Thapar Summer School 2022

(5th July 2022)

NLP with NLTK and PYTHON

Dr. Nitin Arvind Shelke

Corpus

A text corpus is a large, structured collection of texts.

Examples are

- Gutenberg Corpus
- Web and Chat Text Corpus
- Brown Corpus
- Reuters Corpus
- Inaugural Address Corpus

Gutenberg Corpus

NLTK includes a small selection of texts from the Project Gutenberg electronic text archive, which contains some 25,000 free electronic books, hosted at http://www.gutenberg.org/.

To import this corpus, we use following command:

>>>>from nltk.corpus import gutenberg

Gutenberg Corpus

These methods associated with this corpus are:

>>>gutenberg.fileids()- gives file identifiers

>>>gutenberg.raw()- the contents of the file without any linguistic processing.

>>> gutenberg.words()-divides the text up into its words

>>>gutenberg.sents()-divides the text up into its sentences where each sentence is a list of words.

Gutenberg Corpus

```
>>> emma = gt.words('austen-emma.txt')
>>> len(emma)



>>> emma = nltk.Text(gt.words('austen-emma.txt'))
>>> emma.concordance("surprize")
```

Gutenberg Corpus

Method details

raw(fileids=[f1,f2,f3])[:] The raw content of the specified files

words(fileids=[f1,f2,f3])[:] The words of the specified fileids

sents(fileids=[f1,f2,f3])[:] The sentences of the specified fileids

Gutenberg Corpus

Write a code to find the Number of characters, words and sentences for the respective fileids

Gutenberg Corpus

Write a program which gives three important statistics required in number of applications: average word length, average sentence length, and the lexical diversity.

Web Text corpus

NLTK's small collection of web text includes content from a Firefox discussion forum, conversations overheard in New York, the movie script of Pirates of the Carribean, personal advertisements, and wine reviews.

```
>>>from nltk.corpus import webtext
>>> for fileid in webtext.fileids():
... print fileid, webtext.raw(fileid)[:65], '...'
...
```

Web Text corpus

Write a program which gives three important statistics required in number of applications: average word length, average sentence length, and the lexical diversity.

Chat Text Corpus

- It is a corpus of instant messaging chat sessions, originally collected by the Naval Postgraduate School for research on automatic detection of Internet predators.
- The corpus contains over 10,000 posts. where each file contains several hundred posts collected on a given date, for an age-specific chatroom (teens, 20s, 30s, 40s, plus a generic adults chatroom).
- The filename contains the date, chatroom, and number of posts; e.g., 10-19-20s_706posts.xml contains 706 posts gathered from the 20s chat room on 10/19/2006.

>>>>from nltk.corpus import nps_chat

Chat Text Corpus

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

Brown Corpus

The Brown Corpus was the first million-word electronic corpus of English, created in 1961 at Brown University.

This corpus contains text from 500 sources, and the sources have been categorized by genre, such as news, editorial, and so on.

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

Brown Corpus

Write a program to count the model verb such as 'can', 'could', 'may', 'might', 'must', 'will' from the category's news

Conditional frequency distribution

- When the texts of a corpus are divided into several categories (by genre, topic, author, etc.), we can maintain separate frequency distributions for each category.
- A conditional frequency distribution is a collection of frequency distributions, each one for a different "condition."
- A conditional frequency distribution needs to pair each event with a condition.

CFD

Most Commonly used methods

cfdist = ConditionalFreqDist(pairs) //create a conditional frequency distribution from a list of pairs

cfdist.conditions() //the conditions

cfdist[condition] //the frequency distribution for this condition

cfdist[condition][sample] //frequency for the given sample for this condition

cfdist.tabulate(samples, conditions) //tabulation limited to the specified samples and conditions

cfdist.plot(samples, conditions) //graphical plot limited to the specified samples and conditions

CFD

Write a program to count the model verb such as 'can', 'could', 'may', 'might', 'must', 'will' from the list of categories such as

'news', 'religion', 'hobbies', 'science_fiction', 'romance', 'humor'

Reuters Corpus (HW)

The Reuters Corpus contains 10,788 news documents totaling 1.3 million words.

```
>>> from nltk.corpus import reuters
>>> reuters.fileids()
>>> reuters.categories()
>>> reuters.categories('training/9865')
>>> reuters.categories(['training/9865', 'training/9880'])
>>> reuters.fileids('barley')
>>> reuters.fileids(['barley', 'corn'])
```

Inaugural Address Corpus (HW)

This corpus is actually a collection of 55 texts, one for each presidential address. An interesting property of this collection is its time dimension:

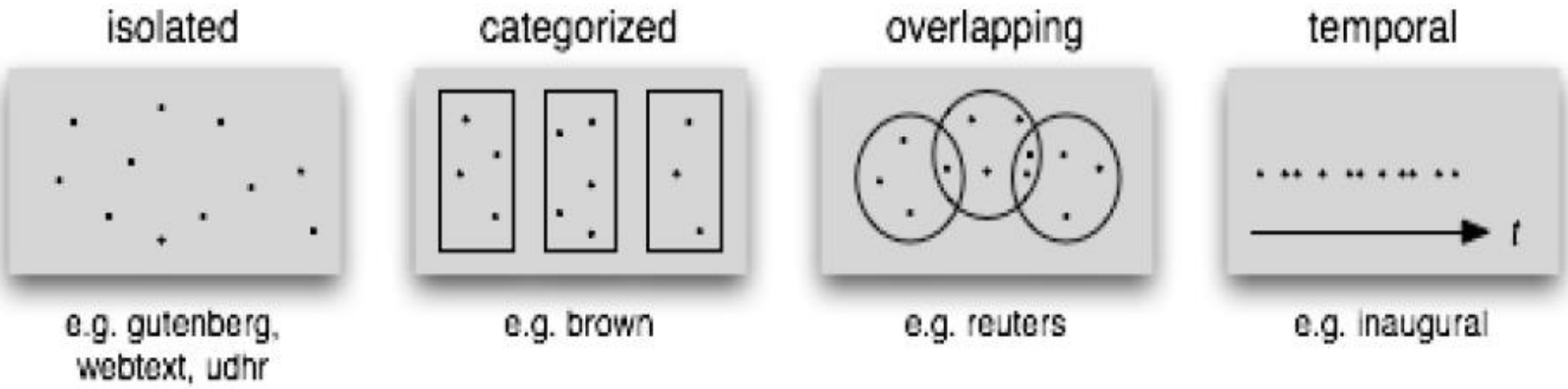
```
>>> from nltk.corpus import inaugural
>>> inaugural.fileids()
```

Inaugural Address Corpus and CFD (HW)

Lets look how the words America and citizen are used over time in Inagural Corpus. WAP to converts the words in the Inaugural corpus in lower case and checks whether they start with either of the "targets" america or citizen.

```
>>> 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()
```

Structure of Text Corpus



Loading your own Corpus

If you have your own collection of text files that you would like to access, then you can easily load them with the help of NLTK's PlaintextCorpusReader.

Extraction of Text Data from PDF file

Reading and printing single page

Pip install PyPDF2

```
>>> import PyPDF2
>>> path = 'C:\\Users\\Nitin Arvind Shelke\\Desktop\\tod.pdf'
>>> read_pdf = PyPDF2.PdfFileReader(path)
>>> page = read_pdf.getPage(0)
>>> page_content = page.extractText()
>>> print (page_content)
```

Extraction of Text Data from PDF file

Reading and printing multiple pages

```
>>> for i in range(read_pdf.getNumPages()):
...     page = read_pdf.getPage(i)
...     print ('Page No - ' + str(1+read_pdf.getPageNumber(page)))
...     page_content = page.extractText()
...     print (page_content)
...
```

Extraction of Text Data from DOC file

printing out the text

```
pip install python-docx
>>> import docx
>>> def readdoc(filename):
...     doc = docx.Document(filename)
...     fullText = []
...     for para in doc.paragraphs:
...             fullText.append(para.text)
...     return '\n'.join(fullText)
...
>>> print (readdoc('C:\\Users\\Nitin Arvind
   Shelke\\Desktop\\Types of Databases.docx'))
```

Lexical Resources

A lexicon, or lexical resource, is a collection of words and/or phrases along with associated information such as part of speech and sense definitions.

- Wordlist Corpora
- Name Corpus
- Comparative Wordlists
- WordNet
- Stopword

Wordlist Corpus

- NLTK includes some corpus that are nothing more than wordlists.
- We can use it to find unusual or misspelled words in a text corpus
- words is the keyword

```
>>>from nltk.corpus import words
```

fileids()

words()

Wordlist Corpus(Program 1)

Write a Code for Filtering a Text that computes the vocabulary of a text, then removes all items that occur in an existing wordlist, leaving just the uncommon or misspelled words.

wordlist (Program 2)

Puzzle code : A wordlist is useful for solving word puzzles. How many words of six letters or more can you make from those shown (agivrnlpe)? With condition that each letter must be used once only and must contain center letter

a	g	i
v	r	n
l	p	e

wordlist (HW)

Puzzle code : A wordlist is useful for solving word puzzles. How many words of six letters or more can you make from those shown (agivrnlpe)? With condition that each letter must be used once only and must contain center letter

A	J	K	L
I	B	F	M
H	Y	E	D
G	O	N	C

Name Corpus

One more wordlist corpus is the Names corpus, containing 8,000 first names categorized by gender. The male and female names are stored in separate files.

```
>>> names = nltk.corpus.names
>>> names.fileids()
>>> male_names = names.words('male.txt')
>>> female_names = names.words('female.txt')
```

Let's find names which appear in both files, i.e. names that are ambiguous for gender:

```
>>> [w for w in male_names if w in female_names]
```

Name Corpus and CFD

Draw a plot that shows the number of female and male names ending with each letter of the alphabet.

```
cfd = nltk.ConditionalFreqDist(
        (fileid, name[-1])
        for fileid in names.fileids()
        for name in names.words(fileid))
>>> cfd.plot()
```

most names ending with a, e or i are female; names ending in h and l are equally likely to be male or female; names ending in k, o, r, s, and t are likely to be male.

Comparative Wordlists

Another example of a tabular lexicon is the comparative wordlist. NLTK includes so-called Swadesh wordlists, lists of about 200 common words in several languages.

```
>>> from nltk.corpus import swadesh
>>> swadesh.fileids()
>>> swadesh.words('en')
```

Comparative Wordlists

We can access cognate words from multiple languages using the entries() method, specifying a list of languages.

```
>>> fr2en = swadesh.entries(['fr', 'en'])
>>> fr2en
>>> translate = dict(fr2en)
>>> translate['chien']
'dog'
>>> translate['jeter']
'throw'
```

WordNet

WordNet is a lexical database for the English language, which was created by Princeton, and is part of the NLTK corpus.

WordNet is a semantically oriented dictionary of English, similar to a traditional thesaurus but with a richer structure. NLTK includes the English WordNet, with 155,287 words and 117,659 synonym sets.

```
>>>from nltk.corpus import wordnet
>>>syn = wordnet.synsets("pain")
>>> print(syn[0].lemma_names())
>>>print(syn[0].definition())
>>>print(syn[0].examples())
```

WordNet

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
[Synset('car.n.01')]   // having only one synset
```

The entity car.n.01 is called a synset, or "synonym set", a collection of synonymous words (or "lemmas"):

```
>>> wn.synset('car.n.01').lemma_names()
['car', 'auto', 'automobile', 'machine', 'motorcar']
```

WordNet

```
>>> wn.synset('car.n.01').lemma_names()
>>> wn.synset('car.n.02').lemma_names()
>>> wn.synset('car.n.03').lemma_names()
>>> wn.synset('car.n.04').lemma_names()
>>> wn.synset('car.n.05').lemma_names()
```

WordNet

the word *car* is ambiguous, having five synsets

```
>>> wn.synsets(“car”)
>>> for synset in wn.synsets('car'):
... print(synset.lemma_names())
..
>>> for synset in wn.synsets('car'):
...     print(synset.definition())
...
>>> for synset in wn.synsets('car'):
...     print(synset.examples())
...
```

Example with word ‘program’

WordNet

compare the similarity index of any two words

```
import nltk
from nltk.corpus import wordnet

w1 = wordnet.synset('run.v.01') # v here denotes the tag verb
w2 = wordnet.synset('sprint.v.01')
print(w1.wup_similarity(w2))

# Let's compare the noun of "ship" and "boat:"  ‘car’ and ‘motorcar’
```

Stopwords

- There is also a corpus of stopwords, that is, high-frequency of words like “the, to and also” that we sometimes want to filter out from a document before further processing.
- Stopwords usually have little lexical content, and their presence in a text fails to distinguish it from other texts.

```
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
```

Stopwords (Program 1)

WAP to define a function that filter the words in a text are not in the stopwords list (remove the stopwords from the text file)

Stopwords (Program 2)

WAP to define a function that compute what fraction of words in a text are not in the stopwords list:

```
>>> def content_fraction(text):
...     stopwords = nltk.corpus.stopwords.words('english')
...     content = [w for w in text if w.lower() not in stopwords]
...     return len(content) / len(text)
...
```

To call function :

```
>>> content_fraction(nltk.corpus.gutenberg.words())
```

Stopwords (Program 3)

Remove the stopowords from the input_text

```
def filter_content(text):
    stopwords = set(nltk.corpus.stopwords.words('english'))
    filter_content = [w for w in text if w.lower() not in stopwords]
    fraction= len(filter_content) / len(text)
    return filter_content,fraction


input_text = "Game of Thrones……"
text = nltk.Text( input_text.split() )


filter_content, fraction=filter_content(text)
print(filter_content)
print(fraction)
```

Thank You

nitinashelke@gmail.com