Handout 5: Corpora and Lexica

Corpora

1. Corpus objects

a. Reside in module `nltk.corpus`

```
>>> from nltk.corpus import gutenberg
>>> from nltk.corpus import brown
```

b. Corpus description: `print(gutenberg.readme())`

2. Characters, words, sentences

```
>>> gutenberg.raw()
'[Emma by Jane Austen ...'
>>> gutenberg.words()
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
>>> gutenberg.sents()
[['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']'], ['VOLUME', 'I'], ...]
```

3. How to avoid typing "gutenberg" all the time?

```
>>> from nltk.corpus import gutenberg as gb
>>> gb.words()
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
```

4. Selecting a single file

```
>>> gb.words('melville-moby_dick.txt')
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', ...]
>>> gb.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', ...]
```

5. Instead of doing `from nltk.book import *`

```
>>> from nltk import Text
>>> text1 = Text(gb.words('melville-moby_dick.txt'))
```

6. *Review.* How do we do the following?

a. Get the number of tokens.

b. Get the wordlist in alphabetic order.

c. Get the word frequency distribution.

d. Get the relative frequency of word w.

e. Get the most-frequent n words.

f. Make a concordance of word w.

7. Categories

a. How do I get a genre of the Brown corpus?

```
from nltk.corpus import brown
brown.words(categories='news')
```

b. What if I forget what the genres are?

```
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', ..., 'science_fiction']
```

c. Making one big text out of multiple selected genres

```
brown.words(categories=['news','editorial'])
```

d. Files in a category: `brown.fileids(categories='romance')`

e. Categories for a file: `brown.categories(fileids='cp27')`

f. Not all corpora have categories: `gutenberg.categories()` – error

g. Reuters corpus has overlapping categories

```
>>> reuters.categories('training/9865')
['barley', 'corn', 'grain', 'wheat']
>>> reuters.fileids(categories='corn')
['test/14832', 'test/14858', 'test/15033', ..., 'training/9989']
```

h. Temporal information in filename:

```
>>> [fid[:4] for fid in inaugural.fileids()]
['1789', '1793', '1797', '1801', '1805', '1809', ..., '2009']
```

8. Corpora discussed in the chapter

Corpus	Files	FileIDs	Categories
`gutenberg`	books	*author*-*titleword*`.txt`	–
`webtext`	misc from web	*titleword*`.txt`	–
`brown`	representative corpus	`c` *genre no*	genres
`reuters`	newswire articles	`test/`*no*, `training/`*no*	topics
`inaugural`	inaugural addresses	*year*-*pres*`.txt`	–
`cess_esp`	spanish	*code*`.tbf`	–
`floresta`	portuguese	[only one file]	–
`indian`	one file per language	*lang*`.pos`	–
`udhr`	one file per language	*lang*-*charenc*	–

Lexical resources

9. English wordlist

a. `nltk.corpus.words`

```
>>> from nltk.corpus import words as wordlist
>>> wordlist.words()
['A', 'a', 'aa', 'aal', 'aalii', 'aam', 'Aani', 'aardvark', ...]
```

b. Example:

```
>>> eng_vocab = set(w.lower() for w in wordlist.words())
```

c. Finding words that are not in the dictionary

```
>>> [w for w in set(text1) if w.lower() not in eng_vocab]
['woods', 'clotted', 'plaudits', 'marching', 'disobeying', 'canes', ...]
```

d. How do we strip the endings? → *stemmers*

10. Other wordlists

a. Recall the highest-frequency pairs in *Moby Dick*: `(',', 'and')`, `('of', 'the')`, `...` We'd like to get rid of the "empty" words.

b. Stopwords

```
>>> from nltk.corpus import stopwords
>>> stopwords.fileids()
['danish', 'dutch', 'english', 'finnish', 'french', 'german', ...]
>>> stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', ..., 'now']
```

c. Names

```
>>> from nltk.corpus import names
>>> names.fileids()
['female.txt', 'male.txt']
>>> names.words('female.txt')
['Abagael', 'Abagail', 'Abbe', 'Abbey', 'Abbi', 'Abbie', ...]
```

11. CMU pronunciation dictionary

a.

```
>>> from nltk.corpus import cmudict
>>> cmudict.words()
['a', 'a.', 'a', 'a42128', 'aaa', 'aaberg', ...]
```

b. New method: `entries()`

```
>>> cmudict.entries()[:5]
[('a', ['AH0']), ('a.', ['EY1']), ('a', ['EY1']), ('a42128', ['EY1',
'F', 'AO1', 'R', 'T', 'UW1', 'W', 'AH1', 'N', 'T', 'UW1', 'EY1',
'T']), ('aaa', ['T', 'R', 'IH2', 'P', 'AH0', 'L', 'EY1'])]
```

c. Each entry is a pair `(word, pronunc)`

```
>>> for (word, pronunc) in cmudict.entries():
...     if word.endswith('tale'):
...         print(word, pronunc)
...
artale ['AA0', 'R', 'T', 'AA1', 'L', 'IY0']
denatale ['D', 'IH0', 'N', 'AA0', 'T', 'AA1', 'L', 'IY0']
...
vitale ['V', 'IH0', 'T', 'AE1', 'L', 'IY0']
vitale ['V', 'AY2', 'T', 'AE1', 'L']
```

d. The numbers represent stress: $1 > 2 > 0$

```
aaa       ['T', 'R', 'IH2', 'P', 'AH0', 'L', 'EY1']
fairytale ['F', 'EH1', 'R', 'IY0', 'T', 'EY2', 'L']
```

12. Using a dict for look-up

a. Create from a list of pairs

```
>>> pron_dict = dict(cmudict.entries())
>>> pron_dict['apple']
['AE1', 'P', 'AH0', 'L']
```

b. `cmudict.dict()` – values are *lists* of pronunciations

13. Example: rhymes with

a. Words that rhyme with "cat":

```
>>> pron_dict['cat']
['K', 'AE1', 'T']
>>> [w for (w,p) in cmudict.entries()
...    if len(p) > 2 and p[-2:] == ['AE1', 'T']]
['arnatt', 'at-bat', 'balyeat', 'bat', 'batt', 'batte', 'begat', ...]
```

b. The "rhyming suffix" of a word starts with the (last) stressed vowel

```
>>> pron = ['T', 'AH1', 'M', 'K', 'AE1', 'T']
>>> list(enumerate(pron))
[(0, 'T'), (1, 'AH1'), (2, 'M'), (3, 'K'), (4, 'AE1'), (5, 'T')]
>>> [i for (i,ph) in enumerate(pron) if '1' in ph]
[1,4]
>>> _[-1]
4
>>> i = _
>>> suf = pron[i:]
>>> suf
['AE1', 'T']
```

c. Find words that end with the rhyming suffix

```
>>> [w for (w,p) in cmudict.entries() if p[-len(suf):] == suf]
['arnatt', 'at', 'at-bat', 'balyeat', 'bat', 'batt', ..., 'vat']
```

14. Swadesh lists

a. 100 common words in multiple languages

```
>>> from nltk.corpus import swadesh
>>> swadesh.fileids()
['be', 'bg', 'bs', 'ca', 'cs', 'cu', 'de', 'en', 'es', 'fr', 'hr',
'it', 'la', 'mk', 'nl', 'pl', 'pt', 'ro', 'ru', 'sk', 'sl', 'sr',
'sw', 'uk']
```

b. Each entry is a tuple corresponding to one meaning

```
>>> swadesh.entries()
[('\xd1\x8f', '\xd0\xb0\xd0\xb7', 'ja', ..., 'I', ...),
 ('\xd1\x82\xd1\x8b', '\xd1\x82\xd0\xb8', 'ti', ..., 'you (singular), thou', .
 ...]
```

c. Can select just some languages

```
>>> swadesh.entries(['en', 'de'])[:5]
[('I', 'ich'), ('you (singular), thou', 'du, Sie'), ('he', 'er'), ...]
```

d. Make a table for translation

```
>>> trans = dict(swadesh.entries(['en', 'de']))
>>> trans['dog']
'Hund'
```

Wordnet

15. Words, synsets, lemmas

a. Each word has some number of "meanings," called **synsets**

```
>>> from nltk.corpus import wordnet as wn
>>> horse = wn.synsets('horse')
>>> horse
[Synset('horse.n.01'), Synset('horse.n.02'), Synset('cavalry.n.01'),
Synset('sawhorse.n.01'), Synset('knight.n.02'), Synset('horse.v.01')]
```

b. Meanings represented indirectly by the **set** of **synonyms** that share the meaning

```
>>> horse[0].lemma_names()
['horse', 'Equus_caballus']
>>> horse[1].lemma_names()
['horse', 'gymnastic_horse']
>>> horse[4].lemma_names()
['knight', 'horse']
```

c. Synonyms share a synset

```
>>> knight = wn.synsets('knight')
>>> [s.lemma_names() for s in knight]
[['knight'], ['knight', 'horse'], ['knight', 'dub']]
>>> knight[1] == horse[4]
True
```

d. Each synset has a unique part of speech

```
>>> horse[0].pos()
'n'
>>> horse[5].pos()
'v'
```

e. Restricting synsets to a pos

```
>>> wn.synsets('horse', 'v')
[Synset('horse.v.01')]
```

16. Relations between synsets

a. Hypernyms and hyponyms

```
>>> horse[0].hypernyms()
[Synset('equine.n.01')]
>>> horse[0].hyponyms()
[Synset('hack.n.07'), Synset('stalking-horse.n.04'),
Synset('saddle_horse.n.01'), Synset('pacer.n.02'), ...]
```

b. Part meroynyms and holonyms

```
>>> cell = wn.synsets('cell', 'n')
>>> cell[1].part_meronyms()
[Synset('cytoplasm.n.01'), Synset('nucleus.n.01'), Synset('organelle.n.01'),
Synset('cell_membrane.n.01'), Synset('energid.n.01'), Synset('vacuole.n.01')]
>>> _[0].part_holonyms()
[Synset('cell.n.02')]
```

c. Substance meronyms and holonyms

```
>>> animal = wn.synsets('animal', 'n')
>>> animal[0].substance_meronyms()
[Synset('animal_tissue.n.01')]
```

d. Member meronyms and holonyms

```
>>> oscines = wn.synsets('oscines','n')
>>> oscines[0].member_meronyms()
[Synset('xenicidae.n.01'), Synset('hirundinidae.n.01'), Synset('oscine.n.01'),
```

e. Entailments

```
>>> snore = wn.synsets('snore', 'v')
>>> snore[0].entailments()
[Synset('sleep.v.01')]
```

f. Attribute

```
>>> fast = wn.synsets('fast', 'a')
>>> fast[0].attributes()
[Synset('speed.n.02')]
```

17. Lemmas

a. A lemma is the pairing of a word and a synset: it is a **word sense**

```
>>> horse_senses = wn.lemmas('horse')
>>> horse_senses[4]
Lemma('knight.n.02.horse')
>>> horse_senses[4].name()
'horse'
>>> horse_senses[4].synset()
Synset('knight.n.02')
```

b. From synset to lemmas expressing it

```
>>> horse[4].lemmas()
[Lemma('knight.n.02.knight'), Lemma('knight.n.02.horse')]
```

c. Antonym relation—applies to a *lemma*

```
>>> fast[0].lemmas()[0]
Lemma('fast.a.01.fast')
>>> fast[0].lemmas()[0].antonyms()
[Lemma('slow.a.01.slow')]
```

18. Iterating over all (these are generators):

a. `wn.all_synsets()`

b. `wn.all_synsets('n')`

c. `wn.all_lemma_names()`

d. `wn.all_lemma_names('n')`