Natural Language Processing with Python

CS372: Spring, 2013

Lecture 3

Jong C. Park
Department of Computer Science
Korea Advanced Institute of Science and Technology

ACCESSING TEXT CORPORA AND LEXICAL RESOURCES

Accessing Text Corpora
Conditional Frequency Distributions

Introduction

- Questions
  - 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?

Accessing Text Corpora

- Gutenberg Corpus
- Web and Chat Text
- Brown Corpus
- Reuters Corpus
- Inaugural Address Corpus
- Annotated Text Corpora
- Corpora in Other Languages
- Text Corpus Structure
- Loading Your Own Corpus

Gutenberg Corpus

- The Project Gutenberg electronic text archive contains some 25,000 free electronic books. 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-a
lice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.t
xt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-paradise.txt', '
shakespeare-caesar.txt', 'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'w
hitman-leaves.txt']
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
>>>
```

2013-03-12 CS372: NLP with Python 5

Gutenberg Corpus

```
>>> from nltk.corpus import gutenberg
>>> gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt'
, 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-a
lice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.t
xt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-paradise.txt', '
shakespeare-caesar.txt', 'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'w
hitman-leaves.txt']
>>> emma = gutenberg.words('austen-emma.txt')
>>>
```

```
>>> for fileid in gutenberg.fileids():
        num_chars = len(gutenberg.raw(fileid))
        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 int(num_chars/num_words), int(num_words/num_sents), int(num_words/
num_vocab), fileid


4 21 26 austen-emma.txt
4 23 16 austen-persuasion.txt
4 23 22 austen-sense.txt
4 33 79 bible-kjv.txt
4 18 5 blake-poems.txt
4 17 14 bryant-stories.txt
4 17 12 burgess-busterbrown.txt
4 16 12 carroll-alice.txt
4 17 11 chesterton-ball.txt
4 19 11 chesterton-brown.txt
4 16 10 chesterton-thursday.txt
4 17 24 edgeworth-parents.txt
4 24 15 melville-moby_dick.txt
4 52 10 milton-paradise.txt
4 11 8 shakespeare-caesar.txt
4 12 7 shakespeare-hamlet.txt
4 12 6 shakespeare-macbeth.txt
4 35 12 whitman-leaves.txt
>>>
```

2013-03-12 CS372: NLP with Python 6

Gutenberg Corpus

```
>>> macbeth_sentences = gutenberg.sents('shakespeare-macbeth.txt')
>>> macbeth_sentences[1037]
['Good', 'night', ',', 'and', 'better', 'health', 'Attend', 'his', 'Maiesty']
>>> longest_len = max([len(s) for s in macbeth_sentences])
>>> [s for s in macbeth_sentences if len(s) == longest_len]
[['Doubtfull', 'it', 'stood', ',', 'As', 'two', 'spent', 'Swimmers', ',', 'that'
, 'doe', 'cling', 'together', ',', 'And', 'choake', 'their', 'Art', ':', 'The',
'mercilesse', 'Macdonwald', '(', 'Worthie', 'to', 'be', 'a', 'Rebell', ',', 'for
', 'to', 'that', 'The', 'multiplying', 'Villanies', 'of', 'Nature', 'Doe', 'swar
me', 'vpon', 'him', ')', 'from', 'the', 'Westerne', 'Isles', 'Of', 'Kernes', 'an
d', 'Gallowgrosses', 'is', 'supply', "'", 'd', ',', 'And', 'Fortune', 'on', 'his
', 'damned', 'Quarry', 'smiling', ',', 'Shew', "'", 'd', 'like', 'a', 'Rebells',
 'Whore', ':', 'but', 'all', "'", 's', 'too', 'weake', ':', 'For', 'braue', 'Mac
beth', '(', 'well', 'hee', 'deserues', 'that', 'Name', ')', 'Disdayning', 'Fortu
ne', ',', 'with', 'his', 'brandisht', 'Steele', ',', 'Which', 'smoak', "'", 'd',
 'with', 'bloody', 'execution', '(', 'Like', 'Valours', 'Minion', ')', 'caru', "
'", 'd', 'out', 'his', 'passage', ',', 'Till', 'hee', 'fac', "'", 'd', 'the', 'S
laue', ':', 'Which', 'neu', "'", 'r', 'shooke', 'hands', ',', 'nor', 'bad', 'far
well', 'to', 'him', ',', 'Till', 'he', 'vnseam', "'", 'd', 'him', 'from', 'the',
 'Naue', 'toth', "'", 'Chops', ',', 'And', 'fix', "'", 'd', 'his', 'Head', 'vpon
', 'our', 'Battlements']]
>>>
```


Web and Chat Text

- 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 Carribean*, personal advertisements, and wine reviews.

Web and Chat Text

```
>>> 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 ...
>>>
```


Web and Chat Text

- A corpus of instant messaging chat sessions, originally collected by the Naval Postgraduate School for research on automatic detection of Internet predators

```
>>> 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', '.']
>>>
```

Brown Corpus

- The Brown Corpus was the first million-word electronic corpus of English, created in 1961 at Brown University.

Brown Corpus

```
>>> from nltk.corpus import brown
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies',
 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance'
, 'science_fiction']
>>> brown.words(categories='news')
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
>>> brown.words(fileids=['cg22'])
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
>>> brown.sents(categories=['news', 'editorial', 'reviews'])
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investiga
tion', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no
', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place', '.'], ['T
he', 'jury', 'further', 'said', 'in', 'term-end', 'presentments', 'that', 'the',
 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-all', 'charge', 'o
f', 'the', 'election', ',', '``', 'deserves', 'the', 'praise', 'and', 'thanks',
'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the', 'manner', 'in', 'which
', 'the', 'election', 'was', 'conducted', '.'], ...]
>>> news_text = brown.words(categories='news')
>>> fdist = nltk.FreqDist([w.lower() for w in news_text])
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> for m in modals:
        print m + ':', fdist[m],


can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
>>>
```

Brown Corpus

```
>>> cfd = nltk.ConditionalFreqDist(
        (genre, word)
        for genre in brown.categories()
        for word in brown.words(categories=genre))
>>> genres = ['news', 'religion', 'hobbies', 'science_fiction', 'romance', 'humo
r']
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> cfd.tabulate(conditions=genres, samples=modals)
                  can could  may might must will
            news   93   86   66   38   50  389
        religion   82   59   78   12   54   71
         hobbies  268   58  131   22   83  264
science_fiction   16   49    4   12    8   16
         romance   74  193   11   51   45   43
           humor   16   30    8    8    9   13
>>>
```


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”.

Reuters Corpus

```
>>> from nltk.corpus import reuters
>>> reuters.categories()
['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconut', 'c
ocoanut-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', 'ins
tal-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', 'pa
lladium', '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']
>>> reuters.categories('training/9865')
['barley', 'corn', 'grain', 'wheat']
>>> reuters.categories(['training/9865', 'training/9880'])
['barley', 'corn', 'grain', 'money-fx', 'wheat']
>>> reuters.fileids(['barley', 'corn'])
['test/14832', 'test/14858', 'test/15033', 'test/15043', 'test/15106', 'test/152
87', 'test/15341', 'test/15618', 'test/15648', 'test/15649', 'test/15676', 'test
/15686', 'test/15720', 'test/15728', 'test/15845', 'test/15856', 'test/15860', '
test/15863', 'test/15871', 'test/15875', 'test/15877', 'test/15890', 'test/15904
', 'test/15906', 'test/15910', 'test/15911', 'test/15917', 'test/15952', 'test/1
5999', 'test/16012', 'test/16071', 'test/16099', 'test/16147', 'test/16525', 'te
st/16624', 'test/16751', 'test/16765', 'test/17503', 'test/17509', 'test/17722',
 'test/17767', 'test/17769', 'test/18024', 'test/18035', 'test/18263', 'test/184
```

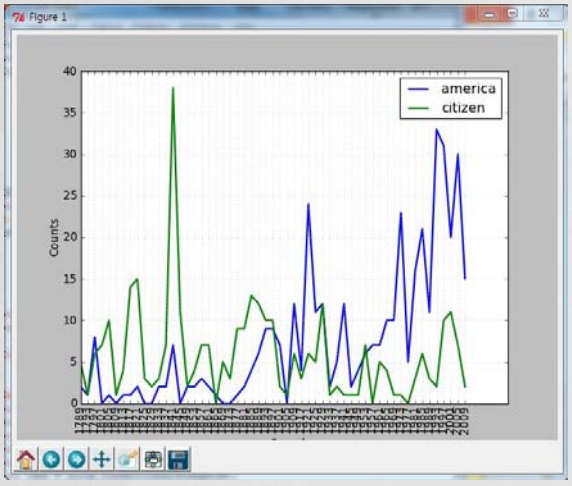

Reuters Corpus

```
>>> reuters.words('training/9865')[:14]
['FRENCH', 'FREE', 'MARKET', 'CEREAL', 'EXPORT', 'BIDS', 'DETAILED', 'French', '
operators', 'have', 'requested', 'licences', 'to', 'export']
>>> reuters.words(['training/9865', 'training/9880'])
['FRENCH', 'FREE', 'MARKET', 'CEREAL', 'EXPORT', ...]
>>> reuters.words(categories='barley')
['FRENCH', 'FREE', 'MARKET', 'CEREAL', 'EXPORT', ...]
>>> reuters.words(categories=['barley', 'corn'])
['THAI', 'TRADE', 'DEFICIT', 'WIDENS', 'IN', 'FIRST', ...]
>>>
```

Inaugural Address Corpus

```
>>> import nltk
>>> 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))
>>> cfd.plot()
```


Inaugural Address Corpus

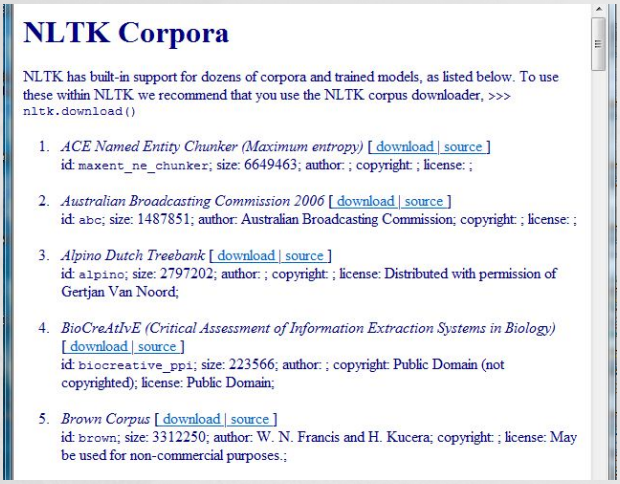


forward

Annotated Text Corpora

- Many text corpora contain linguistic annotations represent part-of-speech tags, named entities, syntactic structures, semantic roles, and so forth.
- Consult http://www.nltk.org/data for information about downloading them.

Annotated Text Corpora



NLTK Corpora

NLTK has built-in support for dozens of corpora and trained models, as listed below. To use these within NLTK we recommend that you use the NLTK corpus downloader, `>>> nltk.download()`

1. *ACE Named Entity Chunker (Maximum entropy)* [download | source]
   id: `maxent_ne_chunker`; size: 6649463; author: ; copyright: ; license: ;
2. *Australian Broadcasting Commission 2006* [download | source]
   id: `abc`; size: 1487851; author: Australian Broadcasting Commission; copyright: ; license: ;
3. *Alpino Dutch Treebank* [download | source]
   id: `alpino`; size: 2797202; author: ; copyright: ; license: Distributed with permission of Gertjan Van Noord;
4. *BioCreAtIvE (Critical Assessment of Information Extraction Systems in Biology)* [download | source]
   id: `biocreative_ppi`; size: 223566; author: ; copyright: Public Domain (not copyrighted); license: Public Domain;
5. *Brown Corpus* [download | source]
   id: `brown`; size: 3312250; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;

Annotated Text Corpora

72. *Treebank Part of Speech Tagger (HMM)* [download | source]
id: hmm_treebank_pos_tagger; size: 750857; author: ; copyright: ; license: ;

73. *Treebank Part of Speech Tagger (Maximum entropy)* [download | source]
id: maxent_treebank_pos_tagger; size: 5031883; author: ; copyright: ; license: ;

74. *Punkt Tokenizer Models* [download | source]
id: punkt; size: 5920457; author: Jan Strunk; copyright: ; license: ;

Corpora in Other Languages

```
>>> 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')
['\xe0\xa4\xaa\xe0\xa5\x82\xe0\xa4\xb0\xe0\xa5\x8d\xe0\xa4\xa3', '\xe0\xa4\xaa\x
e0\xa5\x8d\xe0\xa4\xb0\xe0\xa4\xa4\xe0\xa4\xbf\xe0\xa4\xac\xe0\xa4\x82\xe0\xa4\x
a7', ...]
>>> nltk.corpus.udhr.fileids()
['Abkhaz-Cyrillic+Abkh', 'Abkhaz-UTF8', 'Achehnese-Latin1', 'Achuar-Shiwiar-Lati
n1', 'Adja-UTF8', 'Afaan_Oromo_Oromiffa-Latin1', 'Afrikaans-Latin1', 'Aguaruna-L
atin1', 'Akuapem_Twi-UTF8', 'Albanian_Shqip-Latin1', 'Amahuaca', 'Amahuaca-Latin
1', 'Amarakaeri-Latin1', 'Amharic-Afenegus6..60375', 'Amuesha-Yanesha-UTF8', 'Ar
abela-Latin1', 'Arabic_Alarabia-Arabic', 'Armenian-DallakHelv', 'Asante-UTF8', '
```

Corpora in Other Languages

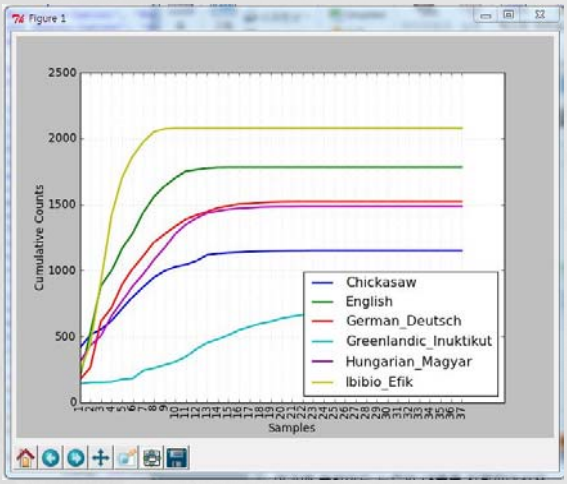
```
_Gaeilge-Latin1', 'Italian-Latin1', 'Italian_Italiano-Latin1', 'Japanese_Nihongo
-EUC', 'Japanese_Nihongo-JIS', 'Japanese_Nihongo-SJIS', 'Japanese_Nihongo-UTF8',
 'Javanese-Latin1', 'Jola-Fogny_Diola-UTF8', 'Kabye-UTF8', 'Kannada-UTF8', 'Kaon
de-Latin1', 'Kapampangan-Latin1', 'Kasem-UTF8', 'Kazakh-Cyrillic', 'Kazakh-UTF8'
, 'Kiche_Quiche-Latin1', 'Kicongo-Latin1', 'Kimbundu_Mbundu-Latin1', 'Kinyamwezi
_Nyamwezi-Latin1', 'Kinyarwanda-Latin1', 'Kituba-Latin1', 'Korean_Hankuko-UTF8',
 'Kpelewo-UTF8', 'Krio-UTF8', 'Kurdish-UTF8', 'Lamnso_Lam-nso-UTF8', 'Lao-UTF8',
 'Latin_Latina-Latin1', 'Latin_Latina-v2-Latin1', 'Latvian-Latin1', 'Limba-UTF8'
```

```
', 'Yoruba-UTF8', 'Zapoteco-Latin1', 'Zapoteco-SanLucasQuiavini-Latin1', 'Zhuang
-Latin1', 'Zulu-Latin1']
>>> nltk.corpus.udhr.words('Korean_Hankuko-UTF8')[11:]
[u'\ucc9c\ubd80\uc758', u'\uc874\uc5c4\uc131\uacfc', ...]
>>> from nltk.corpus import udhr
>>> languages = ['Chickasaw', 'English', 'German_Deutsch',
            'Greenlandic_Inuktikut', 'Hungarian_Magyar', 'Ibibio_Efik']
>>> cfd = nltk.ConditionalFreqDist(
        (lang, len(word))
        for lang in languages
        for word in udhr.words(lang + '-Latin1'))
>>> cfd.plot(cumulative=True)
```

```
>>> for w in nltk.corpus.udhr.words('Korean_Hankuko-UTF8')[11:50]:
        print w,

천부의 존엄성과 동등하고 양도할 수 없는 권리를 인정하는 것이 세계의 자유 , 정의 및 평화의 기초
이며 , 인권에 대한 무시와 경멸이 인류의 양심을 격분시키는 만행을 초래하였으며 , 인간이 언론과
신앙의 자유 , 그리고 공포와 결립으로부터의 자유를 누릴 수 있는
>>>
```


Corpora in Other Languages



Text Corpus Structure

- Common structures
  - Isolated, Categorized, Overlapping, Temporal

```
>>> from nltk.corpus import gutenberg
>>> raw = gutenberg.raw("burgess-busterbrown.txt")
>>> raw[1:20]
'The Adventures of B'
>>> words = gutenberg.words("burgess-busterbrown.txt")
>>> words[1:20]
['The', 'Adventures', 'of', 'Buster', 'Bear', 'by', 'Thornton', 'W', '.', 'Burge
ss', '1920', ']', 'I', 'BUSTER', 'BEAR', 'GOES', 'FISHING', 'Buster', 'Bear']
>>> sents = gutenberg.sents("burgess-busterbrown.txt")
>>> sents[1:20]
[['I'], ['BUSTER', 'BEAR', 'GOES', 'FISHING'], ['Buster', 'Bear', 'yawned', 'as'
, 'he', 'lay', 'on', 'his', 'comfortable', 'bed', 'of', 'leaves', 'and', 'watche
d', 'the', 'first', 'early', 'morning', 'sunbeams', 'creeping', 'through', 'the'
, 'Green', 'Forest', 'to', 'chase', 'out', 'the', 'Black', 'Shadows', '.'], ['On
ce', 'more', 'he', 'yawned', ',', 'and', 'slowly', 'got', 'to', 'his', 'feet', '
and', 'shook', 'himself', '.'], ['Then', 'he', 'walked', 'over', 'to', 'a', 'big
', 'pine', '-', 'tree', ',', 'stood', 'up', 'on', 'his', 'hind', 'legs', ',', 'r
eached', 'as', 'high', 'up', 'on', 'the', 'trunk', 'of', 'the', 'tree', 'as', 'h
e', 'could', ',', 'and', 'scratched', 'the', 'bark', 'with', 'his', 'great', 'cl
aws', '.'], ['After', 'that', 'he', 'yawned', 'until', 'it', 'seemed', 'as', 'if
```


Loading Your Own Corpus

- Load your own collection of text files.

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

Loading Your Own Corpus

- Another example

```
>>> from nltk.corpus import BracketParseCorpusReader
>>> corpus_root  = r"C:\corpora\penntreebank\parsed\mrg\wsj"
>>> file_pattern = r".*/wsj_.*\.mrg"
>>> ptb = BracketParseCorpusReader(corpus_root, file_pattern)
>>> ptb.fileids()
['00/wsj_0001.mrg', '00/wsj_0002.mrg', '00/wsj_0003.mrg', ...]
>>> len(ptb.sents())
49208
>>> ptb.sents(fileids='20/wsj_2013.mrg')[19]
['The', '55-year-old', 'Mr.', 'Noriega', 'is', "n't", 'as', 'smooth', 'as',
'the', 'shah', 'of', 'Iran', ',', 'as', 'well-born', 'as', 'Nicaragua', "'s", ...]
```


Conditional Frequency Distributions

- Conditions and Events
- Counting Words by Genre
- Plotting and Tabulating Distributions
- Generating Random Text with Bigrams

Conditions and Events

- While a frequency distribution counts observable events, a conditional frequency distribution needs to pair each event with a condition.

>>> text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

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

Counting Words by Genre

```
>>> from nltk.corpus import brown
>>> cfd = nltk.ConditionalFreqDist(
        (genre, word)
        for genre in brown.categories()
        for word in brown.words(categories=genre))
>>>
>>> genre_word = [(genre, word)
              for genre in ['news', 'romance']
              for word in brown.words(categories=genre)]
>>> len(genre_word)
170576
>>> genre_word[:4]
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
>>> genre_word[-4:]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "''"), ('romance', '.')]
>>>
```

Counting Words by Genre

```
>>> cfd = nltk.ConditionalFreqDist(genre_word)
>>> cfd
<ConditionalFreqDist with 2 conditions>
>>> cfd.conditions()
['news', 'romance']
>>>
>>> cfd['news']
<FreqDist with 100554 outcomes>
>>> cfd['romance']
<FreqDist with 70022 outcomes>
>>> cfd['romance']['could']
193
```

```
>>> list(cfd['romance'])
[',', '.', 'the', 'and', 'to', 'a', 'of', '``', "''", 'was', 'I', 'in', 'he', 'h
ad', '?', 'her', 'that', 'it', 'his', 'she', 'with', 'you', 'for', 'at', 'He', '
on', 'him', 'said', '!', '--', 'be', 'as', ';', 'have', 'but', 'not', 'would', '
She', 'The', 'out', 'were', 'up', 'all', 'from', 'could', 'me', 'like', 'been',
'so', 'there', 'they', 'one', 'about', 'my', 'an', 'is', 'or', 'this', 'It', 'if
', 'them', 'into', 'But', 'And', 'down', 'back', 'when', 'no', 'what', 'did', 't
heir', 'do', 'by', 'only', 'your', 'thought', 'which', 'You', "didn't", 'then',
'just', 'little', 'time', 'get', 'too', 'got', 'who', 'before', 'know', 'over',
'man', 'because', 'more', 'never', 'now', 'way', 'went', 'we', "I'm", 'eyes', 'g
o', 'came', 'can', 'see', 'come', 'even', 'old', 'are', 'looked', 'other', 'They
', 'its', 'knew', 'much', 'some', 'around', 'any', 'There', 'good', 'here', 'lon
g', 'than', 'away', 'felt', 'day', 'own', "don't", 'made', 'still', 'take', 'goi
ng', 'how', 'say', 'something', 'after', ':', 'off', 'think', 'through', 'In', '
right', 'look', 'night', 'where', "I'll", 'again', 'himself', 'those', 'first',
```


Plotting and Tabulating Distributions

- Pages 17 and 18 of this lecture note.
- Pages 23 and 24 of this lecture note.

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

Generating Random Text with Bigrams

- Create a table of bigrams using a conditional frequency distribution.

```
>>> sent = ['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven',
        'and', 'the', 'earth', '.']
>>> nltk.bigrams(sent)
[('In', 'the'), ('the', 'beginning'), ('beginning', 'God'), ('God', 'created'),
('created', 'the'), ('the', 'heaven'), ('heaven', 'and'), ('and', 'the'), ('the'
, 'earth'), ('earth', '.')]
>>>
```


Generating Random Text with Bigrams

- Example 2-1. Generating random text

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


>>> text = nltk.corpus.genesis.words('english-kjv.txt')
>>> bigrams = nltk.bigrams(text)
>>> cfd = nltk.ConditionalFreqDist(bigrams)
>>>
>>> print cfd['living']
<FreqDist: u'creature': 7, u'thing': 4, u'substance': 2, u',': 1, u'.': 1, u'sou
l': 1>
>>> generate_model(cfd, 'living')
living creature that he said , and the land of the land of the land
>>>
```

Summary

- Accessing Text Corpora
  - Gutenberg Corpus
  - Web and Chat Text
  - Brown Corpus
  - Reuters Corpus
  - Inaugural Address Corpus
  - Annotated Text Corpora
  - Corpora in Other Languages
  - Text Corpus Structure
  - Loading Your Own Corpus

Summary

- Conditional Frequency Distributions
  - Conditions and Events
  - Counting Words by Genre
  - Plotting and Tabulating Distributions
  - Generating Random Text with Bigrams

Homework #1

- Due: 22 March, 2013 (midnight)
- Problems
  - Exercises: 1.4, 1.19, 1.20, 1.22, 1.24, 1.26
  - Your Turn: Pages 6, 8, 24
- Submission
  - Send a message to cs372@nlp.kaist.ac.kr with a Word file that includes answers and/or Python codes, with Subject: [CS372] HW#1, Your Name.