

# Natural Language Processing With Python

## Chapter2-Accessing Text Corpora and Lexical Resources

Jianzhang Zhang

[jianzhang.zhang@foxmail.com](mailto:jianzhang.zhang@foxmail.com)



# Goals of This Chapter

- What are some **useful text corpora** and **lexical resources**, and how can we access them with Python?
- Which Python constructs are most helpful for this work?
- How do we avoid repeating ourselves when writing Python code?

# 1 Accessing Text Corpora

# 1.1 Gutenberg Corpus

- NLTK includes a small selection of texts from the Project Gutenberg electronic text archive, which contains some 25,000 free electronic books, hosted at <http://www.gutenberg.org/>.

```
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt',
'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt',
'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt',
'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt',
'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt',
'shakespeare-macbeth.txt', 'whitman-leaves.txt']
```



# 1.1 Gutenberg Corpus (contd.)

- Observe that average word length appears to be a general property of English, since it has a recurrent value of 4. (In fact, the average word length is really 3 not 4, since the num\_chars variable counts space characters.) By contrast average sentence length and lexical diversity appear to be characteristics of particular authors.

```
>>> for fileid in gutenber.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 gutenber.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
```

# 1.2 Web and Chat Text

- It is important to consider **less formal language** as well. NLTK's small collection of web text includes content from a Firefox discussion forum, conversations overheard in New York, the movie script of Pirates of the Caribbean, personal advertisements, and wine reviews

```
>>> from nltk.corpus import webtext
>>> for fileid in webtext.fileids():
...     print(fileid, webtext.raw(fileid)[:65], '...')
...
firefox.txt Cookie Manager: "Don't allow sites that set removed cookies to se...
grail.txt SCENE 1: [wind] [clop clop clop] KING ARTHUR: Whoa there! [clop...
overheard.txt White guy: So, do you have any plans for this evening? Asian girl...
pirates.txt PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terr...
singles.txt 25 SEXY MALE, seeks attrac older single lady, for discreet encoun...
wine.txt Lovely delicate, fragrant Rhone wine. Polished leather and strawb...
```

```
>>> from nltk.corpus import nps_chat
>>> chatroom = nps_chat.posts('10-19-20s_706posts.xml')
>>> chatroom[123]
['i', 'do', "n't", 'want', 'hot', 'pics', 'of', 'a', 'female', ',',
'I', 'can', 'look', 'in', 'a', 'mirror', '.']
```

# 1.3 Brown Corpus

- The Brown Corpus was the first million-word electronic corpus of English, created in 1961 at Brown University. This corpus contains text from 500 sources, and the sources have been categorized by genre, such as news, editorial, and so on.

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

# 1.4 Reuters Corpus

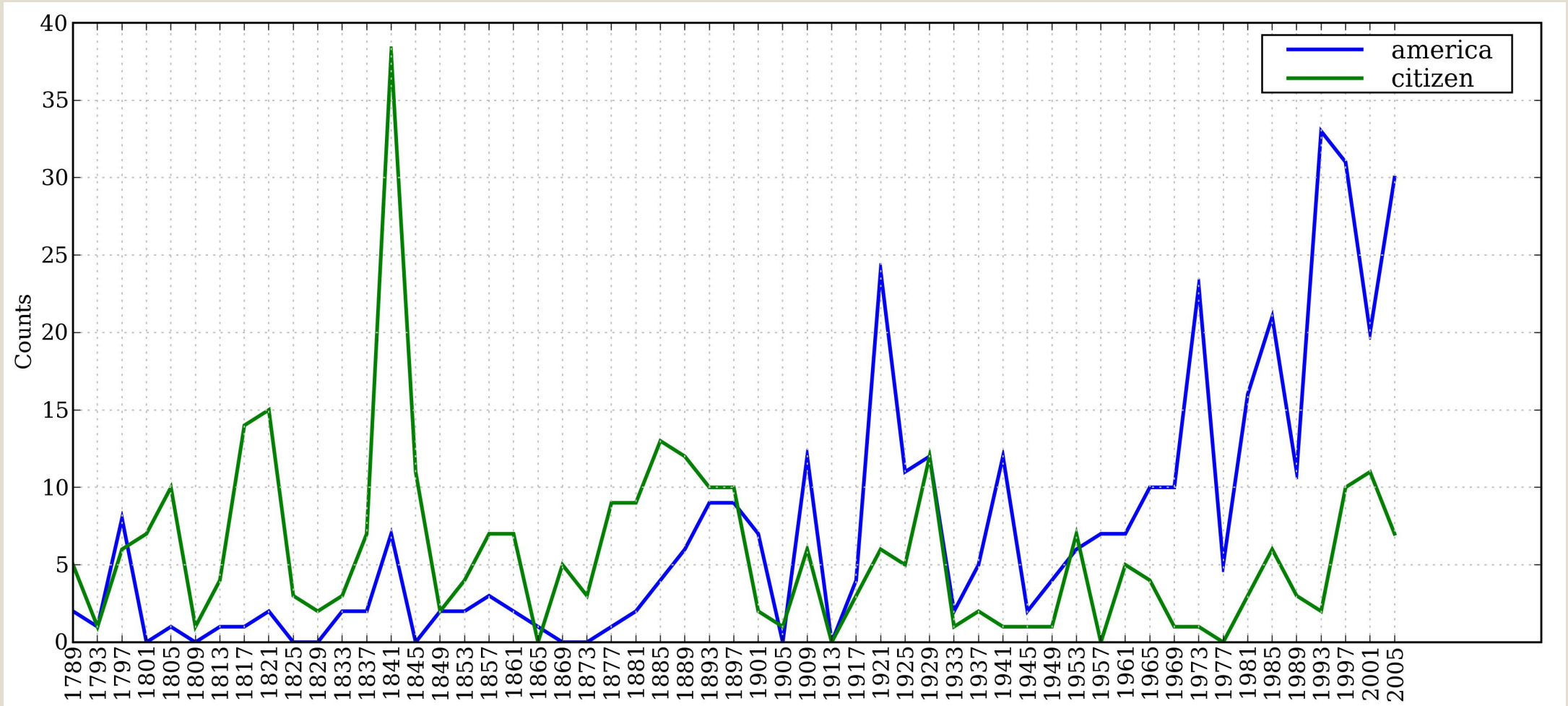
- The Reuters Corpus contains 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics, and grouped into two sets, called "training" and "test"; thus, the text with fileid 'test/14826' is a document drawn from the test set. This split is for training and testing algorithms that automatically detect the topic of a document.

```
>>> from nltk.corpus import reuters
>>> reuters.fileids()
['test/14826', 'test/14828', 'test/14829', 'test/14832', ...]
>>> 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', ...]
```



# 1.5 Inaugural Address Corpus

- An interesting property of this collection is its **time dimension**.



# 1.6 Annotated Text Corpora

- Many text corpora contain **linguistic annotations**, representing POS tags, named entities, syntactic structures, semantic roles, and so forth. NLTK provides convenient ways to access several of these corpora, and has data packages containing corpora and corpus samples, freely downloadable for use in teaching and research.
- <https://www.nltk.org/howto/corpus.html>

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



## 1.7 Corpora in Other Languages

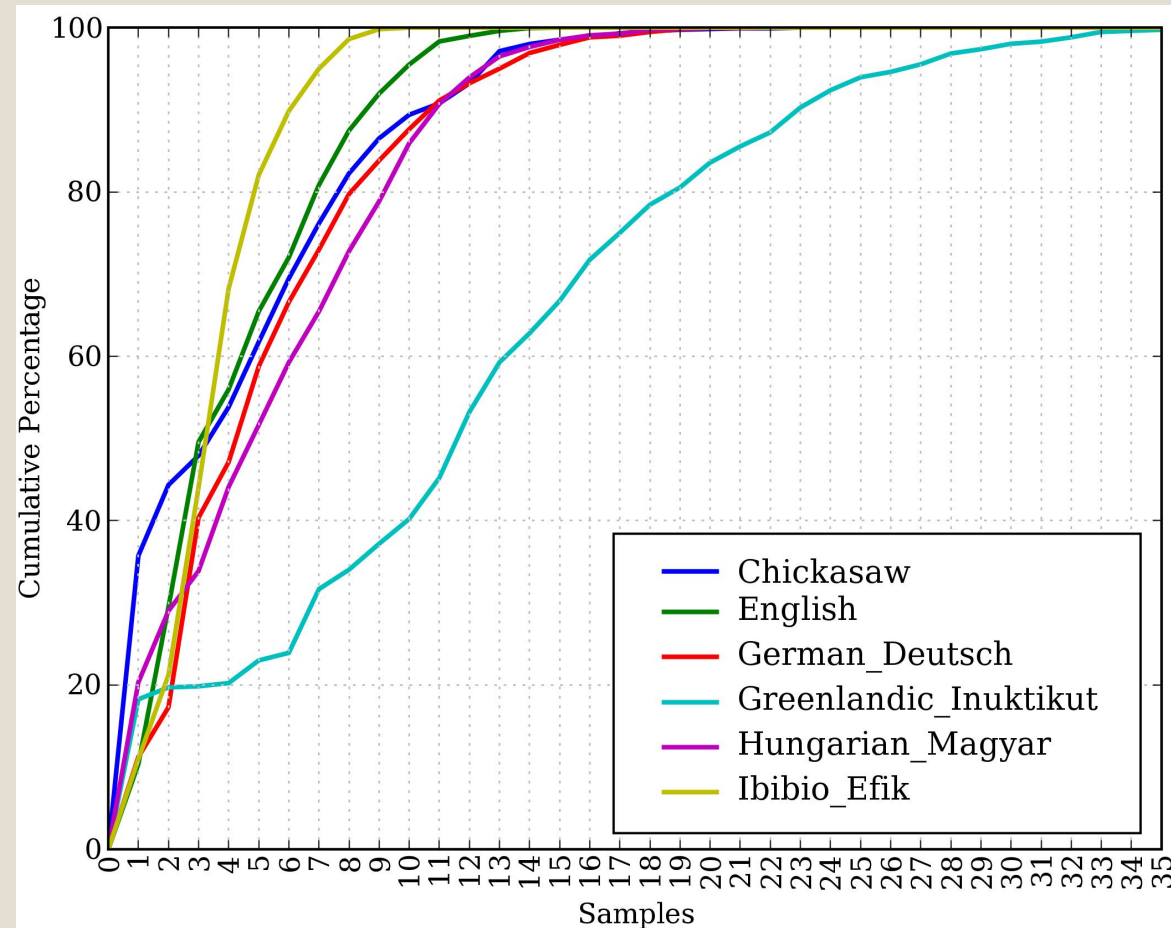
- NLTK comes with corpora for many languages, though in some cases you will need to learn how to manipulate **character encodings** in Python before using these corpora.

```
>>> nltk.corpus.cess_esp.words()
['El', 'grupo', 'estatal', 'Electricit\xe9_de_France', ...]
>>> nltk.corpus.floresta.words()
['Um', 'revivalismo', 'refrescante', 'O', '7_e_Meio', ...]
>>> 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', ...]
>>> nltk.corpus.udhr.words('Javanese-Latin1')[11:]
['Saben', 'umat', 'manungsa', 'lair', 'kanthi', 'hak', ...]
```



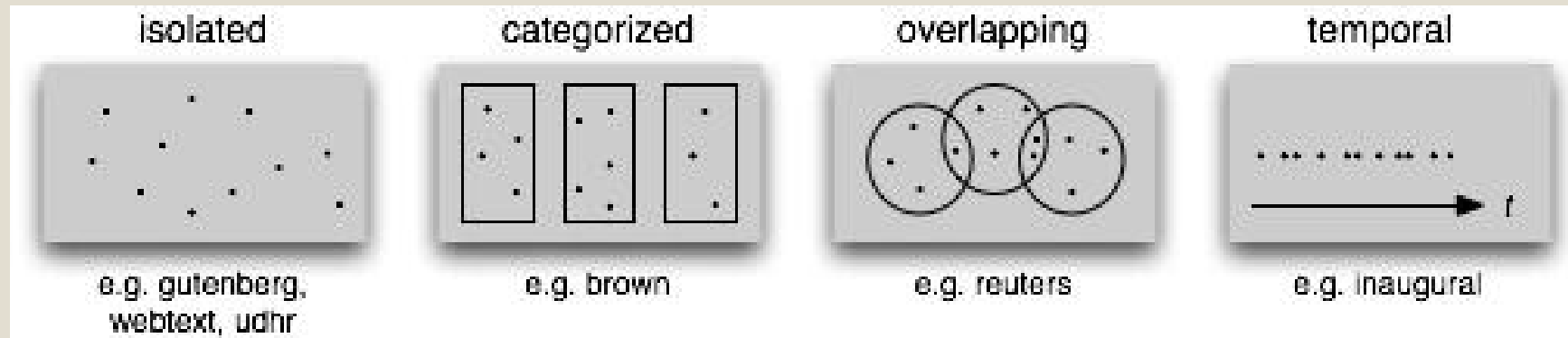
## 1.7 Corpora in Other Languages (contd.)

- Cumulative Word Length Distributions: Six translations of the Universal Declaration of Human Rights are processed; this graph shows that words having 5 or fewer letters account for about 80% of Ibibio text, 60% of German text, and 25% of Inuktitut text.



# 1.8 Text Corpus Structure

- Common Structures for Text Corpora: The simplest kind of corpus is a collection of **isolated texts** with no particular organization; some corpora are structured into categories like **genre** (Brown Corpus); some categorizations overlap, such as **topic** categories (Reuters Corpus); other corpora represent language use over **time** (Inaugural Address Corpus).



## 1.8 Text Corpus Structure (contd.)

### ➤ Basic Corpus Functionality defined in NLTK

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

## 1.9 Loading your own Corpus

- If you have your own collection of text files that you would like to access using the above methods, you can easily load them with the help of NLTK's PlaintextCorpusReader.

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = '/usr/share/dict' ❶
>>> wordlists = PlaintextCorpusReader(corpus_root, '.*') ❷
>>> wordlists.fileids()
['README', 'connectives', 'propernames', 'web2', 'web2a', 'words']
>>> wordlists.words('connectives')
['the', 'of', 'and', 'to', 'a', 'in', 'that', 'is', ...]
```



## 2 Conditional Frequency Distributions

## 2.1 Conditions and Events

- A **conditional frequency distribution** is a **collection of frequency distributions**, each one for a different "condition". The condition will often be the **category** of the text.

Condition: News		Condition: Romance	
the		the	
cute		cute	
Monday		Monday	
could		could	
will		will	

- Counting Words Appearing in a Text Collection (a conditional frequency distribution)

## 2.1 Conditions and Events (contd.)

- A **frequency distribution** counts observable **events**, such as the appearance of words in a text. A conditional frequency distribution needs to **pair each event with a condition**. So instead of processing a sequence of words, we have to process **a sequence of pairs**.

```
>>> text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...] ❶  
>>> pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...] ❷
```

- Each pair has the form **(condition, event)**. If we were processing the entire Brown Corpus by genre there would be 15 conditions (one per genre), and 1,161,192 events (one per word).

## 2.2 Counting Words by Genre

- Whereas `FreqDist()` takes a simple list as input, `ConditionalFreqDist()` takes a list of pairs.

```
>>> from nltk.corpus import brown
>>> cfd = nltk.ConditionalFreqDist(
...     (genre, word)
...     for genre in brown.categories()
...     for word in brown.words(categories=genre))
```

- Let's break this down, and look at just two genres, news and romance. For each genre, we loop over every word in the genre, producing pairs consisting of the genre and the word:

```
>>> genre_word = [(genre, word) ❶
...               for genre in ['news', 'romance'] ❷
...               for word in brown.words(categories=genre)] ❸
>>> len(genre_word)
170576
```



## 2.2 Counting Words by Genre (contd.)

```
>>> genre_word[:4]
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')] # [_start-genre]
>>> genre_word[-4:]
[('romance', 'afraid'), ('romance', 'not'), ('romance', ''), ('romance', '.')] # [_end-genre]
```

```
>>> cfd = nltk.ConditionalFreqDist(genre_word)
>>> cfd
<ConditionalFreqDist with 2 conditions>
>>> cfd.conditions()
['news', 'romance'] # [_conditions-cfd]
```

- Let's access the two conditions, and satisfy ourselves that each is just a frequency distribution:

```
>>> print(cfd['news'])
<FreqDist with 14394 samples and 100554 outcomes>
>>> print(cfd['romance'])
<FreqDist with 8452 samples and 70022 outcomes>
>>> cfd['romance'].most_common(20)
[(',', 3899), (',.', 3736), ('the', 2758), ('and', 1776), ('to', 1502),
('a', 1335), ('of', 1186), ('`', 1045), ('"', 1044), ('was', 993),
('I', 951), ('in', 875), ('he', 702), ('had', 692), ('?', 690),
('her', 651), ('that', 583), ('it', 573), ('his', 559), ('she', 496)]
>>> cfd['romance']['could']
193
```

## 2.3 Plotting and Tabulating Distributions

- Apart from combining two or more frequency distributions, and being easy to initialize, a `ConditionalFreqDist` provides some useful methods for **tabulation** and **plotting**.

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

```
>>> from nltk.corpus import udhr
>>> languages = ['Chickasaw', 'English', 'German_Deutsch',
...             'Greenlandic_Inuktitut', 'Hungarian_Magyar', 'Ibibio_Efik']
>>> cfd = nltk.ConditionalFreqDist(
...     (lang, len(word))
...     for lang in languages
...     for word in udhr.words(lang + '-Latin1'))
```

## 2.3 Plotting and Tabulating Distributions (contd.)

- In the `plot()` and `tabulate()` methods, we can optionally specify which conditions to display with a `conditions= parameter`. Similarly, we can limit the samples to display with a `samples= parameter`.
- It also gives us full control over the `order` of conditions and samples in any displays.

```
>>> cfd.tabulate(conditions=['English', 'German_Deutsch'],  
...               samples=range(10), cumulative=True)
```

	0	1	2	3	4	5	6	7	8	9
English	0	185	525	883	997	1166	1283	1440	1558	1638
German_Deutsch	0	171	263	614	717	894	1013	1110	1213	1275

To be continued