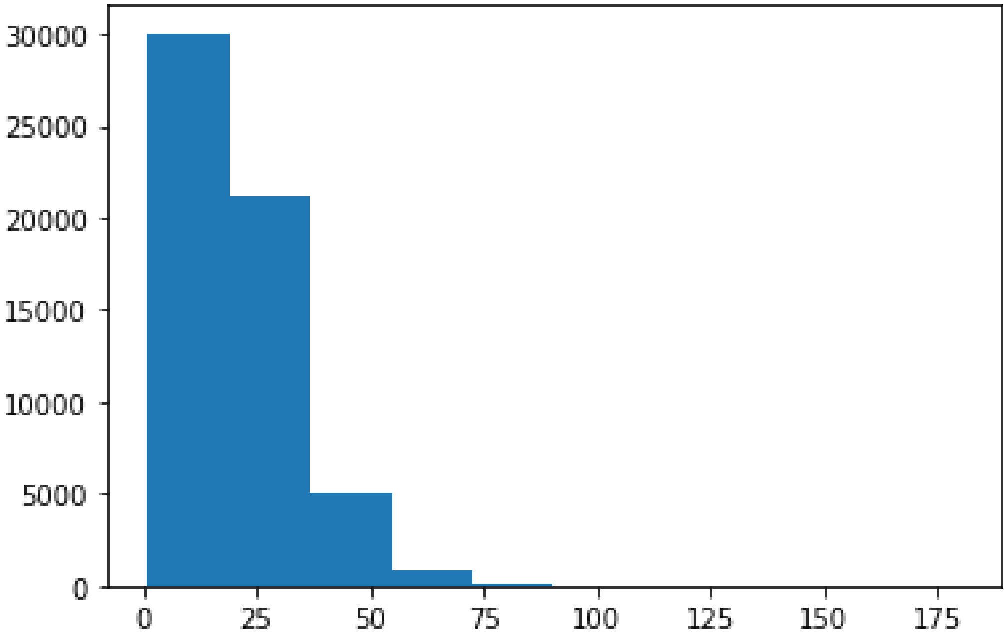
Observe that *I* is sorted before *he* because of the capital.

```
[40]: plt.bar( np.arange(len(brown_freq)),
              [f for w,f in brown_freq.most_common()],
              tick_label=[w for w,f in brown_freq.most_common()])
      plt.show()
```

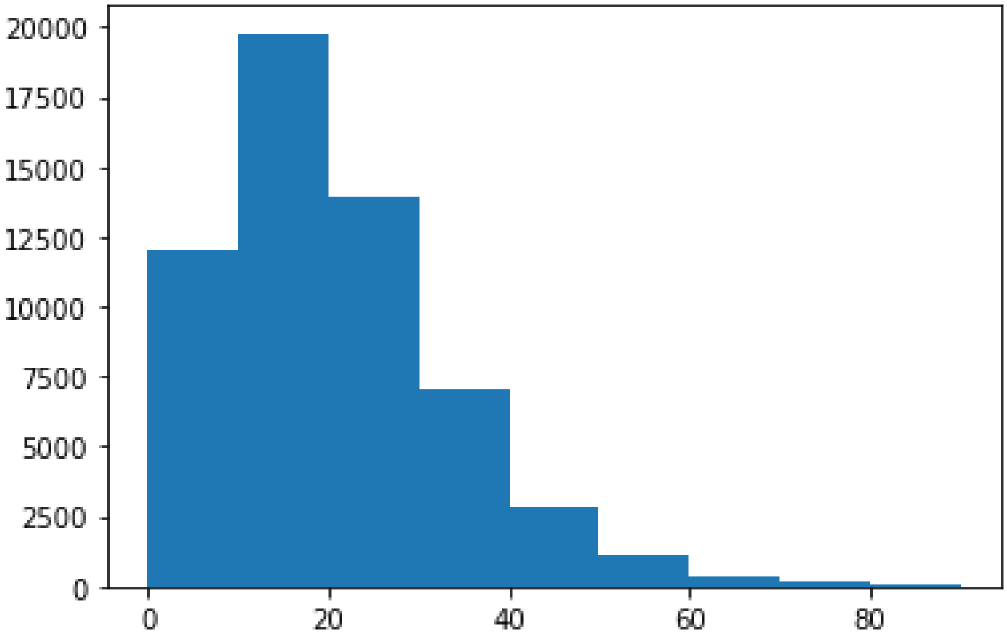


0.5.3 Exercise 7

[41]:
```
plt.hist([len(sent) for sent in brown.sents()])
plt.show()
```

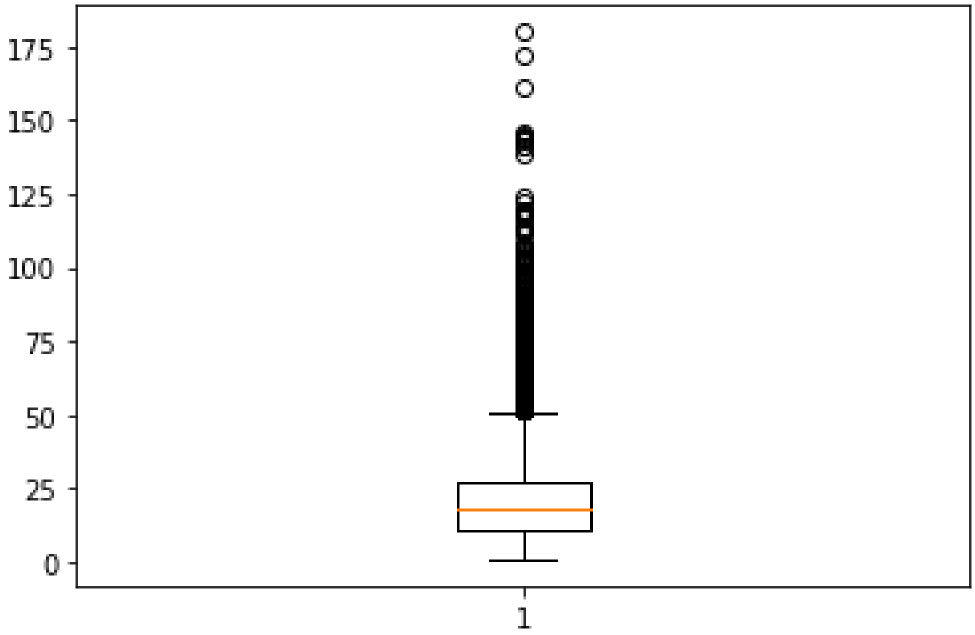


[42]:
```
plt.hist([len(sent) for sent in brown.sents()], range(0,100,10))
plt.show()
```



0.5.4 Exercise 8

[43]:
```
plt.boxplot([len(sent) for sent in brown.sents()])
plt.show()
```


175
150
125
100
75
50
25
0
1