Untitled

September 17, 2018

In [90]:
```
#Question 4
import nltk
from nltk.corpus import state_union
cfd = nltk.ConditionalFreqDist(
      (fileid,word)
      for fileid in state_union.fileids()
      for word in state_union.words(fileids=fileid))
fileids=state_union.fileids()
word=['men','women','people']
cfd.tabulate(conditions=fileids,samples=word)

#The trend is that over the years, the writers still tend to mention "people" more th
#and "women". Also, after 1978, writers mentioned women more than men
```

	men	women	people
1945-Truman.txt	2	2	10
1946-Truman.txt	12	7	49
1947-Truman.txt	7	2	12
1948-Truman.txt	4	1	22
1949-Truman.txt	2	1	15
1950-Truman.txt	6	2	15
1951-Truman.txt	8	2	9
1953-Eisenhower.txt	3	0	17
1954-Eisenhower.txt	2	0	15
1955-Eisenhower.txt	4	0	26
1956-Eisenhower.txt	2	2	30
1957-Eisenhower.txt	5	2	11
1958-Eisenhower.txt	2	1	19
1959-Eisenhower.txt	4	1	11
1960-Eisenhower.txt	2	0	10
1961-Kennedy.txt	6	0	10
1962-Kennedy.txt	6	2	10
1963-Johnson.txt	0	0	3
1963-Kennedy.txt	8	5	12
1964-Johnson.txt	3	1	3
1965-Johnson-1.txt	7	0	16
1965-Johnson-2.txt	11	3	14
1966-Johnson.txt	12	1	35

```
  1967-Johnson.txt     11      1     25
  1968-Johnson.txt      4      0     17
  1969-Johnson.txt      5      2      6
    1970-Nixon.txt      2      0     23
    1971-Nixon.txt      1      0     31
    1972-Nixon.txt      1      0      7
    1973-Nixon.txt      0      0      9
    1974-Nixon.txt      0      0     19
     1975-Ford.txt      0      0     13
     1976-Ford.txt      3      1     18
     1977-Ford.txt      2      1     17
   1978-Carter.txt      0      1     26
   1979-Carter.txt      0      1     15
   1980-Carter.txt      1      2     11
   1981-Reagan.txt      1      1     11
   1982-Reagan.txt      1      1     17
   1983-Reagan.txt      3      7     19
   1984-Reagan.txt      3      5     23
   1985-Reagan.txt      1      1     12
   1986-Reagan.txt      2      2     14
   1987-Reagan.txt      1      0     24
   1988-Reagan.txt      1      0     16
     1989-Bush.txt      2      3     13
     1990-Bush.txt      3      2      9
   1991-Bush-1.txt      2      2     13
   1991-Bush-2.txt      7      7     13
     1992-Bush.txt      4      4     26
  1993-Clinton.txt      1      2     45
  1994-Clinton.txt      1      1     63
  1995-Clinton.txt      1      3     73
  1996-Clinton.txt      2      3     40
  1997-Clinton.txt      1      2     30
  1998-Clinton.txt      2      2     22
  1999-Clinton.txt      2      3     22
  2000-Clinton.txt      5      6     41
 2001-GWBush-1.txt      3      3     14
 2001-GWBush-2.txt      1      2     12
   2002-GWBush.txt      3      5     14
   2003-GWBush.txt      6      4     33
   2004-GWBush.txt      6      8     21
   2005-GWBush.txt      8     11     18
   2006-GWBush.txt      7      7     22
```

```
In [28]: #Question 8
         from nltk.corpus import names
         #names.fileids()
         cfd = nltk.ConditionalFreqDist(
```

```
            [(fileid, word[0])
             for fileid in names.fileids()
             for word in names.words(fileids=fileid)])
        cfd.tabulate()

        #Therefore we can conclude that words beginning with H, Q, U, W, X are more frequent
```

```
              A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U
female.txt 443 246 469 308 251 144 213 124  83 293 276 332 484 158  66 121   9 247 309 198  14
  male.txt 213 173 166 146 119  87 156 163  45 144  70 113 200  77  52 101  15 200 238 188  22
```

In [29]:
```
        #Question 16
        import nltk
        from nltk.corpus import brown

        def lds(corpus):
            for genre in corpus.categories():
                lds = len(set(corpus.words(categories=genre)))/len(corpus.words(categories=ge
                print (genre + ':', lds)

        lds(brown)

        #Therefore we find out the learned type has the lowest lexical diversity score
```

```
adventure: 0.1279743878169075
belles_lettres: 0.10642071451679992
editorial: 0.16054152327770924
fiction: 0.1358194136199042
government: 0.11667641228232811
hobbies: 0.14493897625842492
humor: 0.23125144042406084
learned: 0.09268890745953554
lore: 0.13148804612915801
mystery: 0.12212912592488936
news: 0.14314696580941583
religion: 0.1617553745018909
reviews: 0.21192020440251572
romance: 0.12070492131044529
science_fiction: 0.22342778161713892
```

In [30]:
```
        #Question 17
        import nltk
        from nltk.corpus import stopwords

        def content_fraction(text):
            stopwords=nltk.corpus.stopwords.words('english')
            content=[w for w in text if w.lower() not in stopwords and any(c.isalpha() for c
```

```
    fdist=nltk.FreqDist(content)
    #return fdist.most_common(50)
    return [w for w, num in fdist.most_common(50)]

#Test for reuters
content_fraction(nltk.corpus.reuters.words())
```

Out[30]:
```
['said',
 'mln',
 'vs',
 'dlrs',
 'pct',
 'lt',
 'cts',
 'U',
 'year',
 'billion',
 'would',
 'loss',
 'company',
 'Net',
 'Shr',
 'last',
 'share',
 'profit',
 'Inc',
 'one',
 'shares',
 'oil',
 'market',
 'also',
 'two',
 'Corp',
 'tonnes',
 'stock',
 'Revs',
 'trade',
 'QTR',
 'net',
 'per',
 'April',
 'prices',
 'March',
 'quarter',
 'Bank',
 'new',
 'price',
 'February',
```

```
 'rate',
 'January',
 'Japan',
 'government',
 'NET',
 'week',
 'offer',
 'three',
 'bank']
```

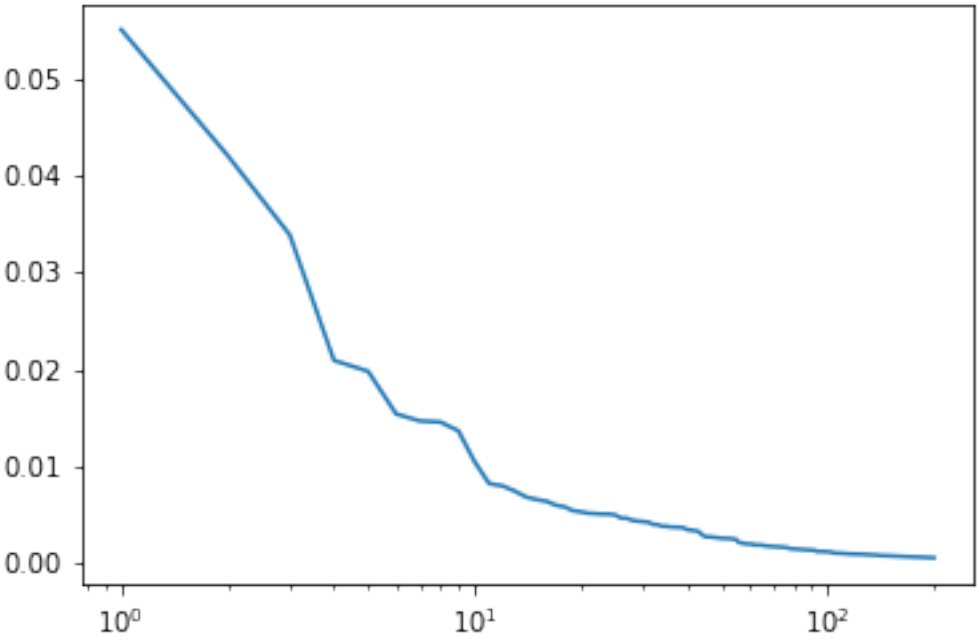
In [31]:
```
#Question 18
import nltk
from nltk.corpus import stopwords

def bigrams(text):
    stopwords=nltk.corpus.stopwords.words('english')
    content=[w for w in nltk.bigrams(text)
             if w[0] not in stopwords
             and w[1] not in stopwords
             and any(c.isalpha() for c in w[0])
             and any(c.isalpha() for c in w[1])]
    fdist=nltk.FreqDist(content)
    #return fdist.most_common(50)
    return [w for w, num in fdist.most_common(50)]

#Test for reuters
bigrams(nltk.corpus.reuters.words())
```

Out[31]:
```
[('mln', 'dlrs'),
 ('mln', 'vs'),
 ('cts', 'vs'),
 ('cts', 'Net'),
 ('vs', 'loss'),
 ('billion', 'dlrs'),
 ('last', 'year'),
 ('company', 'said'),
 ('The', 'company'),
 ('dlrs', 'vs'),
 ('NET', 'Shr'),
 ('Avg', 'shrs'),
 ('vs', 'profit'),
 ('per', 'share'),
 ('It', 'said'),
 ('He', 'said'),
 ('QTR', 'NET'),
 ('mln', 'stg'),
 ('mln', 'tonnes'),
 ('Inc', 'said'),
```

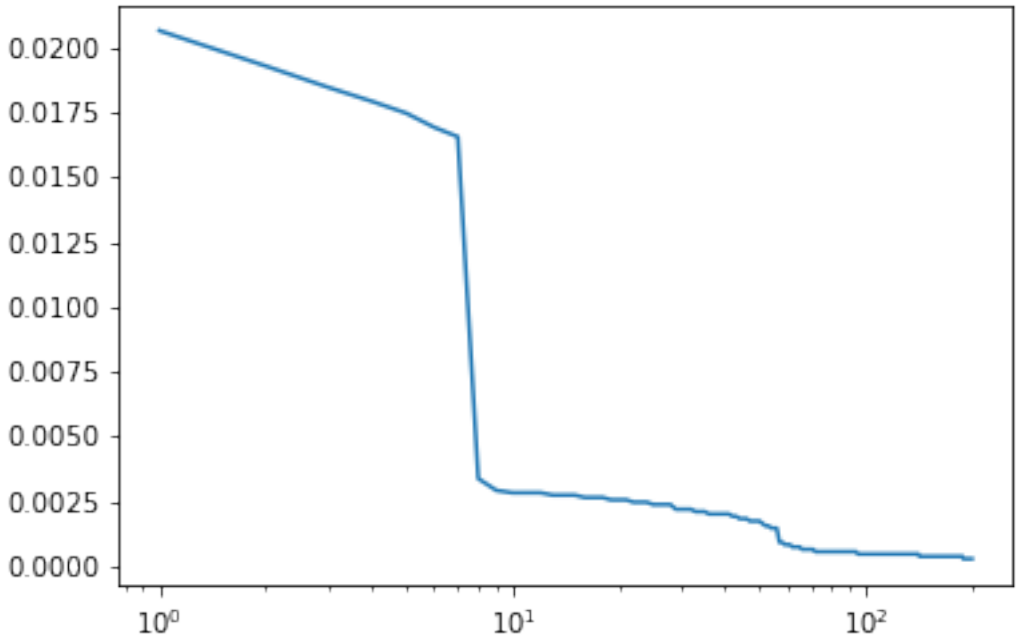
```
 ('4TH', 'QTR'),
 ('Net', 'loss'),
 ('Shr', 'loss'),
 ('mln', 'Avg'),
 ('United', 'States'),
 ('sources', 'said'),
 ('mln', 'NOTE'),
 ('also', 'said'),
 ('Corp', 'said'),
 ('dlrs', 'per'),
 ('dlrs', 'Net'),
 ('billion', 'vs'),
 ('mths', 'Shr'),
 ('3RD', 'QTR'),
 ('1ST', 'QTR'),
 ('Oper', 'shr'),
 ('Oper', 'net'),
 ('spokesman', 'said'),
 ('first', 'quarter'),
 ('mln', 'Revs'),
 ('common', 'stock'),
 ('Net', 'profit'),
 ('New', 'York'),
 ('last', 'month'),
 ('told', 'Reuters'),
 ('Year', 'Shr'),
 ('cts', 'prior'),
 ('Nine', 'mths'),
 ('central', 'bank'),
 ('Qtly', 'div')]
```

In [85]:
```
#Question 23
#a.
import pylab

def zipf(text):
    fd = nltk.FreqDist(text)
    rank = 1
    freqs = []
    ranks = []
    for sample, count in fd.most_common(200):
        if any(c.isalpha() for c in sample):
            freqs.append(fd.freq(sample))
            ranks.append(rank)
            rank = rank + 1
    pylab.plot(ranks, freqs)
    pylab.xscale("log")
```

```
zipf(nltk.corpus.reuters.words())

#Therefore, from the plot we can see it confirms the Zipf's law, and the frequency of
#rank is infinitely close to 0
```



In [86]:
```
#Question 23
#b. 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 st
#Then tokenize this string, and generate the Zipf plot as before, and compare the two
#What do you make of Zipf's Law in the light of this?

import random
import nltk
random_text = ''
count = 0
while count < 100001:
    random_text = random_text + random.choice('abcdefg ')
    count = count + 1
text_split = random_text.split()
zipf(random_text.split())

#From the plot we see that with random text,
#as rank goes up, the frequency of the word becomes less however there is a steep dec
#the 10th word. Therefore within random text, the Zipf may not be true
```



In [81]:
```
#Question 28
import nltk
from nltk.corpus import wordnet as wn

def polysemy(category):
    allsyns=[]
    count=0
    sum=0
    for synset in wn.all_synsets(category):
        if count>10000:
            break;
        for lemma in synset.lemmas():
            lemma_name=lemma.name()
            if lemma_name not in allsyns:
                allsyns.append(lemma_name)
                count=count+1
                sum=sum + len(wn.synsets(lemma_name,category))
    return sum/count

def print_polysemy():
    print("Noun: ", polysemy('n'))
    print("Verb: ", polysemy('v'))
    print("Adjective: ", polysemy('a'))
    print("Adverb: ", polysemy('r'))

print_polysemy()
```

```
Noun:  1.9563043695630438
Verb:  2.3236676332366764
Adjective:  1.6914308569143086
Adverb:  1.2532916759651864
```