Introduction to Python Programming

(17) Introduction to NLTK

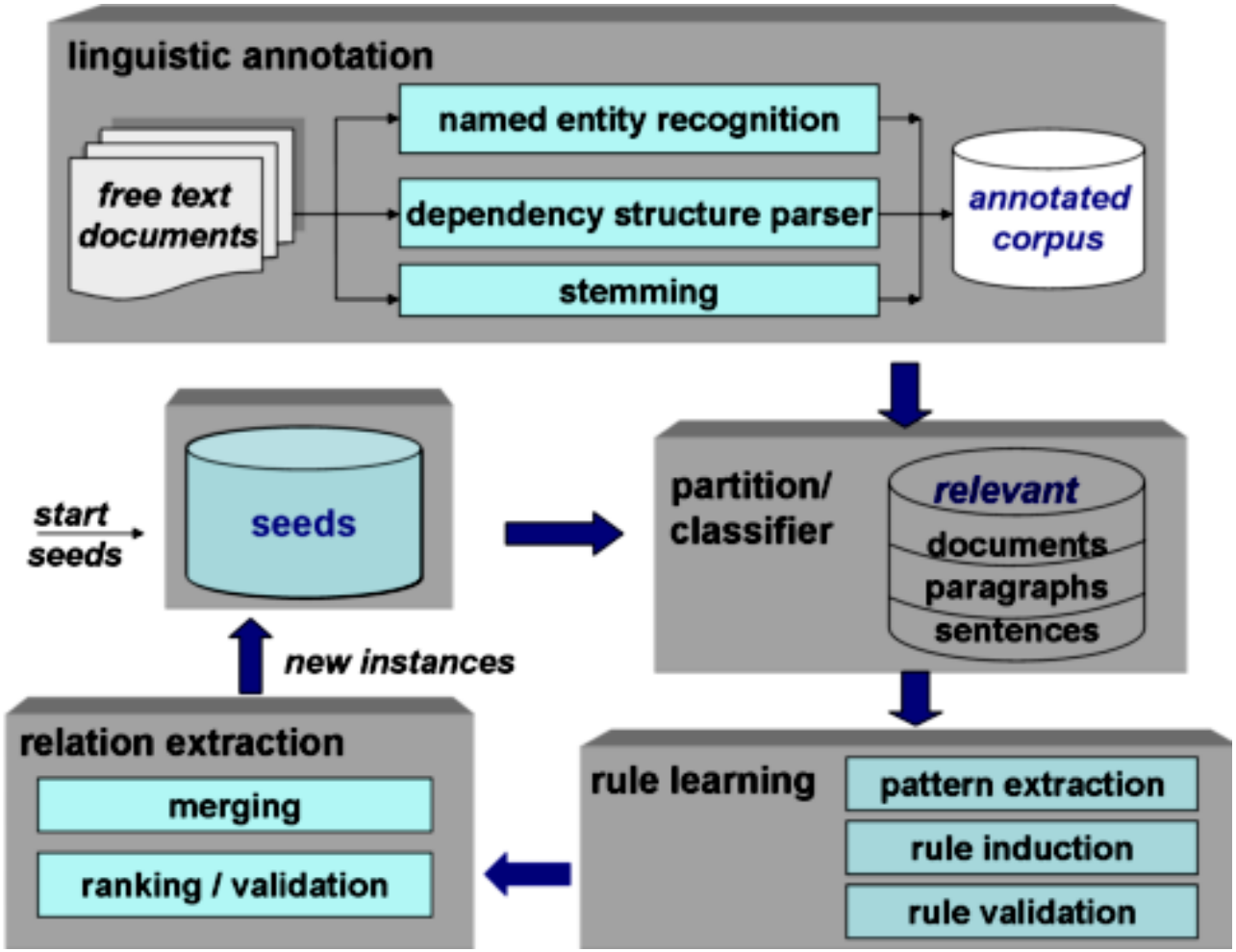
S. Thater and A. Friedrich

Saarland University

Winter Semester 2011/2012

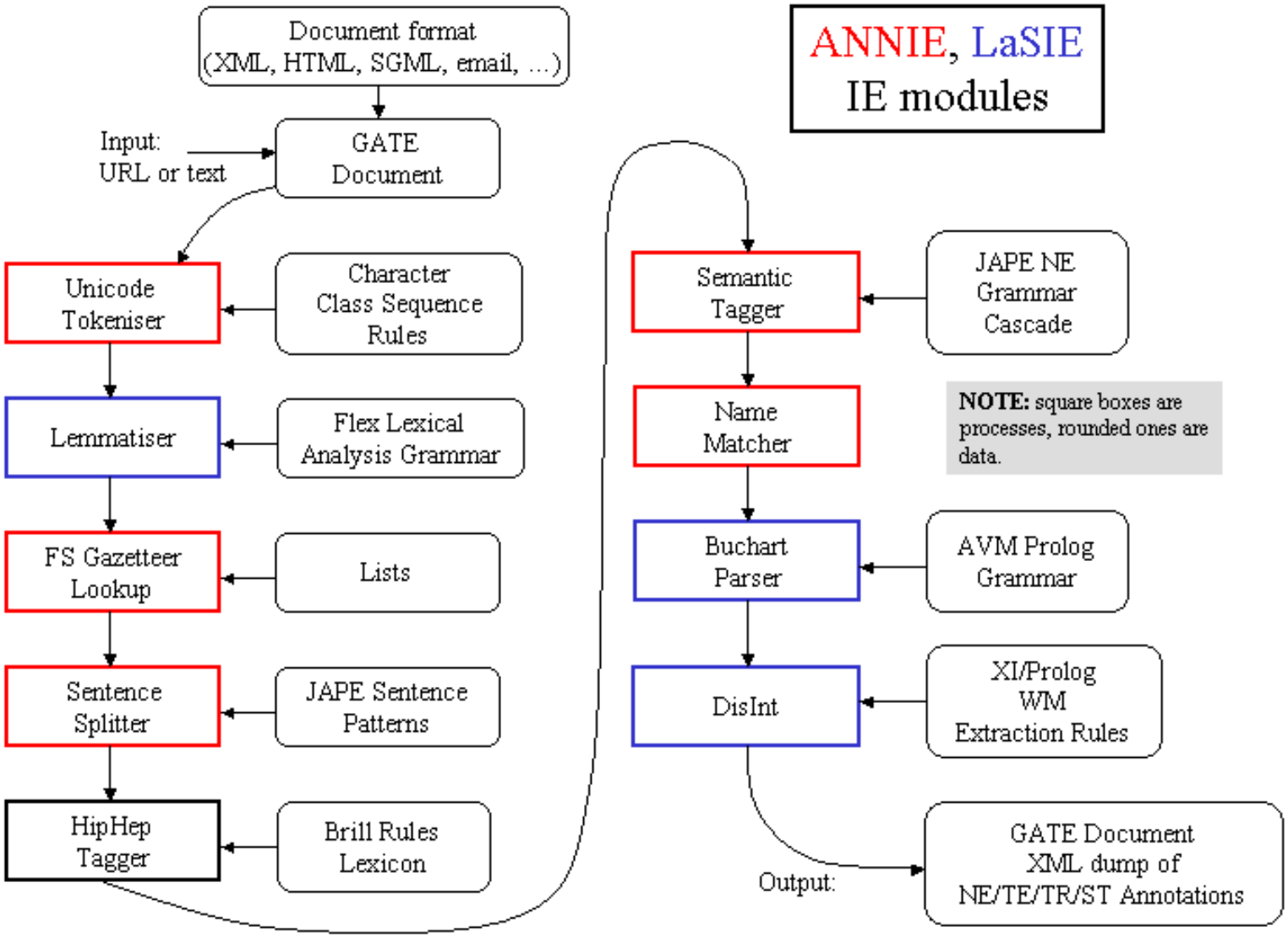
DARE (Relation Extraction System)

developed at DFKI by Feiyu Xu (source of image: http://dare.dfki.de/)



ANNIE (Information Extraction System)

developed at U' of Sheffield (source of image: http://gate.ac.uk/sale/tao/annie.png)



Common Natural Language Processing Tasks

- Tokenization
- POS Tagging
- Stemming / Lemmatisation
- Named Entity Recognition
- Parsing
- ...

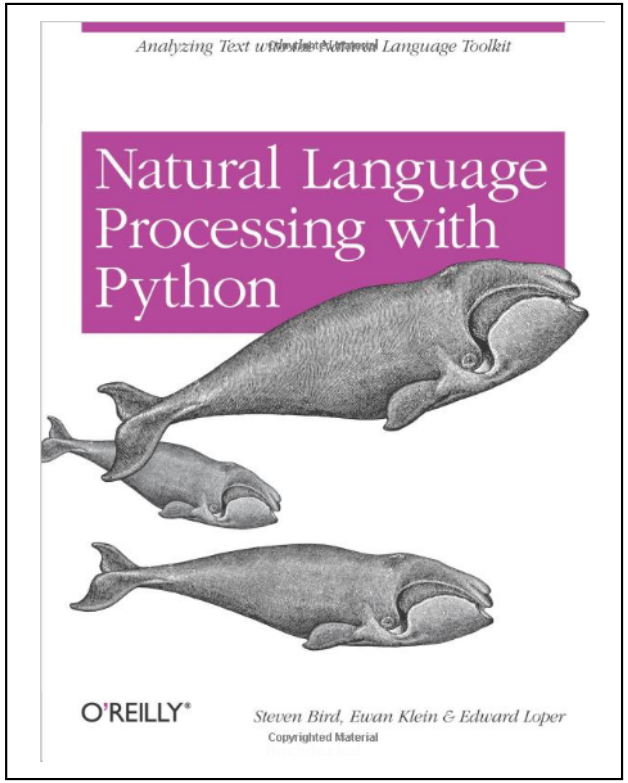
Natural Language Toolkit (NLTK)

- open source Python library
- software, data & documentation
- originates from U'Penn, developed for educational purposes
- Natural Language Processing (NLP):
  computer manipulation of natural language
  - counting word frequencies
  - ...
  - 'understanding' human utterances

Documentation

- Website: `http://www.nltk.org`: Download, Documentation etc.

- NLTK Book: Natural Language Processing with Python
  - Authors: Steven Bird, Ewan Klein, and Edward Loper
  - O'Reilly 2009
  - available online `http://www.nltk.org/book`



NLTK Library

NLTK Modules	NLP Task
nltk.corpus	Accessing corpora and lexicons
nltk.tokenize, nltk.stem	String processing, tokenizer, stemmers
nltk.collocations	Collocation discovery (t-test, PMI, chi-squared)
nltk.tag	Part-of-speech tagging (n-gram, HMM,...)
nltk.classify, nltk.cluster	Classification (decision trees, Bayes, k-means...)
nltk.chunk	Chunking (regex, n-gram, named entity)
nltk.parse	Parsing
nltk.sem, nltk.inference	Semantic interpretation (lambda, FOL, ...)
nltk.metrics	Evaluation metrics (precision, recall, agreement)
nltk.probability	Frequency distributions
nltk.app, nltk.chat	Applications, parsers, WordNet browser, chatbots, ...
nltk.toolbox	Linguistic fieldwork

Software Requirements

- **Python** 2.X (to date, NLTK has not been ported to Python 3.X)
- **NLTK**
- **NLTK-Data**
- **NumPy**: linear algebra, needed for tasks involving probability, clustering, classification etc.
- **Matplotlib**: 2D plotting library for data visualization

- Check `http://www.nltk.org/download` how to install this software on your computer.
- In the lab, nltk is already installed.

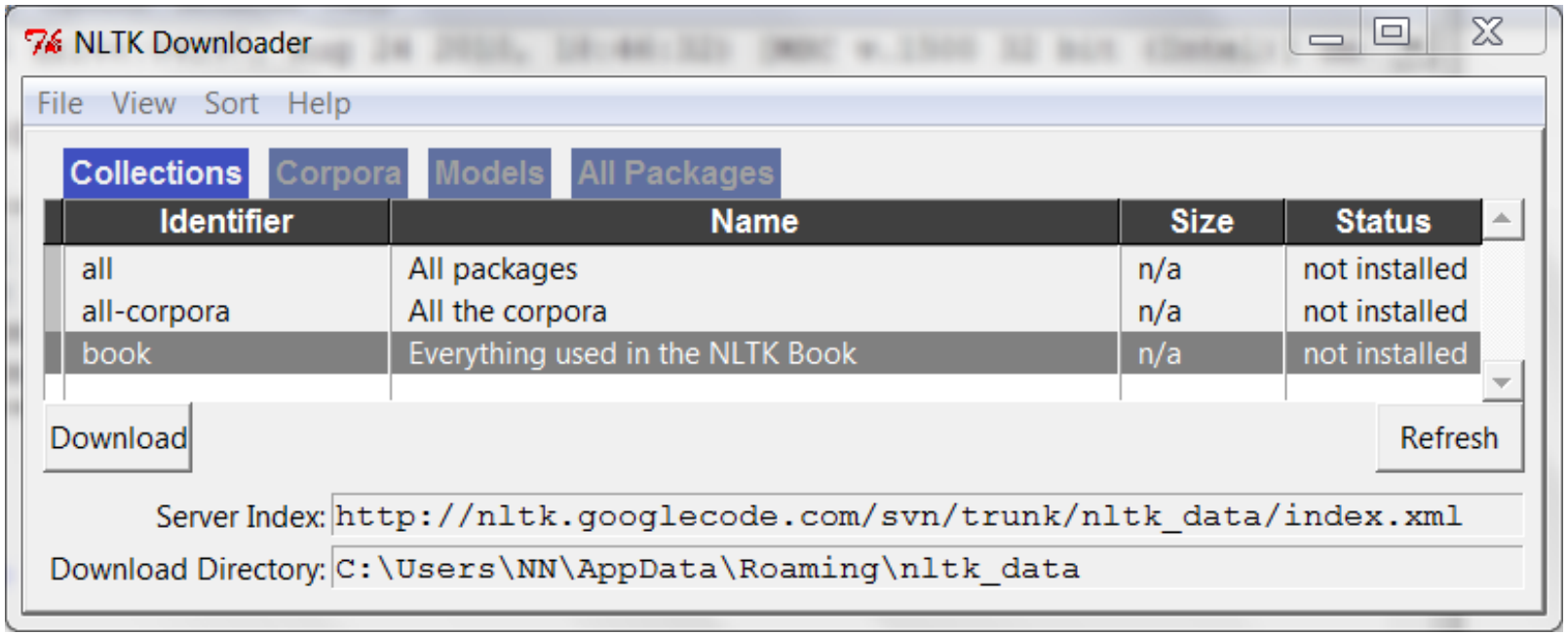
Text Corpus

- large body (collection) of text
- may be open-domain or concentrate on a genre
- may be raw text or annotated / categorized

gutenberg	selection of e-books from Project Gutenberg
webtext	forum discussions, reviews, movie script
nps_chat	anonymized chats
brown	1 million word corpus, categorized by genre
reuters	news corpus
inaugural	inaugural addresses of US presidents
udhr	multilingual corpus

Downloading Data I (your own machine)

```
1  # importing the nltk library
2  import nltk
3
4  # download some of the data
5  nltk.download()
```



Downloading Data II (Coli machine)

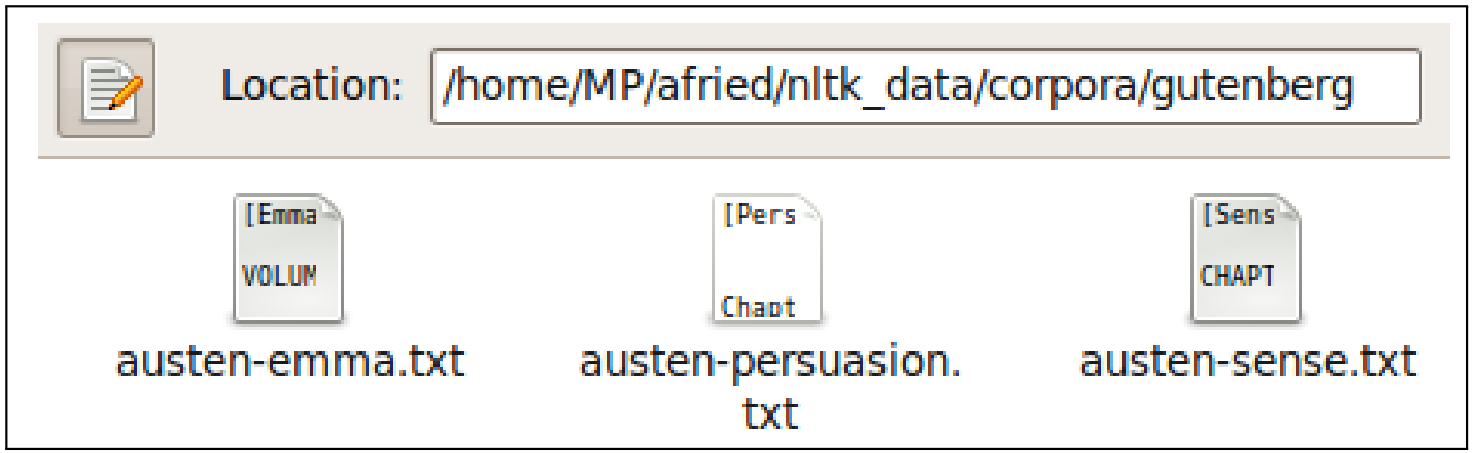
1. Create a folder `nltk_data` in your home directory.
2. Inside this folder, create the folders: `corpora`, `tokenizers` and `taggers`.
3. Go to `http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml` and download the corpora / taggers you need. Unzip them into the `corpora`, `tokenizers` or `taggers` folder.
4. You're done - now you can access these corpora/tokenizers/taggers in your Python code via the NLTK library.

Accessing Corpora

- Corpora on disk: text file(s)
- NLTK provides Python modules / functions / classes that allow for accessing the corpora in a convenient way
- This example shows why frameworks are useful:
  - It's quite an effort to write functions that read in a corpus (especially when it comes with annotations).
  - The task of reading in a corpus is needed in many NLP projects.
  - As the code for this has been written, you can simply use it!

Accessing Corpora

```
1  # tell Python we want to use the Gutenberg corpus
2  from nltk.corpus import gutenberg
3
4  # which files are in this corpus?
5  print(gutenberg.fileids())
6
7  >>> ['austen-emma.txt', 'austen-persuasion.txt',
8  'austen-sense.txt', 'bible-kjv.txt', ...]
```



Accessing Corpora: Raw text

```
1  # get the raw text of a corpus = one string
2  >>> emmaText = gutenberg.raw("austen-emma.txt")
3
4  # print the first 289 characters of the text
5  >>> emmaText = gutenberg.raw("austen-emma.txt")
6  >>> emmaText[:289]
7  '[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER
8  I\n\n\nEmma Woodhouse, handsome, clever, and rich,
9  with a comfortable home\nand happy disposition,
10 seemed to unite some of the best blessings\nof
11 existence; and had lived nearly twenty-one years in
12 the world\nwith very little to distress or vex her.'
```

Accessing Corpora: Words

```
# get the words of a corpus as a list
emmaWords = gutenberg.words("austen-emma.txt")

# print the first 30 words of the text
>>> print(emmaWords[:30])
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']',
'VOLUME', 'I', 'CHAPTER', 'I', 'Emma', 'Woodhouse',
',', 'handsome', ',', 'clever', ',', 'and', 'rich',
',', 'with', 'a', 'comfortable', 'home', 'and',
'happy', 'disposition', ',', 'seemed']
```

Accessing Corpora: Sentences

```
# get the sentences of a corpus as a list of lists
# (one list of words per sentence)
>>> senseSents = gutenberg.sents("austen-sense.txt")

# print out the first four sentences
>>> print(senseSents[:4])
[['[', 'Sense', 'and', 'Sensibility', 'by', 'Jane',
'Austen', '1811', ']'],

['CHAPTER', '1'],

['The', 'family', 'of', 'Dashwood', 'had', 'long',
'been', 'settled', 'in', 'Sussex', '.'],

['Their', 'estate', 'was', 'large', ',', 'and',
'their', 'residence', 'was', 'at', ...]]
```

Brown Corpus

- the first million-word electronic corpus of English
- created at Brown University in 1961
- text from 500 sources, categorized by genre

```
>>> from nltk.corpus import brown

>>> print(brown.categories())
['adventure', 'belles_lettres', 'editorial',
'fiction', 'government', 'hobbies', 'humor',
'learned', 'lore', 'mystery', 'news', 'religion',
'reviews', 'romance', 'science_fiction']
```

Brown Corpus

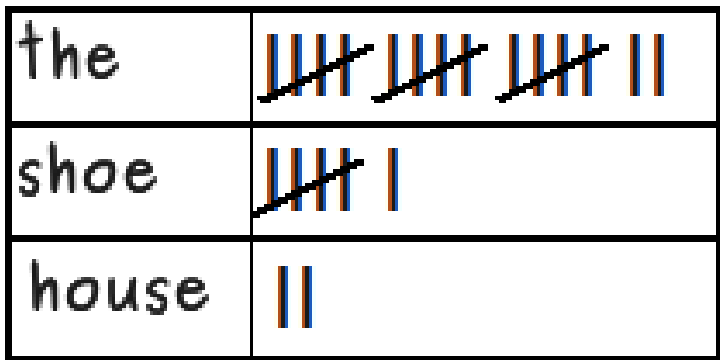
- You can retrieve the words by category

```
>>> from nltk.corpus import brown
>>> news_words = brown.words(categories = "news")
>>> print(news_words)
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said',
'Friday', 'an', 'investigation', 'of', "Atlanta's",
'recent', 'primary', 'election', 'produced', ...]

>>> adv_words = brown.words(categories = "adventure")
>>> print(adv_words)
['Dan', 'Morgan', 'told', 'himself', 'he', 'would',
'forget', 'Ann', 'Turner', '.', ...]

>>> reli_words = brown.words(categories = "religion")
>>> print(reli_words)
['As', 'a', 'result', ',', 'although', 'we', 'still',
'make', 'use', 'of', 'this', 'distinction', ',',...]
```

Frequency Distribution

- records how often each item occurs in a list of words
- frequency distribution over words
- basically a dictionary with some extra functionality
- `init` creates a frequency distribution from a list of words

```
1  news_words = brown.words(categories = "news")
2  fdist = nltk.FreqDist(news_words)
3  print("shoe:", fdist["shoe"])
4  print("the: ", fdist["the"])
```




```
('shoe:', 1)
('the: ', 5580)
```

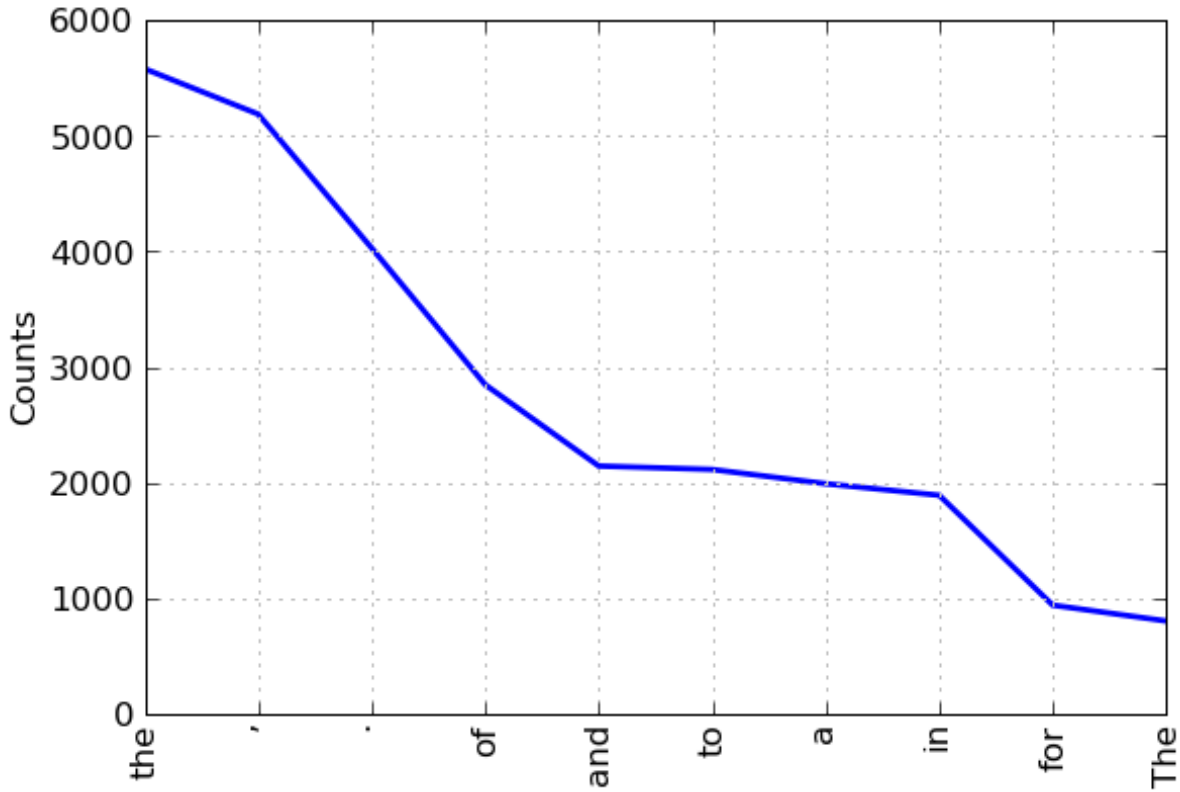
Frequency Distribution

```
1  # show the 10 most frequent words & frequencies
2  fdist.tabulate(10)
```

```
 the    ,    .   of  and   to    a   in  for  The
5580 5188 4030 2849 2146 2116 1993 1893  943  806
```

Frequency Distribution

```
1  # create a plot of the 10 most frequent words
2  fdist.plot(10)
```



Stylistics

- systematic differences between genres
- Brown corpus with its categories is a convenient resource
- E.g. is there a difference in how the modal verbs (can, could, may, might, must, will) are used in the genres?

Frequency Distributions of Modal Verbs in various Genres

```
1  from nltk import FreqDist
2  # Define modals of interest
3  modals = ["may", "could", "will"]
4  # Define genres of interest
5  genres = ["adventure", "news", "government", "romance"]
6
7  # count how often they occur in the genres of interest
8  for g in genres:
9      words = brown.words(categories = g)
10     fdist = FreqDist([w.lower() for w in words
11                       if w.lower() in modals])
12     print g, fdist
```

Frequency Distributions of Modal Verbs in various Genres

```
1  # count how often they occur in the genres of interest
2  for g in genres:
3      words = brown.words(categories = g)
4      fdist = FreqDist([w.lower() for w in words
5                        if w.lower() in modals])
6      print g, fdist
```

```
adventure  <'will': 51,  'could': 154, 'may': 7>
news       <'will': 389, 'could': 87,  'may': 93>
government <'will': 244, 'could': 38,  'may': 179>
romance    <'will': 49,  'could': 195, 'may': 11>
```

Conditional Frequency Distributions

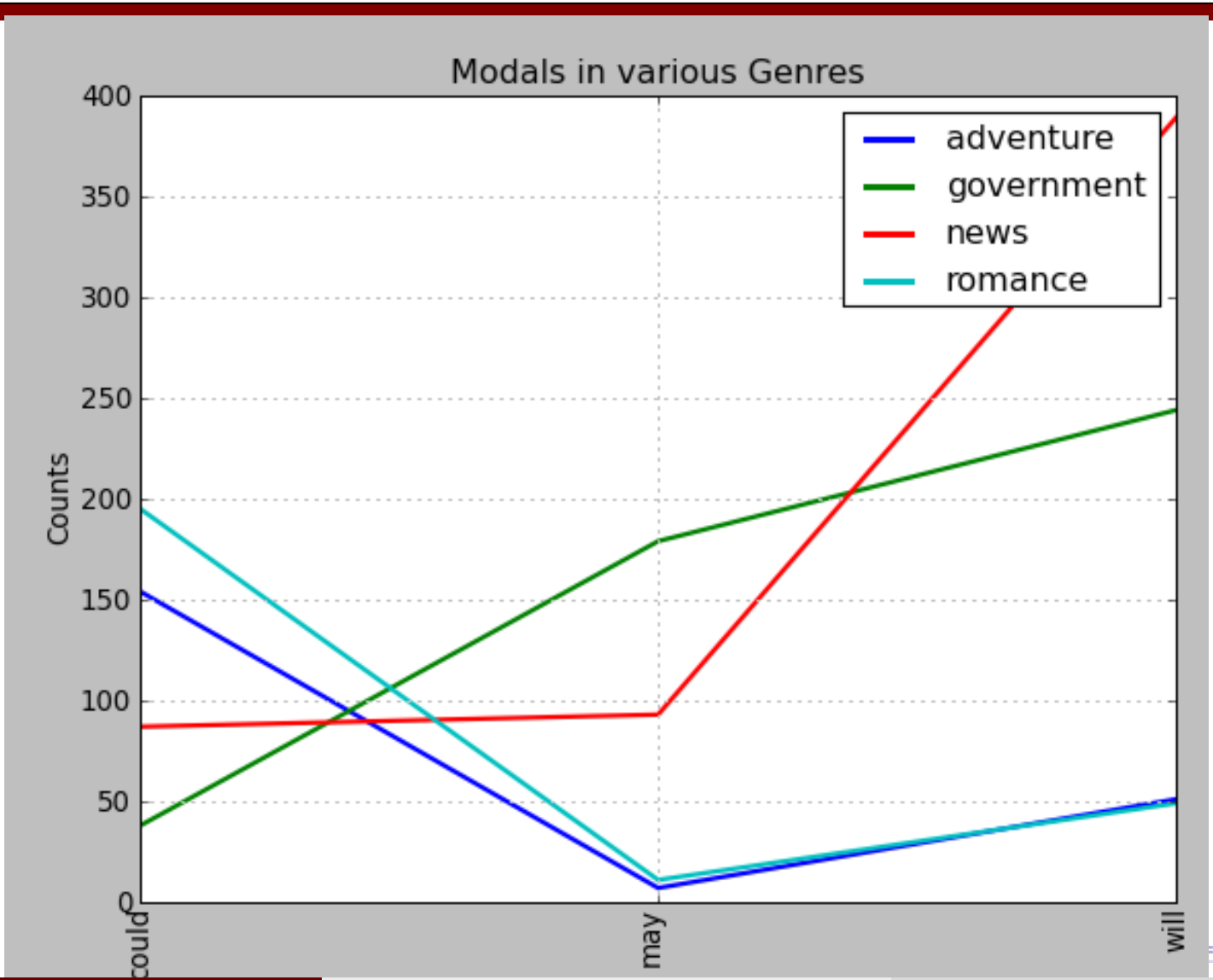
news	the	𝍸 𝍸 𝍸 ||
	president	𝍸 |||
	earthquake	||
adventure	the	𝍸 𝍸 ||
	president	|||
religion	the	𝍸 𝍸 |||
	god	𝍸 𝍸
	pray	𝍸 𝍸 ||

Conditional Frequency Distributions

```
 1  from nltk import ConditionalFreqDist
 2  cfdist = ConditionalFreqDist()
 3  for g in genres:
 4      words = brown.words(categories = g)
 5      for w in words:
 6          if w.lower() in modals:
 7              cfdist[g].inc(w.lower())
 8
 9  >>> cfdist.tabulate()
10                could  may will
11   adventure  154    7   51
12  government   38  179  244
13        news   87   93  389
14     romance  195   11   49
```

Conditional Frequency Distributions

```
1  cfdist.plot(title="Modals in various Genres")
```



Processing raw Text

- Assume you have a text file on your disk...

```
1  # Read the text (as we know it already)
2  >>> path = "holmes.txt"
3  >>> f = open(path)
4  >>> rawText = f.read()
5  >>> f.close()
6  >>> print(rawText[:165])
7  THE ADVENTURES OF SHERLOCK HOLMES
8  by
9  SIR ARTHUR CONAN DOYLE
10     I. A Scandal in Bohemia
11    II. The Red-headed League
```

Sentence Tokenization

```
# Split the text up into sentences
""" sent_tokenize calls the NLTK's currently
recommended sentence tokenizer to tokenize sentences
in the given text. Currently, this uses
PunktSentenceTokenizer."""
>>> sents = nltk.sent_tokenize(raw)
>>> print(sents[20:22])
['I had seen little of Holmes lately.',
'My marriage had drifted us\r\naway from each other.',
...]
```

Word Tokenization

```
1  # Tokenize the sentences using nltk
2  """ Use NLTK's currently recommended word
3  tokenizer to tokenize words in the given sentence.
4  Currently, this uses TreebankWordTokenizer.
5  This tokenizer should be fed a single sentence
6  at a time. """
7  tokens = []
8  for sent in sents:
9      tokens += nltk.word_tokenize(sent)
10
11 print(tokens[300:350])
```

```
['such', 'as', 'his', '.', 'And', 'yet', 'there',
'was', 'but', 'one', 'woman', 'to', 'him', ',',
'and', 'that', 'woman', 'was', 'the', 'late',
'Irene', 'Adler', ',', 'of', 'dubious', 'and',
'questionable', 'memory', ...]
```

Creating a Text object

- Using a list of tokens, we can create an `nltk.Text` object for a document.
- Collocations = terms that occur together unusually often.
- Concordance view = shows the contexts in which a token occurs.

```
1  # Create a text object
2  text = nltk.Text(tokens)
3
4  # Do stuff with the text object
5  print(text.collocations())
6  print(text.concordance("Irene"))
```

Collocations

```
Sherlock Holmes; said Holmes; St. Simon;
Baker Street; Lord St.; St. Clair; Mr. Holmes;
Hosmer Angel; Irene Adler; Miss Hunter; young lady;
Briony Lodge; Stoke Moran; Neville St.; Miss Stoner;
Scotland Yard; could see; Mr. Holmes.; Boscombe Pool;
Mr. Rucastle
```

Concordance View

```
Building index...
Displaying 17 of 17 matches:
 to love for Irene Adler . All emotions , and that one
was the late Irene Adler , of dubious and questionable
dventuress , Irene Adler . The name is no doubt familia
nd . " " And Irene Adler ? " " Threatens to send them t
se , of Miss Irene Adler . " " Quite so ; but the seque
 And what of Irene Adler ? " I asked . " Oh , she has t
 tying up of Irene Adler , spinster , to Godfrey Norton
ction . Miss Irene , or Madame , rather , returns from
...
```

Documentation

Overview: `http://www.nltk.org/documentation`

API: `http://nltk.github.com/api/nltk.html`

Look at API documentation...

Annotated Corpora

Example

```
The/at Fulton/np-tl County/nn-tl Grand/jj-tl
Jury/nn-tl said/vbd Friday/nr an/at
investigation/nn ...
```

- Some corpora come with annotations, e.g. POS tags, parse trees,...
- NLTK provides convenient access to these corpora (get the text + annotations) *see exercise sheet*
- Dependency Tree Bank (e.g. Penn): collection of (dependency-)parsed sentences (manually annotated), can be used for training a statistical parser or parser evaluation. (Exercise Sheet: We will only look at POS tags, not at the dependency parses.)

Exercises

- **Please** submit 5 files and don't change the file names.
- **Please** don't forget to put your name in each file.
- Accessing WordNet using NLTK will be presented on Monday!