The Natural Language Toolkit (NLTK)

NLTK

Texts

Distributions

New data

Built-In Corpora

Getting Started

Download the materials from the NLTK book (if you have not done so already):

```python
>>> import nltk
>>> nltk.download()
```

Assuming that book material have been downloaded, for today do:

```python
>>> import nltk
>>> from nltk.book import *
```

This last command loads various texts to work with.

Searching Text

We now have texts available:

```python
>>> text1
<Text: Moby Dick by Herman Melville 1851>
```

Methods:

- `concordance`
  ```python
text1.concordance("monstrous")
  ```
- `similar`
  ```python
text1.similar("monstrous")
text2.similar("monstrous")
  ```
- `common_contexts`
  ```python
text2.common_contexts(["monstrous", "very"])
  ```

Texts as Lists of Words

NLTK treats texts as lists of words

Here are the first 20 words of *Moby Dick*:

```python
>>> text1[:20]
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(',
'Supplied', 'by', 'a', 'Late', 'Consumptive', 'Usher', 'to', 'a', 'Grammar']
```

Counting Vocabulary

Because it's Python-based, it's easy to create functions to analyze the texts

```python
>>> def lexical_diversity(text):
...         return len(text) / len(set(text))
... >>> lexical_diversity(text1)
13.502044830977896
>>> lexical_diversity(text2)
20.71949729255086
```

Note: `set()` converts a list to a set

- More on sets and functions later this semester...
The Natural Language Toolkit (NLTK)

**Simple Statistics**

**Frequency Distributions**

NLTK has pre-built packages for creating distributions

```python
>>> fdist1 = FreqDist(text1)
```

```python
>>> fdist1
<FreqDist with 19317 samples and 260819 outcomes>
```

```python
>>> fdist1['whale']
906
```

We will build our own dictionaries, but some capabilities are quickly calculated with `FreqDist()`:

```python
>>> fdist1.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024), ('a', 4569), ('to', 4542), (';', 4072), ...
```

**Organizing by word length**

```python
>>> fdist = FreqDist([len(w) for w in text1])
```

```python
>>> fdist
<FreqDist with 19 samples and 260819 outcomes>
```

```python
>>> list(fdist.keys())
[3, 1, 4, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
```

```python
>>> list(fdist.items())
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), ...
```

```python
>>> fdist.max()
3
```

```python
>>> fdist[3]
50223
```

```python
>>> fdist.freq(3)
0.19255882431878046
```

**Notes on previous slide**

- To create an empty frequency distribution, use `FreqDist()` with no arguments
- To add to item frequencies, use the `+=` notation
  - Access each word's frequency using square brackets
  - We'll see this type of notation with dictionaries
- If you have only done `import nltk` (and not from `nltk_book` `import *`), you need to use `nltk.FreqDist()` (and not just `FreqDist()`)

This type of procedure is useful when creating distributions as you iterate over new texts

**Functions**

- **bigrams**
  ```python
  >>> list(bigrams(text1[:10]))
  [['', 'Moby'], ('Moby', 'Dick'), ('Dick', 'by'), ('by', 'Herman'), ('Herman', 'Melville'), ('Melville', '1851'), ('1851', ''), ('', 'ETYMOLOGY'), ('ETYMOLOGY', '')]
  ```
  - **collocations**

```python
>>> text1.collocations()
Building collocations list
```

Building collocations list
Sperm Whale; Moby Dick; White Whale; old man; Captain Ahab; sperm whale; Right Whale; Captain Peleg; New Bedford; Cape Horn; cried Ahab; years ago; lower jaw; never mind; Father Mapple; cried Stubbs; chief mate; white whale; ivory leg; one hand
#7. Find the collocations in text5.

#18. Using list addition, and the set and sorted operations, compute the vocabulary of the sentences sent1 ... sent8. Which one will give a larger value? Will this be the case for other texts?

```python
>>> sorted(set([w.lower() for w in text1]))
```

```python
>>> sorted([w.lower() for w in set(text1)])
```

#19. What is the difference between the following two lines? Which one will give a larger value? Will this be the case for other texts?

```python
>>> sorted(set([w.lower() for w in text1]))
>>> sorted([w.lower() for w in set(text1)])
```

#22. Find all the four-letter words in the Chat Corpus (text5). With the help of a frequency distribution (FreqDist), show these words in decreasing order of frequency.

```python
>>> nltk.Text(tokens).collocations()
```

Creating an NLTK text

```python
>>> text = nltk.Text(tokens)
>>> type(text)
<class 'nltk.text.Text'>
```

Stemming (prep)

```python
>>> raw = """DENNIS: Listen, strange women lying in ponds... distributing swords is no basis for a system of government. Supreme executive power derives from... a mandate from the masses, not from some farcical aquatic ceremony.""
>>> tokens = word_tokenize(raw)
```
There are options for normalizing words, as well:

```python
>>> porter = nltk.PorterStemmer()
>>> lancaster = nltk.LancasterStemmer()
>>> [porter.stem(t) for t in tokens]
['DENNI', ':', 'Listen', ',', 'strang', 'women', 'lie', ...]
>>> [lancaster.stem(t) for t in tokens]
['den', ':', 'list', ',', 'strange', 'wom', 'lying', ...]
```

It gets better if you pass in a part-of-speech!

**Exercises (2)**

#18. Read in some text from a corpus, tokenize it, and print the list of all \(\text{wh}\)-word types that occur. (\(\text{wh}\)-words in English are used in questions, relative clauses and exclamations: who, which, what, and so on.) Print them in order. Are any words duplicated in this list, because of the presence of case distinctions or punctuation?

#30. Use the Porter Stemmer to normalize some tokenized text, calling the stemmer on each word. Do the same thing with the Lancaster Stemmer and see if you observe any differences.

**Brown Corpus**

```python
>>> from nltk.corpus import brown
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', ...]
```

Note that these categories correspond to sections (e.g., section f (or cf) is "lore")

- See section 1.3 of chapter 2 of the NLTK book

**Built-In Corpora**

NLTK has a variety of built-in corpora, which allow you to work with different kinds of (somewhat standard) data:

- Gutenberg Corpus
- Web and Chat Text
- Brown Corpus
- Reuters Corpus
- Inaugural Address Corpus

There are functions to load in your own corpus

- Note: a corpus typically has sub-structure & meta-data, whereas a text is simply a text

(http://www.nltk.org/book/ch02.html)