The Natural Language Toolkit (NLTK)

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"])
  ```

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))
...     ...
```

```python
>>> lexical_diversity(text1)
13.502044830977896
```

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

- More on sets and functions later this semester
NLTK has pre-built packages for creating distributions

```python
>>> fdist1 = FreqDist(text1)
>>> fdist1
<FreqDist with 19317 samples and 260819 outcomes>
```

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])
>>> fdist
<FreqDist with 19 samples and 260819 outcomes>
```

```python
>>> fdist.max()
3
>>> fdist[3]
50223
>>> 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
#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?

>>> 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.

Creating an NLTK text

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

Stemming (prep)

>>> 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)

More on regular expressions in a couple weeks ...
Stemming

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', ...]
```

(Let's get 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 wh-word types that occur. (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

Lemmatization

```python
>>> wn1 = nltk.WordNetLemmatizer()
>>> [wn1.lemmatize(t) for t in tokens]
['DENNIS', ':', 'Listen', ',', 'strange', 'woman', 'lying', 'in', 'pond', 'distributing', 'sword', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government', '.', ...]
```

(Brown Corpus)

Data access

Access words (with .words()) in various ways:

```python
>>> brown.words(categories='news')
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
>>> brown.words(fileids=['cg22'])
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
>>> brown.words(fileids=['cm01','cm02'])
['Now', 'that', 'he', 'knew', 'himself', 'to', 'be', ...]
```

Access sentences in the same ways but with .sents(), e.g.:

```python
>>> brown.sents(categories=['news', 'editorial', 'reviews'])
[['The', 'Fulton', 'County'...], ['The', 'jury', 'further'...]
```