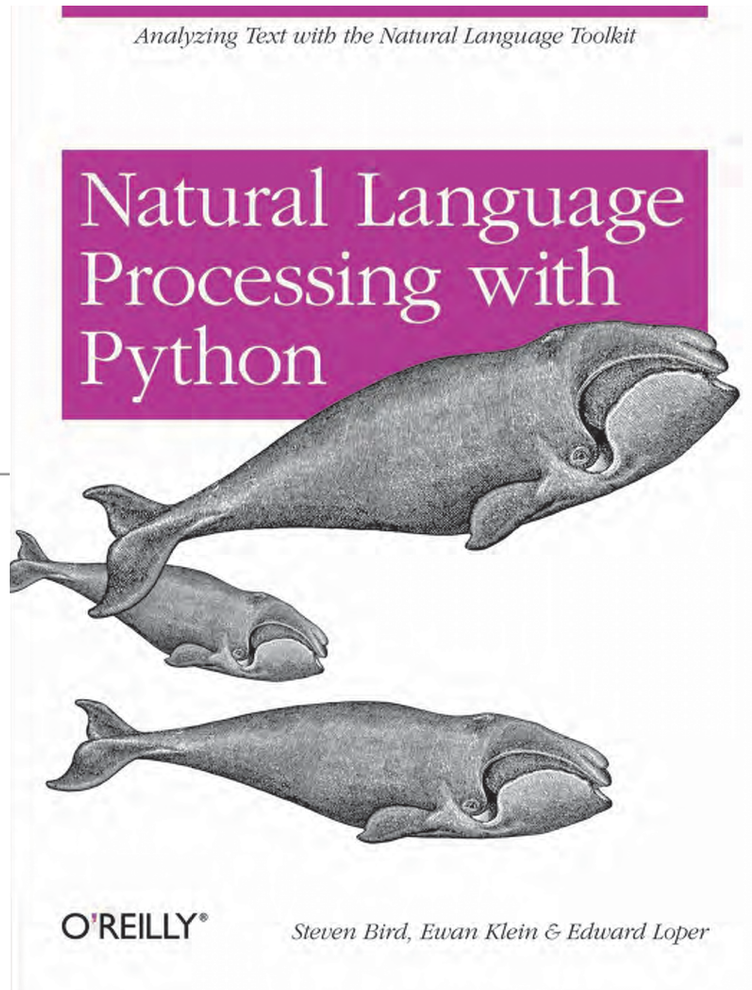
Accessing Text Corpora and Lexical Resources

Sutee Sudprasert

เนื้อหาจากบทที่ 2 ของหนังสือ
Natural Language Processing with Python



คำถามก่อนเรื่อง

- อะไรคือประโยชน์ของคลังข้อความ (text corpora) และทรัพยากรคำศัพท์ (lexical resources) และเราจะใช้ Python เข้าถึงข้อมูลเหล่านี้ได้อย่างไร?
- เราจะใช้ Python อย่างไร ให้เหมาะสมสำหรับงานนี้?
- ทำอย่างไรจึงจะลดการทำซ้ำเมื่อเขียนโค้ดด้วย Python?

Accessing Text Corpora

- คลังข้อความคือการรวบรวมเอกสารจำนวนมากไว้ด้วยกัน และอาจมีการแบ่งหมวดหมู่ เช่น คลังประโยคข่าว อาจแบ่งเป็นหมวด กีฬา บันเทิง อาชญากรรม และ การเมือง

- ในบทที่ 1 เราได้เห็น text1, text2, ..., text9 แล้ว ซึ่งนั้นเป็นตัวอย่างหนึ่งของคลังข้อความ ที่ nltk.book ได้นำเข้ามาใช้เป็นตัวอย่าง

- ต่อไปเราจะได้เรียนรู้วิธีเรียกใช้งานคลังข้อความและวิธีการทำงานกับคลังข้อความโดยละเอียด

Gutenberg Corpus

- Project Gutenberg เป็นโครงการรวบรวมหนังสือที่ไม่มีลิขสิทธิ์แล้วจัดเก็บไว้ในรูปแบบดิจิทัล http://www.gutenberg.org/ ซึ่งปัจจุบันมีหนังสือมากกว่า 25,000 เล่ม

- NLTK ได้รวมคลังข้อความบางส่วนจาก Project Gutenberg เอาไว้

```
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
```

- การดึงข้อความบางส่วนมาใช้

```
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
```

Gutenberg Corpus

- หา concordance จากคลังข้อความ

```
>>> emma = nltk.Text(nltk.corpus.gutenberg.words \
... ('austen-emma.txt'))
>>> emma.concordance('surpize')
```

- หากต้องการใช้เฉพาะส่วนใดส่วนหนึ่งของ NLTK เท่านั้นสามารถเขียนย่อได้ดังนี้

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

Gutenberg Corpus

- ตัวอย่างโปรแกรมสำหรับแสดงข้อมูลของแต่ละเอกสารในคลังข้อความ

```
>>> for fileid in gutenberg.fileids():
...     number_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_sents),
        int(num_words/num_vocab), fileid
...
```

- โปรแกรมแสดงข้อมูลสามอย่างคือ ความยาวเฉลี่ยของคำ ความยาวเฉลี่ยของประโยค และ จำนวนครั้งโดยเฉลี่ยที่คำศัพท์หนึ่งคำจะถูกใช้

Web and Chat Text

- webtext เป็นคลังข้อความใน NLTK ที่รวบรวมข้อความที่มาจากอินเทอร์เน็ต

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

- nps_chat เป็นคลังข้อความที่ถูกรวบรวมจากบทสนทนาผ่านทาง instant messaging โดยเนื้อหาจะปกปิดชื่อผู้ใช้ และ แบ่งหมวดหมู่ตามอายุ (teens, 20s, 30s, 40s, adults)

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

Brown Corpus

- Brown Corpus เป็นคลังข้อความดิจิทัลสำหรับภาษาอังกฤษอันแรกที่มีขนาด 1 ล้านคำ ซึ่งถูกสร้างขึ้นในปี 1961 ที่ Brown University

- คลังข้อความได้มาจาก 500 แหล่งข้อมูลและได้ถูกแบ่งแยกเป็นหมวดหมู่ เช่น news, editorials

- http://icame.uib.no/brown/bcm-los.html

Brown Corpus

ID	File	Genre	Description
A16	*ca16*	news	Chicago Tribune: *Society Reportage*
B02	*cb02*	editorial	Christian Science Monitor: *Editorials*
C17	*cc17*	reviews	Time Magazine: *Reviews*
D12	*cd12*	religion	Underwood: *Probing the Ethics of Realtors*
E36	*ce36*	hobbies	Norling: *Renting a Car in Europe*
F25	*cf25*	lore	Boroff: *Jewish Teenage Culture*
G22	*cg22*	belles_lettres	Reiner: *Coping with Runaway Technology*
H15	*ch15*	government	US Office of Civil and Defence Mobilization: *The Family Fallout Shelter*
J17	*cj19*	learned	Mosteller: *Probability with Statistical Applications*
K04	*ck04*	fiction	W.E.B. Du Bois: *Worlds of Color*
L13	*cl13*	mystery	Hitchens: *Footsteps in the Night*
M01	*cm01*	science_fiction	Heinlein: *Stranger in a Strange Land*
N14	*cn15*	adventure	Field: *Rattlesnake Ridge*
P12	*cp12*	romance	Callaghan: *A Passion in Rome*
R06	*cr06*	humor	Thurber: *The Future, If Any, of Comedy*

Brown Corpus

- สำหรับ Brown Corpus เราสามารถเลือกข้อความตามหมวดหมู่ได้

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

- ตรวจสอบสไตล์การเขียนของแต่ละหมวดหมู่

```
>>> from nltk.corpus import brown
>>> news_text = brown.words(categories='news')
>>> fdist = nltk.FreqDist([w.lower() for w in news_text])
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> for m in modals:
...     print m + ':', fdist[m],
...
```

Brown Corpus

- สามารถใช้ ConditionalFreqDist มาช่วยได้

```
>>> 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']
>>> cfd.tabulate(conditions=genres, samples=modals)
```

Inaugural Address Corpus

- คลังข้อความที่รวบรวมสุนทรพจน์ของประธานาธิบดีของอเมริกาในวันแรกที่ขึ้นรับตำแหน่งอย่างเป็นทางการ ซึ่งมีทั้งหมด 55 ข้อความ

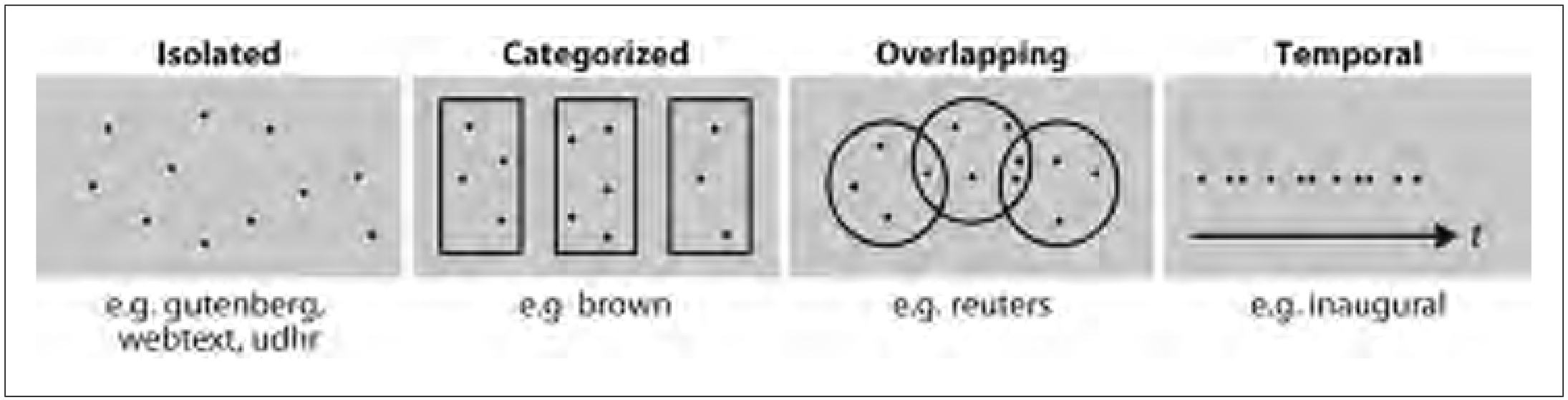
```
>>> from nltk.corpus import inaugural
>>> inaugural.fileids()
>>> [fileid[:4] for fileid in inaugural.fileids()]
```

- วิเคราะห์ว่าสุนทรพจน์ที่กล่าวมักใช้คำว่า America และ citizen มากขนาดไหน

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

Text Corpus Structure

- โครงสร้างของคลังข้อความมีหลายรูปแบบ ปกติจะมีการแบ่งตามหมวดหมู่ แบบง่ายสุดก็คือนำเอกสารมารวมๆ กันไว้เฉยๆ



Example	Description
`fileids()`	The files of the corpus
`fileids([categories])`	The files of the corpus corresponding to these categories
`categories()`	The categories of the corpus
`categories([fileids])`	The categories of the corpus corresponding to these files
`raw()`	The raw content of the corpus
`raw(fileids=[f1,f2,f3])`	The raw content of the specified files
`raw(categories=[c1,c2])`	The raw content of the specified categories
`words()`	The words of the whole corpus
`words(fileids=[f1,f2,f3])`	The words of the specified fileids
`words(categories=[c1,c2])`	The words of the specified categories
`sents()`	The sentences of the specified categories
`sents(fileids=[f1,f2,f3])`	The sentences of the specified fileids
`sents(categories=[c1,c2])`	The sentences of the specified categories
`abspath(fileid)`	The location of the given file on disk
`encoding(fileid)`	The encoding of the file (if known)
`open(fileid)`	Open a stream for reading the given corpus file
`root()`	The path to the root of locally installed corpus
`readme()`	The contents of the README file of the corpus

Text Corpus Structure

- ความแตกต่างระหว่าง raw() words() และ sents()

```
>>> raw = gutenberg.raw("burgess-busterbrown.txt")
>>> raw[1:20]
>>> words = gutenberg.words("burgess-busterbrown.txt")
>>> words[1:20]
>>> sents = gutenberg.sents("burgess-busterbrown.txt")
>>> sents[1:20]
```

Loading Your Own Corpus

- ในกรณีที่เรามีคลังข้อความของตัวเอง เราสามารถใช้ NLTK โหลดคลังข้อความนั้นเข้ามาใช้ได้ โดยที่ NLTK ได้เตรียมคลาสสามารถคลังข้อความไว้หลายตัว เช่น

  - PlaintextCorpusReader, BracketParseCorpusReader, TaggedCorpusReader, etc.

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

Conditional Frequency Distributions

- FreqDist ใช้ในการนับคำทั้งหมดในลิสต์ที่ใส่เข้าไป แต่ในบางกรณีคลังข้อความได้ถูกแบ่งเป็นหมวดหมู่ หากเราต้องการนับคำแยกตามหมู่หมวด FreqDist ไม่สามารถทำได้

- ConditionalFreqDist เป็นตัวที่เหมาะสมกับงานนี้ เราสามารถมองได้ว่า CFD คือกลุ่มของ FD ที่แยกไปตามแต่ละ condition

Condition: News

the	𝍸 𝍸 𝍸 ||||
cute	
Monday	𝍸 ||||
could	|
will	𝍸 |||

Condition: Romance

the	𝍸 𝍸 |||
cute	|||
Monday	|
could	𝍸 𝍸 𝍸
will	||||

Conditions and Events

- FD จะนับเหตุการณ์ (event) ที่ต้องการสังเกต เช่น การเกิดขึ้นของคำในเอกสาร

```
>>> text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

- CFD จะสนใจที่เงื่อนไข (condition) ของเหตุการณ์ด้วย ดังนั้นข้อมูลนำเข้าจึงเป็นคู่ลำดับของเงื่อนไขและเหตุการณ์ (condtion, event)

```
>>> pairs = [('news','The'),('news','Fulton'),('news','County'), ...]
```

Counting Words by Genre

- จากที่ผ่านมาเราได้เห็นการใช้ CFD ในการนับ Brown Corpus แล้ว ซึ่งเราจะมาดูแต่ละส่วนของโค้ดโดยละเอียด

```
>>> from nltk.corpus import brown
>>> cfd = nltk.ConditionalFreqDist(
...             (genre, word)
...             for genre in brown.categories()
...             for word in brown.words(categories=genre))
```

- ส่วนแรกคือส่วนการวนลูปเพื่อสร้างคู่ลำดับของหมวดหมู่กับคำ

```
>>> genre_word = [(genre, word)
...               for genre in ['news', 'romance']
...               for word in brown.words(categories=genre)]
>>> len(genre_word)
>>> genre_word[:4]
>>> genre_word[-4:]
```

Counting Words by Genre

- จากนั้นเราสามารถใช้ลิสต์ genre_word ในการสร้าง CFD ได้

```
>>> cfd = nltk.ConditionalFreqDist(genre_word)
>>> cfd
>>> cfd.conditions()
```

- หากลองเข้าไปดูที่แต่ละ condition จะเห็นได้ว่าแต่ละอันคือหนึ่ง FD

```
>>> cfd['news']
>>> cfd['romance']
>>> list(cfd['romance'])
>>> cfd['romance']['could']
```

Plotting and Tabulating Distributions

- CFD มีฟังก์ชั่น plot และ tabulate เพื่อใช้ในการแสดงข้อมูล ซึ่งเราได้เห็นไปแล้ว ตอนนี้เราจะมีดูในรายละเอียดกันอีกที

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

สร้างคู่ของ target **และ ปี สำหรับทุกครั้งที่เจอ** target **ใน** file

```
>>> 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']
>>> cfd.tabulate(conditions=genres, samples=modals)
```

สามารถกำหนดขอบเขตโดยใช้ conditions **และ** samples

Plotting and Tabulating Distributions

- หากสังเกตดีๆ จะเห็นว่า เราไม่ได้ใช้ list comprehension

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

- ความแตกต่าง

```
>>> set([w.lower for w in t])
>>> set(w.lower() for w in t)
```

- ลักษณะแบบนี้เรียกว่า generator expression

Generating Random Text with Bigrams

- เราสามารถใช้ CFD ในการสร้างตารางสำหรับ bigrams (คำสองคำที่เกิดติดกัน) โดยใช้คำก่อนหน้าเป็น condition ส่วนคำที่ตามมาเป็น event

```
def generate_model(cfdist, word, num=15):
    for i in range(num):
        print word,
        word = cfdist[word].max()

text = nltk.corpus.genesis.words('english-kjv.txt')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)

>>> print cfd['living']
>>> generate_model(cfd, 'living')
```

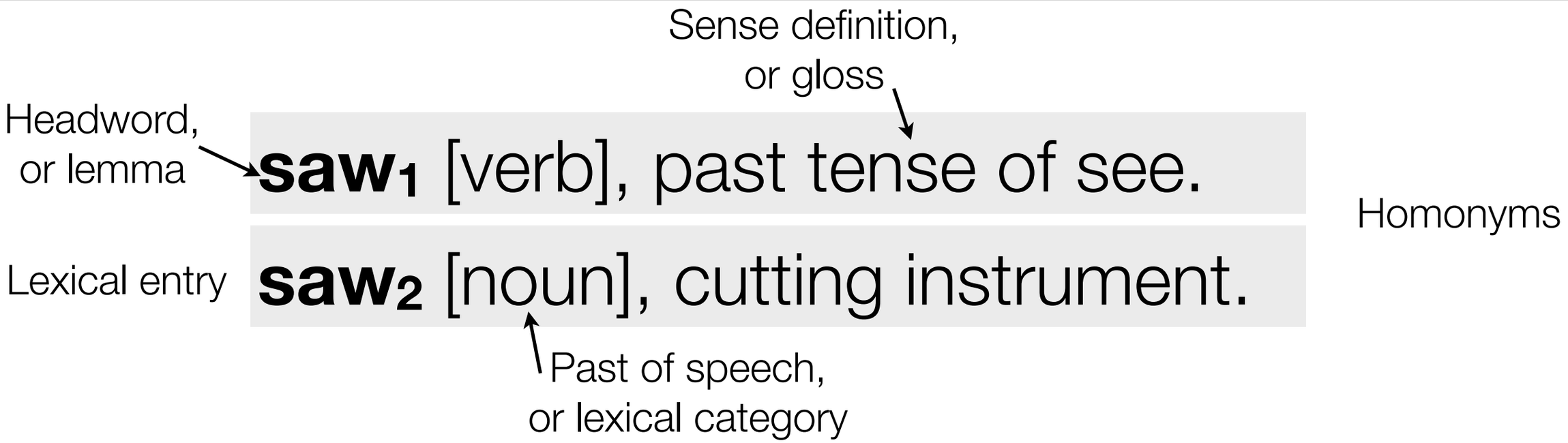
Commonly used methods

Example	Description
`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 specified samples and conditions
`cfdist.plot()`	Graphical plot of the conditional frequency distribution
`cfdist.plot(samples, conditions)`	Graphical plot limited to the specified samples and conditions
`cfdist1 < cfdist2`	Test if samples in `cfdist1` occur less frequently than in `cfdist2`

Lexical Resources

- lexicon หรือ lexical resources คือ การรวบรวมกลุ่มคำ และ/หรือ วลี เข้าด้วยกัน พร้อมกับข้อมูลที่เกี่ยวข้อง
    - ตัวอย่าง: การเก็บคำพร้อมกับความถี่ที่เกิดของคำ
        - vocab = sorted(set(my_text))
        - word_freq = FreqDist(my_text)

Lexicon terminology



- A **lexical entry** consists of a **headword** (also known as a **lemma**) along with additional information, such as the part-of-speech and the sense definition.

- Two distinct words having the same spelling are called **homonyms**.

Wordlist Corpora

- wordlist เป็น corpora อย่างง่าย โดยรวบรวมรายการของคำเอาไว้ ซึ่ง เราสามารถนำมาใช้ในการตรวจสอบคำไม่รู้จัก หรือ คำที่สะกดผิดในเอกสารได้

```
def unusual_words(text):
  text_vocab = set(w.lower() for w in text if w.isalpha())
  english_vocab = set(w.lower() for w in nltk.corpus.words.words())
  unusual = text_vocab.difference(english_vocab)
  return sorted(unusual)

>>> unusual_words(nltk.corpus.gutenberg.words('austen-sense.txt'))
```

Wordlist Corpora

- NLTK ได้เตรียม corpora ที่เรียกว่า stopwords ซึ่งรวบรวมคำที่ความถี่สูงๆ แต่เรามักจะลบทิ้งเมื่อนำเอกสารมาประมวลผล (โดยเฉพาะใช้งานด้าน IR)

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

- เมื่อเปรียบเทียบสัดส่วนระหว่างคำปกติกับ stopwords ในเอกสารจะพบว่า

```
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)

>>> content_fraction(nltk.corpus.reuters.words())
0.6599769539328526l
```

Wordlist Corpora

- Target puzzle:

E	G	I
V	R	V
O	N	L

```
puzzle_letters = nltk.FreqDist('egivrvonl')
obligatory = 'r'
wordlist = nltk.corpus.words.words()
[w for w in wordlist if len(w) >= 6
                     and obligatory in w
                     and nltk.FreqDist(w) <=  puzzle_letters]
```

Wordlist Corpora

- nltk.corpus.names เป็น corpora ที่รวบรวมชื่อคนทั้งหมดเอาไว้ โดยแบ่งแยกตามเพศ

- ตัวอย่าง: หาชื่อที่ใช้ได้ทั้งผู้ชายและผู้หญิง

```
names = nltk.corpus.names
male_names = names.words('male.txt')
female_names = names.words('female.txt')
[w for w in male_names if w in female_names]
```

- ตัวอย่าง: วาดกราฟเพื่อดูว่าตัวอักษรที่ลงท้ายชื่อของผู้ชายและผู้หญิง

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

WordNet

- WordNet คือพจนานุกรมภาษาอังกฤษที่แต่ละคำถูกกำกับด้วยข้อมูลโครงสร้างทางความหมายเพิ่มเข้าไปด้วย ซึ่งอาจมองว่าเป็น thesaurus ที่มีข้อมูลเยอะกว่า thesaurus ปกติ

- ใน NLTK ได้รวม WordNet เข้าไว้ด้วย โดยมีคำทั้งหมด 155,287 คำ และ 117,659 synonym sets (synset)

  - synset = กลุ่มของคำที่มีความหมายคล้ายกัน

Senses and Synonyms

- พิจารณาประโยค (a) ถ้าเราแทนคำว่า motorcar ด้วย automobile ซึ่งจะได้ประโยค (b) ซึ่งความหมายของทั้งสองประโยคใกล้เคียงกันมาก
  - a) Benz is credited with the invention of the motorcar.
  - b) Benz is credited with the invention of the automobile.
- เราสามารถสรุปได้ว่า motorcar และ automobile มีความหมายคล้ายกัน (synonyms)

Senses and Synonyms

- ทดลอง WordNet บน NLTK

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motocar')
[Synset('car.n.01')]
```

- car.n.01 หมายถึง ความหมายของ car ที่เป็นคำนามตัวที่ 1 ซึ่งจะถูกเรียกว่า synset

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

- car มีหลายความหมาย ดังนั้นสำหรับคำว่า car ที่ใช้เป็น synset นี้ จะมีความหมายตรงกับกลุ่มคำที่มีใน synset นั้น ซึ่งเราสามารถดูความหมายและตัวอย่างของ synset ได้

```
>>> wn.synset('car.n.01').definition
>>> wn.synset('car.n.01').examples
```

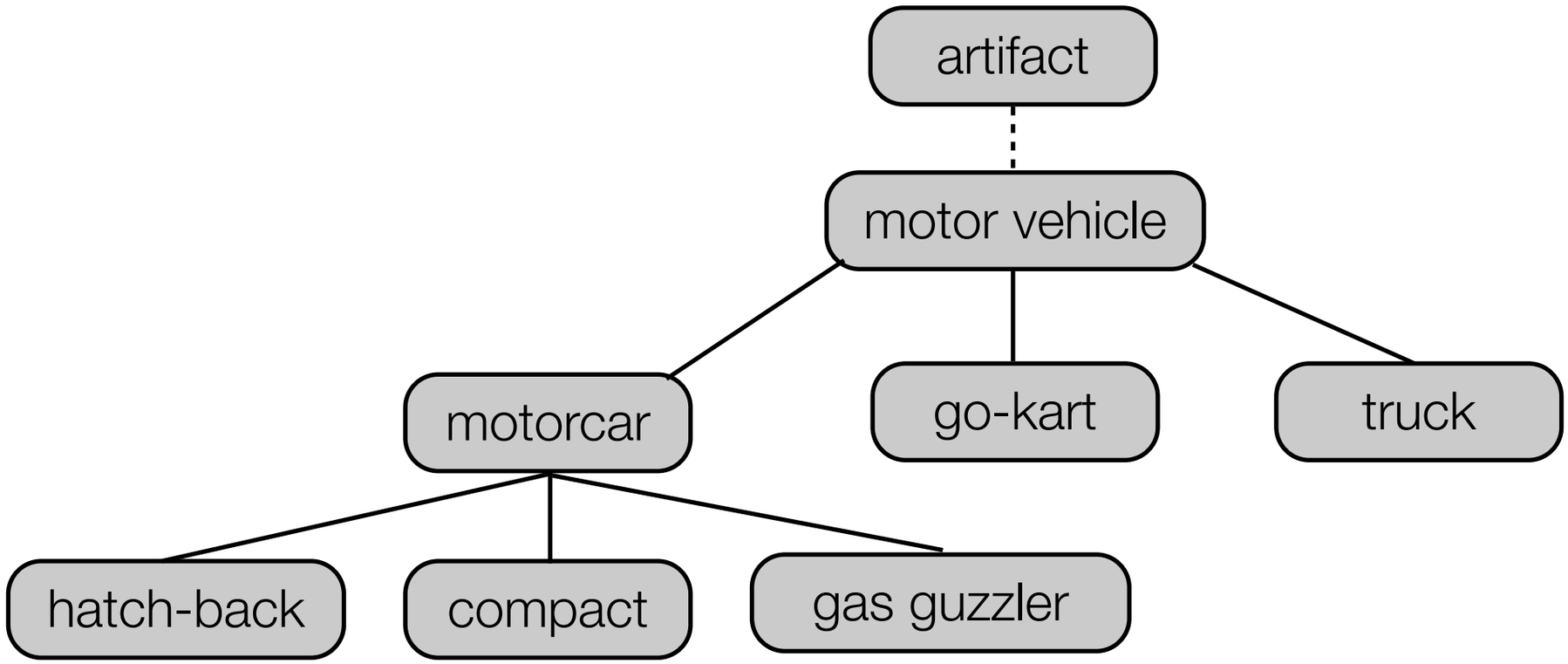
Senses and Synonyms

- เนื่องจากหนึ่งคำมีได้หลายความหมาย เพื่อลดความกำกวมโดยการแทนคำด้วย lemma ซึ่งมีการรวมคำเข้ากับ synset เช่น car.n.01.automobile, car.n.01.motocar, ...

```
>>> wn.synset('car.n.01').lemmas # หา lemma จาก synset

>>> wn.lemma('car.n.01.automobile') # หา lemma แบบเจาะจง

>>> wn.lemma('car.n.01.automobile').synset # หา synset ของ lemma

>>> wn.lemma('car.n.01.automobile').name # หา name ของ lemma

>>> wn.lemmas('car') # หา lemma ทั้งหมดที่เกี่ยวข้องกับ car
```

- ทดลองสำรวจคำว่า `dish` โดยใช้คำสั่งต่างๆ ที่กล่าวไปแล้ว

The WordNet Hierarchy



- WordNet จัดเรียงแต่ละ synset ตามลำดับตามความชี้เฉพาะเจาะจงของความหมาย ซึ่งความสัมพันธ์นี้เรียกว่า hyponyms

- synset ที่มีความหมายถึงทั่วไปที่สุดจะเรียกว่า unique beginners

The WordNet Hierarchy

```
>>> motorcar = wn.synset('car.n.01')
>>> types_of_motorcar = motorcar.hyponyms()
>>> types_of_motorcar[26]
>>> sorted([lemma.name for synset in types_of_motorcar
            for lemma in synset.lemmas])
```

- เราสามารถหาตามการสืบทอดขึ้นไปข้างบนได้ โดยหนึ่งความหมายอาจถูกแบ่งกลุ่มไว้มากกว่าหนึ่งได้

```
>>> motorcar.hypernyms()
>>> paths = motorcar.hypernym_paths()
>>> len(paths)
>>> [synset.name for synset in paths[0]]
>>> [synset.name for synset in paths[1]]
```

- หาความหมายที่ทั่วไปที่สุดของ synset นั้น

```
>>> motorcar.root_hypernyms()
```

More Lexical Relations

- ทั้ง hypernyms hyponyms และ synonyms นั้นเรียกว่า **lexical relations** ในส่วนของ hypernyms และ hyponyms จะมีชื่อเฉพาะว่า **is-a** hierarchy

- WordNet ได้เตรียมความสัมพันธ์แบบอื่นไว้ให้ด้วยเช่น

  - meronyms: แสดงการเป็นส่วนประกอบ
  - holonyms: แสดงการเข้าไปอยู่ข้างใน
  - entails: แสดงความเกี่ยวเนื่องกันของกริยา
  - antonymy: แสดงความหมายที่ตรงข้ามกันระหว่าง lemma

More Lexical Relations

```
>>> wn.synset('tree.n.01').part_meronyms()
>>> wn.synset('tree.n.01').substance_meronyms()

>>> wn.synset('tree.n.01').member_holonyms()

>>> wn.synset('walk.v.01').entailments()
>>> wn.synset('eat.v.01').entailments()
>>> wn.synset('tease.v.03').entailments()

>>> wn.lemma('supply.n.02.supply').antonyms()
>>> wn.lemma('rush.v.01.rush').antonyms()
>>> wn.lemma('horizontal.a.01.horizontal').antonyms()
>>> wn.lemma('staccato.r.01.staccato').antonyms()
```

- เราสามารถใช้ `dir()` ในการตรวจสอบว่ามี lexical relations หรือ method อะไรบ้างที่ประกาศไว้สำหรับ synset เช่น `dir(wn.synset('harmony.n.02'))`

Semantic Similarity

- เราสามารถใช้ WordNet ในการเปรียบเทียบคำได้ว่า คำนั้นมีความหมายสัมพันธ์กันขนาดไหน โดยการเทียบดูว่า hypernym ตัวที่เหมือนกันที่ใกล้ที่สุดคืออะไร

```
>>> right = wn.synset('right_whale.n.01')
>>> orca = wn.synset('orca.n.01')
>>> minke = wn.synset('minke_whale.n.01')
>>> tortoise = wn.synset('tortoise.n.01')
>>> novel = wn.synset('novel.n.01')

>>> right.lowest_common_hypernyms(minke)
[Synset('baleen_whale.n.01')]
>>> right.lowest_common_hypernyms(orca)
[Synset('whale.n.02')]
>>> right.lowest_common_hypernyms(tortoise)
[Synset('vertebrate.n.01')]
>>> right.lowest_common_hypernyms(novel)
[Synset('entity.n.01')]
```

Semantic Similarity

- เราสามารถดูได้ว่าระยะระหว่าง synset กับ ความหมายที่ทั่วไปที่สุดคือ entity ว่าเป็นเท่าไร

```
>>> wn.synset('baleen_whale.n.01').min_depth()
14
>>> wn.synset('whale.n.02').min_depth()
13
>>> wn.synset('vertebrate.n.01').min_depth()
8
>>> wn.synset('entity.n.01').min_depth()
0
```

Semantic Similarity

- เราหาคะแนนความคล้ายกันระหว่าง synset โดยใช้ระยะของเส้นทาง hypernym โดยที่สูตรคือ 1/(1+distance)

```
>>> right.path_similarity(minke)
0.25
>>> right.path_similarity(orca)
0.16666666666666666
>>> right.path_similarity(tortoise)
0.07692307692307692
>>> right.path_similarity(novel)
0.043478260869565216
```

- การหาคะแนนความคล้ายมีอีกหลายสูตร สามารถดูได้จากคู่มือการใช้
  - `help(nltk.corpus.wordnet)`