Practical Python

6: Text Analytics and aspects of NLTK

Dafydd Gibbon

First South African Workshop in Digital Humanities
North-Western University, Potchefstroom, SA
2015-04-04 to 2015-04-05

Natural Language Processing

- Text generation
- Machine translation
- Dialogue systems
- Terminology extraction
- Spam filtering
- Summarisation

- Document similarity:
  - plagiarism detection
  - document categorisation
  - text classification
  - topic detection
  - authorship attribution
  - language identification

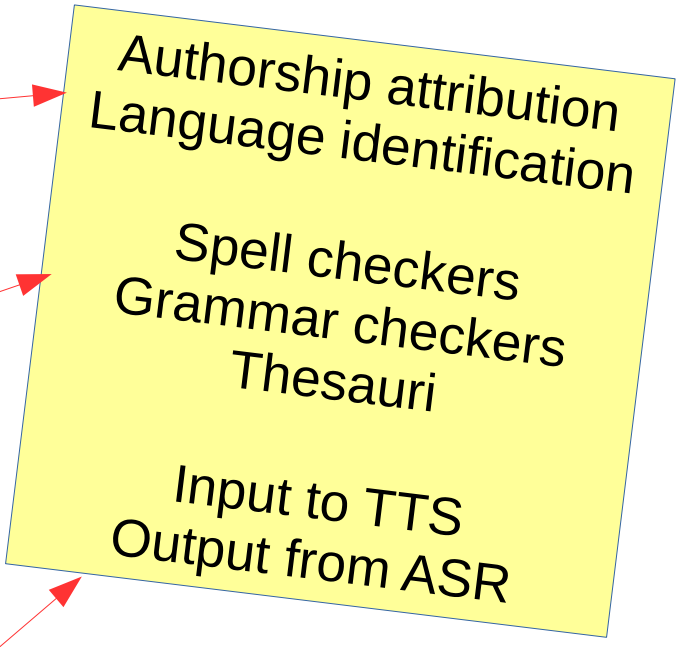
- Keyword spotting
- Part of Speech (POS) tagging
- Tokenisation, stemming
- Chunking
- Parsing
- Word sense disambiguation
- Pronoun resolution
- Textual entailment
- Sentiment analysis
- Named entity recognition (NER)

NLP: some details

- Text analytics:
  - Text retrieval
  - Text properties
  - *n*-gram tagging
  - Text classification
- Lexicon construction
  - wordlist extraction
  - concordance construction
  - POS, G2P, valency, semantics
- Parsing, text and speech understanding
  - grammar and automaton induction
  - mapping to situation models, databases
- Machine translation
  - analysis of parallel and comparable texts
  - translator's workbench
  - translation services

NLP: some details

- Text analytics:
  - Text retrieval
  - Text properties
  - *n*-gram tagging
  - Text classification
- Lexicon construction
  - wordlist extraction
  - concordance construction
  - POS, G2P, valency, semantics
- Parsing, text and speech understanding
  - grammar and automaton induction
  - mapping to situation models, databases
- Machine translation
  - analysis of parallel and comparable texts
  - translator's workbench
  - translation services

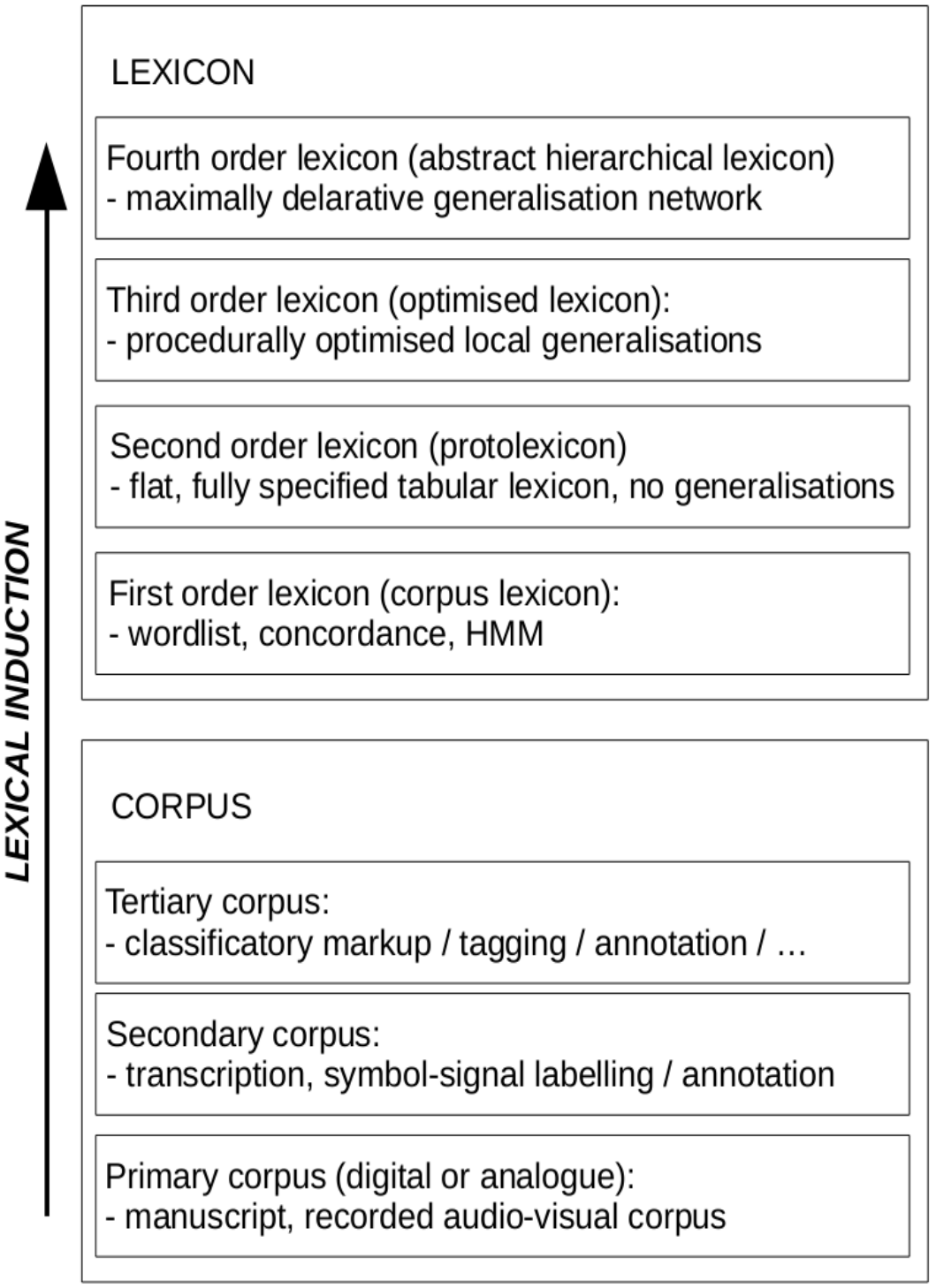


Corpus Processing:

Data Abstraction and Lexical Induction

Increase in structural complexity

Decrease in data complexity



Data sources and tools

Common digital text sources:

Custom OCR

Project Gutenberg:

- tens of thousands of books in many formats, including basic unformatted text
- metadata at start and end may need treating
- download a collection of texts from here

Web pages: Python has suitable packages for these:

- News etc., Facebook, Twitter, ...

NLTK

- a large number of ready-formatted digital texts in various languages
- access to Project Gutenberg and the Web
- plus other Python libraries

Aspects of lexicography: KWIC (KeyWord In Context) Concordance

- Purpose:
  - Clarification of word usage
  - Analysis of grammatical word category
  - Analysis of meaning in context
  - Comparison of texts
  -
- Behaviour of a typical KWIC concordance:
  - Input: selection of text, keyword
  - Output: list of contexts in which the keyword occurs, with contexts defined
    - either as *n* characters left and right of the word concerned
    - or as *n* words left and right of the word concerned

Aspects of lexicography: KWIC (KeyWord In Context) Concordance

Text: Joseph Conrad, Lord Jim (Gutenberg Project)
Keyword: “fast”

```
            my back , all fast asleep in that fore-
    gently , and holding fast to some deep idea
 the twilight was ebbing fast from the sky above
      . they would stand fast on their hill and
' henceforth events move fast without a check ,
   began to whisper very fast , touching his elbow
    voice , but speaking fast , he began to
```

Comments:

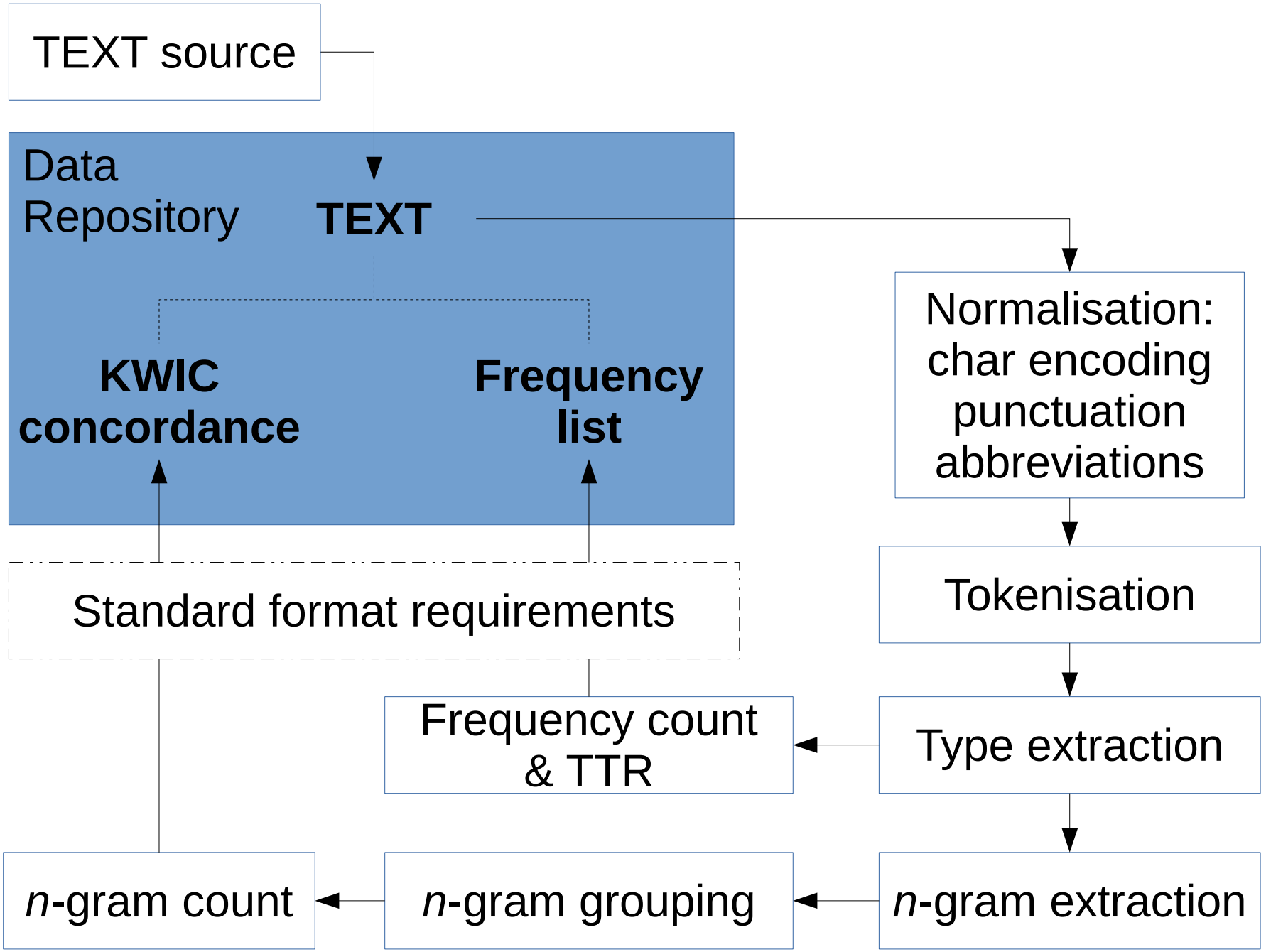
- “fast” occurs 7 times, in 3 types of contexts:
  - idiomatic collocation / fixed expression: ‘fast asleep’
  - (archaic?) adverb meaning ‘firmly’: ‘holding fast’, ‘stand fast’
  - adverb meaning ‘quickly’: ‘ebbing fast’ (of tide), ‘move fast’ (of events), ‘whisper fast’, ‘speaking fast’ (of speech), the speech contexts each before comma

Question:

Which is the most well-known concordance in the world?

Try googling this ...

Design of a basic KWIC concordance based on raw text



Words in context – implementation of a simple KWIC

```
#!/usr/bin/python
# kwic.py
# D. Gibbon
# 2015-02-16

# Import system and regular expression modules

import os, sys, re
```

Preamble containing:

1. Shebang line to run python script under Linux
2. 3 lines with minimal documentation
3. Minimal inline documentation
4. Import of modules:
   - os (operating system file handling)
   - sys (for command line handling)
   - re (for regular expression handling)

Words in context – implementation of a simple KWIC

```
# Main definition of processing sequence
# Input: raw text
# Output: formatted KWIC, with tokens, types and ttr

def kwickie():

    text,length,pos,lenerr,fname = kwicinput()
    tokens = tokenise(text)
    ngrams = makengrams(tokens,length)
    kwictionary = makekwic(ngrams,length,pos)
    kwicstr = formatkwic(kwictionary,pos)
    types,ttr = typesandtokens(tokens)
    toklen = len(tokens)
    typlen = len(types)
    kwicoutput(kwicstr,toklen,typlen,ttr,lenerr,fname)

    return
```

Words in context – implementation of a simple KWIC

```
# Tokenisation

def tokenise(text):

    for c in '\n\r\t\f':
        text = re.sub(c,' ',text)

    text = tokenisepunc(text)
    text = re.sub('  *',' ',text)
    tokens = text.split(' ')
    tokens = [t for t in tokens if t != '']

    return tokens
```

Words in context – implementation of a simple KWIC

```
# Tokenise punctuation

def tokenisepunc(text):
    puncstr = ',;:!/\'"!-=]{}%~'
    for c in puncstr:
        text = re.sub(c,' '+c+' ',text)
    text = re.sub('\.',' . ',text)
    text = re.sub('\?',' ? ',text)
    text = re.sub('\(',' ( ',text)
    text = re.sub('\)',' ) ',text)
    text = re.sub('\[',' [ ',text)
    text = re.sub('\^',' [ ',text)
    text = re.sub('\$',' [ ',text)
    text = re.sub('\*',' [ ',text)
    text = re.sub('\&',' [ ',text)
    text = re.sub('\#',' [ ',text)
    return text
```

Python regex special characters treated separately

Words in context – implementation of a simple KWIC

```
# Collect types and type-token ratio from list of tokens
# ttr as percentage of types in relation to tokens

def typesandtokens(tokens):

    types = sorted(list(set(tokens)))
    ttr = 100.0 * len(types) / len(tokens)
    ttr = "%6.2f" % ttr

    return types, ttr
```

Words in context – implementation of a simple KWIC

```
# Collect ngrams with sliding window of specified length

def makengrams(tokens,length):
    ngrams = [tokens[i:i+length] for i in
range(len(tokens)-6)]
    return ngrams

# Construct Python dictionary from ngrams:

def makekwic(ngrams,length,pos):
    kwictionary = {}
    for ng in ngrams:
        if ng[pos] not in kwictionary:
            kwictionary[ng[pos]] = [ng]
        else:
            kwictionary[ng[pos]].append(ng)
    return kwictionary
```

Words in context – implementation of a simple KWIC

```
# Format Python dictionary into KWIC concordance:

def formatkwic(kwictionary,pos):
    kwicstr = ""
    for key in sorted(kwictionary.keys()):
        freq = str(len(kwictionary[key]))
        kwicstr += key+" (freq: " + freq + ")\n"
        for pr in kwictionary[key]:
            outstr = " ".join(pr[:pos]).rjust(30)
            outstr += str(pr[pos]).center(len(pr[pos])+4)
            outstr += " ".join(pr[pos+1:])
            kwicstr += outstr + '\n'
        kwicstr += '\n'
    return kwicstr
```

Words in context – implementation of a simple KWIC

```
# Input and check command line parameters

def kwicinput():
    if len(sys.argv) < 3:
        print "Usage: kwic.py <fname> <ngramlen>"
        exit()
    fname = sys.argv[1]
    if not os.path.isfile(fname):
        print "File not found." ; exit()
    lenstr = sys.argv[2]
    if len(lenstr) == 1 and lenstr in "123456789":
        length = int(lenstr) ; lenerr = ""
    else:
         length = 5 ; lenerr = "Length corrected to 5!"
    pos = int(length/2)
    text = open(fname, 'r').read()
    return text, length, pos, lenerr, fname
```

1. Check for text filename and *n*-gram length parameters

2. Check for existence of text file

3. Check length parameter for freakish input, using Python try-except construct

4. Open and read the text file

Words in context – implementation of a simple KWIC

```
# Main caller


kwickie()
```

The main KWIC concordance function is called directly from the top level of the Python script

Words in context – implementation of a simple **KWIC**

- Get a text from somewhere, in a standard text coding.
- Write the KWIC program with an editor
- Test the KWIC program.
- Rewrite the linear scripting parts in functional style.

NLTK toolbox library

Preliminaries

- Full documentation at http://www.nltk.org
- NLTK is included in Anaconda:

  https://www.continuum.io/downloads
- If you are not using Anaconda, then (under Ubuntu)

  sudo apt-get install python-nltk
- NLTK uses other python libraries, e.g.

  numpy

  matplotlib

  NetworkX

  Prover9
- But you will need to download NLTK data (it is stored in your your home directory):

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

Initialising NLTK with Anaconda installation

```
(trusty)gibbon@localhost:~/Desktop/DH-Potch-2016-Python-Tutorial$ python

Python 2.7.11 |Continuum Analytics, Inc.| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org

>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
True
```

Initialising NLTK with Anaconda installation

```
>>> 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
>>>
```

Text analytics with NLTK - Chapter 1

```
>>> texts()
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
>>>
```

Text analytics with NLTK - Chapter 1

```
>>> text1
<Text: Moby Dick by Herman Melville 1851>
>>> 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
>>> text1.concordance('monstrous')
```

Check the text concordance method with other texts and other search keys, e.g.
text4 (Inaugural Addresses) for ‘nation’, ‘citizen’, ‘America’, ‘terror’, ‘god’ ...

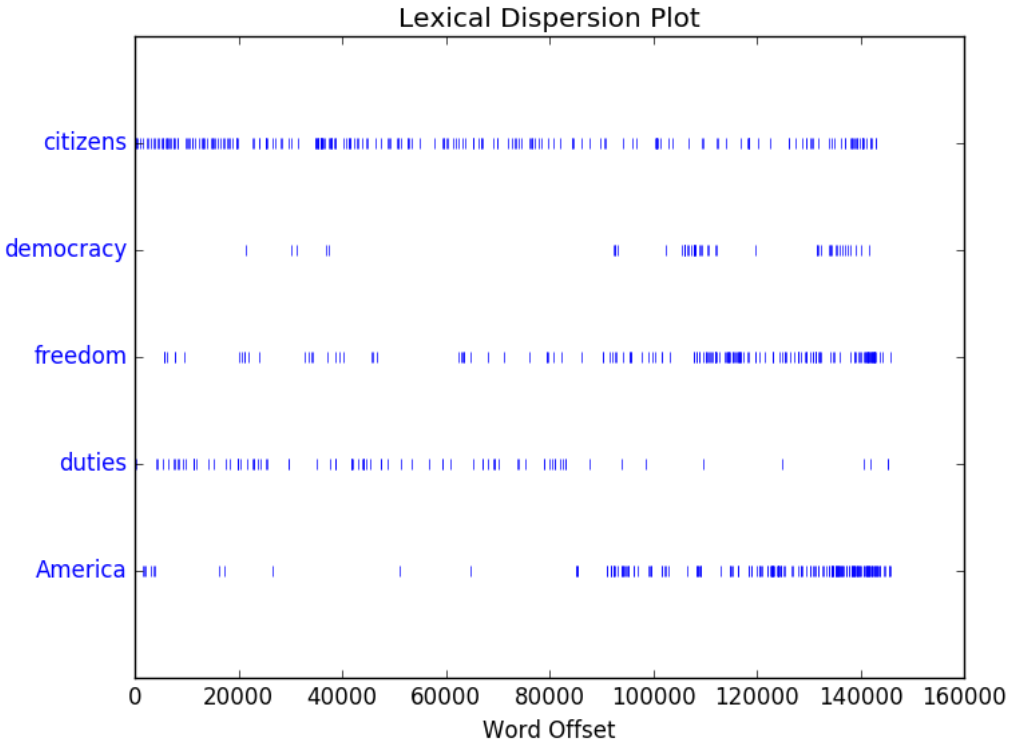
- Check other text methods:

```
>>> text1.similar('monstrous')
imperial subtly impalpable pitiable curious
abundant perilous trustworthy untoward singular
lamentable few determined maddens horrible
tyrannical lazy mystifying christian exasperate
```

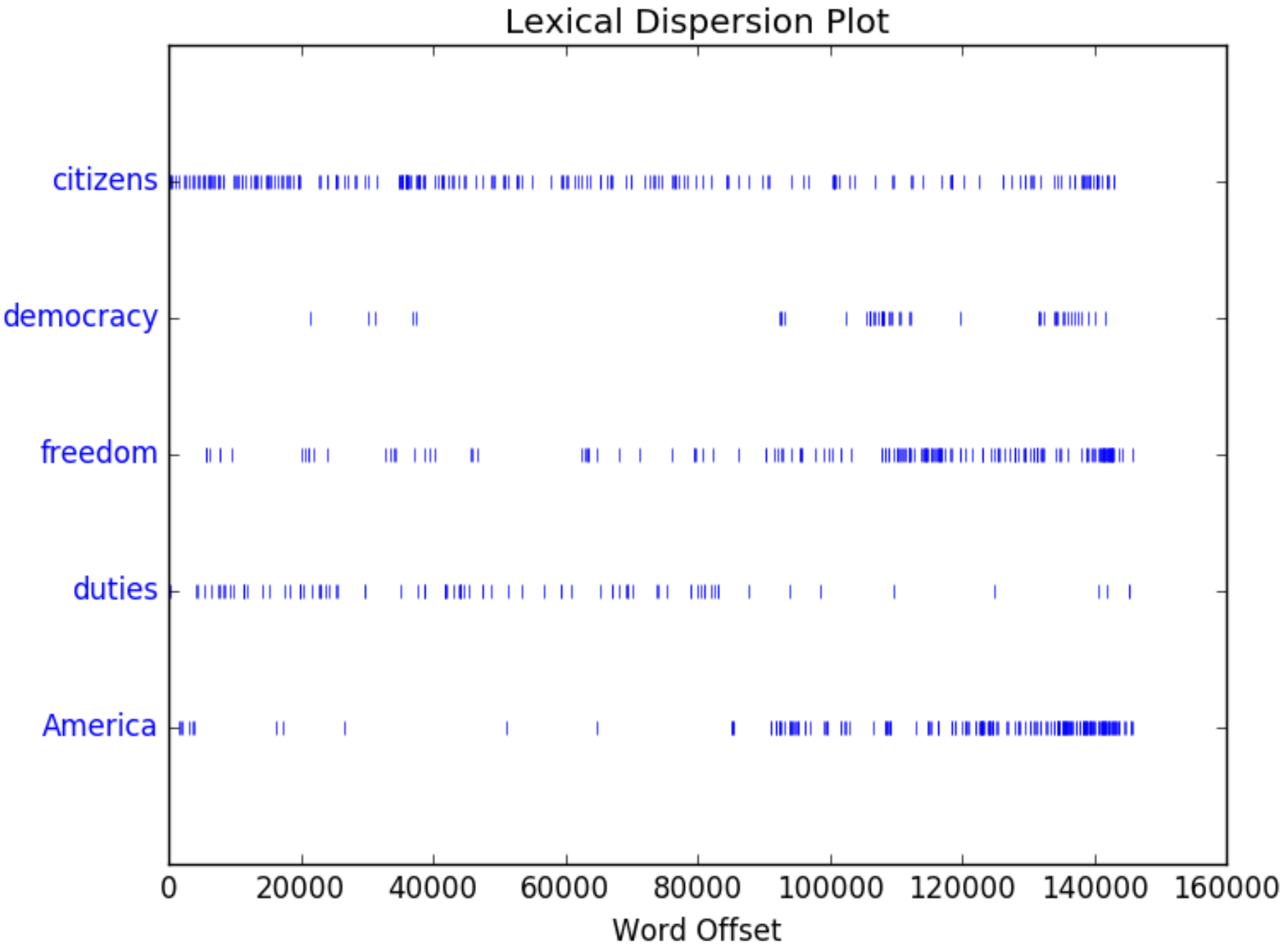
```
>>> text2.similar('monstrous')
very exceedingly so heartily a great good amazingly as sweet
remarkably extremely vast
```

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

```
>>> text4.dispersion_plot(["citizens", "democracy",
"freedom","duties", "America"])
/home/gibbon/miniconda2/lib/python2.7/site-
packages/matplotlib/font_manager.py:273: UserWarning:
Matplotlib is building the font cache using fc-list.
This may take a moment.
  warnings.warn('Matplotlib is building the font cache
using fc-list. This may take a moment.')
```



Text analytics with NLTK - Chapter 1



Text analytics with NLTK - Chapter 1

```
len(text3)
sorted(set(text3))
len(set(text3))
   float(len(set(text3))) / len(text3)
   1.0 * len(set(text3)) / len(text3)

text3.count('smote')
100 * text4.count('a') / len(text4)

def lexical_diversity(text):
    return 1.0 * len(text) / len(set(text))
def percentage(count,total):
    return 100.0 * count / total

lexical_diversity(text3)
percentage(len(set(text3)) / len(text3))
```

or:

```
from __future__ import division
```

- More text functions and methods:

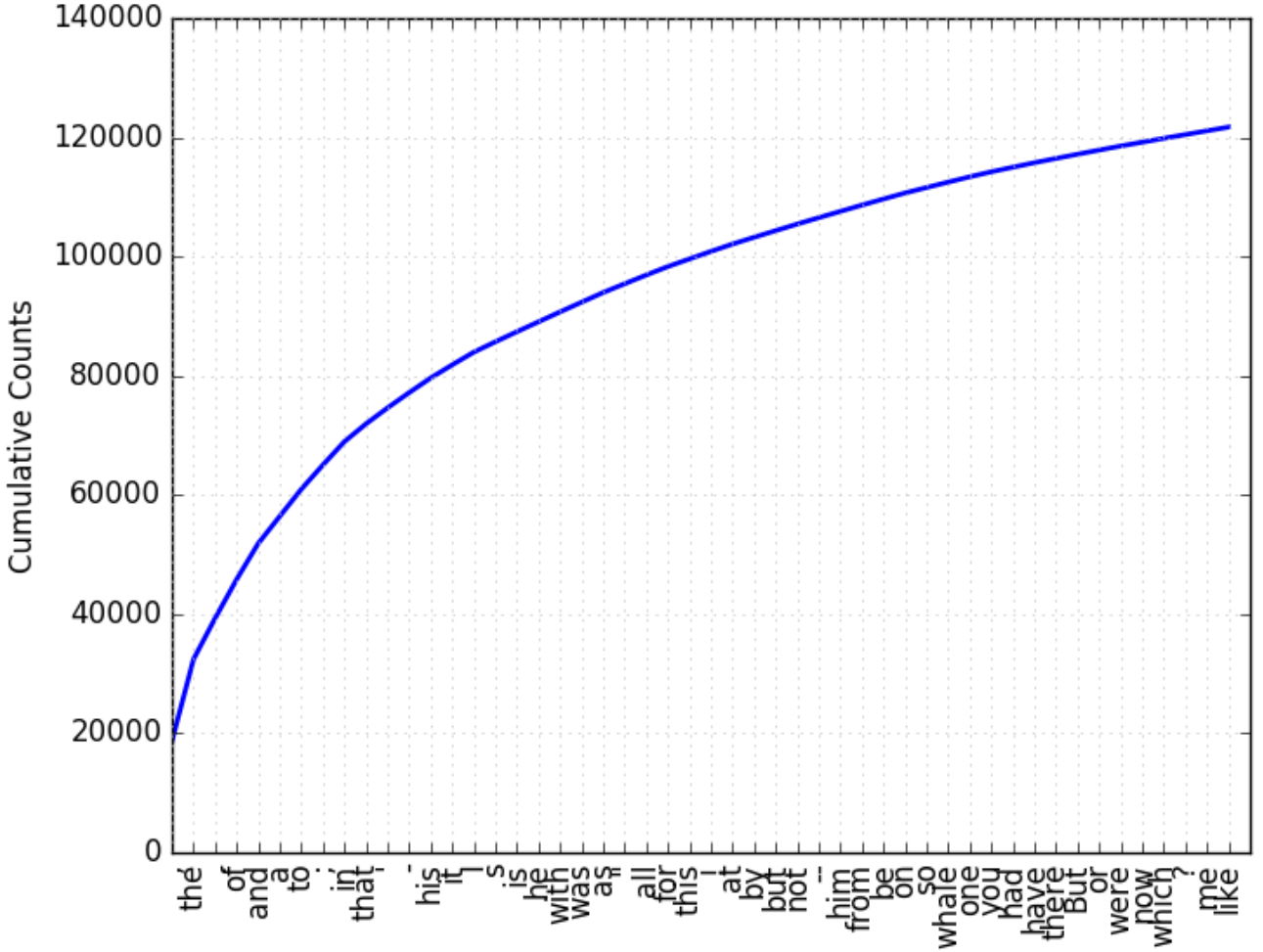
```
text4[173]
text4.index('awaken')
text5[16715:16735]
text6[1600:1625]
text2[:141525]
text2[141525:]
```

- Frequency distribution dictionary:

```
fdist1 = FreqDist(text1)
vocabulary1 = sorted(fdist1.keys())
```

Frequency distribution dictionary:

```
fdist1 = FreqDist(text1)
vocabulary1 = sorted(fdist1.keys())
fdist1.plot(50,cumulative=True)
```



- Selecting sub-vocabularies:

```
V = set(text1)
long_words = [w for w in V if len(w) > 15]
sorted(long_words)
```

- Check long words for all texts

```
fdist5 = FreqDist(text5)
sorted([w for w in set(text5) if len(w) > 7 and
fdist5[w] > 7])
```

- Difference?

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

- Work out the average word length in a text.

- Words in context:

```
nltk.bigrams(['more','is','said','than','done])

text4.collocations()
text8.collocations()
```

- Word lengths

```
fdist = FreqDist([len(w) for w in text1])
fdist
sorted(fdist.keys())
fdist.items()
fdist.max()
fdist[3]
fdist.freq(3)
help(FreqDist)
```

Text analytics with NLTK - Chapter 2

Text analytics with NLTK – Chapter 2

- Gutenberg corpus

```
nltk.corpus.gutenberg.fileids()
emma = nltk.corpus.gutenberg.words('austen-emma.txt')
emma.concordance('surprize')
```

- Or:

```
from nltk.corpus import gutenberg
gutenberg.fileids()
emma = gutenberg.words('austen-emma.txt')
```

- Statistical text features:

```
for fileid in gutenberg.fileids():
    num_chars = len(gutenberg.raw(fileid))
    num_words = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    num_vocab = len(set([w.lower() for w in
gutenberg.words(fileid)]))
print int(num_chars/num_words),
int(num_words/num_sentes), int(num_words/num_vocab),
fileid
```

Statistical text features:

```
for fileid in gutenberg.fileids():
    num_chars = len(gutenberg.raw(fileid))
    num_words = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    num_vocab = len(set([w.lower() for w in
gutenberg.words(fileid)]))
    print fileid
    print int(num_chars/num_words)
    print int(num_words/num_sents)
    print int(num_words/num_vocab)
```

Corpora

```
from nltk.corpus import webtext
for fileid in webtext.fileids():
    print fileid, webtext.raw(fileid)[:tr], ' ...'


from nltk.corpus import nps_chat
chatroom = nps_chat.posts('10-19-20s_706posts.xml')
chatroom[123]

from nltk.corpus import brown
brown.categories()
brown.words(fileids=['cg22'])
brown.sents(categories=['news','editorial','reviews'])
news_text = brown.words(categories='news')
fdist = nltk.FreqDist([w.lower() for w in news_text])
```

Corpora

```
from nltk.corpus import brown
brown.categories()
brown.words(fileids=['cg22'])
brown.sents(categories=['news','editorial','reviews'])
news_text = brown.words(categories='news')

fdist = nltk.FreqDist([w.lower() for w in news_text])

modals = ['can', 'could', 'may', 'might', 'must',
'will', 'would']

for m in modals:
    print m + ':', fdist[m]
```

Text analytics with NLTK – Chapter 2

```
from nltk.corpus import brown
cfd = nltk.ConditionalFreqDist((genre,word)
     for genre in brown.categories()
     for word in brown.words(categories=genre))
genres = ['news','religion', 'hobbies', 'science_fiction',
'romance', 'humor']
modals = ['can', 'could', 'may', 'might', 'must', 'will',
'would']
cfd.tabulate(conditions=genres, samples=modals)
```

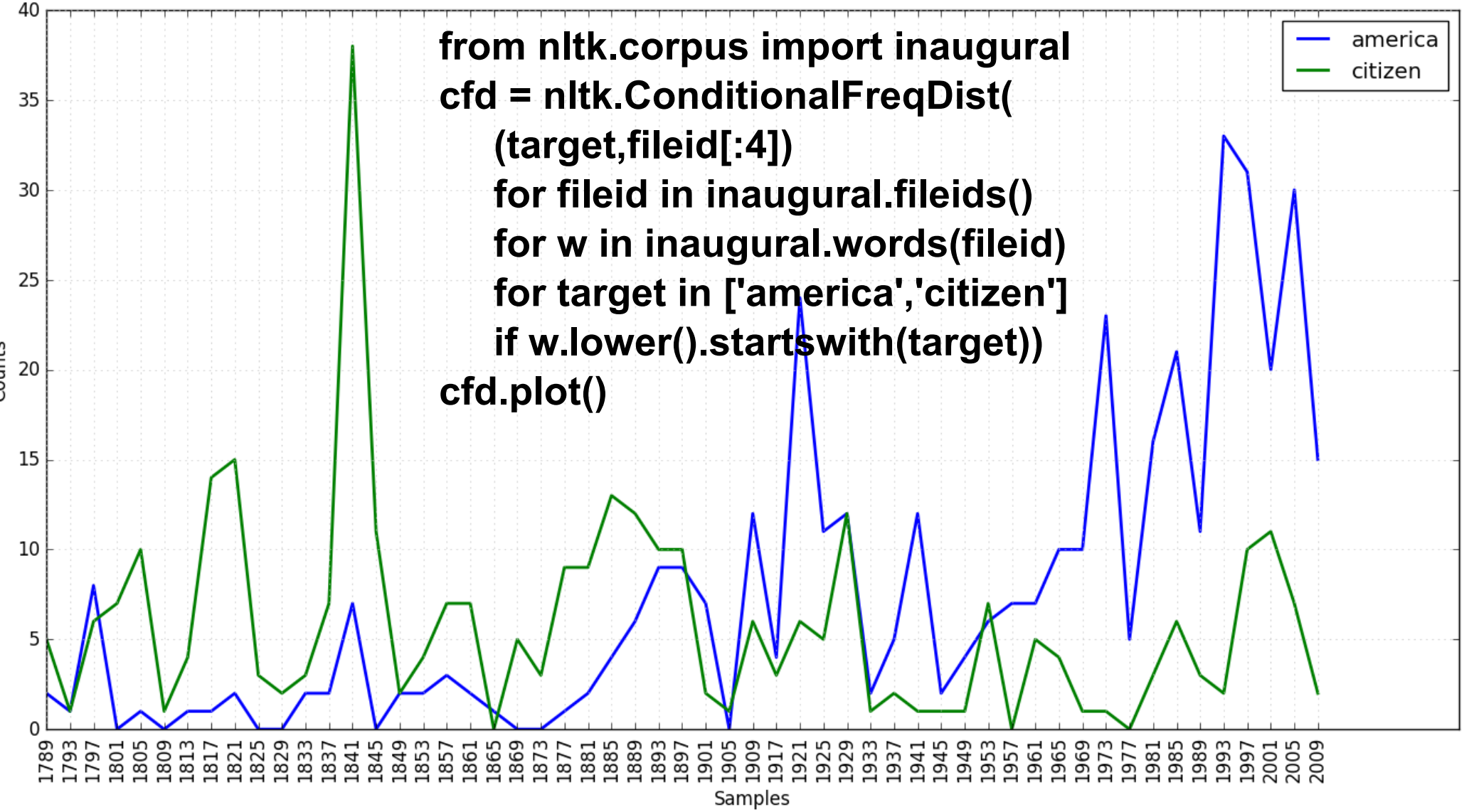
Text analytics with NLTK – Chapter 2

```
from nltk.corpus import brown
cfd = nltk.ConditionalFreqDist((genre,word)
     for genre in brown.categories()
     for word in brown.words(categories=genre))
genres = ['news','religion', 'hobbies', 'science_fiction',
'romance', 'humor']
modals = ['can', 'could', 'may', 'might', 'must', 'will',
'would']
cfd.tabulate(conditions=genres, samples=modals)
                  can could   may might  must  will would
            news    93    86    66    38    50   389   244
        religion    82    59    78    12    54    71    68
         hobbies   268    58   131    22    83   264    78
science_fiction    16    49     4    12     8    16    79
         romance    74   193    11    51    45    43   244
           humor    16    30     8     8     9    13    56
```

Text analytics with NLTK – Chapter 2

Inaugural Address corpus: Conditional Frequency Distribution

```
from nltk.corpus import inaugural
cfd = nltk.ConditionalFreqDist(
    (target,fileid[:4])
    for fileid in inaugural.fileids()
    for w in inaugural.words(fileid)
    for target in ['america','citizen']
    if w.lower().startswith(target))
cfd.plot()
```



Corpus functionality:

```
fileids()
fileids([categories])
categories()
categories([fileids])
raw(), raw(fileids=[...]), raw(categories=[...])
words(), words(fileids=[...]), words(categories=[...])
sents()
encoding(fileid)
```

Your own corpus

```
corpus_root = "/home/gibbon/mycorpus"
texts = PlaintextCorpusReader(corpus_root,".*")
texts.fileids()

# Convert into NLTK Text format:
words = nltk.Text(texts.words("goldi.txt"))

words.concordance("porridge")
Displaying 8 of 8 matches:
                                    Goldilocks and the Three Bears Once upon a t
e , there was a little girl named Goldilocks . She went for a walk in the fore
re were three bowls of porridge . Goldilocks was hungry . She tasted the porri
room where she saw three chairs . Goldilocks sat in the first chair to rest he
 to rest , it broke into pieces ! Goldilocks was very tired by this time , so
third bed and it was just right . Goldilocks fell asleep . As she was sleeping
exclaimed Baby bear . Just then , Goldilocks woke up and saw the three bears .
mped up and ran out of the room . Goldilocks ran down the stairs , opened the
```

- Conditional Frequency Distributions
  -
  - List of pairs: [(condition, event), …]
  - Word lengths per language:

```
from nltk.corpus import udhr
languages = ['Chickasaw','English','German_Deutsch',
'Greenlandic_Inuktikut', 'Hungarian_Magyar',
'Ibibio_Efik']


cfd = nltk.ConditionalFreqDist((lang,len(word))
    for lang in languages
    for word in udhr.words(lang + '-Latin1'))

cfd.tabulate(languages, samples=range(10))
cfd.tabulate(languages, samples=range(10),
    cumulative=True)
```

Text analytics with NLTK – Chapter 2

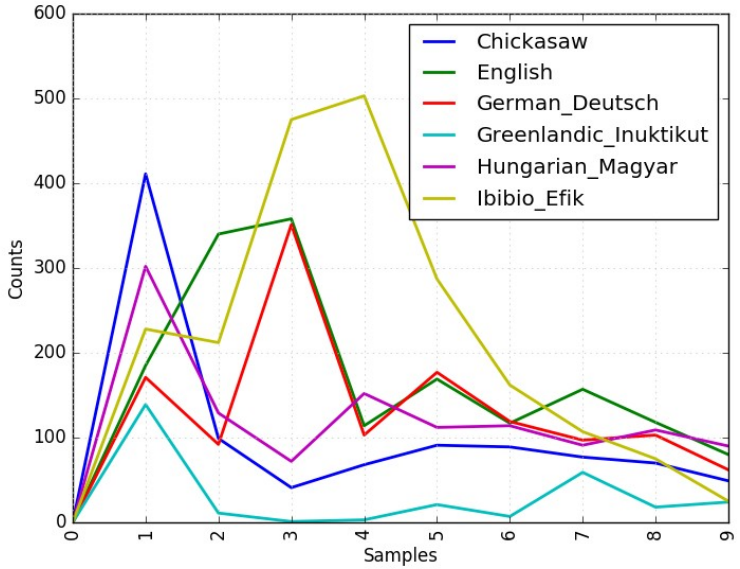
```
cfd.tabulate(languages, samples=range(10))
                      0   1   2   3   4   5   6   7   8   9
           Chickasaw  0 411  99  41  68  91  89  77  70  49
             English  0 185 340 358 114 169 117 157 118  80
      German_Deutsch  0 171  92 351 103 177 119  97 103  62
Greenlandic_Inuktikut 0 139  11   1   3  21   7  59  18  24
    Hungarian_Magyar  0 302 129  72 152 112 114  91 109  90
         Ibibio_Efik  0 228 212 475 503 287 162 107  75  25
```

```
cfd.tabulate(languages, samples=range(10), cumulative=True)
                      0    1    2    3    4    5    6    7    8    9
           Chickasaw  0  411  510  551  619  710  799  876  946  995
             English  0  185  525  883  997 1166 1283 1440 1558 1638
      German_Deutsch  0  171  263  614  717  894 1013 1110 1213 1275
Greenlandic_Inuktikut 0  139  150  151  154  175  182  241  259  283
    Hungarian_Magyar  0  302  431  503  655  767  881  972 1081 1171
         Ibibio_Efik  0  228  440  915 1418 1705 1867 1974 2049 2074
```

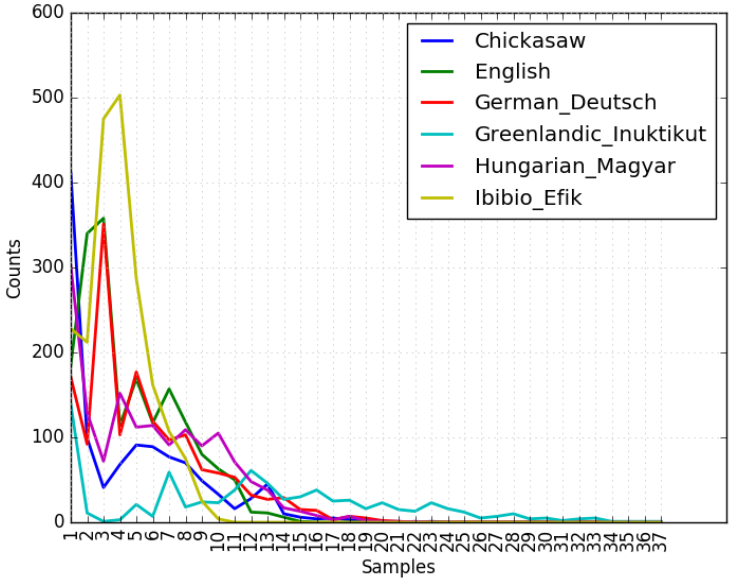
Text analytics with NLTK – Chapter 2

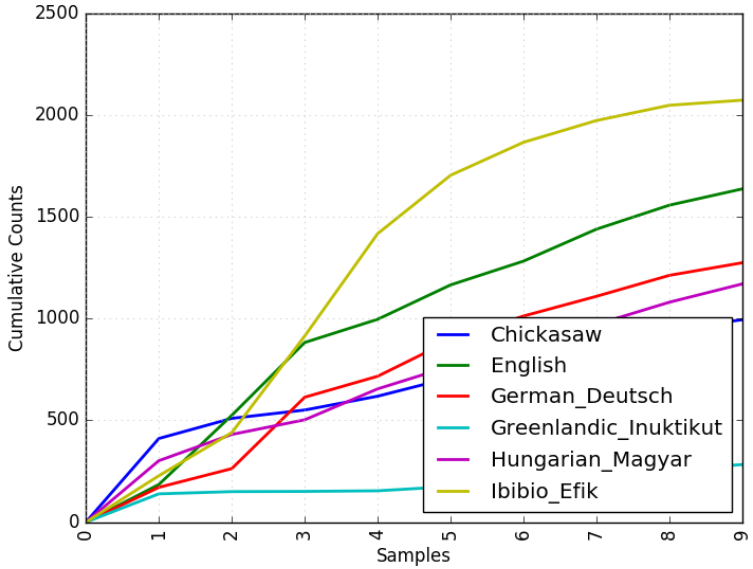
cfd.plot(conditions=languages, samples=range(10))

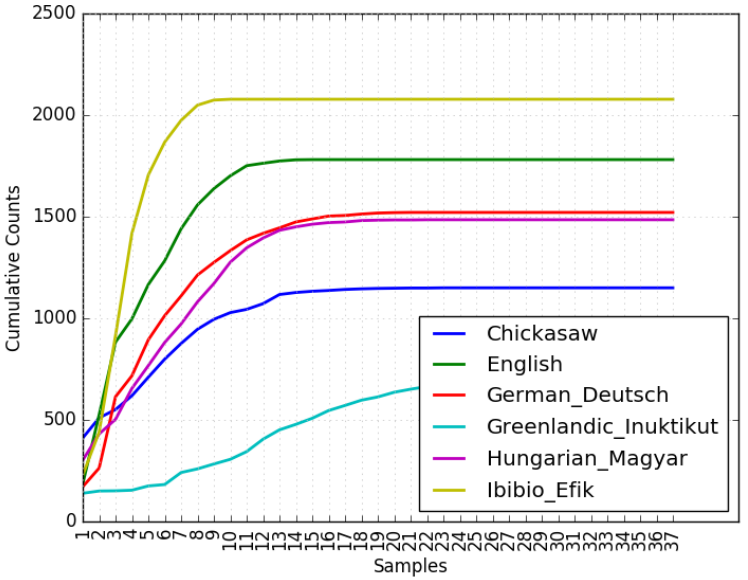
right-hand side:
all samples



ample







Text analytics with NLTK - tokenisation

```
#!/home/gibbon/miniconda2/bin/python
# Simple POS tagging from NLTK Ch. 5

import nltk

text = word_tokenize("And now for something completely different")
# Output wordds and tags as pairs
nltk.pos_tag(text)
# e.g.: [('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]

text = word_tokenize("They refuse to permit us to obtain the refuse
permit")
nltk.pos_tag(text)
# e.g. [('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us',
'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN' ('permit',
'NN')]
```

End of Unit 6