LING 408/508: Computational Techniques for Linguists

Lecture 30

NLTK Book

```
>>> for fileid in gutenberg.fileids():
...     num_chars = len(gutenberg.raw(fileid)) ❶
...     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), round(num_words/num_sents), round(num_words/num_vocab), fileid)
...
5 25 26 austen-emma.txt
5 26 17 austen-persuasion.txt
5 28 22 austen-sense.txt
4 34 79 bible-kjv.txt
5 19 5 blake-poems.txt
4 19 14 bryant-stories.txt
4 18 12 burgess-busterbrown.txt
4 20 13 carroll-alice.txt
5 20 12 chesterton-ball.txt
5 23 11 chesterton-brown.txt
5 18 11 chesterton-thursday.txt
4 21 25 edgeworth-parents.txt
5 26 15 melville-moby_dick.txt
5 52 11 milton-paradise.txt
4 12 9 shakespeare-caesar.txt
4 12 8 shakespeare-hamlet.txt
4 12 7 shakespeare-macbeth.txt
5 36 12 whitman-leaves.txt
```

filename is the fileid

Statistics:

1. average # letters per word
2. average # words per sentence
3. average # times each word is used

NLTK Book: Chapter 2

Other types of corpora:

- `from nltk.corpus import webtext` — Web text corpus
- `from nltk.corpus import nps_chat` — NPS chat corpus
- `from nltk.corpus import brown` — Brown corpus

ID	File	Genre	Description
A16	ca16	news	Chicago Tribune: *Society Reportage*
B02	cb02	editorial	Christian Science Monitor: *Editorials*
C17	cc17	reviews	Time Magazine: *Reviews*
D12	cd12	religion	Underwood: *Probing the Ethics of Realtors*
E36	ce36	hobbies	Norling: *Renting a Car in Europe*
F25	cf25	lore	Boroff: *Jewish Teenage Culture*
G22	cg22	belles_lettres	Reiner: *Coping with Runaway Technology*
H15	ch15	government	US Office of Civil and Defence Mobilization: *The Family Fallout Shelter*
J17	cj19	learned	Mosteller: *Probability with Statistical Applications*
K04	ck04	fiction	W.E.B. Du Bois: *Worlds of Color*
L13	cl13	mystery	Hitchens: *Footsteps in the Night*
M01	cm01	science_fiction	Heinlein: *Stranger in a Strange Land*
N14	cn15	adventure	Field: *Rattlesnake Ridge*
P12	cp12	romance	Callaghan: *A Passion in Rome*
R06	cr06	humor	Thurber: *The Future, If Any, of Comedy*

"*A balanced corpus*"

NLTK Book: Chapter 2

```
modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> from nltk.corpus import brown
>>> len(brown.fileids())          #number of files = 500
500
>>> brown.categories()            #number of categories = 15
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government',
'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion',
'reviews', 'romance', 'science_fiction']
>>> news = brown.words(categories='news')
>>> import nltk
>>> fd = nltk.FreqDist(w.lower() for w in news)
>>> for m in modals:
...     print('{}: {}'.format(m,fd[m]), end=' ')
...
can: 94 could: 87 may: 93 might: 38 must: 53 shall: 5 should: 61 will: 389
would: 246 >>>
```

NLTK Book: Chapter 2

```
>>> modals = ['can', 'could', 'may', 'might', 'must', 'shall', 'should', 'will', 'would']
>>> cfd = nltk.ConditionalFreqDist((cat,word) for cat in brown.categories() for word in brown.words(categories=cat))
>>> cfd.tabulate(conditions=brown.categories(), samples=modals)
                  can  could    may  might   must  shall should   will  would
      adventure    46    151      5     58     27      7     15     50    191
 belles_lettres   246    213    207    113    170     34    102    236    392
      editorial   121     56     74     39     53     19     88    233    180
        fiction    37    166      8     44     55      3     35     52    287
     government   117     38    153     13    102     98    112    244    120
        hobbies   268     58    131     22     83      5     73    264     78
          humor    16     30      8      8      9      2      7     13     56
        learned   365    159    324    128    202     40    171    340    319
           lore   170    141    165     49     96     12     76    175    186
        mystery    42    141     13     57     30      1     29     20    186
           news    93     86     66     38     50      5     59    389    244
       religion    82     59     78     12     54     21     45     71     68
        reviews    45     40     45     26     19      1     18     58     47
        romance    74    193     11     51     45      3     32     43    244
science_fiction    16     49      4     12      8      3      3     16     79
```

NLTK book says, "Observe that the most frequent modal in the news genre is *will*, while the most frequent modal in the romance genre is *could* (actually *would*). Would you have predicted this?'

NLTK Book: Chapter 2

Other types of corpora:

- `from nltk.corpus import reuters` Reuters news corpus

```
>>> reuters.categories()
['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconut',
'coconut-oil', 'coffee', 'copper', 'copra-cake', 'corn', 'cotton', 'cotton-
oil', 'cpi', 'cpu', 'crude', 'dfl', 'dlr', 'dmk', 'earn', 'fuel', 'gas',
'gnp', 'gold', 'grain', 'groundnut', 'groundnut-oil', 'heat', 'hog',
'housing', 'income', 'instal-debt', 'interest', 'ipi', 'iron-steel', 'jet',
'jobs', 'l-cattle', 'lead', 'lei', 'lin-oil', 'livestock', 'lumber', 'meal-
feed', 'money-fx', 'money-supply', 'naphtha', 'nat-gas', 'nickel', 'nkr',
'nzdlr', 'oat', 'oilseed', 'orange', 'palladium', 'palm-oil', 'palmkernel',
'pet-chem', 'platinum', 'potato', 'propane', 'rand', 'rape-oil', 'rapeseed',
'reserves', 'retail', 'rice', 'rubber', 'rye', 'ship', 'silver', 'sorghum',
'soy-meal', 'soy-oil', 'soybean', 'strategic-metal', 'sugar', 'sun-meal',
'sun-oil', 'sunseed', 'tea', 'tin', 'trade', 'veg-oil', 'wheat', 'wpi', 'yen',
'zinc']
>>> len(reuters.categories())
90
```

NLTK Book: Chapter 2

- File IDs are divided into test and training datasets (for machine learning purposes)

```
>>> reuters.fileids('strategic-metal')
['test/15420', 'test/15838', 'test/15872', 'test/17480',
'test/17486', 'test/17783', 'test/17805', 'test/18348',
'test/18466', 'test/18872', 'test/18944', 'training/10151',
'training/11999', 'training/12007', 'training/13052',
'training/13251', 'training/14719', 'training/2186',
'training/2936', 'training/2942', 'training/3010', 'training/309',
'training/346', 'training/3460', 'training/3497', 'training/5693',
'training/7775']
>>> reuters.categories('test/15420')
['palladium', 'platinum', 'strategic-metal']
```

NLTK Book: Chapter 2

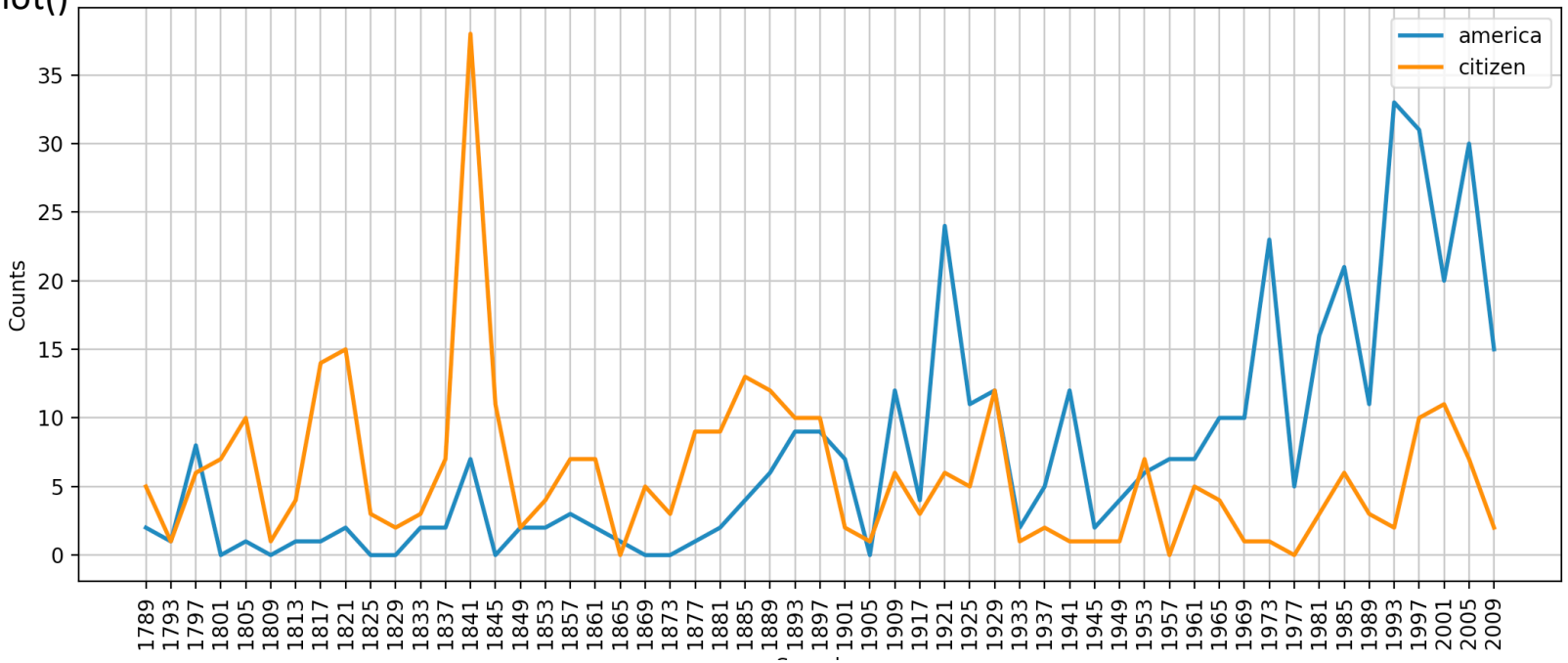
Other types of corpora:

- `from nltk.corpus import inaugural` Inaugural address corpus

```
>>> inaugural.fileids()
['1789-Washington.txt', '1793-Washington.txt', '1797-Adams.txt', '1801-Jefferson.txt',
'1805-Jefferson.txt', '1809-Madison.txt', '1813-Madison.txt', '1817-Monroe.txt', '1821-
Monroe.txt', '1825-Adams.txt', '1829-Jackson.txt', '1833-Jackson.txt', '1837-VanBuren.txt',
'1841-Harrison.txt', '1845-Polk.txt', '1849-Taylor.txt', '1853-Pierce.txt', '1857-
Buchanan.txt', '1861-Lincoln.txt', '1865-Lincoln.txt', '1869-Grant.txt', '1873-Grant.txt',
'1877-Hayes.txt', '1881-Garfield.txt', '1885-Cleveland.txt', '1889-Harrison.txt', '1893-
Cleveland.txt', '1897-McKinley.txt', '1901-McKinley.txt', '1905-Roosevelt.txt', '1909-
Taft.txt', '1913-Wilson.txt', '1917-Wilson.txt', '1921-Harding.txt', '1925-Coolidge.txt',
'1929-Hoover.txt', '1933-Roosevelt.txt', '1937-Roosevelt.txt', '1941-Roosevelt.txt', '1945-
Roosevelt.txt', '1949-Truman.txt', '1953-Eisenhower.txt', '1957-Eisenhower.txt', '1961-
Kennedy.txt', '1965-Johnson.txt', '1969-Nixon.txt', '1973-Nixon.txt', '1977-Carter.txt',
'1981-Reagan.txt', '1985-Reagan.txt', '1989-Bush.txt', '1993-Clinton.txt', '1997-
Clinton.txt', '2001-Bush.txt', '2005-Bush.txt', '2009-Obama.txt']
>>> [fileid[:4] for fileid in inaugural.fileids()]
['1789', '1793', '1797', '1801', '1805', '1809', '1813', '1817', '1821', '1825', '1829',
'1833', '1837', '1841', '1845', '1849', '1853', '1857', '1861', '1865', '1869', '1873',
'1877', '1881', '1885', '1889', '1893', '1897', '1901', '1905', '1909', '1913', '1917',
'1921', '1925', '1929', '1933', '1937', '1941', '1945', '1949', '1953', '1957', '1961',
'1965', '1969', '1973', '1977', '1981', '1985', '1989', '1993', '1997', '2001', '2005',
'2009']
```

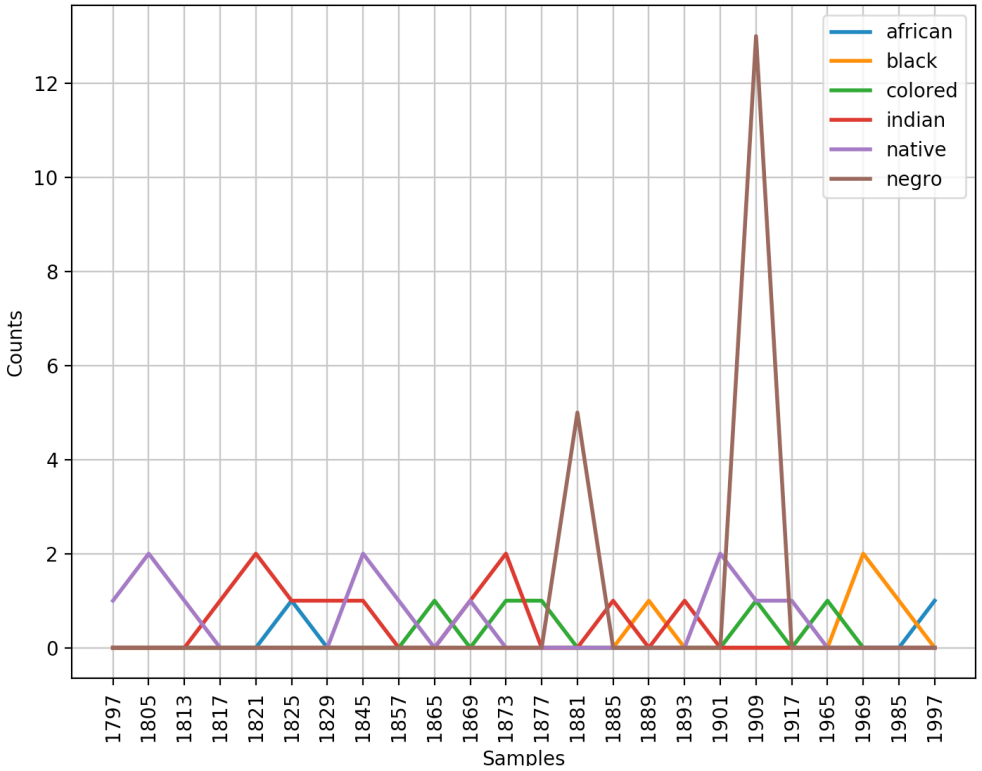
NLTK Book: Chapter 2


```
>>> cfd = nltk.ConditionalFreqDist((word,fileid[:4]) for fileid in inaugural.fileids() for w in inaugural.words(fileid)
for word in ['america','citizen'] if w.lower().startswith(word))
>>> cfd.plot()
```



NLTK Book: Chapter 2

```
cfd =
nltk.ConditionalFreqDist((w
ord,fileid[:4]) for fileid
in inaugural.fileids() for
w in
inaugural.words(fileid) for
word in
['black','negro','african',
'mexican','hispanic','color
ed','native','indian'] if
w.lower().startswith(word))
>>> cfd.plot()
```



some of these terms are obviously outdated (and offensive today)

NLTK Book: Chapter 2

Corpus	Compiler	Contents
Brown Corpus	Francis, Kucera	15 genres, 1.15M words, tagged, categorized
CESS Treebanks	CLiC-UB	1M words, tagged and parsed (Catalan, Spanish)
Chat-80 Data Files	Pereira & Warren	World Geographic Database
CMU Pronouncing Dictionary	CMU	127k entries
CoNLL 2000 Chunking Data	CoNLL	270k words, tagged and chunked
CoNLL 2002 Named Entity	CoNLL	700k words, pos- and named-entity-tagged (Dutch, Spanish)
CoNLL 2007 Dependency Treebanks (sel)	CoNLL	150k words, dependency parsed (Basque, Catalan)
Dependency Treebank	Narad	Dependency parsed version of Penn Treebank sample
FrameNet	Fillmore, Baker et al	10k word senses, 170k manually annotated sentences
Floresta Treebank	Diana Santos et al	9k sentences, tagged and parsed (Portuguese)
Gazetteer Lists	Various	Lists of cities and countries
Genesis Corpus	Misc web sources	6 texts, 200k words, 6 languages
Gutenberg (selections)	Hart, Newby, et al	18 texts, 2M words
Inaugural Address Corpus	CSpan	US Presidential Inaugural Addresses (1789-present)
Indian POS-Tagged Corpus	Kumaran et al	60k words, tagged (Bangla, Hindi, Marathi, Telugu)
MacMorpho Corpus	NILC, USP, Brazil	1M words, tagged (Brazilian Portuguese)
Movie Reviews	Pang, Lee	2k movie reviews with sentiment polarity classification
Names Corpus	Kantrowitz, Ross	8k male and female names
NIST 1999 Info Extr (selections)	Garofolo	63k words, newswire and named-entity SGML markup
Nombank	Meyers	115k propositions, 1400 noun frames
NPS Chat Corpus	Forsyth, Martell	10k IM chat posts, POS-tagged and dialogue-act tagged
Open Multilingual WordNet	Bond et al	15 languages, aligned to English WordNet
PP Attachment Corpus	Ratnaparkhi	28k prepositional phrases, tagged as noun or verb modifiers
Proposition Bank	Palmer	113k propositions, 3300 verb frames
Question Classification	Li, Roth	6k questions, categorized
Reuters Corpus	Reuters	1.3M words, 10k news documents, categorized
Roget's Thesaurus	Project Gutenberg	200k words, formatted text
RTE Textual Entailment	Dagan et al	8k sentence pairs, categorized
SEMCOR	Rus, Mihalcea	880k words, part-of-speech and sense tagged
Senseval 2 Corpus	Pedersen	600k words, part-of-speech and sense tagged
SentiWordNet	Esuli, Sebastiani	sentiment scores for 145k WordNet synonym sets
Shakespeare texts (selections)	Bosak	8 books in XML format
State of the Union Corpus	CSPAN	485k words, formatted text
Stopwords Corpus	Porter et al	2,400 stopwords for 11 languages
Swadesh Corpus	Wiktionary	comparative wordlists in 24 languages
Switchboard Corpus (selections)	LDC	36 phonecalls, transcribed, parsed
Univ Decl of Human Rights	United Nations	480k words, 300+ languages
Penn Treebank (selections)	LDC	40k words, tagged and parsed
TIMIT Corpus (selections)	NIST/LDC	audio files and transcripts for 16 speakers
VerbNet 2.1	Palmer et al	5k verbs, hierarchically organized, linked to WordNet
Wordlist Corpus	OpenOffice.org et al	960k words and 20k affixes for 8 languages
WordNet 3.0 (English)	Miller, Fellbaum	145k synonym sets

Not an exhaustive list…

NLTK Book: Chapter 2

Methods:

Example	Description
`fileids()`	the files of the corpus
`fileids([categories])`	the files of the corpus corresponding to these categories
`categories()`	the categories of the corpus
`categories([fileids])`	the categories of the corpus corresponding to these files
`raw()`	the raw content of the corpus
`raw(fileids=[f1,f2,f3])`	the raw content of the specified files
`raw(categories=[c1,c2])`	the raw content of the specified categories
`words()`	the words of the whole corpus
`words(fileids=[f1,f2,f3])`	the words of the specified fileids
`words(categories=[c1,c2])`	the words of the specified categories
`sents()`	the sentences of the whole corpus
`sents(fileids=[f1,f2,f3])`	the sentences of the specified fileids
`sents(categories=[c1,c2])`	the sentences of the specified categories
`abspath(fileid)`	the location of the given file on disk
`encoding(fileid)`	the encoding of the file (if known)
`open(fileid)`	open a stream for reading the given corpus file
`root`	if the path to the root of locally installed corpus
`readme()`	the contents of the README file of the corpus

NLTK Book: Chapter 2

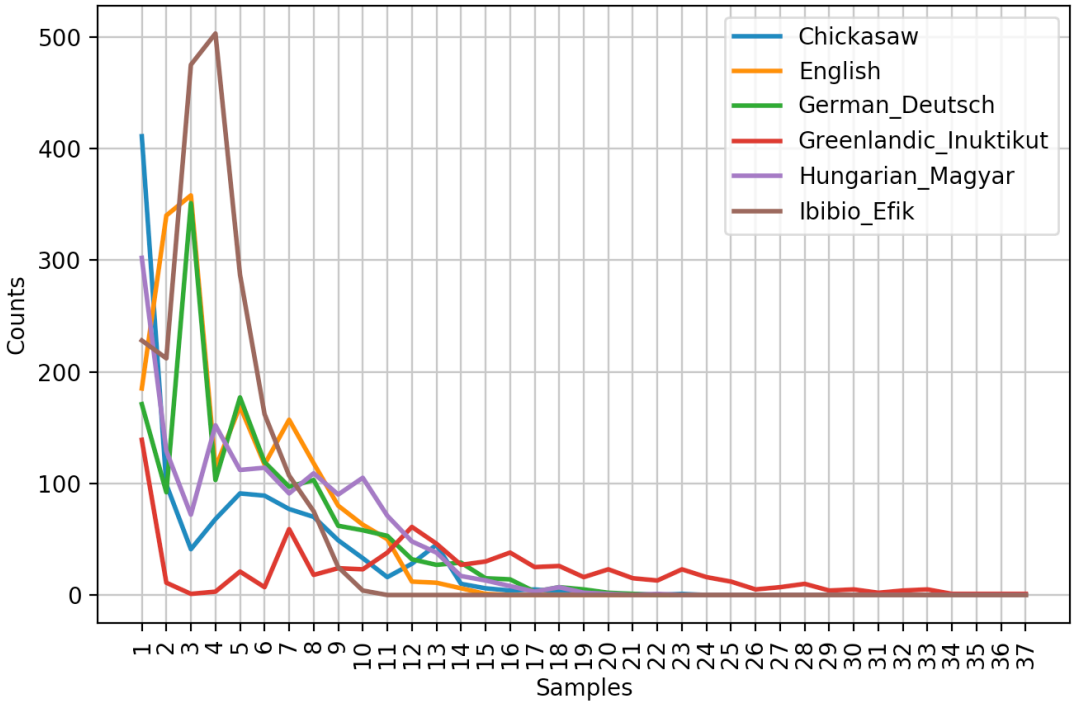
UDHR = Universal Declaration of Human Rights

```
>>> nltk.corpus.indian.words('hindi.pos')
['पूर्ण', 'प्रतिबंध', 'हटाओ', ':', 'इराक', 'संयुक्त', ...]
>>> nltk.corpus.udhr.fileids()
['Abkhaz-Cyrillic+Abkh', 'Abkhaz-UTF8', 'Achehnese-Latin1', 'Achuar-Shiwiar-Latin1', 'Adja-UTF8', 'Afaan_Oromo_Oromiffa-Latin1', 'Afrikaans-Latin1', 'Aguaruna-
Latin1', 'Akuapem_Twi-UTF8', 'Albanian_Shqip-Latin1', 'Amahuaca', 'Amahuaca-Latin1', 'Amarakaeri-Latin1', 'Amuesha-Yanesha-UTF8', 'Arabela-Latin1',
'Arabic_Alarabia-Arabic', 'Asante-UTF8', 'Ashaninca-Latin1', 'Asheninca-Latin1', 'Asturian_Bable-Latin1', 'Aymara-Latin1', 'Balinese-Latin1', 'Bambara-UTF8',
'Baoule-UTF8', 'Basque_Euskara-Latin1', 'Batonu_Bariba-UTF8', 'Belorus_Belaruski-Cyrillic', 'Belorus_Belaruski-UTF8', 'Bemba-Latin1', 'Bengali-UTF8', 'Beti-
UTF8', 'Bichelamar-Latin1', 'Bikol_Bicolano-Latin1', 'Bora-Latin1', 'Bosnian_Bosanski-Cyrillic', 'Bosnian_Bosanski-Latin2', 'Bosnian_Bosanski-UTF8', 'Breton-
Latin1', 'Bugisnese-Latin1', 'Bulgarian_Balgarski-Cyrillic', 'Bulgarian_Balgarski-UTF8', 'Cakchiquel-Latin1', 'Campa_Pajonalino-Latin1', 'Candoshi-Shapra-
Latin1', 'Caquinte-Latin1', 'Cashibo-Cacataibo-Latin1', 'Cashinahua-Latin1', 'Catalan-Latin1', 'Catalan_Catala-Latin1', 'Cebuano-Latin1', 'Chamorro-Latin1',
'Chayahuita-Latin1', 'Chechewa_Nyanja-Latin1', 'Chickasaw-Latin1', 'Chinanteco-Ajitlan-Latin1', 'Chinanteco-UTF8', 'Chinese_Mandarin-GB2312', 'Chuuk_Trukese-
Latin1', 'Cokwe-Latin1', 'Co …

… tin1', 'WesternSotho_Tswana-Setswana-Latin1', 'Wolof-Latin1', 'Xhosa-Latin1', 'Yagua-Latin1', 'Yao-Latin1', 'Yapese-Latin1', 'Yoruba-UTF8', 'Zapoteco-
Latin1', 'Zapoteco-SanLucasQuiavini-Latin1', 'Zhuang-Latin1', 'Zulu-Latin1']
>>> len(nltk.corpus.udhr.fileids())
310
>>> nltk.corpus.udhr.words('Chinese_Mandarin-GB2312')
['世界人权宣言', '联合国大会一九四八年十二月十日第217A', '(', 'III', ')', ...]
>>> nltk.corpus.udhr.words('English-Latin1')[:20]
['Universal', 'Declaration', 'of', 'Human', 'Rights', 'Preamble', 'Whereas', 'recognition', 'of',
'the', 'inherent', 'dignity', 'and', 'of', 'the', 'equal', 'and', 'inalienable', 'rights', 'of']
```

NLTK Book: Chapter 2

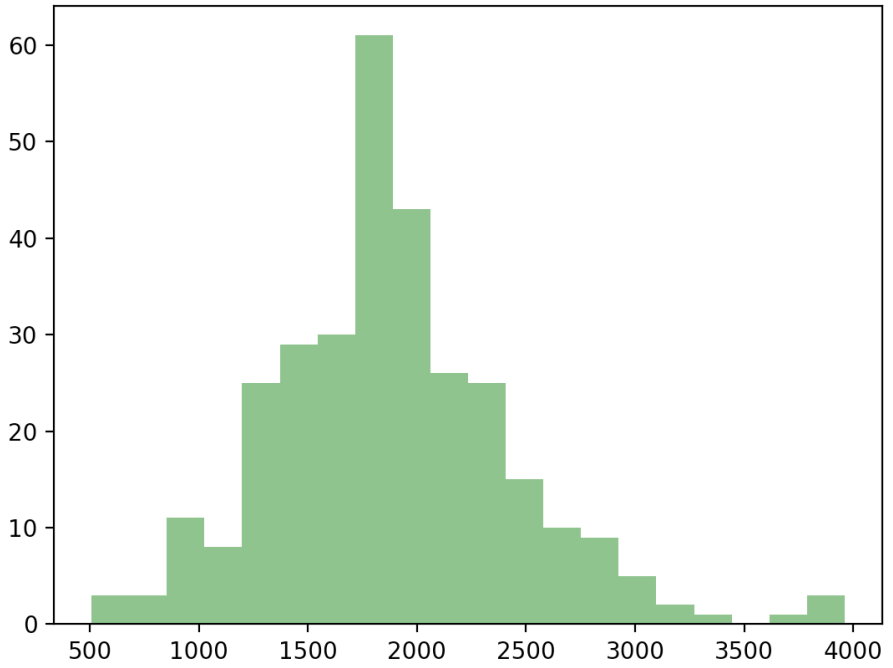
```
cfd = nltk.ConditionalFreqDist((lang, len(word)) for lang in
languages for word in nltk.corpus.udhr.words(lang + '-Latin1'))
>>> cfd.plot()
```



NLTK Book: Chapter 2

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from nltk.corpus import udhr
>>> udhr_len = [len(udhr.words(fid)) for fid in
udhr.fileids()]
>>> plt.hist(udhr_len, 20, facecolor='green',
alpha=0.5)
(array([  3.,   3.,  11.,   8.,  25.,  29.,  30.,
  61.,  43.,  26.,  25.,
         15.,  10.,   9.,   5.,   2.,   1.,   0.,
  1.,   3.]), array([  508.  ,   680.65,  853.3
,  1025.95,  1198.6 ,  1371.25,
        1543.9 ,  1716.55,  1889.2
,  2061.85,  2234.5 ,  2407.15,
        2579.8 ,  2752.45,  2925.1
,  3097.75,  3270.4 ,  3443.05,
        3615.7 ,  3788.35,  3961.  ]), <a list of
20 Patch objects>)
>>> plt.show()
```

```
>>> len(udhr.words('English-Latin1'))
1781
```



NLTK Book: Chapter 2

Your Turn: Working with the news and romance genres from the Brown Corpus, find out which days of the week are most newsworthy, and which are most romantic. Define a variable called `days` containing a list of days of the week, i.e. `['Monday', ...]`. Now tabulate the counts for these words using `cfd.tabulate(samples=days)`. Now try the same thing using `plot` in place of `tabulate`. You may control the output order of days with the help of an extra parameter: `samples=['Monday', ...]`.

- http://www.nltk.org/book/ch02.html

```
import nltk
from nltk.corpus import brown
days = set(['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
categories = ['news', 'romance']
cfd = nltk.ConditionalFreqDist((category, w) for category in categories for w in
brown.words(categories=category) if w in days)
cfd.tabulate(conditions=categories, samples=days)
```

Collect counts for pairs: conditions samples

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
news	54	43	22	20	41	33	51
romance	2	3	3	1	3	4	5

NLTK Book: Chapter 2

Table 2.1:

NLTK's Conditional Frequency Distributions: commonly-used methods and idioms for defining, accessing, and visualizing a conditional frequency distribution of counters.

Example	Description
`cfdist = ConditionalFreqDist(pairs)`	create a conditional frequency distribution from a list of pairs
`cfdist.conditions()`	the conditions
`cfdist[condition]`	the frequency distribution for this condition
`cfdist[condition][sample]`	frequency for the given sample for this condition
`cfdist.tabulate()`	tabulate the conditional frequency distribution
`cfdist.tabulate(samples, conditions)`	tabulation limited to the specified samples and conditions
`cfdist.plot()`	graphical plot of the conditional frequency distribution
`cfdist.plot(samples, conditions)`	graphical plot limited to the specified samples and conditions
`cfdist1 < cfdist2`	test if samples in `cfdist1` occur less frequently than in `cfdist2`

NLTK Book: Chapter 2

- **2.4 Generating Random Text with Bigrams**

```
>>> g = nltk.corpus.genesis.words('english-kjv.txt')
>>> g
['In', 'the', 'beginning', 'God', 'created', 'the', ...]
>>> g2 = nltk.bigrams(g)
>>> g2
<generator object bigrams at 0x111d6df68>
>>> l2 = list(g2)
>>> l2[:12]
[('In', 'the'), ('the', 'beginning'), ('beginning', 'God'),
('God', 'created'), ('created', 'the'), ('the', 'heaven'),
('heaven', 'and'), ('and', 'the'), ('the', 'earth'),
('earth', '.'), ('.', 'And'), ('And', 'the')]
```

NLTK Book: Chapter 2

- **2.4 Generating Random Text with Bigrams**

```
>>> g2 = nltk.bigrams(g)
>>> cfd = nltk.ConditionalFreqDist(g2)
>>> cfd['beginning'].max()
'of'
>>> cfd['beginning']
FreqDist({'of': 2, ',': 1, '.': 1, 'God': 1})
```

NLTK Book: Chapter 2

- **2.4 Generating Random Text with Bigrams**

```
>>> cfd['.']
FreqDist({'And': 1038, 'Then': 34, 'Now': 24,
'But': 23, 'So': 17, 'These': 16, 'The': 15,
'For': 13, 'Thus': 9, 'Therefore': 7, ...})
>>> cfd['.'].N()
1314
```

NLTK Book: Chapter 2

```
def next_word(w):
    return choice([k for k in cfd[w].keys() for i in range(cfd[w][k])])
def nwords(n, w):
    for i in range(n):
        print(w, end=' ')
        w = next_word(w)
    print()
>>> nwords(5,'The')
The purchase of the LORD
>>> nwords(5,'The')
The children of Egypt to
>>> nwords(5,'The')
The Angel which he would
```

NLTK Book: Chapter 2

```
def next_word(w):
    return choice([k for k in cfd[w].keys() for i in range(cfd[w][k])])
```

Function explained:

```
>>> cfd['beginning']
FreqDist({'of': 2, ',': 1, '.': 1, 'God': 1})
>>> cfd['beginning'].keys()
dict_keys(['God', 'of', '.', ','])
>>> cfd['beginning']['God']
1
>>> cfd['beginning']['of']
2
>>> [k for k in cfd['beginning'].keys() for i in range(cfd['beginning'][k])]
['God', 'of', 'of', '.', ',']
```

NB. Python 3.6 has a function called `choices()`

random.choice(*seq*)

Return a random element from the non-empty sequence *seq*. If *seq* is empty, raises IndexError.

NLTK Book: Chapter 2

```
>>> from random import *
>>> choice(['a','a','b'])
'a'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'tails'
>>> count = 0
>>> for i in range(1000):
...     if choice(['heads','heads','tails','heads','tails']) == 'heads':
...             count += 1
...
>>> count
580
>>>
```

- 1000 trials:
  - 580 heads
- Expected number of heads?
  - 600
  - Since 3 out of 5 in ['heads','heads','tails','heads','tails'] are heads

NLTK Book: Chapter 2

```
def nwords(n, w):
    for i in range(n):
        print(w, end=' ')
        w = next_word(w)
    print()
>>> nwords(5,'The')
The purchase of the LORD
>>> nwords(5,'The')
The children of Egypt to
>>> nwords(5,'The')
The Angel which he would
```

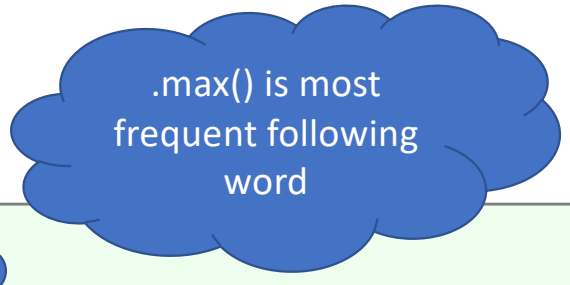
i in range(5)	w
0	The
1	purchase
2	of
3	the
4	LORD

NLTK Book: Chapter 2

```
def nwords(n, w):
    result = []
    for i in range(n):
        result.append(w)
        w = next_word(w)
    return result
>>> nwords(5,'The')
['The', 'sons', 'and', 'steal', 'away']
>>> nwords(5,'And')
['And', 'Laban', 'said', 'unto', 'Abraham']
>>> nwords(5,'And')
['And', 'Abraham', 'his', 'hand', 'of']
```

`*More useful function…*

NLTK Book: Chapter 2




```
def generate_model(cfdist, word, num=15):
    for i in range(num):
        print(word, end=' ')
        word = cfdist[word].max()

text = nltk.corpus.genesis.words('english-kjv.txt')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams) ❶
```

```
>>> cfd['living']
FreqDist({'creature': 7, 'thing': 4, 'substance': 2, ',': 1, '.': 1, 'soul': 1})
>>> generate_model(cfd, 'living')
living creature that he said , and the land of the land of the land
```

NLTK Book: Chapter 2

- **4.1 Wordlist Corpora**

```
def unusual_words(text):
    text_vocab = set(w.lower() for w in text if w.isalpha())
    english_vocab = set(w.lower() for w in nltk.corpus.words.words())
    unusual = text_vocab - english_vocab
    return sorted(unusual)
>>> len(unusual_words(nltk.corpus.gutenberg.words('austen-emma.txt')))
1934
… 'wives', 'women', 'wondered', 'wonderings', 'wonders', 'woodhouses',
'woods', 'woollen', 'words', 'workbags', 'workmen', 'worlds',
'worshipped', 'worshipping', 'worthies', 'wrapt', 'wretchedest',
'wretches', 'writes', 'xii', 'xiii', 'xiv', 'xix', 'xv', 'xvi',
'xvii', 'xviii', 'yards', 'years', 'yielded', 'youngest', 'zigzags']
```

```
wc -l /usr/share/dict/words
  235886
```

NLTK Book: Chapter 2

- Stopwords usually have little lexical content, and their presence in a text fails to distinguish it from other texts.

```
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

NLTK Book: Chapter 2

```
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
>>> vocab = set(emma)
>>> len(vocab)
7811
>>> from nltk.corpus import stopwords
>>> len(stopwords.words('english'))
179                                        (Note: 7811 - 179 = 7632)
>>> type(stopwords.words('english'))
<class 'list'>
>>> len(vocab - set(stopwords.words('english')))
7682
```

NLTK Book: Chapter 2

```
>>> set(stopwords.words('english')) - vocab
{'hasn', 'doesn', 'weren', 'm', 'shouldn', 'mightn', 'wouldn', 'didn', 'until',
"hasn't", 'haven', 'won', "should've", "aren't", "haven't", 'wasn', "mustn't", "you'd",
'needn', 've', "hadn't", 'y', "doesn't", "you're", 'aren', 'yourselves', "wasn't",
"you'll", 'isn', "shouldn't", "mightn't", "isn't", "it's", "needn't", 'll', "weren't",
"couldn't", 'mustn', 'ain', 'couldn', "didn't", "don't", "wouldn't", "won't", 'hadn',
"you've", "that'll", "shan't", 'i', "she's"}
>>> len(set(stopwords.words('english')) - vocab)
50
```

- **Why**?

```
>>> cfd = nltk.ConditionalFreqDist(nltk.bigrams(emma))
>>> cfd["'"]
FreqDist({'s': 932, 't': 19, 'am': 11, 'clock': 8, 'ye': 6, 'd': 4, 'absence': 2, 'coming': 1,
'imaginations': 1, 'understanding': 1, ...})
```

- **Note**: tokenization turned *Emma's* into *Emma* + ' + *s*

```
>>> nltk.corpus.gutenberg.sents('austen-emma.txt')[9]
['The', 'real', 'evils', ',', 'indeed', ',', 'of', 'Emma', "'", 's', 'situation', 'were', 'the',
'power', 'of', 'having', 'rather', 'too', 'much', 'her', 'own', 'way', ',', 'and', 'a',
'disposition', 'to', 'think', 'a', 'little', 'too', 'well', 'of', 'herself', ';', 'these', 'were',
'the', 'disadvantages', 'which', 'threatened', 'alloy', 'to', 'her', 'many', 'enjoyments', '.']
```

NLTK Book: Chapter 2

- One more wordlist corpus is the Names corpus, containing 8,000 first names categorized by gender. The male and female names are stored in separate files.

```
>>> names = nltk.corpus.names
>>> males = set(names.words('male.txt'))
>>> len([name for name in names.words('female.txt') if name in males])
365
>>> [name for name in names.words('female.txt') if name in males][:50]
['Abbey', 'Abbie', 'Abby', 'Addie', 'Adrian', 'Adrien', 'Ajay', 'Alex',
'Alexis', 'Alfie', 'Ali', 'Alix', 'Allie', 'Allyn', 'Andie', 'Andrea',
'Andy', 'Angel', 'Angie', 'Ariel', 'Ashley', 'Aubrey', 'Augustine',
'Austin', 'Averil', 'Barrie', 'Barry', 'Beau', 'Bennie', 'Benny', 'Bernie',
'Bert', 'Bertie', 'Bill', 'Billie', 'Bil
'Bobbie', 'Bobby', 'Brandy', 'Brett', 'B
'Bryn', 'Cal', 'Cam']
```

```
>>> len(names.words("female.txt"))
5001
>>> len(names.words("male.txt"))
2943
```

NLTK Book: Chapter 2

Names ending in?

```
>>> cfd = nltk.ConditionalFreqDist((fileid, name[-1]) for fileid in names.fileids() for
name in names.words(fileid))
>>> cfd.plot()
```