Lecture 24

408/508 *Computational Techniques for Linguists*

Last Time

- Started playing with nltk
- Python list comprehensions
  - conditional form:

```
[word for word in alice if len(word) == 14]
 alice2 = [word for word in alice if word != ',' and word !=
'.' and word != '?']
```

Today's Topics

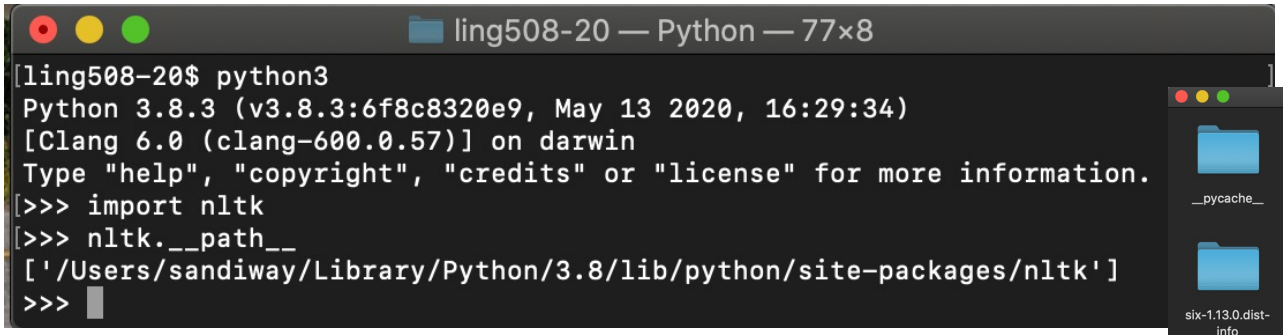
- Homework 11
- More cool stuff with `nltk` today …
  1. `nltk.word_tokenize(`*`string`*`)`
  2. `nltk.pos_tag(`*`list`*`)`
  3. `treebank.parsed_sents() and .draw()`
  4. `nltk.chunk.ne_chunk(`*`tuples`*`)`
  5. *`text`*`.concordance(`*`word`*`)`
  6. *`text`*`.similar(`*`word`*`)`
  7. *`text`*`.common_contexts(`*`list`*`)`
  8. *`text`*`.dispersion_plot()`

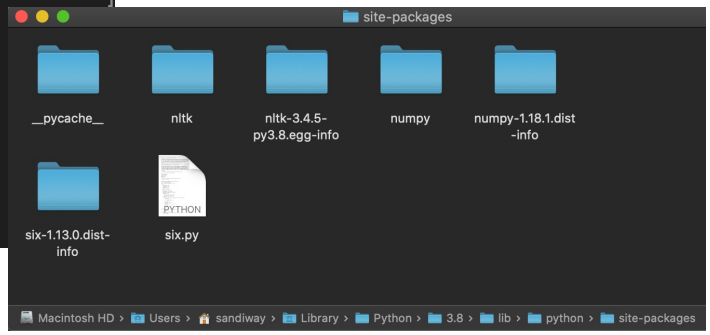
nltk

- Where is it installed on my computer?

```
(base) ~$ python
Python 3.9.12 (main, Jun  1 2022, 06:34:44)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.__path__
['/Users/sandiway/opt/anaconda3/lib/python3.9/site-packages/nltk']
>>>
```

Anaconda distribution comes with over 250 packages automatically installed





Tokenization: `nltk.word_tokenize()`

Recall:

```
>>> text = 'Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do. Once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?"\nSo she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her. '
>>> text.split()
['Alice', 'was', 'beginning', 'to', 'get', 'very', 'tired', 'of',
'sitting', 'by', 'her', 'sister', 'on', 'the', 'bank,', 'and', 'of',
'having', 'nothing', 'to', 'do.', 'Once', 'or', 'twice', 'she', 'had',
'peeped', 'into', 'the', 'book', 'her', 'sister', 'was', 'reading,',
'but', 'it', 'had', 'no', 'pictures', 'or', 'conversations', 'in',
'it,', '"and', 'what', 'is', 'the', 'use', 'of', 'a', 'book,"',
'thought', 'Alice,', '"without', 'pictures', 'or', 'conversations?"',
'So', 'she', 'was', 'considering', 'in', 'her', 'own', 'mind', '(as',
'well', 'as', 'she', 'could,', 'for', 'the', 'hot', 'day', 'made',
'her', 'feel', 'very', 'sleepy', 'and', 'stupid),', 'whether', 'the',
'pleasure', 'of', 'making', 'a', 'daisy-chain', 'would', 'be',
'worth', 'the', 'trouble', 'of', 'getting', 'up', 'and', 'picking',
'the', 'daisies,', 'when', 'suddenly', 'a', 'White', 'Rabbit', 'with',
'pink', 'eyes', 'ran', 'close', 'by', 'her.']
```

- Compare (punctuation are words):

```
>>> nltk.word_tokenize(text)
['Alice', 'was', 'beginning', 'to', 'get', 'very', 'tired', 'of',
'sitting', 'by', 'her', 'sister', 'on', 'the', 'bank', ',', 'and',
'of', 'having', 'nothing', 'to', 'do', '.', 'Once', 'or', 'twice',
'she', 'had', 'peeped', 'into', 'the', 'book', 'her', 'sister', 'was',
'reading', ',', 'but', 'it', 'had', 'no', 'pictures', 'or',
'conversations', 'in', 'it', ',', '``', 'and', 'what', 'is', 'the',
'use', 'of', 'a', 'book', ',', "''", 'thought', 'Alice', ',', '``',
'without', 'pictures', 'or', 'conversations', '?', "''", 'So', 'she',
'was', 'considering', 'in', 'her', 'own', 'mind', '(', 'as', 'well',
'as', 'she', 'could', ',', 'for', 'the', 'hot', 'day', 'made', 'her',
'feel', 'very', 'sleepy', 'and', 'stupid', ')', ',', 'whether', 'the',
'pleasure', 'of', 'making', 'a', 'daisy-chain', 'would', 'be', 'worth',
'the', 'trouble', 'of', 'getting', 'up', 'and', 'picking', 'the',
'daisies', ',', 'when', 'suddenly', 'a', 'White', 'Rabbit', 'with',
'pink', 'eyes', 'ran', 'close', 'by', 'her', '.']
>>> len(nltk.word_tokenize(text))
129
```

```
>>> len(text.split())
112
```

Part of Speech Tagging: `nltk.pos_tag()`

- Once tokenized, we can apply POS tagging
- `>>> words = nltk.word_tokenize(text)`
- `>>> nltk.pos_tag(words)` produces list of tuples (word, tag)

- [('Alice', 'NNP'), ('was', 'VBD'), ('beginning', 'VBG'), ('to', 'TO'), ('get', 'VB'), ('very', 'RB'), ('tired', 'JJ'), ('of', 'IN'), ('sitting', 'VBG'), ('by', 'IN'), ('her', 'PRP$'), ('sister', 'NN'), ('on', 'IN'), ('the', 'DT'), ('bank', 'NN'), (',', ','), ('and', 'CC'), ('of', 'IN'), ('having', 'VBG'), ('nothing', 'NN'), ('to', 'TO'), ('do', 'VB'), ('.', '.'), ('Once', 'VB'), ('or', 'CC'), ('twice', 'VB'), ('she', 'PRP'), ('had', 'VBD'), ('peeped', 'VBN'), ('into', 'IN'), ('the', 'DT'), ('book', 'NN'), ('her', 'PRP$'), ('sister', 'NN'), ('was', 'VBD'), ('reading', 'VBG'), (',', ','), ('but', 'CC'), ('it', 'PRP'), ('had', 'VBD'), ('no', 'DT'), ('pictures', 'NNS'), ('or', 'CC'), ('conversations', 'NNS'), ('in', 'IN'), ('it', 'PRP'), (',', ','), ('``', '``'), ('and', 'CC'), ('what', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('use', 'NN'), ('of', 'IN'), ('a', 'DT'), ('book', 'NN'), (',', ','), ("''", "''"), ('thought', 'VBD'), ('Alice', 'NNP'), (',', ','), ('``', '``'), ('without', 'IN'), ('pictures', 'NNS'), ('or', 'CC'), ('conversations', 'NNS'), ('?', '.'), ("''", "''"), ('So', 'IN'), ('she', 'PRP'), ('was', 'VBD'), ('considering', 'VBG'), ('in', 'IN'), ('her', 'PRP$'), ('own', 'JJ'), ('mind', 'NN'), ('(', '('), ('as', 'RB'), ('well', 'RB'), ('as', 'IN'), ('she', 'PRP'), ('could', 'MD'), (',', ','), ('for', 'IN'), ('the', 'DT'), ('hot', 'JJ'), ('day', 'NN'), ('made', 'VBD'), ('her', 'PRP$'), ('feel', 'JJ'), ('very', 'RB'), ('sleepy', 'JJ'), ('and', 'CC'), ('stupid', 'JJ'), (')', ')'), (',', ','), ('whether', 'IN'), ('the', 'DT'), ('pleasure', 'NN'), ('of', 'IN'), ('making', 'VBG'), ('a', 'DT'), ('daisy-chain', 'NN'), ('would', 'MD'), ('be', 'VB'), ('worth', 'IN'), ('the', 'DT'), ('trouble', 'NN'), ('of', 'IN'), ('getting', 'VBG'), ('up', 'RP'), ('and', 'CC'), ('picking', 'VBG'), ('the', 'DT'), ('daisies', 'NNS'), (',', ','), ('when', 'WRB'), ('suddenly', 'RB'), ('a', 'DT'), ('White', 'NNP'), ('Rabbit', 'NN'), ('with', 'IN'), ('pink', 'JJ'), ('eyes', 'NNS'), ('ran', 'VBD'), ('close', 'RB'), ('by', 'IN'), ('her', 'PRP'), ('.', '.')]

Part of Speech Tagging: `nltk.pos_tag()`

- Output:

```
[('Alice', 'NNP'), ('was', 'VBD'),
('beginning', 'VBG'), ('to', 'TO'),
('get', 'VB'), ('very', 'RB'), ('tired',
'JJ'), ('of', 'IN'), ('sitting', 'VBG'),
('by', 'IN'), ('her', 'PRP$'), ('sister',
'NN'), ('on', 'IN'), ('the', 'DT'),
('bank', 'NN'), (',', ','), ('and', 'CC'),
('of', 'IN'), ('having', 'VBG'),
('nothing', 'NN'), ('to', 'TO'), ('do',
'VB'), ('.', '.')
```

…

Tagset (Penn Treebank):

- NN/NNS/NNP: common noun/NN plural/proper noun
- VB/VBD/VBG/VBZ/VBN: verb nonfinite form/past tense/gerund/3rd person singular present/ past participle
- PRP$: possessive pronoun
- DT: determiner
- IN: preposition
- JJ: adjective
- CC: coordinating conjunction
- TO: the word *to*

```
>>> nltk.help.upenn_tagset('RB')
RB: adverb
    occasionally unabatingly maddeningly adventurously
professedly
    stirringly prominently technologically magisterially
predominately
    swiftly fiscally pitilessly ...
```

Part of Speech Tagging: `nltk.pos_tag()`

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	+,%, &	WRB	wh-adverb	*how, where*

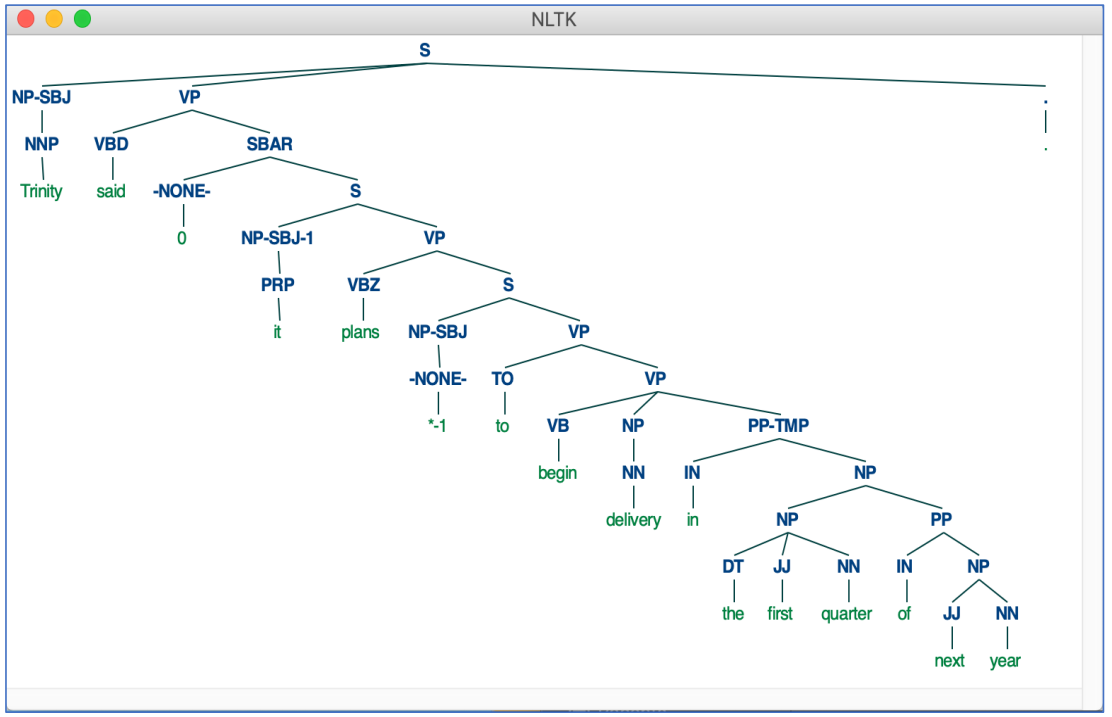
Figure 8.2 Penn Treebank part-of-speech tags.

from Jurafsky and Martin (*draft edition* 3)

Penn Treebank

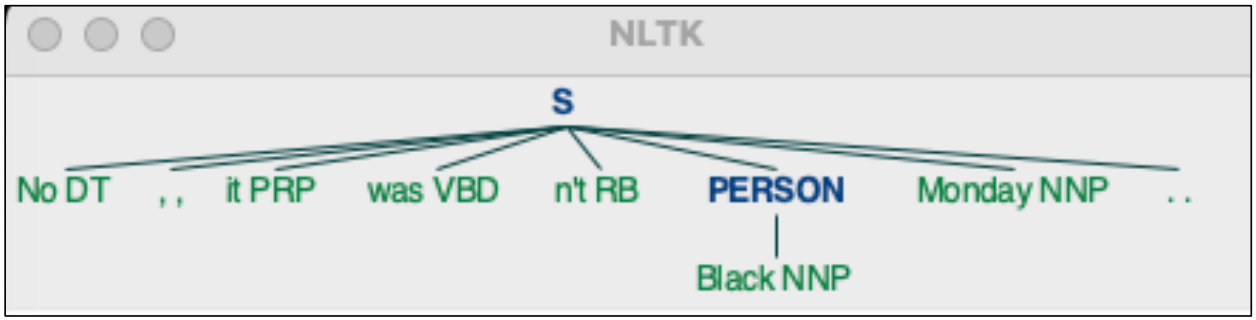
- There is a *sample* of the Penn Treebank Wall Street Journal (WSJ) corpus included
  - 3,914 parsed sentences out of
  - 49,000+ parsed sentences

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents()
>>> len(t)
3914
>>> t[-1].draw()
```



Named Entity (NE) chunking

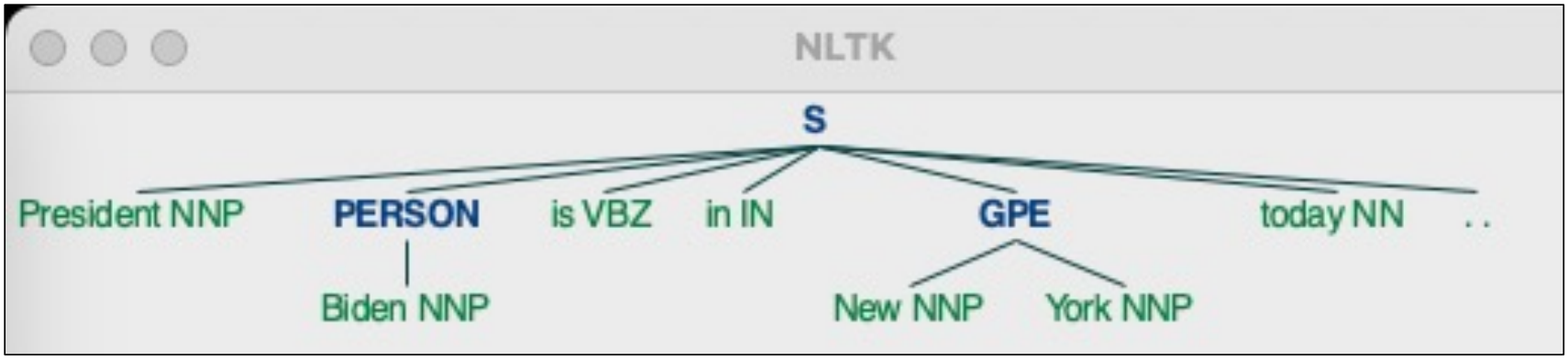
```
import nltk
tuples = nltk.pos_tag(nltk.word_tokenize("No, it wasn't Black Monday."))
tuples
[('No', 'DT'), (',', ','), ('it', 'PRP'), ('was', 'VBD'), ("n't", 'RB'),
('Black', 'NNP'), ('Monday', 'NNP'), ('.', '.')]
nltk.chunk.ne_chunk(tuples)
Tree('S', [('No', 'DT'), (',', ','), ('it', 'PRP'), ('was', 'VBD'),
("n't", 'RB'), Tree('PERSON', [('Black', 'NNP')]), ('Monday', 'NNP'),
('.', '.')])
>>> nltk.chunk.ne_chunk(tuples).draw()
```



Named Entity (NE) chunking

```
nltk.chunk.ne_chunk(nltk.pos_tag(nltk.word_tokenize("President Biden is in New York today.")))
```

```
Tree('S', [('President', 'NNP'), Tree('PERSON', [('Biden', 'NNP')]), ('is', 'VBZ'), ('in', 'IN'), Tree('GPE', [('New', 'NNP'), ('York', 'NNP')]), ('today', 'NN'), ('.', '.')]
```



nltk book: Language Processing and Python

1 Computing with Language: Texts and Words: http://www.nltk.org/book/ch01.html

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

```
>>> text1
<Text: Moby Dick by Herman Melville 1851>
>>> text2
<Text: Sense and Sensibility by Jane Austen 1811>
>>>
```

nltk book: Language Processing and Python

```
>>> text1.concordance("monstrous")
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
>>>
```

nltk book: Language Processing and Python

monstrous

```
>>> text1.concordance("monstrous")
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

contemptible

```
>>> text1.concordance("contemptible")
Displaying 4 of 4 matches:
, we shall find they will appear contemptible in the comparison . The whale is
 God ? Miserable man ! Oh ! most contemptible and worthy of all scorn ; with s
s these pig - fish are a noisy , contemptible set , mostly lurking in the mout
 harm , it may possibly be of no contemptible advantage ; considering that oil
```

```
>>> text1.similar("monstrous")
mean part maddens doleful gamesome subtly uncommon careful untoward
exasperate loving passing mouldy christian few true mystifying
imperial modifies contemptible
>>> text2.similar("monstrous")
very heartily so exceedingly remarkably as vast a great amazingly
extremely good sweet
>>>
```

nltk book: Language Processing and Python

- On my mac
  - your order may be different depending on dict implementation:

```
>>> text1.similar("monstrous")
true contemptible christian abundant few part mean careful puzzled
mystifying passing curious loving wise doleful gamesome singular
delightfully perilous fearless
>>> text2.similar("monstrous")
very so exceedingly heartily a as good great extremely remarkably
sweet vast amazingly
>>>
```

nltk book: Language Processing and Python

monstrous

```
>>> text1.concordance("monstrous")
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

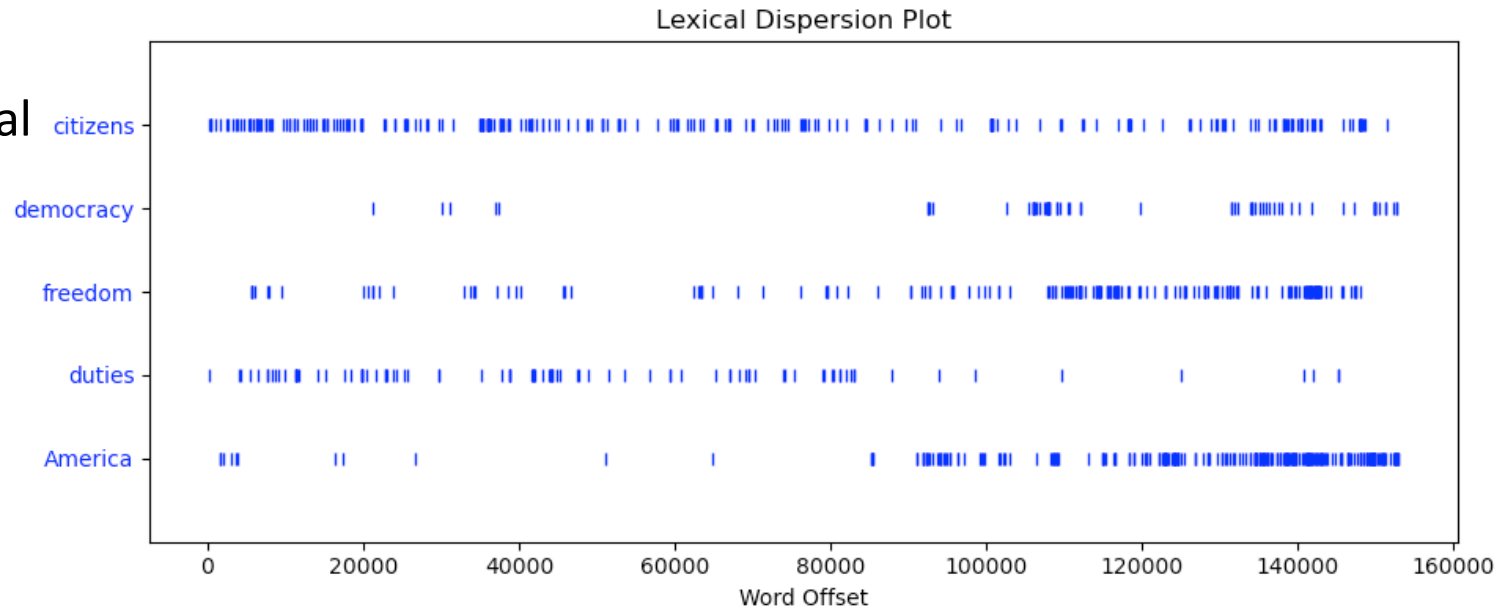
contemptible

```
>>> text1.concordance("contemptible")
Displaying 4 of 4 matches:
, we shall find they will appear contemptible in the comparison . The whale is
 God ? Miserable man ! Oh ! most contemptible and worthy of all scorn ; with s
s these pig - fish are a noisy , contemptible set , mostly lurking in the mout
 harm , it may possibly be of no contemptible advantage ; considering that oil
```

```
>>> text1.common_contexts(["monstrous","contemptible"])
most_and
>>>
```

nltk book: Language Processing and Python

Inaugural Presidential Addresses




```
text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])
```

nltk book: Language Processing and Python

- 1.4 Counting Vocabulary
  - `<text>` placeholder for some text object
  - `<word>` placeholder for a word

1.	`len(<text>)`	word count
2.	`set(<text>)`	no duplicate words
3.	`len(set(<text>))`	no. of different words
4.	`len(set(<text>)) / len(<text>)`	lexical diversity
5.	`<text>.count(<word>)`	# of times <word> occurs in <text>
6.	`100 * <text>.count(<word>) / len(<text>)`	% of <text> taken up by <word>

nltk book: Language Processing and Python

```
>>> def lex_diversity(text):
...     return len(set(text)) / len(text)
...
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

nltk book: Language Processing and Python

```
>>> for i in range(1,10):
...     name = "text" + str(i)
...     print(name, '{:.3f}'.format(lex_diversity(eval(name))))
...
text1 0.074
text2 0.048
text3 0.062
text4 0.066
text5 0.135
text6 0.128
text7 0.123
text8 0.228
text9 0.098
```

text2: Sense and Sensibility by Jane Austen 1811

text8: Personals Corpus

eval()

eval(*expression, globals=None, locals=None*)

The arguments are a string and optional globals and locals. If provided, *globals* must be a dictionary. If provided, *locals* can be any mapping object.

The *expression* argument is parsed and evaluated as a Python expression (technically speaking, a condition list) using the *globals* and *locals* dictionaries as global and local namespace. If the *globals* dictionary is present and does not contain a value for the key `__builtins__`, a reference to the dictionary of the built-in module `builtins` is inserted under that key before *expression* is parsed. That way you can control what builtins are available to the executed code by inserting your own `__builtins__` dictionary into *globals* before passing it to `eval()`. If the *locals* dictionary is omitted it defaults to the *globals* dictionary. If both dictionaries are omitted, the expression is executed with the *globals* and *locals* in the environment where `eval()` is called. Note, *eval()* does not have access to the nested scopes (non-locals) in the enclosing environment.

The return value is the result of the evaluated expression. Syntax errors are reported as exceptions. Example:

```
>>> x = 1
>>> eval('x+1')
2
```

Homework 11

- Term project proposal
- Email me
- Due Sunday midnight
- One paragraph sketch or page: what your project will be on
- Could be a HTML5 or a Webserver project
- nltk: exploratory work/experiment also fine