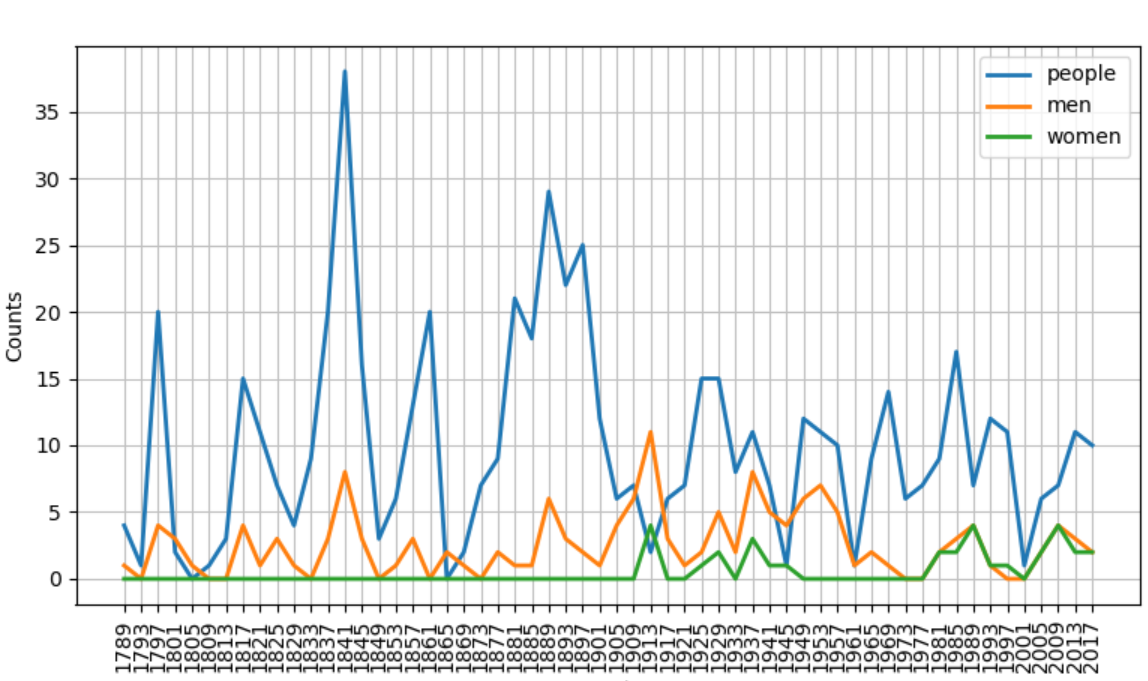
NLTK Required Exercises

Chapter 2:

**4.** As shown in the figure below, the use of “people” and “men” has approximately cycled over time. The plot for “people,” as well as the plot for “men” to a lesser extent, takes on an irregular but distinctly sinusoidal shape. Usage of both periodically undergoes rapid peaks and drops. In contrast, the plot for “women” mostly stays at the 0 count mark with only occasional peaks. Of all 3 words, “people” demonstrates the widest range in count frequency (>35) while “women” demonstrates the smallest (<5). The ranking of count frequency ranges also reflects the words’ overall usage frequencies with “people” as the most frequent and “women” as the least.



**5.** I chose to analyze the meronym-holonym relationships for 5 nouns: plant, house, computer, human, and water. The tables below list the numbers (i.e. distinct lemmas) of part, substance, and member meronyms and holonyms found in each word’s noun-related synsets.

Overall, the nouns with several common meanings (house and water) had many more synsets than those with just a single common usage. This finding is expected, given that WordNet’s synsets are generated through grouping words by definitions. Across all analyzed synsets, part meronyms were the most frequent by a wide margin. The relatively high frequency of part meronyms is likely a result of the specific nouns I chose to analyze. Several words in my sample (most notably, human and computer) describe nouns with many distinct parts or portions, logically linking them to large sets of part meronyms. Conversely, the generally low frequencies of all types of holonyms reflects the nouns’ (barring “water”) lack of usage as basic “building materials” in collections or classifications.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Plant** | | | | | | |
| Synset | Part Meronyms | Substance Meronyms | Member Meronyms | Part Holonyms | Substance Holonyms | Member Holonyms |
| plant.n.01 | 0 | 0 | 0 | 0 | 0 | 0 |
| plant.n.02 | 4 | 0 | 0 | 0 | 0 | 3 |
| plant.n.03 | 0 | 0 | 0 | 0 | 0 | 0 |
| plant.n.04 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **House** | | | | | | |
| Synset | Part Meronyms | Substance Meronyms | Member Meronyms | Part Holonyms | Substance Holonyms | Member Holonyms |
| house.n.01 | 6 | 0 | 0 | 0 | 0 | 0 |
| firm.n.01 | 0 | 0 | 0 | 0 | 0 | 0 |
| house.n.03 | 0 | 0 | 0 | 0 | 0 | 0 |
| house.n.04 | 1 | 0 | 0 | 0 | 0 | 0 |
| house.n.05 | 0 | 0 | 0 | 0 | 0 | 0 |
| house.n.06 | 0 | 0 | 0 | 0 | 0 | 0 |
| house.n.07 | 0 | 0 | 0 | 0 | 0 | 0 |
| sign\_of\_the\_zodiac.n.01 | 0 | 0 | 0 | 1 | 0 | 0 |
| house.n.09 | 0 | 0 | 0 | 0 | 0 | 0 |
| family.n.01 | 0 | 0 | 0 | 0 | 0 | 0 |
| theater.n.01 | 18 | 0 | 0 | 0 | 0 | 0 |
| house.n.12 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Computer** | | | | | | |
| Synset | Part Meronyms | Substance Meronyms | Member Meronyms | Part Holonyms | Substance Holonyms | Member Holonyms |
| computer.n.01 | 36 | 0 | 0 | 1 | 0 | 0 |
| calculator.n.01 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Human** | | | | | | |
| Synset | Part Meronyms | Substance Meronyms | Member Meronyms | Part Holonyms | Substance Holonyms | Member Holonyms |
| homo.n.02 | 30 | 0 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Water** | | | | | | |
| Synset | Part Meronyms | Substance Meronyms | Member Meronyms | Part Holonyms | Substance Holonyms | Member Holonyms |
| water.n.01 | 0 | 6 | 0 | 0 | 18 | 0 |
| body\_of\_water.n.01 | 0 | 2 | 0 | 3 | 0 | 0 |
| water.n.03 | 0 | 0 | 0 | 0 | 0 | 0 |
| water\_system.n.02 | 4 | 0 | 0 | 2 | 0 | 0 |
| urine.n.01 | 0 | 0 | 0 | 0 | 0 | 0 |
| water.n.06 | 0 | 0 | 0 | 0 | 0 | 0 |

**7.** I chose the Gutenberg texts to for my analysis of “however” usage. Based on both the linked “Fossilized Prejudices About ‘However’” and my own observations of concordances, I created 3 categories for types of usage: beginning, parenthetical, and other. Beginning refers to the presence of “however” at the start of an independent clause (i.e. after a period or semicolon); parenthetical refers to “second-position” usage (i.e. flanked by commas); other refers to any usage that did not fit the 2 previous categories. The table below displays frequency counts of each category for all texts that contain “however.”

Overall, the parenthetical, or “second-position,” usage appears to be the most common. This finding supports the historical information in “Fossilized Prejudices About ‘However.’” According to the post, William Strunk’s *Elements of Style* and thereby his preference for the parenthetical “however” were published in 1869. Strunk’s “prejudice” is attributed to the language of the literature of the early- to mid-1800s. As many of the analyzed texts were published within that period, their preference for the parenthetical “however” may well have influenced Strunk.

|  |  |  |  |
| --- | --- | --- | --- |
| Text | Type of Usage | | |
| Beginning | Parenthetical | Other |
| austen-emma.txt | 13 | 97 | 21 |
| austen-persuasion.txt | 6 | 67 | 16 |
| austen-sense.txt | 7 | 96 | 52 |
| bryant-stories.txt | 1 | 3 | 1 |
| burgess-busterbrown.txt | 1 | 0 | 0 |
| carroll-alice.txt | 10 | 2 | 8 |
| chesterton-ball.txt | 1 | 30 | 4 |
| chesterton-brown.txt | 1 | 15 | 6 |
| chesterton-thursday.txt | 1 | 24 | 2 |
| edgeworth-parents.txt | 11 | 54 | 7 |
| melville-moby\_dick.txt | 15 | 22 | 58 |
| milton-paradise.txt | 3 | 0 | 6 |
| whitman-leaves.txt | 0 | 0 | 12 |

**9.** I compared the Gutenberg corpus’ *Alice’s Adventures in Wonderland* by Lewis Carroll and *Paradise Lost* by John Milton. The two texts belong to quite different genres and historical periods. *Alice’s Adventures in Wonderland* is a Victorian-era children’s novel. Reflecting Caroll’s intent of light-hearted entertainment, its language is concrete and whimsical, with frequent references to character names and dialogue. *Paradise Lost*, however, is an epic poem written in the 17th century about the Biblical fall of Adam and Eve. As expected from a theological work, its language is far more abstract and complex. References to concepts, including “sin” and “temptation,” are common.

The usage of the words “will,” “even,” and “just” differs across the 2 texts. Whereas Carroll uses “will” only to indicate the future tense, Milton uses it both to modify verbs and as an abstract noun synonymous “intention” or “desire.” Similarly, Carroll’s usage of “even” is restricted to its adverbial meaning, but Milton’s usage includes its noun (i.e. “evening”) and adjective meanings. Continuing the trend, “just” in Carroll’s novel is used only as an adverb (indicating both time and extent) but also appears as an adjective (meaning “fair”) in Milton’s epic.

**12.** The CMU Pronouncing Dictionary contains 123455 distinct words, with 10282/123455 or 0.0833 of the entries having multiple pronunciations.

**17.**

def fifty\_common(text):

from nltk.corpus import stopwords

import re

stop\_w = set(stopwords.words("english"))

non\_stop = nltk.FreqDist(w.lower() for w in text if w.lower() not in stop\_w and re.search("[a-z]", w.lower()))

return non\_stop.most\_common(50)

**18.**

def fifty\_bigrams(text):

from nltk.corpus import stopwords

import re

stop\_w = set(stopwords.words("english"))

bigrams = nltk.bigrams(text)

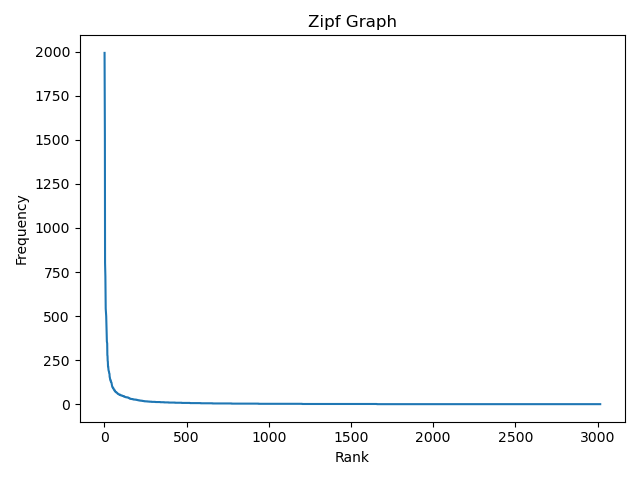
ret = nltk.FreqDist(w for w in bigrams if w[0].lower() not in stop\_w and re.search("[a-z]", w[0].lower()) and w[1].lower() not in stop\_w and re.search("[a-z]", w[1].lower()))

for w in ret:

print(w)

**23.**

**a)** The Zipf graph (generated by the method below) for Lewis Carroll’s *Alice’s Adventures in Wonderland* confirms Zipf’s law. As predicted by Zipf’s law, frequency is inversely proportional to word rank, producing a curve that can be modelled as a transformation of the reciprocal function (y = 1/x). The graph, in accordance with its algebraic definition, illustrates frequency approaching infinity as rank approaches 0 and frequency approaching 0 as rank approaches infinity (horizontal asymptote). This end behavior supports the inverse proportionality expressed in Zipf’s law.

def zipf\_graph(text):

import pylab

freq = nltk.FreqDist(text)

sort = []

ranks = []

for w in freq.most\_common(len(freq)):

sort.append(w[1])

ranks.append(len(sort))

pylab.plot(ranks, sort)

pylab.xlabel("Rank")

pylab.ylabel("Frequency")

pylab.title("Zipf Graph")

pylab.show()

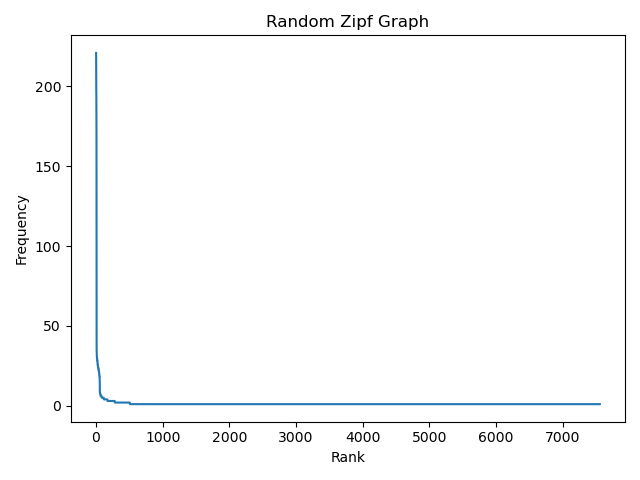
**b)**  The Zipf graph for randomly generated “text” is quite similar to that for Carroll’s novel. Both illustrate a curve and end behavior typical of the reciprocal function family, reflecting the inverse proportionality expressed in Zipf’s law. The major differences between the 2 graphs is in their axes scales. Whereas the graph for Carroll’s work is visible on axes of roughly equal scale, the graph for random text has a “rank-axis” scale roughly 20 times larger than its “frequency-axis” scale. The differences in scale are likely due to the non-random nature of English.

The large degree of similarity between the 2 graphs demonstrates the universality of Zipf’s law. Zipf’s law, as shown by its validity for randomly generated text, is not tied specifically to English or any particular language. Rather, the inverse proportionality between rank and frequency appears to be intrinsic to all texts.

def random\_zipf():

import random

import pylab

text = ""

for k in range(0, 100000):

text += random.choice("abcdefg ")

t\_text = nltk.word\_tokenize(text)

freq = nltk.FreqDist(t\_text)

sort = []

ranks = []

for w in freq.most\_common(len(freq)):

sort.append(w[1])

ranks.append(len(sort))

pylab.plot(ranks, sort)

pylab.xlabel("Rank")

pylab.ylabel("Frequency")

pylab.title("Random Zipf Graph")

pylab.show()

**27.**

|  |  |
| --- | --- |
| **Word Type** | **Average Polysemy** |
| Nouns | 1.2833560159282222 |
| Verbs | 2.1866273523545225 |
| Adjectives | 1.4104837960813446 |
| Adverbs | 1.2532916759651864 |

Chapter 3:

**20.** The code below will print out the newest question on the Stack Overflow questions page.

from urllib.request import urlopen

from bs4 import BeautifulSoup

url = "https://stackoverflow.com/questions"

webpage = urlopen(url).read().decode("utf8")

raw = BeautifulSoup(webpage, "html.parser")

questions = raw.find(id="questions")

top = questions.find(class\_="summary")

text = top.text.split("\n")

print(text[1])

**22.** I analyzed the URL using the following “suggested” regexes from the chapter text: /[A-Z][a-z]\*/, /\w+|\S\w\*/, /\w+(?:[-']\w+)\*|'|[-.(]+|\S\w\*/, and /(?x)(?:[A-Z]\.)+| \w+(?:-\w+)\*| \$?\d+(?:\.\d+)?%?| \.\.\.| [][.,;"'?():-\_`]/. All 4--even the first, which matched only English letters--captured unwanted Javascript and web page data in addition to actual text. However, the 3rd regex, presented in the text as a basic refinement of a naive tokenization regex, performed best in terms of maximizing textual data and minimizing Javascript captures. (Interestingly, the 4th regex, described as NLTK’s tokenization regex, seems to exclude the majority of text in its captures.)

Although not “hard” rules, some general differences between Javascript and English could be used to refine the regexes. Javascript, unlike English, frequently contains uppercase letters in the middle of “words.” At the expense of excluding certain names, a possible regex refinement would be to exclude words with uppercase letters after lowercase letters; the expression /\b(?![A-Z]\*[a-z][A-Z])\w\*/ will do so. A similar observation about the relative frequencies of digit-letter mixtures across the 2 languages rationalizes using expressions like \b[a-zA-Z]+$|^[0-9]+$\b to exclude such mixtures when searching for English text. Likewise, this refinement would exclude the few cases of digit-letter mixtures present in scientific and technical writing.

Chapter 6:

**4.** The table below lists the 30 features the movie review classifier found most informative. Notably, the majority of the features are associated with negativity rather than positivity, which might be explained by the greater applicability of insults as opposed to praise. As predicted from this observation, many features (e.g. “shoddy,” “groan,” and “atrocious”) are words with strongly negative meanings or connotations. Certain features, including “justin,” “turkey,” “schumacher,” “suvari,” and “mena,” were explainable only after some brief research. Such initially-surprising features were typically titles of poorly received movies (e.g. “justin”); movie-critic slang (e.g. “turkey”); or disliked actors, actresses, or directors (e.g. “schumacher,” “mena,” and “suvari”)

|  |  |
| --- | --- |
| **Feature** | **Negative : Positive** |
| justin | 8.9 : 1.0 |
| turkey | 8.3 : 1.0 |
| groan | 7.6 : 1.0 |
| shoddy | 6.9 : 1.0 |
| schumacher | 6.9 : 1.0 |
| suvari | 6.3 : 1.0 |
| mena | 6.3 : 1.0 |
| poorly | 6.2 : 1.0 |
| stretched | 6.1 : 1.0 |
| atrocious | 6.1 : 1.0 |
| miscast | 6.1 : 1.0 |
| wasted | 6.0 : 1.0 |
| ugh | 5.7 : 1.0 |
| singers | 1.0 : 5.7 |
| bronson | 5.6 : 1.0 |
| canyon | 5.6 : 1.0 |
| surveillance | 5.6 : 1.0 |
| waste | 5.4 : 1.0 |
| awful | 5.2 : 1.0 |
| unravel | 1.0 : 5.1 |
| underwood | 4.9 : 1.0 |
| oops | 4.9 : 1.0 |
| ridiculous | 4.8 : 1.0 |
| kudos | 1.0 : 4.8 |
| bland | 4.7 : 1.0 |
| uninspired | 4.6 : 1.0 |
| explores | 1.0 : 4.6 |
| martian | 4.6 : 1.0 |
| welles | 4.6 : 1.0 |
| painfully | 4.5 : 1.0 |