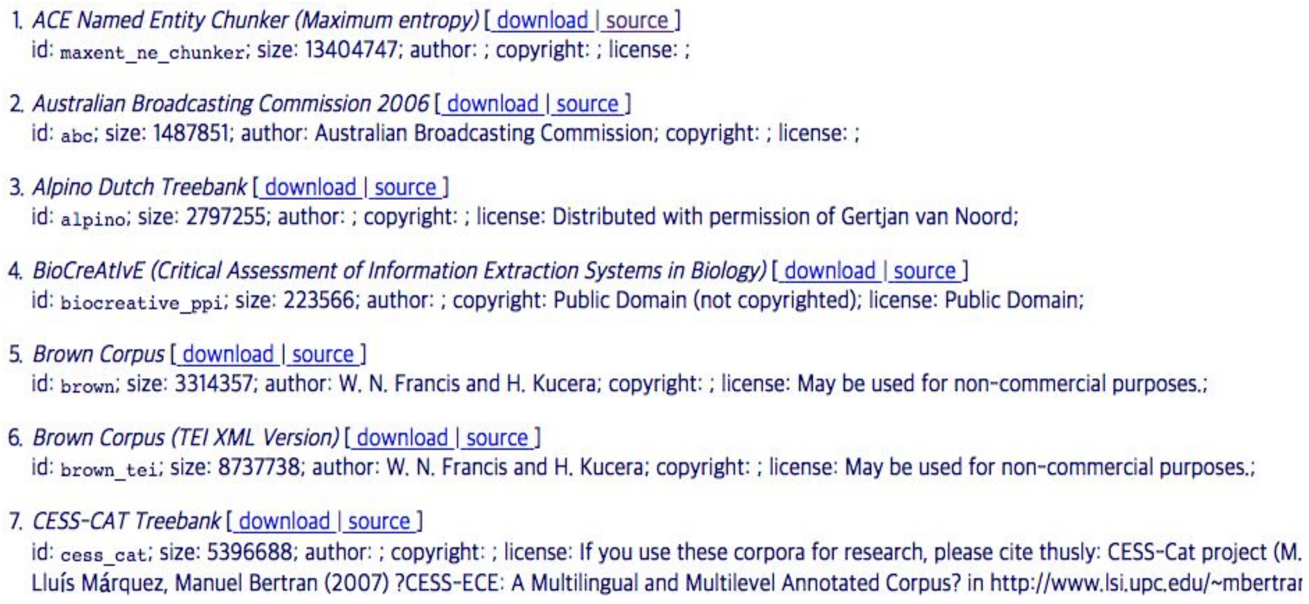
Corpus와 사전

NLTK Ch. 2. Accessing Text Corpora and Lexical Resources
https://www.nltk.org/book/ch02.html

성균관대학교
정윤경

Corpus (pl. corpora)

- a corpus or text corpus is a large and structured set of texts
- http://www.nltk.org/nltk_data/



1. *ACE Named Entity Chunker (Maximum entropy)* [download | source]
   id: `maxent_ne_chunker`; size: 13404747; 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: 2797255; 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: 3314357; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;
6. *Brown Corpus (TEI XML Version)* [download | source]
   id: `brown_tei`; size: 8737738; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;
7. *CESS-CAT Treebank* [download | source]
   id: `cess_cat`; size: 5396688; author: ; copyright: ; license: If you use these corpora for research, please cite thusly: CESS-Cat project (M.
   Lluís Márquez, Manuel Bertran (2007) ?CESS-ECE: A Multilingual and Multilevel Annotated Corpus? in http://www.lsi.upc.edu/~mbertrar

Gutenberg Corpus

- NLTK includes some texts from the Project Gutenberg electronic text archive, which contains 60,000 free e-books 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']
>>> emma =
nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)   #get the number of words in Emma
192427
```

Web Text

- **Webtext** from a *Firefox discussion forum*, *conversations overheard in New York*, the movie script of *Pirates of the Carribean*, 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…
```

Chat Text

- **nps_chat** is a corpus of instant messaging chat sessions (collected by the Naval Postgraduate School) for research on automatic detection of Internet predators
- **nps_chat** contains over 10,000 posts, providing date, an age-specific chatroom (teens, 20s, 30s, 40s, plus a generic adults chatroom)

```
>>> 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 (1961)

- The first million-word electronic corpus of English, created at Brown University
- Contains text from 500 sources, **categorized by genre, such as news, editorial**, etc.
- Resource for studying **differences between genres,** e.g., compare genres in their usage of modal verbs

```
>>> 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'...], ['The', 'jury', 'further'...], ...]

>>> 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], end=' ')
...
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

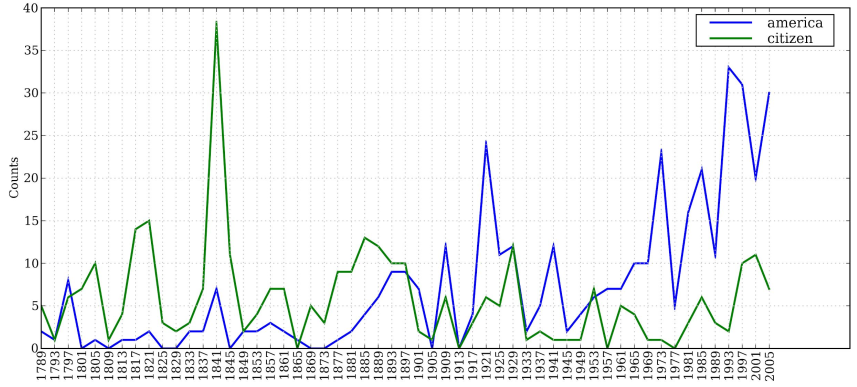
Reuters Corpus

- Contains 10,788 news documents totaling 1.3 million words
- Documents have been classified into **90 topics, and grouped into two sets, called "training" and "test"**
- 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', ...]
```

Inaugural Address Corpus

- Includes 55 presidential address
- An interesting property of this collection is **time dimension**
- How the words *America* and *citizen* are used over time



```
>>> from nltk.corpus import inaugural
>>> inaugural.fileids()
['1789-Washington.txt', '1793-Washington.txt',
'1797-Adams.txt', ...]
>>> [fileid[:4] for fileid in inaugural.fileids()]
['1789', '1793', '1797', '1801', '1805', '1809', '1813',
'1817', '1821', ...]

>>> 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()
```

Annotated Text Corpora

- Contain linguistic annotations, representing POS tags, named entities, syntactic structures, semantic roles, and so forth
- To access NLTK corpora, please consult the Corpus HOWTO at http://nltk.org/howto

Corpus	Compiler	Contents
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)
FrameNet	Fillmore, Baker et al	10k word senses, 170k manually annotated sentences
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

Loading your own Corpus

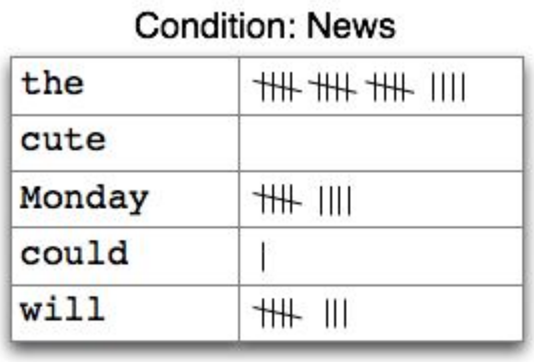
```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = '/usr/share/dict'  #/usr/share/dict 디렉토리의 파일
>>> wordlists = PlaintextCorpusReader(corpus_root, '.*') # ['a.txt', 'test/b.txt'] 파일 이름 리스트도 가능
>>> wordlists.fileids()
['README', 'connectives', 'propernames', 'web2', 'web2a', 'words']
>>> wordlists.words('connectives')
['the', 'of', 'and', 'to', 'a', 'in', 'that', 'is', ...]
```

Basic Corpus Functionality defined in NLTK

fileids()	files of the corpus
fileids([categories])	files of the corpus corresponding to these categories
categories()	categories of the corpus
categories([fileids])	categories of the corpus corresponding to these files
raw()	raw content of the corpus
raw(fileids=[f1,f2,f3])	raw content of the specified files
raw(categories=[c1,c2])	raw content of the specified categories
words()	words of the whole corpus
words(fileids=[f1,f2,f3])	words of the specified fileids
words(categories=[c1,c2])	words of the specified categories
sents()	sentences of the whole corpus
sents(fileids=[f1,f2,f3])	sentences of the specified fileids
sents(categories=[c1,c2])	sentences of the specified categories
encoding(fileid)	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

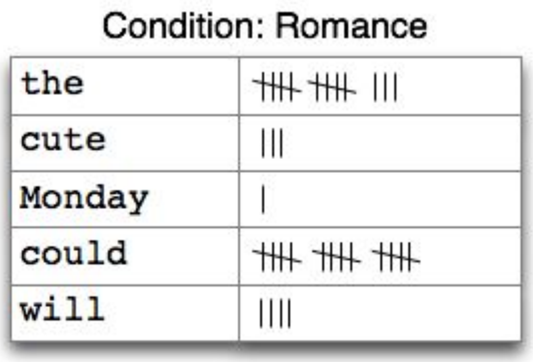
Conditional Frequency Distributions

- **FreqDist(wordlist)** computes the number of occurrences of each word in the list
- A **corpus are divided into several categories**, by genre, topic, author, etc.
- Maintain **frequency distributions for each category**
- **Conditional frequency distribution** is a collection of frequency distributions, each one for a different "condition"



Condition: News

`the`	𝍸 𝍸 𝍸 ||||
`cute`	
`Monday`	𝍸 ||||
`could`	|
`will`	𝍸 |||



Condition: Romance

`the`	𝍸 𝍸 |||
`cute`	|||
`Monday`	|
`could`	𝍸 𝍸 𝍸
`will`	||||

Generating Random Text with Bigrams

1. **bigrams**() takes a list of words and builds a list of consecutive word pairs

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

2. Choose a random **word** as our initial context
3. Repeat printing the value of **word**
4. Reset **word** to be the most likely token in that context (**max()**)

```
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)
>>> 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
```

Counting Words by Genre

- A frequency distribution counts the appearance of words in a text
- **ConditionalFreqDist**() takes a list of pairs (genre, word)

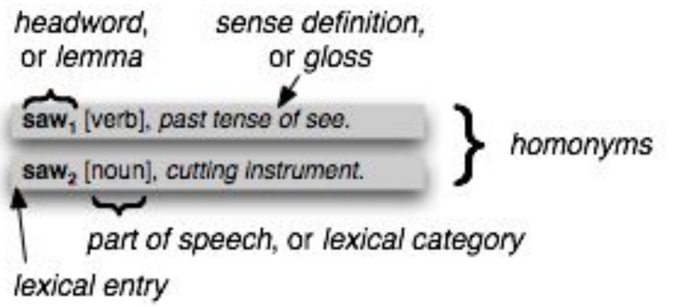
```
from nltk.corpus import brown
>>> genre_word = [(genre, word)
...         for genre in ['news', 'romance']
...         for word in brown.words(categories=genre)]
>>> genre_word[:4]
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news',
'Grand')]
>>> genre_word[-4:]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "''"),
('romance', '.')]
>>> cfd = nltk.ConditionalFreqDist(genre_word)
>>> print(cfd['romance'])
<FreqDist with 8452 samples and 70022 outcomes>
>>> cfd['romance'].most_common(5)
[(',', 3899), ('.', 3736), ('the', 2758), ('and', 1776), ('to', 1502)]
>>> cfd['romance']['could']
193
```

사전

Lexical Resources (lexicon)

- A collection of words and phrases **along with associated information**
- Lexical resources are created and enriched with the help of texts
  - vocab = sorted(set(my_text)) builds the vocabulary of my_text
  - word_freq = FreqDist(my_text) counts the frequency of each word in the text
  - Both of vocab and word_freq are simple lexical resources

A lexical entry consists of a **headword** (also known as a **lemma**) along with information such as the part of speech and the sense definition. Two distinct words having the same spelling are called **homonyms**.


headword, or lemma
sense definition, or gloss
saw₁ [verb], past tense of see.
saw₂ [noun], cutting instrument.
homonyms
part of speech, or lexical category
lexical entry

Wordlist Corpora

- Simplest lexicon
- **Stopwords** are high-frequency words like *the, to* and also that we sometimes want to filter out of a document before further processing. Their presence in a text fails to distinguish it from other texts
- **Names** corpus, containing 8,000 first names categorized by gender

```
>>> 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']

>>> names = nltk.corpus.names
>>> names.fileids()
['female.txt', 'male.txt']
>>> male_names = names.words('male.txt')
>>> female_names = names.words('female.txt')
# 남자인지 여자인지 모호한 이름 찾기
>>> [w for w in male_names if w in female_names]
['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', ...]
```

WordNet

- WordNet is a semantically-oriented dictionary, a **thesaurus** but with a richer structure
- NLTK includes the English WordNet, with 155,287 words and 117,659 synonym sets

Consider the sentence in (a). If we replace the word motorcar in (a) by automobile, to get (b), the meaning of the sentence stays pretty much the same:

a. Benz is credited with the invention of the **motorcar**.
b. Benz is credited with the invention of the **automobile**.

The words motorcar and automobile have the same meaning, i.e. they are **synonyms**.

Senses and Synonyms

- car.**n**.01 is called a **synset** (synonym set), a collection of synonymous words
    - **n:** noun
- Each word of a synset can have several meanings, e.g., car can also signify a train carriage, a gondola, or an elevator car
- We are interested in the **single meaning** that is common to all words of the above synset
- Synsets also come with a definition and example sentences

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
[Synset('car.n.01')]
>>> wn.synset('car.n.01').lemma_names()
['car', 'auto', 'automobile', 'machine', 'motorcar']
>>> wn.synset('car.n.01').definition()
'a motor vehicle with four wheels; usually
propelled by an internal combustion engine'
>>> wn.synset('car.n.01').examples()
['he needs a car to get to work']
```

Senses and Synonyms

- **To eliminate ambiguity**, identify these words as **car.n.01.automobile, car.n.01.motorcar**, etc.
- **lemma**: pairing of a synset with a word
- Unlike the word **motorcar**, which is **unambiguous** and has one synset, the word **car** is **ambiguous**, having five synsets

```
>>> wn.synset('car.n.01').lemmas()
[Lemma('car.n.01.car'), Lemma('car.n.01.auto'),
Lemma('car.n.01.automobile'),
Lemma('car.n.01.machine'), Lemma('car.n.01.motorcar')]
>>> wn.lemma('car.n.01.automobile')
Lemma('car.n.01.automobile')
>>> wn.lemma('car.n.01.automobile').synset()
Synset('car.n.01')
>>> wn.lemma('car.n.01.automobile').name()
'Automobile'

>>> wn.synsets('car')
[Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'),
Synset('car.n.04'), Synset('cable_car.n.01')]
```

sentiWordNet

- SentiWordNet is a lexical resource for **opinion mining**. SentiWordNet assigns to each synset of WordNet three sentiment scores: **positivity, negativity, objectivity**
- Official website: http://sentiwordnet.isti.cnr.it/
- NLTK API: http://www.nltk.org/howto/sentiwordnet.html
- More examples at http://compprag.christopherpotts.net/wordnet.html

```
>>> from nltk.corpus import sentiwordnet as swn
>>> breakdown = swn.senti_synset('breakdown.n.03')
>>> print(breakdown)
<breakdown.n.03: PosScore=0.0 NegScore=0.25>
>>> breakdown.pos_score()
0.0
>>> breakdown.neg_score()
0.25
>>> breakdown.obj_score()
0.75
```

한국어 감성 사전

- KOrean Sentiment Analysis Corpus (Kosac)
  - http://word.snu.ac.kr/kosac/
- KNU 한국어 감성사전
  - http://dilab.kunsan.ac.kr/knusl.html

Summary

- A text **corpus** is a large, structured collection of texts. NLTK comes with many corpora, e.g., the Brown Corpus, nltk.corpus.brown.
- A **conditional frequency distribution** is a collection of frequency distributions, each one for a different condition. They can be used for counting word frequencies, given a context or a genre.
- **WordNet** is a semantically-oriented dictionary of English, consisting of **synonym** sets — or synsets — and organized into a network.