Reilly McBride

Period 5

February 26, 2018

NLTK Chapter 2 Exercises

**Exercise 7:** *According to Strunk and White's Elements of Style, the word however, used at the start of a sentence, means "in whatever way" or "to whatever extent", and not "nevertheless". They give this example of correct usage: However you advise him, he will probably do as he thinks best. (http://www.bartleby.com/141/strunk3.html) Use the concordance tool to study actual usage of this word in the various texts we have been considering.*

emma = nltk.Text(nltk.corpus.gutenberg.words(**"austen-emma.txt"**))  
paradise = nltk.Text(nltk.corpus.gutenberg.words(**"milton-paradise.txt"**))  
sense = nltk.Text(nltk.corpus.gutenberg.words(**"austen-sense.txt"**))  
bible = nltk.Text(nltk.corpus.gutenberg.words(**"bible-kjv.txt"**))  
hamlet = nltk.Text(nltk.corpus.gutenberg.words(**"shakespeare-hamlet.txt"**))  
  
print(nltk.corpus.gutenberg.fileids())  
  
**def** seven():  
 print(**"EMMA"**)  
 print(emma.concordance(**"However"**))  
 print(**"BIBLE"**)  
 print(bible.concordance(**"However"**))  
 print(**"HAMLET"**)  
 print(hamlet.concordance(**"However"**))

Emma exclusively used “however” to mean “nevertheless”, even if it was positioned at the start of a sentence. Neither the Bible nor Hamlet had any matches for “however”, which suggest its use has become more prominent throughout time, and is largely incorrect in texts if it occurs.

**Exercise 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?*

**def** nine():  
 print(**"emma"**)  
 emma.similar(**"love"**)  
 print(**"sense"**)  
 sense.similar(**"love"**)  
 emma.common\_contexts([**"love"**, **"marriage"**])  
 sense.common\_contexts([**"love"**, **"marriage"**])

For Emma, the words used in similar contexts as love were “it her him time highbury general and have life dear coming return short herself which that all consciousness want london”. For Sense and Sensibility, the words were “affection sister heart mother time see town life it dear Elinor marianne me word family her him do regard head”. Sense and Sensibility is more about sisterly and familial love than romantic love, and this is reflected in the names of the family members, and the words sister and mother themselves. The words similar to love in *Emma* are more about her surroundings—love for a place and a situation, less for a person. In *Emma*, love and marriage are both used in the context “the\_of”. In *Sense and Sensibility* love and marriage are both used in the contexts “her\_and” and “my\_i”. Again, love and marriage in *Sense and Sensibility* seems more personal and romantic, attached to a specific person, whereas they are more objective in *Emma.* For the record I have read both of these books so this really does reflect accurately upon them.

**Exercise 12:** *The CMU Pronouncing Dictionary contains multiple pronunciations for certain words. How many distinct words does it contain? What fraction of words in this dictionary have more than one possible pronunciation?*

**def** twelve():  
 entries = nltk.corpus.cmudict.entries()  
 words = set(entry[0] **for** entry **in** entries)  
 print(len(words))  
 dictionary = nltk.corpus.cmudict.dict()  
 unique = 0  
 **for** key **in** dictionary.keys():  
 **if** len(dictionary[key]) > 1:  
 unique += 1  
 print(unique/len(words))

Total Number of Unique Words: 9241

Percent of Number of Unique Words that Have Multiple Pronunciations: 7.49%

**Exercise 13:** *What percentage of noun synsets have no hyponyms? You can get all noun synsets using wn.all\_synsets('n').*

**def** thirteen():  
 number\_with\_none = 0  
 synsets = list(nltk.corpus.wordnet.all\_synsets(**'n'**))  
 **for** syn **in** synsets:  
 **if** len(list(syn.hyponyms())) == 0:  
 number\_with\_none += 1  
 print((number\_with\_none/len(synsets)) \* 100)

Percentage: 79.67%

**Exercise 15:** *Write a program to find all words that occur at least three times in the Brown Corpus.*

**def** fifteen():  
 count = 0  
 fdist = nltk.FreqDist(word **for** word **in** nltk.corpus.brown.words())  
 **for** entry **in** fdist:  
 **if** fdist[entry] >= 3:  
 count += 1  
 print(count)

Answer: 22339

**Exercise 17:** *Write a function that finds the 50 most frequently occurring words of a text that are not stopwords.*

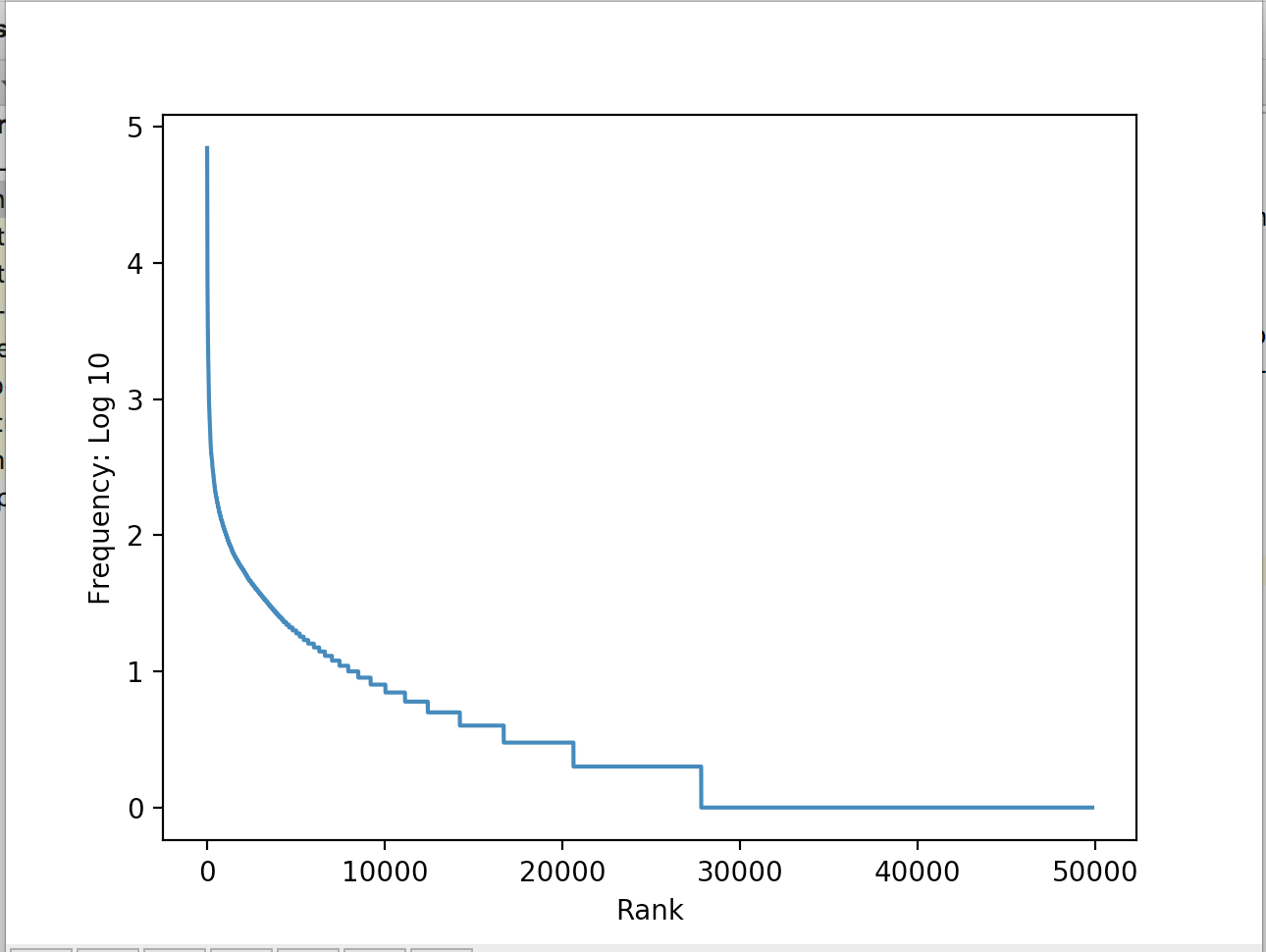
**def** seventeen():  
 fdist = nltk.FreqDist(word **for** word **in** emma **if** word.lower() **not in** nltk.corpus.stopwords.words(**'english'**) **and** word.isalpha())  
 print(fdist.most\_common(50))

Results: [('Mr', 1153), ('Emma', 865), ('could', 825), ('would', 815), ('Mrs', 699), ('Miss', 592), ('must', 564), ('Harriet', 506), ('said', 484), ('much', 478), ('Weston', 439), ('one', 413), ('every', 398), ('thing', 398), ('Knightley', 389), ('Elton', 385), ('think', 380), ('little', 354), ('never', 347), ('good', 340), ('know', 337), ('might', 322), ('well', 315), ('Woodhouse', 313), ('say', 308), ('Jane', 301), ('time', 272), ('quite', 269), ('great', 263), ('Fairfax', 241), ('nothing', 237), ('always', 235), ('man', 233), ('thought', 226), ('Churchill', 223), ('soon', 221), ('see', 220), ('dear', 217), ('may', 213), ('shall', 212), ('without', 211), ('first', 209), ('Frank', 208), ('father', 207), ('sure', 204), ('made', 199), ('like', 199), ('body', 193), ('day', 190), ('young', 190)]

**Exercise 23:** *Zipf's Law: 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.*

1. *Write a function to process a large text and plot word frequency against word rank Do you confirm Zipf's law? What is going on at the extreme ends of the plotted line?*

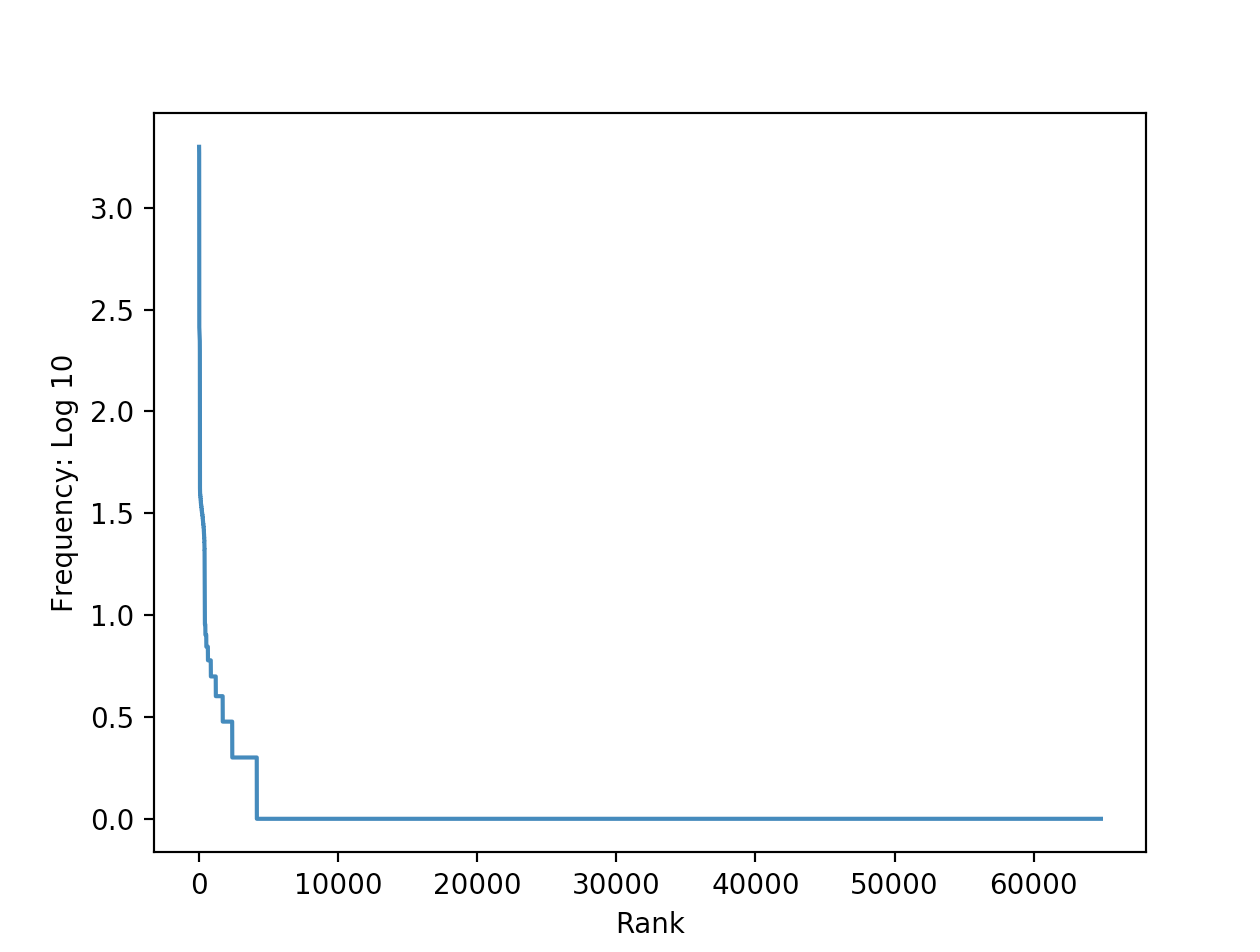
**def** process\_text(text):  
 fdist = nltk.FreqDist(word.lower() **for** word **in** text)  
 common\_ranked = fdist.most\_common(len(fdist))  
 x\_list = []  
 y\_list = []  
 count = 1  
 **for** i **in** common\_ranked:  
 x\_list.append(count)  
 count += 1  
 y\_list.append(math.log10(i[1]))  
 plt.plot(x\_list, y\_list)  
 plt.xlabel(**"Rank"**)  
 plt.ylabel(**"Frequency: Log 10"**)  
 plt.show()  
  
**def** twentythree():  
 process\_text(nltk.corpus.brown.words())



Yes, I have confirmed Zipf’s law, because this is very close to the standard inverse proportional relationship curve. The higher in rank (less frequent), the less it matches the ideal curve, perhaps because there would be a large number of words used at approximately the same low frequency.

1. *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?*

**def** twentythree():  
 *#PART A:  
 #process\_text(nltk.corpus.brown.words())  
 #PART B:* test\_string = **""** chars = [**'a'**, **'b'**, **'c'**, **'d'**, **'e'**, **'f'**, **'g'**, **' '**]  
 **for** i **in** range(1000000):  
 temp = random.choice(chars)  
 test\_string = test\_string + temp  
 test\_text = nltk.word\_tokenize(test\_string)  
 process\_text(test\_text)



This suggests that Zipf’s law is less accurate for random text, as, while this still does approximate the inverse proportional curve, it is less ideal than a real human-produced text. It seems a largely applicable law, but not a perfect fit. Increasing the number of randomly generated characters by a factor of 10 did not noticeably change the graph.

**Exercise 26:** *What is the branching factor of the noun hypernym hierarchy? I.e. for every noun synset that has hyponyms — or children in the hypernym hierarchy — how many do they have on average?*

**def** twentysix():  
 synsets = list(nltk.corpus.wordnet.all\_synsets(**'n'**))  
 count = 0  
 sum = 0  
 **for** syn **in** synsets:  
 hyps = list(syn.hyponyms())  
 **if** len(hyps) != 0:  
 count += 1  
 sum += len(hyps)  
 print(sum/count)

Branching Factor = 4.544