NLTK Corpora

Ling 250/450: Data Science for Linguistics

C.M. Downey

Spring 2025

Review: Corpora

- A **corpus** (plural: corpora) is a (large) **body of language data**
- Corpora come in a **variety of forms**:
  - Digital/digitized or printed **text**
  - Recorded **speech**
  - **Transcribed** speech (printed or digital)
- In this class, we will almost always be working with **digital text**
- **Corpus Linguistics**: answering linguistic questions via **empirical analysis of language corpora**

NLTK: Natural Language Tool-Kit

- An open-source **Python library** for working with human (natural) language
  - "Open-source": the code is **freely available** and maintained by a community
  - Excellent for **exploratory text analysis**
  - Can be **integrated** with custom Python programs
- Provides a **wide range of functionalities** for working with text
  - Access to **existing corpora**, as well as the ability to **create custom corpora**
  - Tools for **discovering and visualizing probabilistic patterns** in text
  - Tools for **segmenting, tagging, parsing, and classifying** text

nltk.corpus

- **`nltk.corpus`** is the primary module for working with corpora
- Example: the **Gutenberg Corpus** can be accessed at `nltk.corpus.gutenberg`
  - Consists of **literature** that is in the **public domain**
  - **Component files** shown with `.fileids()`

```
>>> 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', 'cheste
rton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt
', 'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespear
e-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt']
```

Importing sub-modules

- Note that we can save ourselves typing by using Python's **`from X import Y`** syntax

```
>>> from nltk import corpus
>>> corpus.gutenberg.fileids()[:4]
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt']
>>>
>>> from nltk.corpus import gutenberg
>>> gutenberg.fileids()[:4]
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt']
```

NLTK corpus "cheatsheet"

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

Importing a text file as a corpus

directory

file name

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> nightvale_corpus = PlaintextCorpusReader('CourseCode/ling250', 'Night_Vale.txt')
>>> nightvale_corpus.words()
['A', 'friendly', 'desert', 'community', 'where', ...]
```

can use a RegEx to
load more than one file

```
>>> nightvale_sherlock = PlaintextCorpusReader('CourseCode/ling250', '.*.txt')
>>> nightvale_sherlock.fileids()
['Night_Vale.txt', 'sherlock.txt']
```

nltk.Text

- NLTK defines an object class called **Text**, which enables **extra functionality** when constructed from a list of words
  - Considered a **"wrapper class"**: it is still essentially a list of words, but with **additional features**
  - Can be constructed from **`corpus_name.words()`** or any other **list of strings**

```
>>> toy_corpus = ['This', 'is', 'a', 'toy', 'corpus']
>>> toy_text = nltk.Text(toy_corpus)
>>> toy_text
<Text: This is a toy corpus...>
>>>
>>> nightvale_text = nltk.Text(nightvale_corpus.words())
>>> nightvale_text
<Text: A friendly desert community where the sun is...>
```

Text.concordance()

```
>>> nightvale_text.concordance('desert')
Displaying 25 of 80 matches:
                        A friendly desert community where the sun is hot , th
the work of their bitter rivals the Desert Bluffs Cacti . Desert Bluffs is alw
er rivals the Desert Bluffs Cacti . Desert Bluffs is always trying to show us
eral minutes at least . For shame , Desert Bluffs . For shame . That new scien
t , given we are in the middle of a desert , there is no actual water at the w
he houses in the new development of Desert Creek , out back of the elementary
y to forget in this hot , hot , hot desert climate , but things would actually
 even , yes , extreme heat that our desert community is gifted with . The City
shopping at the Ralph ' s or at the Desert Flower Bowling Alley and Arcade Fun
 perfect symmetry . Speaking of the Desert Flower Bowling Alley and Arcade Fun
night , listeners . Goodnight . The desert seems vast , even endless , and yet
ng Fear will go next — hopefully to Desert Bluffs . It would serve them right
e . But really , as long as we beat Desert Bluffs , fans and Hooded Figures al
tions , and he ' s ready to take on Desert Bluffs , which is probably the wors
 , unlike that hideous sports arena Desert Bluffs built last spring . Desert B
a Desert Bluffs built last spring . Desert Bluffs can ' t do anything right .
her part of a large and featureless desert . I think we can all agree , though
ven as large and featureless as the desert was , the part that would eventuall
e part that would eventually become Desert Bluffs was still probably awful and
gh won the grudge match against the Desert Bluffs Vultures last night . Two —
building waterfront facilities in a desert are fabrications of our collective
the Waterfront buildings out in the desert exactly where you remembered them ,
ll season . Representatives for the Desert Bluffs School District , speaking i
ther . Teddy Williams , over at the Desert Flower Bowling Alley and Arcade Fun
of wings . Mmm , nothing like those Desert Flower wings ! Let me leave you wit
```

Text.collocations()

- Gives the **most common pairs of words** (excluding "stop words")

```
>>> nightvale_text.collocations()
Night Vale; Secret Police; City Council; Old Woman; Dog Park; Glow
Cloud; Stay tuned; Apache Tracker; Tan Jacket; tuned next; John
Peters; Big Rico; Woman Josie; Desert Bluffs; Blinking Light; Faceless
Old; Hooded Figures; press conference; Bowling Alley; Brown Stone
```

Text.common_contexts()

- Gives **any shared contexts** in which all inputted words occur

```
>>> nightvale_text.common_contexts(['red', 'blue'])
,_dots
>>>
>>> nightvale_text.common_contexts(['easy', 'hard'])
s_to be_. is_to
```

Text.findall()

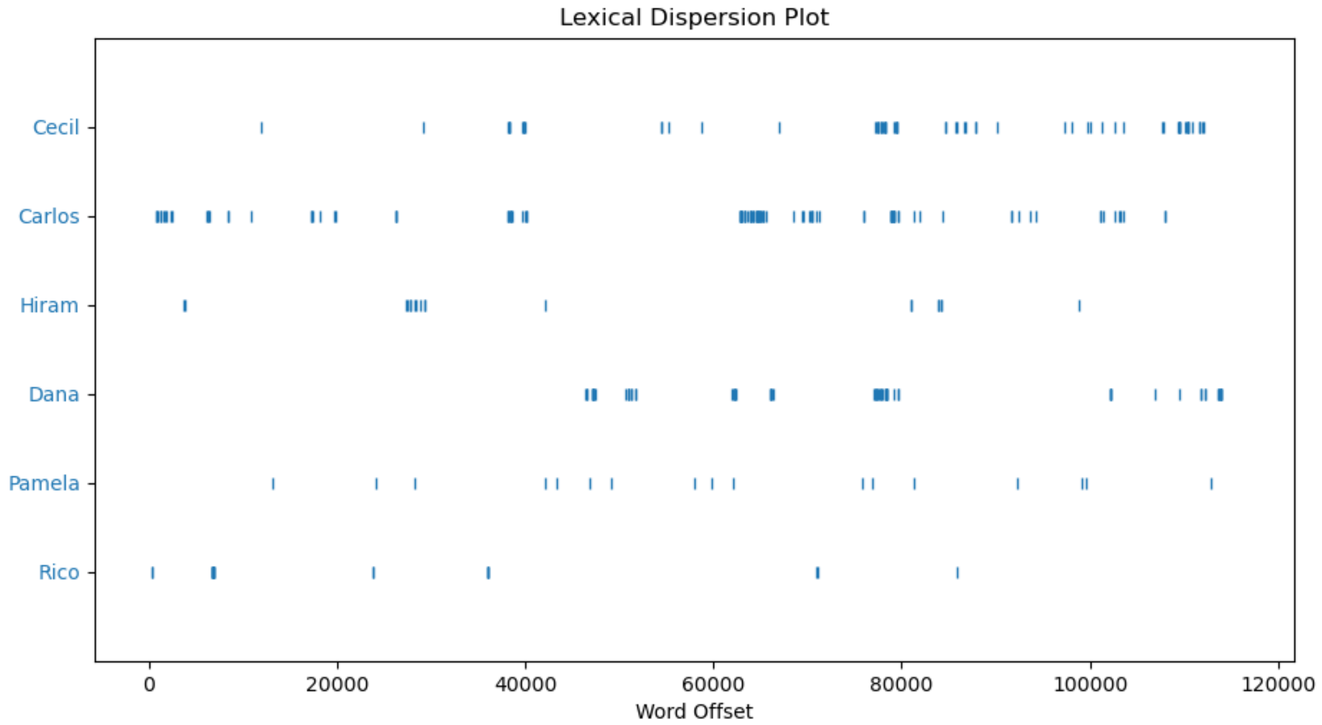
- Searches for all matches of a **regular expression**, where separate **tokens** are indicated with **<angle brackets>**

```
>>> nightvale_text.findall(r'<a><\w+><woman>')
a new woman
>>> nightvale_text.findall(r'<a><\w+><man>')
a running man; a running man; a running man; a twelfth man; a
different man; a grotesque man; a good man; a good man; a good man; a
heavyset man; a grown man; a grown man; a dying man; a good man
```

Text.dispersion_plot()

- Creates a plot showing the **dispersion of certain tokens** within the text

```
>>> import matplotlib.pyplot as plt
>>> nightvale_text.dispersion_plot(['Cecil', 'Carlos', 'Hiram', 'Dana', 'Pamela', 'Rico'])
>>> plt.show()
```



nltk.FreqDist

- Short for **"Frequency Distribution"**; essentially takes in a **list of tokens** and **counts them**

```
>>> fd = nltk.FreqDist(nightvale_corpus.words())
>>> fd.tabulate(samples=['cat', 'dog', 'bird', 'ferret', 'rat'])
   cat    dog   bird ferret    rat
    18     16     10      0      0
```

***Table 3.1**:*

Functions Defined for NLTK's Frequency Distributions

Example	Description
`fdist = FreqDist(samples)`	create a frequency distribution containing the given samples
`fdist[sample] += 1`	increment the count for this sample
`fdist['monstrous']`	count of the number of times a given sample occurred
`fdist.freq('monstrous')`	frequency of a given sample
`fdist.N()`	total number of samples
`fdist.most_common(n)`	the `n` most common samples and their frequencies
`for sample in fdist:`	iterate over the samples
`fdist.max()`	sample with the greatest count
`fdist.tabulate()`	tabulate the frequency distribution
`fdist.plot()`	graphical plot of the frequency distribution
`fdist.plot(cumulative=True)`	cumulative plot of the frequency distribution
`fdist1 |= fdist2`	update `fdist1` with counts from `fdist2`
`fdist1 < fdist2`	test if samples in `fdist1` occur less frequently than in `fdist2`

Our discussion of frequency distributions has introduced some important Python concepts, and we will look at them systematically in 4.

nltk.ConditionalFreqDist

- Counts tokens **within each condition;** for example: categories in the Brown Corpus

```
>>> from nltk.corpus import brown
>>> words_by_genre = [(genre, word) for genre in brown.categories() for word in brown.words(categories=[genre])]
>>> words_by_genre[:8]
[('adventure', 'Dan'), ('adventure', 'Morgan'), ('adventure', 'told'), ('adventure', 'himself'), ('adventure', 'he'), ('adventu
re', 'would'), ('adventure', 'forget'), ('adventure', 'Ann')]
>>> cfd = nltk.ConditionalFreqDist(words_by_genre)
>>> cfd.tabulate(samples=['cat', 'dog', 'bird', 'ferret', 'rat'])
                  cat    dog   bird ferret    rat
      adventure     2      2      1      0      1
 belles_lettres     3      8      2      1      1
      editorial     1      7      4      0      0
        fiction     1      7      7      0      0
     government     0      0      0      0      0
        hobbies     0     19      2      0      0
          humor     1      2      0      0      0
        learned     5      8      3      0      2
           lore     1      1      1      0      1
        mystery     5      0      0      0      0
           news     0      7      2      0      0
       religion     0      0      0      0      0
        reviews     0      1      1      0      1
        romance     3      8      2      0      0
science_fiction     0      0      0      0      0
```

CFD "cheatsheet"

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`