Assignment 2

Dylan Phelan
Working With Corpora
Professor Gregory Crane

September 21, 2018

Exercises for Chapter 2: Accessing Text Corpora and Lexical Resources

Problem 1

Create a variable `phrase` containing a list of words. Review the operations described in the previous chapter, including addition, multiplication, indexing, slicing, and sorting.

Solution:

We can start by defining a variable `phrase` and *add* a bit more detail to it

```
>>> phrase = ["This", "should", "satisfy", "question", "one's", "requirements"]
>>> phrase = phrase + [',' 'I', 'hope', ]
>>> phrase
  ['This', 'should', 'satisfy', 'question', "one's", 'requirements', ',I', 'hope',
      '!']
```

Maybe we want *multiply* our enthusiasm

```
>>> enthusiasm_multiplier = ["!"]
>>> enthusiasm_multiplier *= 5
>>> enthusiasm_multiplier
['!', '!', '!', '!', '!']
>>> phrase += enthusiasm_multiplier
>>> phrase
['This', 'should', 'satisfy', 'question', "one's", 'requirements', ',I', 'hope',
    '!', '!', '!', '!', '!', '!']
```

Let's *slice* out that uncertainty and infuse a bit of confidence into that phrase. You know who has confidence? *Index*-fund traders... a stretch, I know

```
>>> phrase[6:8]
[',I', 'hope']
>>> phrase = phrase[0:6] + phrase[8:]
>>> phrase
['This', 'should', 'satisfy', 'question', "one's", 'requirements', '!', '!', '!',
    '!', '!', '!']
>>> phrase[1]
```

```
'should'
>>> phrase[1] = 'will'
>>> phrase
['This', 'will', 'satisfy', 'question', "one's", 'requirements', '!', '!', '!',
    '!', '!', '!']
>>>
```

And I can't seem to *sort* out a good pun for this last operation... so this meta-pun will have to do.

```
>>> yoda_phrase = sorted(phrase)
>>> yoda_phrase
['!', '!', '!', '!', '!', '!', 'This', "one's", 'question', 'requirements',
    'satisfy', 'will']
```

Problem 2

Use the corpus module to explore austen-persuasion.txt. How many word tokens does this book have? How many word types?

Solution:

```
>>> from nltk.corpus import gutenberg
>>> gutenberg.words('austen-persuasion.txt')
['[', 'Persuasion', 'by', 'Jane', 'Austen', '1818', ...]
>>> words = len(gutenberg.words('austen-persuasion.txt'))
>>> words
98171
>>> word_types = set(gutenberg.words('austen-persuasion.txt'))
>>> len(word_types)
6132
```

Problem 3

Use the Brown corpus reader nltk.corpus.brown.words() or the Web text corpus reader nltk.corpus.webtext.words() to access some sample text in two different genres.

Solution: Let's see what terms lie at the interesection of hobbies and sci_fi, looking only at the first 200 and only at terms that are longer than five characters

```
>>> from nltk.corpus import brown
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies',
    'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews',
    'romance', 'science_fiction']
>>> hobbies_words = brown.words(categories="hobbies")
>>> hobbies_words
['Too', 'often', 'a', 'beginning', 'bodybuilder', ...]
>>> sci_fi_words = brown.words(categories="science_fiction")
```

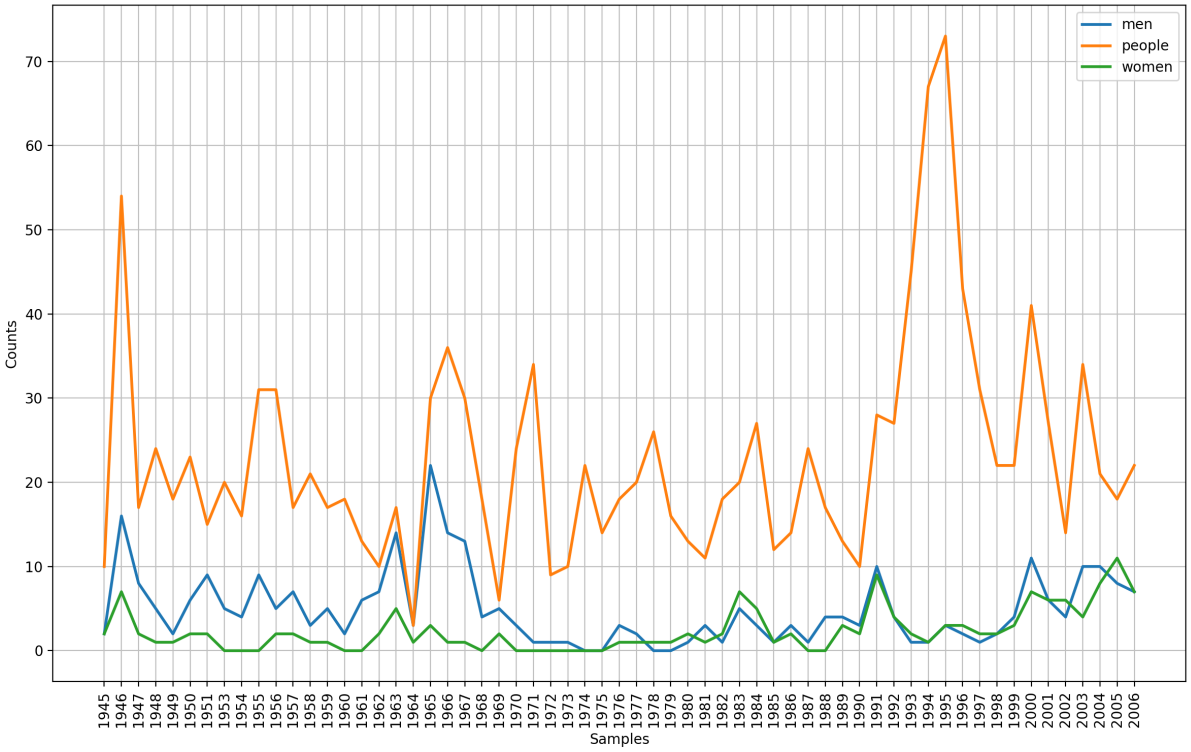
```
>>> sci_fi_words
['Now', 'that', 'he', 'knew', 'himself', 'to', 'be', ...]
>>> sci_fi_hobby_words = set(hobbies_words).intersection(set(sci_fi_words))
>>> fdistf_sci_fi_hobbies_terms = nltk.FreqDist(w.lower() for w in
    sci_fi_hobby_words)
>>> fdist_sci_fi_hobbies_terms = nltk.FreqDist(w.lower() for w in
    sci_fi_hobby_words)
>>> [word for (word, freq) in fdist_sci_fi_hobbies_terms.most_common(200) if
    len(word) > 5]
['school', 'according', 'neither', 'though', 'without', 'sometimes', 'actually',
    "you've", "there's", "that's", 'western', 'system', 'another', 'nevertheless',
    'therefore', 'nobody', 'almost', 'before', 'through', 'perhaps', 'having',
    'animals', 'nations', 'southern', 'behind', 'although', 'between', 'instead',
    'service', 'states', 'however', 'nothing', 'second', 'function', "you're",
    'selective', 'lifted', 'children', 'species', 'special', 'electronic',
    'knotty', 'nerves', 'fertile', 'stayed', 'condition', 'seeing', 'identical',
    "they'll", 'century', 'immediate', 'wanted', 'estimate', 'content', 'vacuum',
    'lessons', 'valuable', 'equipment', 'distinguished', 'gently', 'commonplace',
    'shining', 'contained']
```

Problem 4

Read in the texts of the State of the Union addresses, using the state_union corpus reader. Count occurrences of men, women, and people in each document. What has happened to the usage of these words over time?

Solution: Based on the figure below, we can determine a few insights:

1. The term 'women' is consistently less popular than the term 'men' until the 1970's, after which time the two are much more consistent in frequency.
2. The term 'people' has historically been a more popular term than 'men' or 'women'.
3. The term 'people' had a major spike in popularity the mid 90's.
4. The use of the term 'people' looks like it has been on the decline in the last 10 years of recorded data, whereas the terms 'men' and 'women' are increasing in popularity in that same period.



Generated with the following code:

```
>>> from nltk.corpus import state_union
>>> cfd = nltk.ConditionalFreqDist(
...     (target, fileid[:4])
...     for fileid in state_union.fileids()
...     for w in state_union.words(fileid)
...     for target in ['men', 'women', 'people']
...     if w.lower().startswith(target)
... )
>>> cfd.plot()
```

Problem 5

Investigate the holonym-meronym relations for some nouns. Remember that there are three kinds of holonym-meronym relation, so you need to use: member_meronyms(), part_meronyms(), substance_meronyms(), member_holonyms(), part_holonyms(), and substance_holonyms().

Solution: Originally I was trying individual terms that I thought could have various types of meronyms and holonyms, which was painfully fruitless. To investigate the problem more deeply, I wrote a small script that could print out various terms with holonyms and meronyms from Moby Dick, as provided by the `nltk.corpus.book` module:

```
>>> from nltk.book import *
>>> from nltk.corpus import wordnet as wn
>>> for w in vocab_1[:600]:
...     avail_synsets = wn.synsets(w)
...     if len(avail_synsets) > 0:
...         # For all the synsets, look for holobyms and meronyms
...         for synset in avail_synsets:
...             member_m = synset.member_meronyms()
...             part_m = synset.part_meronyms()
...             substance_m = synset.substance_meronyms()
...             member_h = synset.member_holonyms()
...             part_h = synset.part_holonyms()
...             substance_h = synset.substance_holonyms()
...             # If we have any valid holonyms or meronyms, print them out for
    viewing
...             if (member_m or part_m or substance_m or member_m or part_m or
    substance_m):
...                 print('\n\nSynset: ', synset)
...                 if (member_m or part_m or substance_m):
...                     print("--- meronyms")
...                     if len(member_m) > 0: print("member_m: ", member_m)
...                     if len(part_m) > 0: print("part_m: ", part_m)
...                     if len(substance_m) > 0: print("substance_m: ", substance_m)
...                 if (member_h or part_h or substance_h):
...                     print("--- holonymss")
...                     if len(member_h) > 0: print("member_h: ", member_h)
...                     if len(part_h) > 0: print("part_h: ", part_h)
...                     if len(substance_h) > 0: print("substance_h: ", substance_h)
```

Problem 9

Pick a pair of texts and study the differences between them, in terms of vocabulary, vocabulary richness, genre, etc. Can you find pairs of words which have quite different meanings across the two texts, such as monstrous in Moby Dick and in Sense and Sensibility?

Solution: Solution goes here

Problem 23

Let f(w) be the frequency of a word w in free text. Suppose that all the words of a text are ranked according to their frequency, with the most frequent word first. Zipf's law states that the frequency of a word type is inversely proportional to its rank (i.e. f r = k, for some constant k). For example, the 50th most common word type should occur three times as frequently as the 150th most common word type. Write a function to process a large text and plot word frequency against word rank using pylab.plot. Do you confirm Zipf's law? (Hint: it helps to use a logarithmic scale). What is going on at the extreme ends of the plotted line?

1. Write a function to process a large text and plot word frequency against word rank using pylab.plot. Do you confirm Zipf's law? (Hint: it helps to use a logarithmic scale). What is going on at the extreme ends of the plotted line?

2. Generate random text, e.g., using random.choice("abcdefg "), taking care to include the space character. You will need to import random first. Use the string concatenation operator to accumulate characters into a (very) long string. Then tokenize this string, and generate the Zipf plot as before, and compare the two plots. What do you make of Zipf's Law in the light of this?

Solution: Solution goes here

Research Publication of Interest

Identify a recent research publication that interests you. Write a very short summary and explain why you found it interesting. Be prepared to discuss this in class next week. Suggested publications include (but are by no means limited to):