# Accessing Text Corpora

Lab Session-II (a)

Dr. Jasmeet Singh,
Assistant Professor,
CSED, TIET

# Text Corpora

- In Natural Language Processing typically uses large bodies of linguistic data, or **corpora.**
- In this session we will learn:
  - Some useful text corpora and lexical resources, and how can we access them with Python.
  - The Python constructs are most helpful for this work.
  - All the text corpora are provided in **corpus** package of nltk.

# Gutenberg Corpus

- NLTK includes a small selection of texts from the Project Gutenberg electronic text archive, which contains some 25,000 free electronic books.
- To import this corpus, we use following command:
- from nltk.corpus import gutenberg
- Once, it has been imported we can use various methods with the gutenbetrg object of package corpus. These methods 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.
- raw(fileids=[f1,f2,f3]), words(fileids=[f1,f2,f3]), sents(fileids=[f1,f2,f3]) gives the text, words, and sentences respectively in the specified file ids.

# Example 1

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

```
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 num_chars,num_words,num_sents,num_vocab, int(num_chars/num_words), int(num_words/num_sents), int(num_words/num_vocab),fileid
...
```

# Example 2

- To find longest sentence in a particular file in the corpus
- sentences=gutenberg.sents('austen-emma.txt')
- long_sent=max([len(s) for s in sentences])
- [s for s in gutenberg.sents('austen-emma.txt') if len(s)=long_sent]

- **Try to print long sentences in all the files of gutenberg corpus**

# Web and Chat 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
- There is also 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.
- The corpus is organized into 15 files, 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).
- from nltk.corpus import nps_chat

# 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()- returns the list of categories/genre of brown corpus
- brown.fileids([categories]), brown.raw(categories=[c1,c2,c3]), brown.words(categories=[c1,c2,c3]), brown.sents(categories=[c1,c2,c3]), returns fileids, text, words, sentences of brown corpus from the specified categories.

# Example 3

The Brown Corpus is a convenient resource for studying systematic differences between genres, a kind of linguistic inquiry known as **stylistics**.

For example, the program below compares genres in their usage of modal verbs.

```
cat=brown.categories()
modals=['can','could','may','might','must','will','would']
for category in cat:
    words=brown.words(categories=category)
    fdist=nltk.FreqDist([w.lower() for w in words])
    for modal in modals:
        print(modal,fdist[modal],category)
```

# Reuters Corpus

The Reuters Corpus contains 10,788 news documents totaling 1.3 million words.
The documents have been classified into 90 topics, and grouped into two sets, called “training” and “test”; thus, the text with fileid 'test/14826' is a document drawn from the test set.

```
from nltk.corpus import reuters
reuters.fileids()
reuters.categories()
```

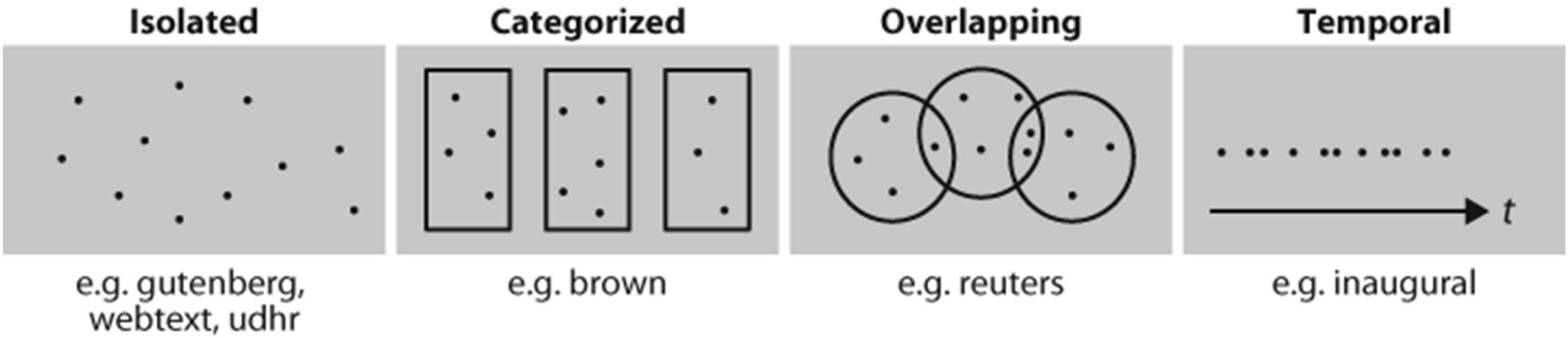
Unlike the Brown Corpus, categories in the Reuters Corpus overlap with each other, simply because a news story often covers multiple topics.

# Inagural Address Corpus

The corpus is 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

# Text Corpus Structure




**Common structures for text corpora:**
**The simplest kind of corpus is a collection of isolated texts with no particular organization**
**some corpora are structured into categories, such as genre (Brown Corpus);**
**some categorizations overlap, such as topic categories (Reuters Corpus);**
**Other corpora represent language use over time (Inaugural Address Corpus)**

# Loading Your Own Corpus

If we have a your own collection of text files that you would like to access using the methods discussed earlier, you can easily load them with the help of NLTK's PlaintextCorpusReader.

It takes two parameters:
1. corpus_root- the location of corpus files.
2. initializer -can be a list of fileids, like ['a.txt', 'test/b.txt'] , or a pattern that matches all fileids, like '[abc]/.*\.txt'

```
from nltk.corpus import PlaintextCorpusReader
corpus_root = '/usr/share/dict'
wordlists = PlaintextCorpusReader(corpus_root, '.*')
```

# Conditional Frequency Distributions

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.

So instead of processing a sequence of words , we have to process a sequence of pairs

Each pair has the form (condition , event) . If we were processing the entire Brown Corpus by genre, there would be 15 conditions (one per genre) and 1,161,192 events (one per word).

# Example 4

Counting words by Genre in brown Corpus

```
cfd = nltk.ConditionalFreqDist(
…       (genre, word)
…       for genre in brown.categories()
…       for word in brown.words(categories=genre))
```

# Example 5

**Counting how the words America and citizen are used over time in Inagural Corpus**
**The following code 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,file[:4])
...     for file in inaugural.fileids()
...     for w in inaugural.words(file)
...     for target in ['america','citizen']
...     if w.lower().startswith(target))

cfd.plot()
```

# Plotting and Tabulating Distributions

A ConditionalFreqDist provides some useful methods for tabulation and plotting namely plot() and tabulate() methods

In these methods, we can optionally specify which conditions to display with a conditions= parameter. When we omit it, we get all the conditions.

Similarly, we can limit the samples to display with a samples= parameter.
For example, cfd.tabulate(condition=['news','religion'],samples = ['can','could'])

# Plotting and Tabulating Distributions

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

cfdist.conditions() //Alphabetically sorted list of conditions

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

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

cfdist.tabulate() //Tabulate the conditional frequency distribution

cfdist.tabulate(samples, conditions) //Tabulation limited to the