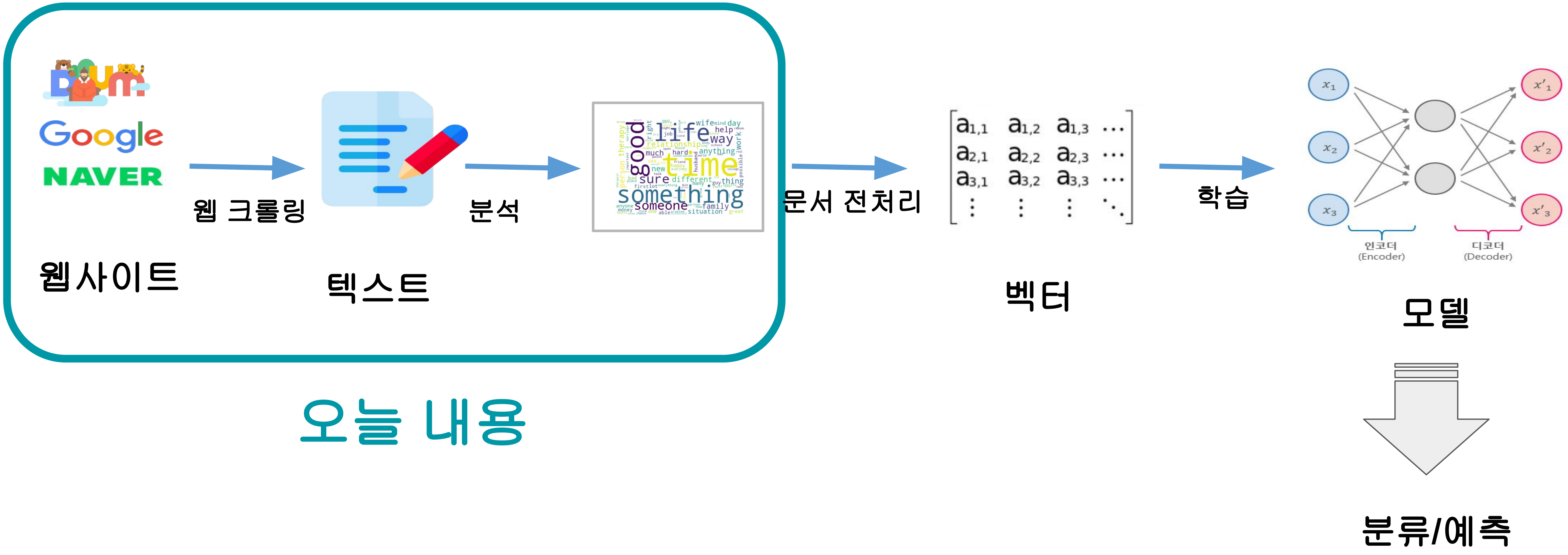
Natural Language Toolkit (NLTK)

성균관대학교
정윤경

한눈에 보는 자연어 처리 과정



웹 크롤링

분석

웹사이트

텍스트

문서 전처리

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

학습

벡터

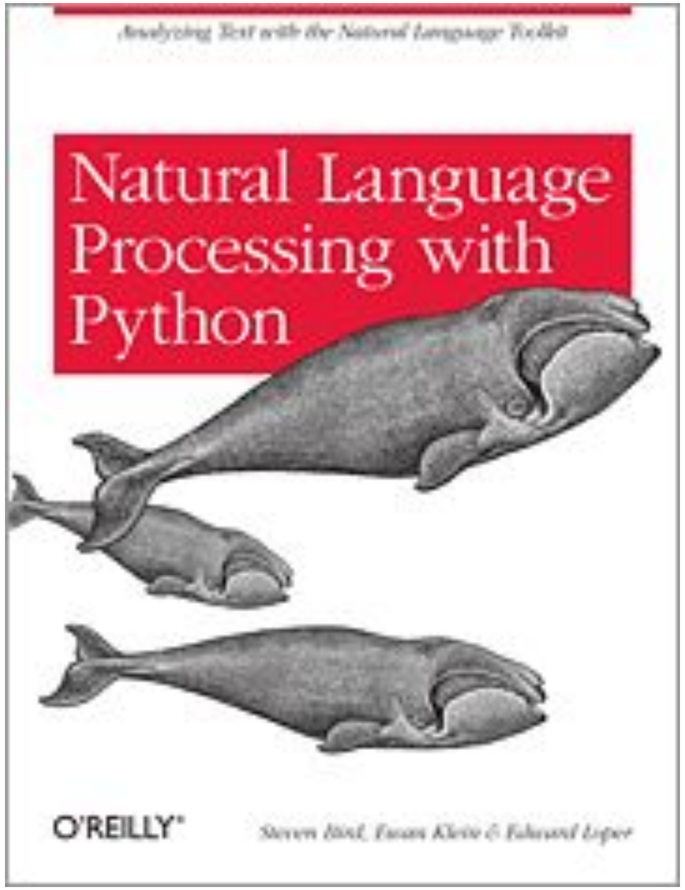
모델

분류/예측

오늘 내용

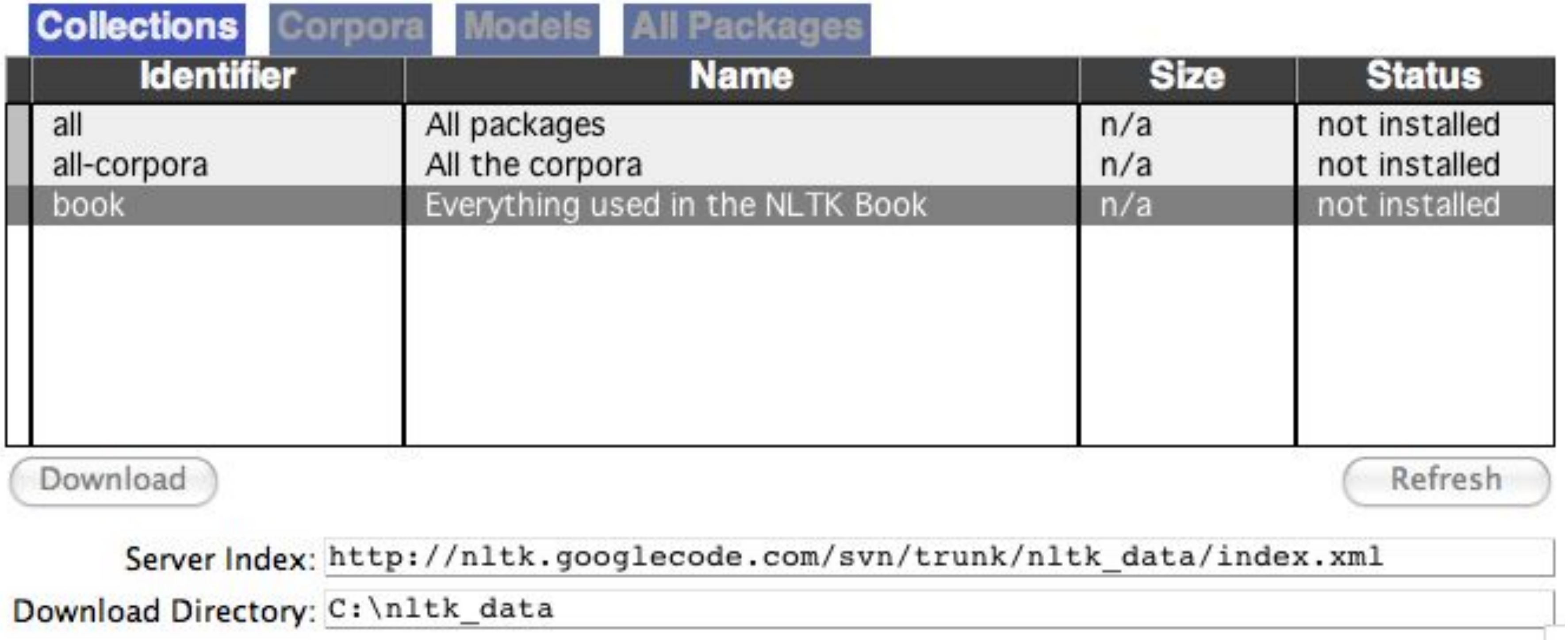
Natural Language Toolkit (NLTK)

- An open source library at http://nltk.org/
- Includes extensive software, data, and documentation
- Natural Language Processing with Python
  – Analyzing Text with the Natural Language Toolkit
  Steven Bird, Ewan Klein, and Edward Loper
- http://www.nltk.org/book/


Analyzing Text with the Natural Language Toolkit
Natural Language Processing with Python
O'REILLY
Steven Bird, Ewan Klein & Edward Loper

NLTK Tool 설치

- Anaconda community version install
- import nltk
- nltk.download() : all



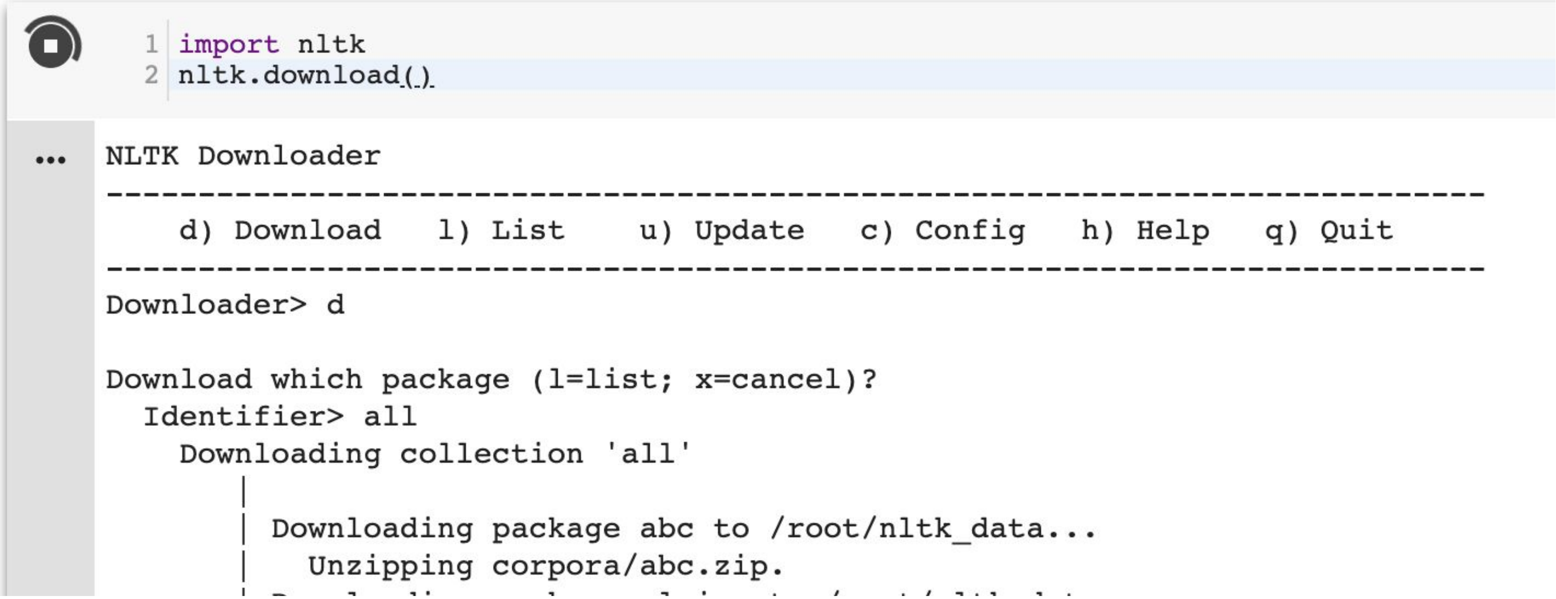
Identifier	Name	Size	Status
all	All packages	n/a	not installed
all-corpora	All the corpora	n/a	not installed
book	Everything used in the NLTK Book	n/a	not installed

Server Index: http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml

Download Directory: C:\nltk_data

Google Colaboratory에서 NLTK 사용

- https://colab.research.google.com 에서 id 만드세요
- import nltk
- nltk.download('all')



```
import nltk
nltk.download()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> all
    Downloading collection 'all'
       |
       | Downloading package abc to /root/nltk_data...
       |   Unzipping corpora/abc.zip.
```

Task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	chunk	regular expression, n-gram, named-entity
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency distributions, smoothed probability distributions
Applications	app, chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

Computing with Language

```
>>> 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>
```

Searching Text

- **concordance(word)** shows every occurrence of *word*, together with some context
    - monstrous in Moby Dick (text1) occurred in contexts such as the ___ pictures and a ___ size

```
>>>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
```

Searching Text

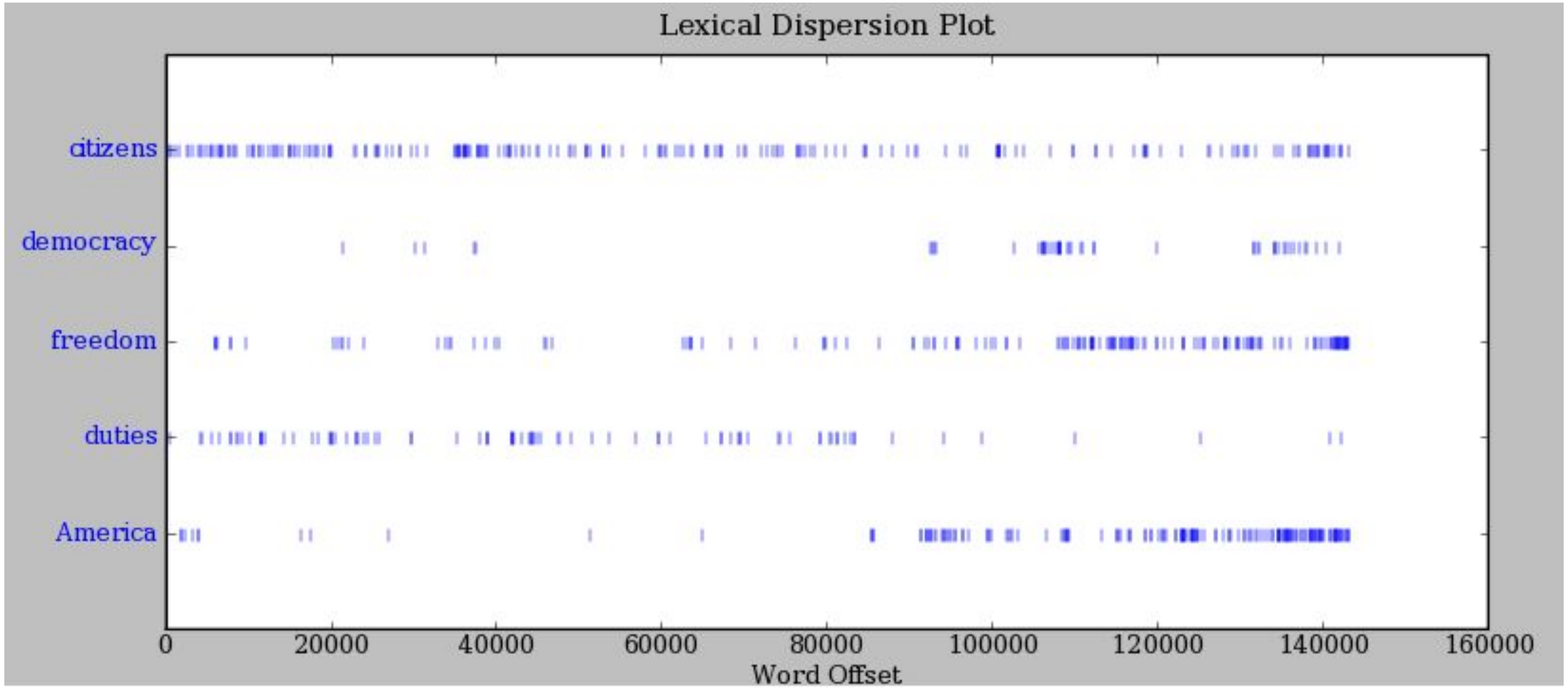
- Other words appear in **similar** contexts
- Different results for different texts
    - For Austen, monstrous has positive connotations, and sometimes functions as an intensifier like the word very
- **common_contexts** allows us to examine the contexts that are shared by two or more words

```
>>> 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

>>> text2.common_contexts(["monstrous", "very"])
a_pretty is_pretty am_glad be_glad a_lucky
```

- Visualize the number of words from the beginning
- Each stripe represents an instance of a word, and each row represents the entire text.
    - Patterns of word usage over the last 220 years (Inaugural Address Corpus)

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



Counting Vocabulary

- **len(text)** returns the length of a text
- *Genesis* has 44,764 tokens
- A **token** is a sequence of characters and punctuation symbols., e.g., his, or :)
- Number of distinct words using **set()**
- Measure the lexical richness of the text
- The number of distinct words is just 6% of the total number of words

```
>>> > len(text3) # number of words
44764
>>> sorted(set(text3))
['!', "'", '(', ')', ',', ',)', '.', '.)', ':', ';', ';)', '?', '?)',
'A', 'Abel', 'Abelmizraim', 'Abidah', 'Abide', 'Abimael',
'Abimelech',
'Abr', 'Abrah', 'Abraham', 'Abram', 'Accad', 'Achbor',
'Adah', ...]
>>> len(set(text3))
2789
>>> len(set(text3)) / len(text3)
0.06230453042623537
>>> text3.count("smote")
5
>>> 100 * text4.count('a') / len(text4)
1.4643016433938312
```

Your Turn: How many times does the word lol appear in text5? How much is this as a percentage of the total number of words in this text?

Lexical Diversity of Various Genres in the Brown Corpus

Genre	Tokens	Types	Lexical diversity
skill and hobbies	82345	11935	0.145
humor	**21695**	**5017**	**0.231**
fiction: science	14470	3233	0.223
press: reportage	100554	14394	0.143
fiction: romance	**70022**	**8452**	**0.121**
religion	39399	6373	0.162

Texts as Lists of Words

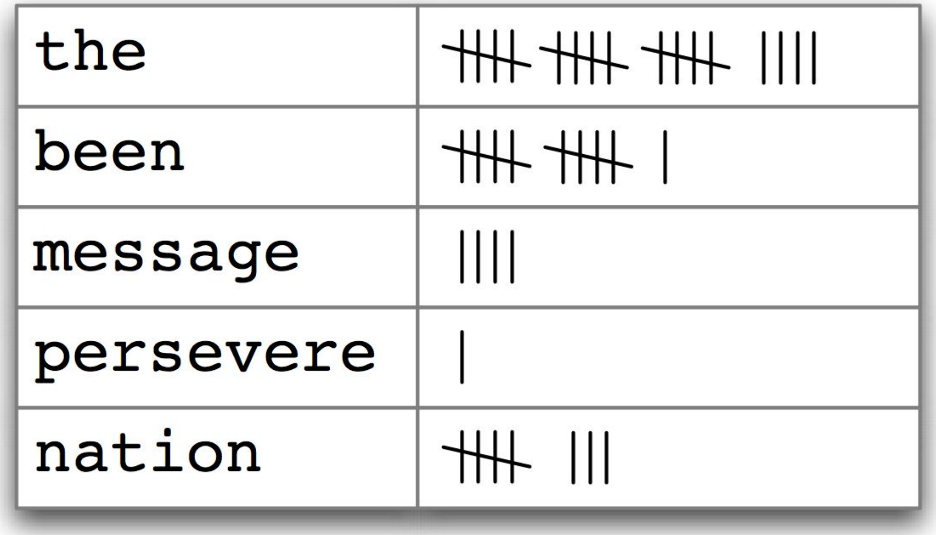
- **Text as a sequence of words and punctuation**

```
>>> sent1 = ['Call', 'me', 'Ishmael', '.']
>>> len(sent1)
4
>>> sent2 = ['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sussex', '.']
>>> sent3 = ['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven', 'and', 'the', 'earth', '.']
```

Simple Statistics

- Identify the words of a text that are **most informative about the topic and genre of the tex**
- One method is to keep a tally for each vocabulary item. It is a "**distribution**" because it tells us **how the total number of word tokens in the text are distributed across the vocabulary items**.

Word Tally

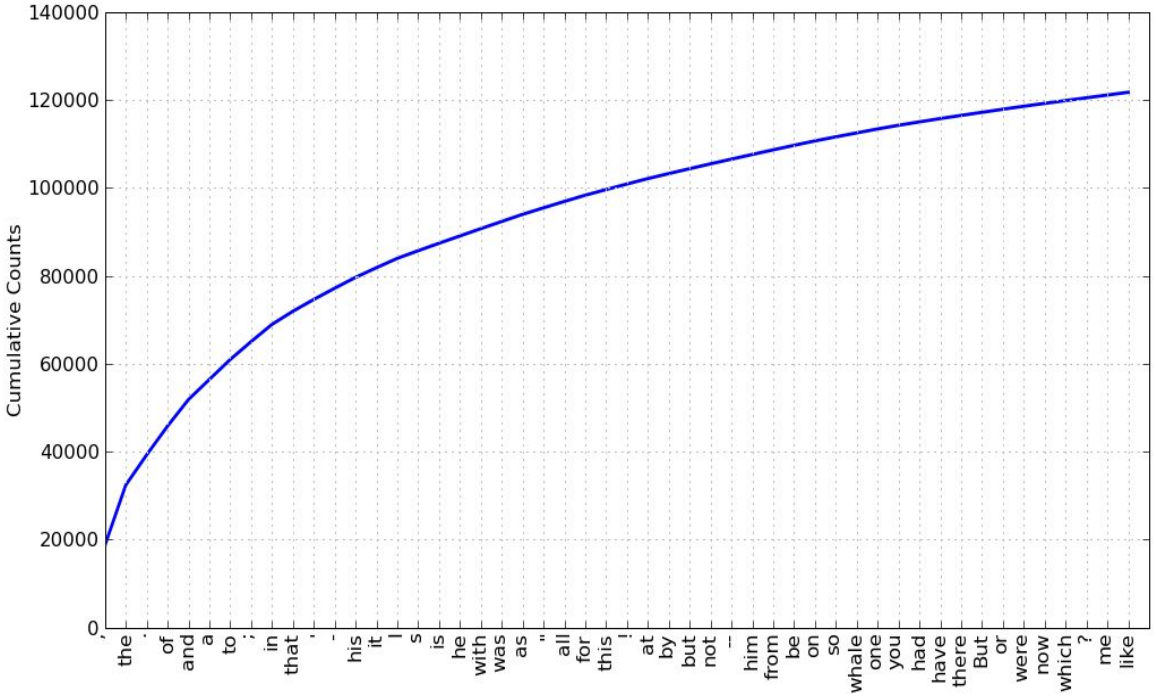


the	𝍸 𝍸 𝍸 ||||
been	𝍸 𝍸 |
message	||||
persevere	|
nation	𝍸 |||

Frequency Distributions

- NLTK provides built-in support for frequency distributions
  - the 50 most frequent words of Moby Dick:

```
>>> fdist1 = FreqDist(text1)
>>> print(fdist1)
<FreqDist with 19317 samples and 260819 outcomes>
>>> fdist1.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024),
('a', 4569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982),
("'", 2684), ('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124),
('s', 1739), ('is', 1695), ('he', 1661), ('with', 1659), ('was', 1632),
>>> fdist1['whale']
906
>>> fdist1.plot(50, cumulative=True)
```



These 50 words account for nearly half the book!

Frequency Distributions Functions

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

Fine-grained Selection of Words

- Find frequently occurring long words

```
>>> fdist5 = FreqDist(text5)
>>> sorted(w for w in set(text5) if len(w) > 7 and fdist5[w] > 7)
['#14-19teens', '#talkcity_adults', '((((((((((', '........', 'Question','actually', 'anything', 'computer', 'cute.-ass', 'everyone',
'football','innocent', 'listening', 'remember', 'seriously', 'something', 'together','tomorrow', 'watching']
```

Collocations and Bigrams

- **Collocation:** a sequence of words that occur together often
    - red wine (O), the wine (X)
- Collocations are resistant to substitution with words that have similar senses
    - maroon wine (X)
- **Bigrams:** a list of word pairs
- **Collocations are frequent bigrams**
- We want to find bigrams that occur more often than we would expect based on the frequency of the words

```
>>> list(bigrams(['more', 'is', 'said', 'than', 'done']))
[('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]

>>> text4.collocations()
United States; fellow citizens; four years; years ago; Federal
Government; General Government; American people; Vice
President; Old World; Almighty God; Fellow citizens; Chief
Magistrate; Chief Justice; God bless; every citizen; Indian
tribes; public debt; one another;
foreign nations; political parties
>>> text8.collocations()
would like; medium build; social drinker; quiet nights; non
smoker; long term; age open; Would like; easy going;
financially secure; fun times; similar interests; Age open;
weekends away; poss rship; well
presented; never married; single mum
```

Counting Other Things

- Distribution of **word lengths**
- **FreqDist** counts the number of times
- There are only 20 different word lengths
- The most frequent word length is 3
- Further analysis of word length might help understand differences between authors, genres, or languages

```
>>> [len(w) for w in text1]   # create a list of the lengths
of words
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1, 1, 8, 2, 1, 4, 11, 5, 2, 1, 7, 6, 1, 3,
4, 5, 2, ...]
>>> fdist = FreqDist(len(w) for w in text1)   # number
of times each of these occurs
>>> fdist
FreqDist({3: 50223, 1: 47933, 4: 42345, 2: 38513, 5: 26597,
6: 17111, 7: 14399, 8: 9966, 9: 6428, 10: 3528, ...})

>>> fdist.most_common()
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), (5, 26597),
(6, 17111), (7, 14399), (8, 9966), (9, 6428), (10, 3528), (11,
1873), (12, 1053), (13, 567), (14, 177), (15, 70), (16, 22),
(17, 12), (18, 1), (20, 1)]
>>> fdist.max()
3
>>> fdist[3]
50223
>>> fdist.freq(3)
0.19255882431878046
```

Summary

- **token** is a particular appearance of a given word in a text
- **sorted(set(t))** builds the vocabulary of a text
- [f(x) for x in text] operates on each item/word of a text
- **set(w.lower() for w in text if w.isalpha()) to derive the vocabulary**, collapsing case distinctions and ignoring punctuation
- A frequency distribution is a collection of tokens along with their frequency counts