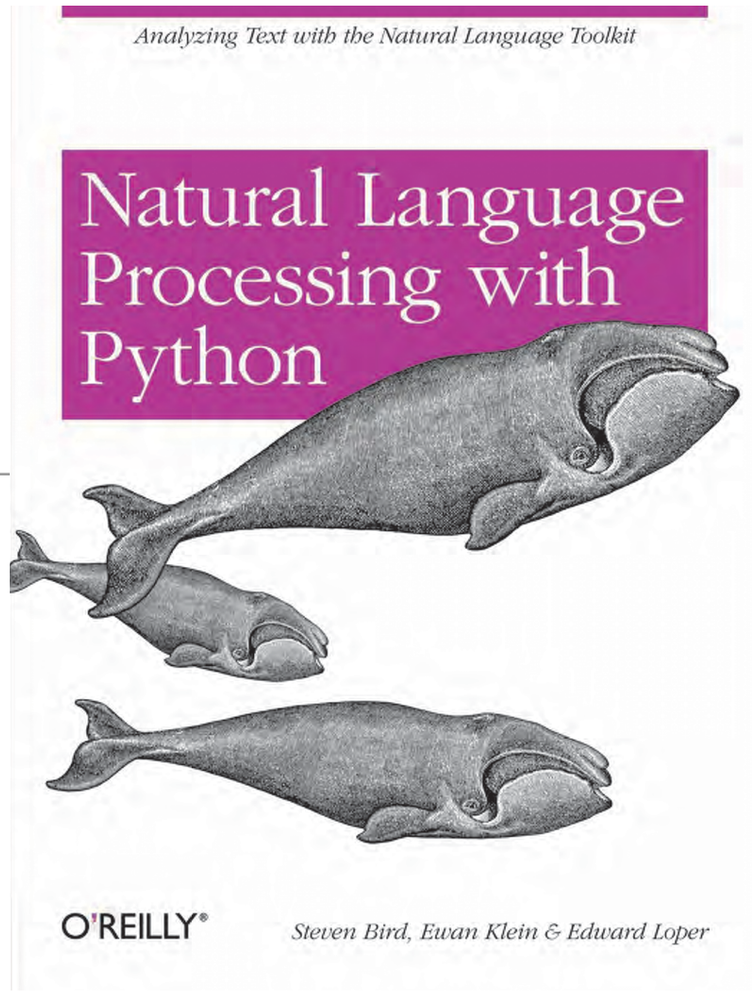
Language Processing and Python

Sutee Sudprasert

เนื้อหาจากบทที่ 1 ของหนังสือ
Natural Language Processing with Python



คำถามก่อนเรียน

- เราจะเขียนโปรแกรมอย่างไรในการจัดการกับเอกสารข้อความขนาดใหญ่?
- เราจะสกัดคำและวลีที่บ่งบอกถึงรูปแบบและเนื้อหาสำคัญของเอกสารโดยอัตโนมัติได้อย่างไร?
- เราจะใช้ภาษา Python เพื่อแก้ไขปัญหาที่กล่าวมาได้อย่างไร?
- อะไรคือความท้าทายในการประมวลผลภาษาธรรมชาติ?

Computing with Language: Texts and Words

- เราเขียนและอ่านเอกสารอยู่เป็นประจำทุกวัน ซึ่งในที่นี้เอกสารคือข้อมูลดิบ (raw data) ที่เราสนใจ เราจะจำกัดและวิเคราะห์เอกสารด้วยวิธีการที่มีประสิทธิภาพ

- ภาษา Python เป็นภาษาที่เหมาะสมในการจัดการกับข้อความ

Getting Started with Python

- Python interactive interpreter
- สร้างความคุ้นเคยในการใช้ interactive interpreter

Getting Started with NLTK

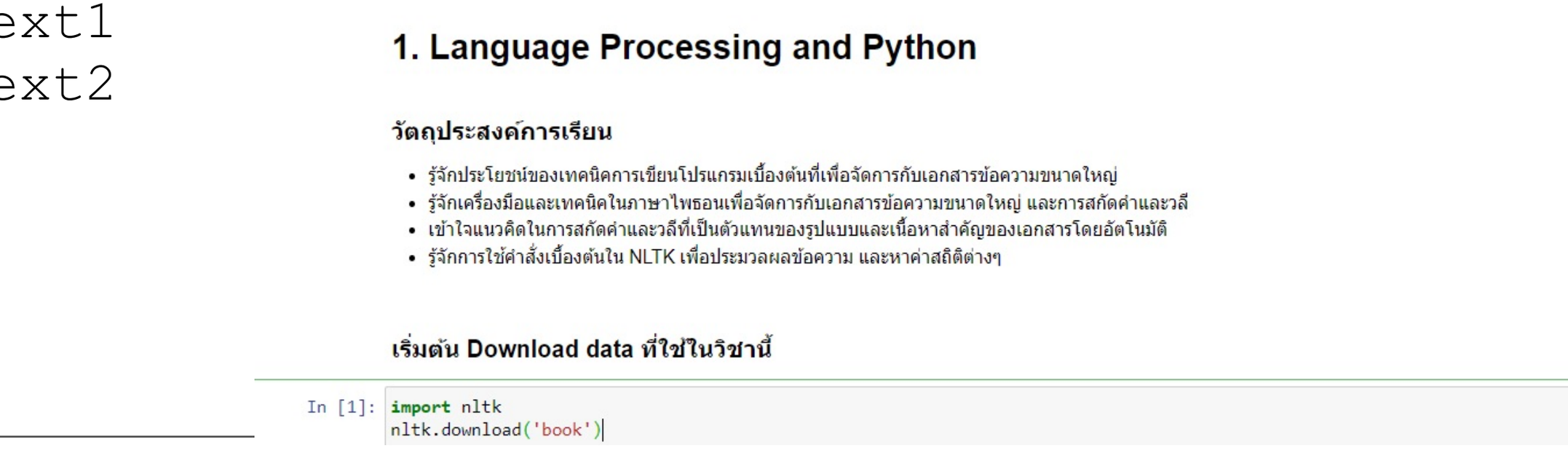
- ดาวน์โหลด NLTK ได้จาก http://www.nltk.org

- พิมพ์ใน interactive interpreter

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

- ให้เลือกดาวน์โหลดข้อมูล book จากนั้นพิมพ์

```
>>> from nltk.book import *
>>> text1
>>> text2
```



1. Language Processing and Python

วัตถุประสงค์การเรียน

- รู้จักประโยชน์ของเทคนิคการเขียนโปรแกรมเบื้องต้นที่เพื่อจัดการกับเอกสารข้อความขนาดใหญ่
- รู้จักเครื่องมือและเทคนิคในภาษาไพธอนเพื่อจัดการกับเอกสารข้อความขนาดใหญ่ และการสกัดคำและวลี
- เข้าใจแนวคิดในการสกัดคำและวลีที่เป็นตัวแทนของรูปแบบและเนื้อหาสำคัญของเอกสารโดยอัตโนมัติ
- รู้จักการใช้คำสั่งเบื้องต้นใน NLTK เพื่อประมวลผลข้อความ และหาค่าสถิติต่างๆ

เริ่มต้น Download data ที่ใช้ในวิชานี้

```
In [1]: import nltk
        nltk.download('book')
```

Searching Text

- มีวิธีการหลายวิธีในการตรวจหาส่วนที่เราต้องการจากเอกสารทั้งหมด ซึ่ง concordance เป็นวิธีหนึ่งที่ใช้ในการแสดงบริบทรอบข้างคำๆ หนึ่งที่เราสนใจ

- ทดลองพิมพ์

```
>>> text1.concordance("monstrous")
```

- หาคำที่อยู่ในบริบทใกล้เคียงกัน

```
>>> text1.similar("monstrous")
>>> text2.similar("monstrous")
```

- หาบริบทที่เหมือนกันระหว่างสองคำ

```
>>> text2.common_contexts(["monstrous", "very"])
```

Searching Text

- หาตำแหน่งของคำในเอกสารแล้วนำมาวาดกราฟเพื่อเปรียบเทียบ

```
>>> text4.dispersion_plot([“citizens”, “democracy”,
    “freedom”, “duties”, “America”])
```

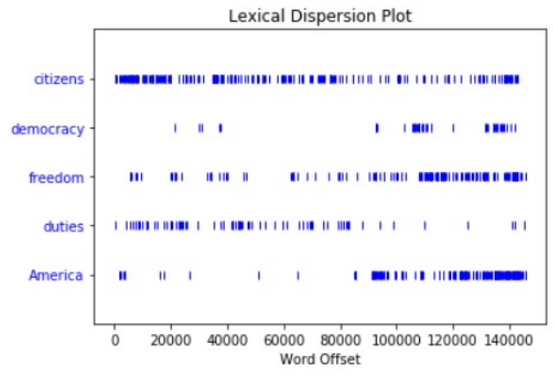
- หากเรานำเอกสารไล่ตามปีมาเรียงต่อกัน เราจะสามารถดูรูปแบบการใช้คำตามยุคสมัยได้

- การวาดกราฟใน nltk จำเป็นต้องลง Numpy และ Matplotlib ก่อน

นอกจากนี้ เรายังสามารถหาตำแหน่งของคำในเอกสารโดยนำมาแสดงผลในรูปของกราฟ เพื่อดูลักษณะการ กระจายตัวในเอกสาร ดังตัวอย่างคำสั่งต่อไปนี้

```
]: print(len(text4))
   text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])
```

145735



Searching Text

- ทดสอบสร้างข้อความแบบอัตโนมัติ

```
>>> text3.generate()
```

- ทดลองหลายๆ ครั้ง แล้วจะเห็นว่าโปรแกรมสร้างข้อความที่ไม่เหมือนกัน

การประมวลผลคำและข้อความ

เรามักจะคุ้นเคยกับข้อความ เนื่องจากเราอ่านและเขียนข้อความอยู่เป็นประจำทุกวัน

ในการเขียนโปรแกรมเพื่อประมวลผลทางภาษา ข้อความ คือ ข้อมูลดิบ (raw data) ที่เราสนใจนำมาจัดการและวิเคราะห์ได้หลากหลายรูปแบบ

ตัวอย่างเช่น ถ้าเราต้องการค้นหาตัวอย่างของการใช้คำๆ หนึ่งที่เราสนใจ โดยอาจจะดูว่าคำนั้นมีการใช้กับ เรื่องใดบ้าง ใช้ในบริบทและความหมายใดได้บ้าง เราใช้วิธีการที่เรียกว่า concordance เพื่อแสดงบริบทรอบข้างของ คำๆ หนึ่งที่เราสนใจได้ดังตัวอย่างต่อไปนี้

:
```
text1.concordance("silently")
```

```
Displaying 15 of 15 matches:
ow came a lull in his look , as he silently turned over the leaves of the Book
draw the poles , ye harpooneers !" Silently obeying the order , the three harp
er from the beginning . For me , I silently recalled the mysterious shadows I
sings of his chip of a craft , and silently eyeing the vast blue eye of the se
ped up in him and the slow - match silently burning along towards them ; as he
 sank . Then once more arose , and silently gleamed . It seemed not a whale ;
d back to the vessel ; the rest as silently following . Whatever superstitions
raceful repose of the line , as it silently serpentines about the oarsmen befo
ales of the boats , we swiftly but silently paddled along ; the calm not admit
d at her helm , and for long hours silently guided the way of this fire - ship
cketer , and a whaleman , you will silently worship there . CHAPTER 105 Does t
straight flame , the Parsee passed silently , and bowing over his head towards
, each of the three tall masts was silently burning in that sulphurous air , l
ng from his enchantment , Gardiner silently hurried to the side ; more fell th
ming like a canopy over the fish , silently perched and rocked on this pole ,
```

Counting Vocabulary

- ฟังก์ชั่น len() ใช้ในการนับ token ทั้งหมดในข้อความ

```
>>> len(text3)
```

- token หมายถึงกลุ่มของตัวอักษรที่ถูกแบ่งด้วย white-spaces

- ฟังก์ชั่น len() จะนับ token ทั้งหมด โดยไม่สนใจว่า token จะซ้ำกันหรือเปล่า เช่น
  - “to be or not to be” จะนับได้ 6 tokens

Counting Vocabulary

- หากต้องการนับเฉพาะคำที่ไม่ซ้ำกัน (หาจำนวนคำศัพท์ทั้งหมดที่ใช้ในเอกสาร)

```
>>> sorted(set(text3))
>>> len(set(text3))
```

- คำที่ไม่เหมือนกันจะเรียกว่า word types หรือ types

- หากต้องการหาว่าแต่ละ word types ถูกใช้เฉลี่ยกี่ครั้งต่อเอกสาร

```
>>> from __future__ import division
>>> len(text3)/len(set(text3))
```

- ค่าที่ได้นั้นเรียกว่า lexical diversity

Counting Vocabulary

- หากต้องการนับเฉพาะคำ

```
>>> text3.count(“smote”)
>>> 100 * text4.count(“a”)/len(text4)
```

Counting Vocabulary

- เราสามารถเขียนฟังก์ชั่น lexical_diversity และ percentage ขึ้นมาเพื่อที่จะได้ไม่ต้องพิมพ์สูตรการคำนวณซ้ำหลายๆ ครั้ง

```
>>> def lexical_diversity(text):
...     return len(text) / len(set(text))
>>> def percentage(count, total):
...     return 100 * count / total
```

- เมื่อประการฟังก์ชั่นแล้ว เราสามารถเรียกฟังก์ชั่นได้ดังนี้

```
>>> lexical_diversity(text3)
>>> lexical_diversity(text5)
>>> percentage(4, 5)
>>> percentage(text4.count("a"), len(text4))
```

Counting Vocabulary

- ตัวอย่างค่า lexical diversity สำหรับแต่ละหมวดหมู่ใน Brown Corpus

Genre	Tokens	Types	Lexical diversity
skill and hobbies	82345	11935	6.9
humor	21695	5017	4.3
fiction:science	14470	3233	4.5
press:reportage	100554	14394	7.0
fiction:romance	70022	8452	8.3
religion	39399	6373	6.2

A Closer Look at Python

- List
  - concatenation and append
  - indexing lists
    - index
    - slicing

```
my_list = [1,2,3,'four',5.5,True]
```

A Closer Look at Python

- Variables
    - assignment
- Strings
    - split and join

Computing with Languages: Simple Statistics

- Frequency Distributions
- Fine-Grained Selection of Words
- Collocations and Bigrams
- Counting Other Things

Frequency Distributions

- หากเราต้องการหาว่าคำใดในเอกสาร ควรจะเป็นคำที่สื่อถึงหัวข้อและหมวดหมู่ของเอกสารนั้น เราอาจทดลองนับคำที่มีความถี่มากที่สุด 50 อันดับแรก

- ถ้าเรานับด้วยมือ เราอาจจะได้ตารางดังนี้

Word Tally

the	𝍸 𝍸 𝍸 ||||
been	𝍸 𝍸 |
message	||||
persevere	|
nation	𝍸 |||

- ตารางนี้เรียกว่า frequency distribution ซึ่งบอกที่การกระจายตัวของ tokens ไปบนคำศัพท์ทั้งหมด

Frequency Distributions

- NLTK มีเครื่องมือสำหรับการหา frequency distributions ที่ใช้ในการประมวลผลภาษาโดยเฉพาะซึ่งคือ FreqDist

```
>>> fdist1 = FreqDist(text1)
>>> fdist1
>>> vocabulary1 = fdist1.keys()
>>> vocabulary1[:50]
>>> fdist1[“whale”]
```

- วาดกราฟเพื่อแสดงความถี่สะสมของคำที่พบมากที่สุด 50 อันดับแรก

```
>>> fdist1.plot(50, cumulative=True)
```

Frequency Distributions

- จากผลลัพธ์ที่ได้จะเห็นว่า 50 คำที่มีความที่มากที่สุด ไม่สามารถใช้เป็นตัวแทนเอกสารได้ (มีคำว่า whale คำเดียวที่ใช้ได้)

- หากทดลองดูคำที่มีความถี่น้อยที่สุดแทน

```
>>> fdist1.hapaxes()
```

- ซึ่งจะเห็นว่าผลลัพธ์ที่ได้ก็ไม่สามารถใช้เป็นตัวแทนเอกสารได้เหมือนกัน

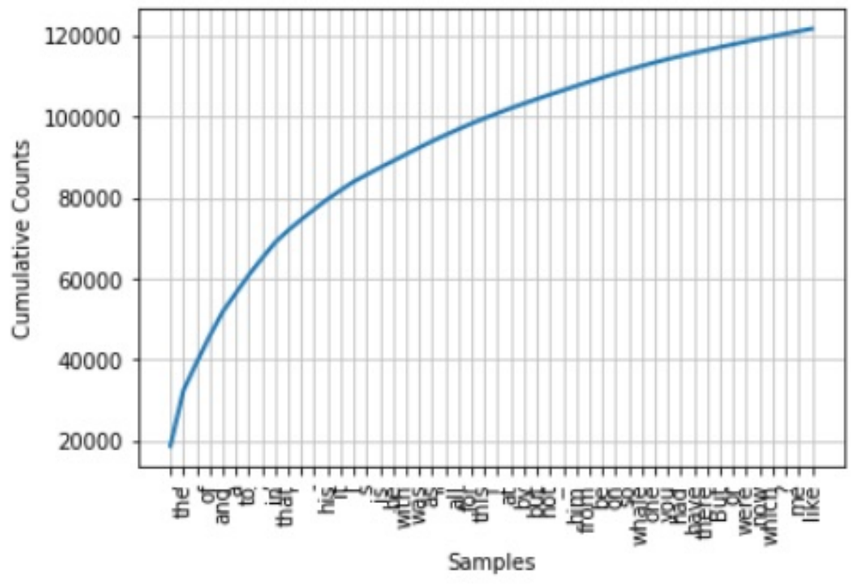
สถิติกับการประมวลผลทางภาษา

การประมวลผลข้อความมักจะเกี่ยวข้องกับการคำนวณเหาค่าทางสถิติต่างๆ สถิติที่มักจะใช้บ่อย ได้แก่ การแจกแจงความถี่ (frequency distribution) เพื่อดูลักษณะการกระจายตัวของโทเค็นบนคำศัพท์ทั้งหมดในเอกสาร

ใน NLTK เราสามารถใช้ชุดคำสั่ง FreqDist เพื่อหาการแจกแจงความถี่ของคำศัพท์ในเอกสารได้ ดังตัวอย่าง

```
]: fdist = FreqDist(text1)
   print(fdist)
   print(len(set(text1)), len(text1))
   vocabs = fdist.keys()
   print(vocabs)
```

```
In [ ]: fdist.plot(50, cumulative=True)
```



จากตัวอย่าง เป็นการใช้ชุดคำสั่ง FreqDist เพื่อแจกแจงความถี่ของเอกสาร text1 แล้วเก็บค่าไว้ในตัวแปร fdist

- เมื่อต้องการหาความถี่ของคำศัพท์ใดๆ สามารถอ้างถึงได้โดยใช้ตัวแปร fdist1 แล้วระบุคำที่ต้องการทราบความถี่ในรูปดัชนี
- ส่วนการหาความถี่ของคำศัพท์ที่พบบ่อยๆ จำนวน n ตัวแรก จะใช้คำสั่ง most_common(n)
- คำสั่งสุดท้ายในตัวอย่างจะแสดงกราฟความถี่สะสมของคำศัพท์ที่พบบ่อย 50 คำแรก
- ถ้าเราสนใจคำศัพท์ที่ปรากฏเพียงครั้งเดียวในเอกสาร จะใช้คำสั่ง hapaxes()

ส่วนตัวอย่างคำสั่งอื่นๆ ใน FreqDist แสดงใน https://kite.com/python/docs/nltk.FreqDist

Fine-Grained Selection of Words

- เราจะทดลองหาคำที่ยาว ซึ่งเป็นไปได้ว่าอาจจากที่คำที่มีสื่อความหมายได้ดีกว่า ในที่นี้ เราจะลองหาคำที่มีความยาวตั้งแต่ 15 ตัวอักษรขึ้นไป

- เราสามารถเขียนเป็นสมการทำคณิตศาสตร์ได้ ดังนี้ กำหนดให้คุณสมบัติ P(w) เป็นจริง เมื่อ w มีความยาวมากกว่า 15 ตัวอักษร และ V เป็นเซตของคำศัพท์ทั้งหมด

$$\{w \mid w \in V \ \& \ P(w)\}$$

$$[\ w \text{ for } w \text{ in } V \text{ if } p(w)\ ]$$

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

Fine-Grained Selection of Words

- ปรากฏว่าคำที่ยาวก็ไม่ได้บ่งบอกถึงลักษณะของเอกสาร เช่น ใน text5 จะปรากฏคำที่ไม่เป็นทางการเช่น “booooooooooglyyyyyy” และ “yuuuuuuuuuuuummmmmmmmmmm”

- สิ่งที่สังเกตได้อีกอย่างหนึ่งคือ คำที่ยาวๆ มักจะเป็นคำที่ไม่ได้เกิดขึ้นบ่อย ดังนั้นการที่หาคำที่ยาวและเกิดขึ้นบ่อย น่าจะพอมีความหวัง

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

ex. อัลกอริทึมที่ใช้สกัดคำ

- TF-IDF (Term Frequency - Inverse Document Frequency) : ใช้วิธีการคำนวณความถี่ของคำศัพท์ในเอกสารแต่ละเรื่อง (Term Frequency) ร่วมกับการชั่งน้ำหนักด้วยความถี่ที่คำนั้นๆ ปรากฏในเอกสารทั้งหมด (IDF)
- คำที่มีความถี่สูงในเอกสารแต่ละเรื่องและมีความน่าจะเป็นที่จะพบในเอกสารทั้งหมดจะถือเป็นคำสำคัญ
- LDA (Latent Dirichlet Allocation) : วิธีการในการจัดกลุ่มเอกสารออกเป็นกลุ่มหลักๆ (Topic) และคำศัพท์ที่เกี่ยวข้องกันแต่ละกลุ่ม โดยพิจารณาถึงการปรากฏของคำศัพท์ในแต่ละกลุ่มเป็นหลัก
- คำที่มีความน่าจะเป็นสูงที่จะปรากฏในกลุ่มใดๆ จะถือเป็นคำสำคัญของกลุ่มนั้น

การสกัดคำหรือวลีสำคัญ (keyword extraction)

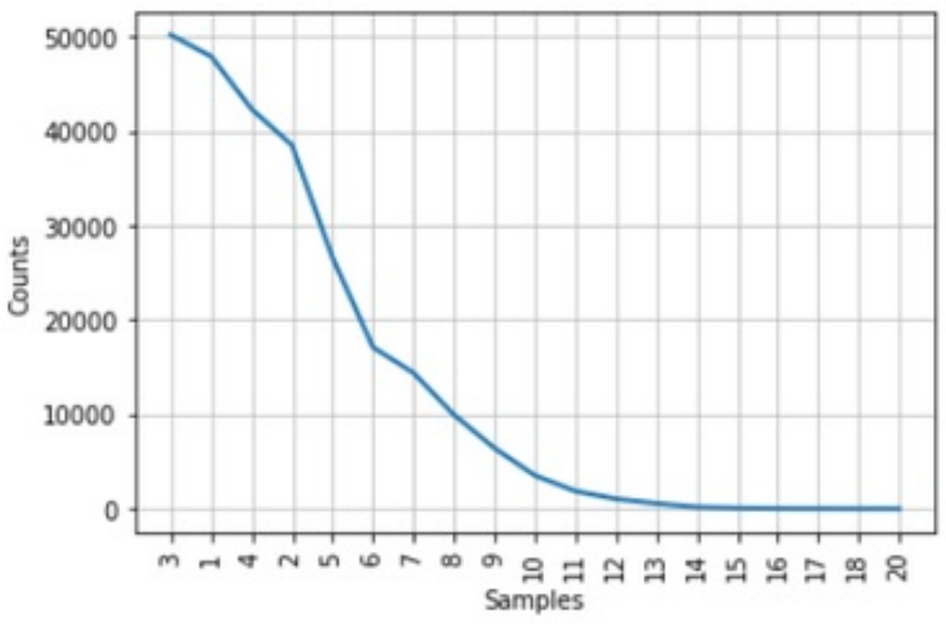
เป็นงานประมวลผลข้อความที่สามารถใช้การแจกแจงความถี่ เพื่อหาลักษณะของคำสำคัญที่สามารถใช้เป็นตัวแทนของเอกสารได้

จากตัวอย่างจะเห็นได้ว่า คำศัพท์ที่พบได้บ่อยที่สุดในเอกสารไม่ใช่คำที่เหมาะสมสำหรับเป็นตัวแทนของเอกสารได้

```
# {w | w ∈ V & P(w)}
# [w for w in V if p(w)]
token = [           ]
voc = set(token)
long_words_2 = [w for w in voc if len(w) >7 and fdist[w] > 7]
long_words_2
```

```
all_wordsLen = [len(w) for w in text1]
print(text1[:20])
print(all_wordsLen[:20])
len_fdist = FreqDist(all_wordsLen)
print(len_fdist.plot())
```

```
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(', 'Supplied', 'by', 'a', 'Late', 'Consumpti
ve', 'Usher', 'to', 'a', 'Grammar']
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1, 1, 8, 2, 1, 4, 11, 5, 2, 1, 7]
```



Collocation and Bigrams

- Collocation คือการที่ลำดับของคำหนึ่งเกิดขึ้นพร้อมกันบ่อยๆ ในลักษณะที่มีความหมายพิเศษแตกต่างไปจากเดิม เช่น red wine หรือ white wine

- เราสามารถหา collocation อย่างง่าย โดยการเริ่มต้นจากการหา bigrams

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

- สิ่งที่ต้องทำต้องไปคือการหาคู่ของคำที่มีความถี่สูง ซึ่งขั้นตอนทั้งหมดสามารถทำได้โดยใช้ฟังก์ชั่น collocations

```
>>> text4.collocations()
```

Collocation

Collocation มักจะเป็นคำที่จำเป็นต้องใช้ร่วมกันในประโยค หรือเป็นคำที่ใช้คู่กันแล้วให้ความหมายพิเศษต่างไปจากเดิม เช่น

1. กลุ่มคำที่ใช้ร่วมกันแล้วมีความหมายพิเศษ และใช้คำอื่นที่มีความหมายทำนองเดียวกันแทนที่ไม่ได้ เช่น
    - fast color หมายถึงสีไม่ตก ถ้าเรานำคำที่มีความหมายทำนองเดียวกันมาแทนที่ เช่น stable color หรือ speed color จะกลายเป็นคำที่มีความหมายผิดไป
    - fast food ไม่นิยมใช้คำว่า quick food
2. กลุ่มคำที่มักใช้ร่วมกันเป็นปกติในภาษานั้นๆ เช่น
    - have lunch, go home, do homework, interested in เป็นกลุ่มคำที่มีจำนวน 2 คำ
    - make a mistake, do your best เป็นกลุ่มคำที่มีจำนวน 3 คำ
    - from dawn till dusk, run out of time, raining cats and dogs กลุ่มคำที่มีมากกว่า 3 คำขึ้นไป มักเป็นสำนวน

ถ้าเป็นคำที่ปรากฏร่วมกันแต่ไม่มีความหมายพิเศษ หรือไม่ได้จำเป็นต้องปรากฏร่วมกันเป็นปกติ จะไม่ถือว่าเป็น collocation เช่น the color หรือ red color

ลองแปลประโยคต่อไปนี้เป็นภาษาอังกฤษดู

1. เธอเป็นคนใจเย็น she is a calm person.
2. เขาผ่านไป he pass by.
3. เขาทานยา he took medicine.
4. บ้านหันหน้าไปทางทิศตะวันออก The house faces east.
5. เธอเป็นคนง่ายๆ he is a chill guy.
   she is a simple person.

Counting Other Things

- นอกจากการนับความถี่ของคำแล้ว เราสามารถพิจารณาเรื่องการกระจายของความยาวของคำได้ โดยใช้ FreqDist บนลิสต์ของความยาวของคำทั้งหมด

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

Functions defined for NLTK's FreqDist

Example	Description
`fdist = FreqDist(samples)`	Create a frequency distribution containing the given samples
`fdist.inc(sample)`	Increment the count for this sample
`fdist['monstrous']`	Count of the number of times a given sample occurred
`fdist.freq('monstrous')`	Frequency of a given sample
`fdist.N()`	Total number of samples
`fdist.keys()`	The samples sorted in order of decreasing frequency
`for sample in fdist:`	Iterate over the samples, in order of decreasing frequency
`fdist.max()`	Sample with the greatest count
`fdist.tabulate()`	Tabulate the frequency distribution
`fdist.plot()`	Graphical plot of the frequency distribution
`fdist.plot(cumulative=True)`	Cumulative plot of the frequency distribution
`fdist1 < fdist2`	Test if samples in `fdist1` occur less frequently than in `fdist2`

Some word comparison operators

Function	Meaning
`s.startswith(t)`	Test if `s` starts with `t`
`s.endswith(t)`	Test if `s` ends with `t`
`t in s`	Test if `t` is contained inside `s`
`s.islower()`	Test if all cased characters in `s` are lowercase
`s.isupper()`	Test if all cased characters in `s` are uppercase
`s.isalpha()`	Test if all characters in `s` are alphabetic
`s.isalnum()`	Test if all characters in `s` are alphanumeric
`s.isdigit()`	Test if all characters in `s` are digits
`s.istitle()`	Test if `s` is titlecased (all words in `s` have initial capitals)

Automatic Natural Language Understanding

- หลังจากเราได้เริ่มเรียนรู้การประมวลข้อความในระดับพื้นฐาน ตอนนี้เราจะข้ามไปในระดับบนสุด เพื่อให้เห็นภาพกว้างของงานด้าน NLP

- หากเราต้องการให้คอมพิวเตอร์ตอบคำถามว่า “มีสถานที่ท่องเที่ยวอะไรบ้างที่น่าสนใจในเชียงใหม่”

- information extraction
- inference
- summarization

Word Sense Disambiguation

- คำหนึ่งคำมีได้หลายความหมายเช่น
  - serve: help with food or drink; hold an office; put ball into play
  - dish: plate; course of a meal; communications device
- หากมีวลี “he served the dish,...” เราสามารถรู้ได้ว่าความหมายของทั้งสองคำนั้นเกี่ยวกับเรื่องอาหาร

Word Sense Disambiguation

- ตัวอย่างคำว่า “by” มีหลายความหมาย ซึ่งสามารถแยกความแตกต่างได้จากบริบทที่ตามมา
  - The lost children were found by the *searchers* (agentive)
  - The lost children were found by the *mountain* (locative)
  - The lost children were found by the *afternoon* (temporal)

Pronoun Resolution

- การวิเคราะห์เพื่อที่จะเข้าถึงความหมายในภาษาระดับลึก สิ่งที่เราต้องการรู้คือ “ใครทำอะไรกับใคร” หรือ หมายความว่าเราต้องรู้ว่า ประธาน กริยา กรรม คืออะไร

- บางทีเราคิดว่าไม่น่าจะเป็นเรื่องยากที่จะตอบคำถามนี้ แต่ในบางกรณีไม่เป็นอย่างนั้น

  - The thieves stole the paintings. They were subsequently *sold*.

  - The thieves stole the paintings. They were subsequently *caught*.

  - The thieves stole the paintings. They were subsequently *found*.

- สิ่งที่ต้องทำคือ anaphora resolution และ semantic role labeling

Generating Language Output

- ในกรณีที่เราสำหรับวิเคราะห์และเข้าใจภาษาได้แล้ว เราจะสามารถทำในส่วนของการสร้างภาษาขึ้นมาได้ เช่นระบบถามตอบ และ ระบบแปลภาษา

- ระบบถามตอบ (question answering) อาจมีลักษณะตัวอย่างดังนี้

  - Text: ... The thieves stole the paintings. They were subsequently sold. ...

  - Human: Who or what was sold?

  - Machine: The paintings.

Machine Translation

- ระบบแปลภาษาต้องสามารถถอดความหมายจากภาษาต้นทางแล้วสร้างข้อความในภาษาปลายทางที่ถูกต้องตามไวยากรณ์พร้อมทั้งยังคงความหมายเช่นเดียวกับภาษาต้นทางด้วย

- ตัวอย่างเช่น ถ้าจะแปลประโยค “The thieves stole the paintings. They were subsequently sold.” ให้เป็นภาษาฝรั่งเศส ในกรณีที่ They = paintings กับ They = thieves ประโยคที่แปลออกมาจะไม่เหมือนกัน เพราะเพศของคำทั้งสองต่างกัน

  - Les voleurs ont volé les peintures. *Ils* ont été *trouvés* plus tard. (the thieves)

  - Les voleurs ont volé les peintures. *Elles* ont été *trouvées* plus tard. (the paintings)

Machine Translation

- ในปัจจุบันวิธีการที่นิยมในการสร้างระบบแปลภาษาคือการสร้างระบบโดยการเรียนรู้จากคลังประโยคขนาน (parallel corpus)

- ในการทำงานจะมีกระบวนการที่เรียกว่า text alignment เพื่อจับคู่คำและวลีระหว่างคู่ประโยคภาษาต้นทางและปลายทาง จากนั้นจึงนำผลลัพธ์ที่ได้ไปสร้างแบบจำลองการแปลต่อไป

Spoken Dialogue Systems

- ในประวัติของวงการ AI ได้มีเสนอวิธีการทดสอบความฉลาดของเครื่องจักร ที่เรียกว่า Turing Test
    - ถ้าเครื่องจักรฉลาดจริงๆ คนต้องแยกความแตกต่างไม่ออก ระหว่างการคุยกับเครื่องจักร กับการคุยกับคนจริงๆ

- ในปัจจุบันระบบโต้ตอบข้อความ สามารถใช้งานได้แค่ในวงจำกัด คือ เรื่องที่โต้ตอบกันต้องเป็นเรื่องเฉพาะเรื่องได้เรื่องหนึ่งเท่านั้น เช่น ระบบถามตอบรอบหนัง

S: How may I help you?
U: When is Saving Private Ryan playing?
S: For what theater?
U: The Paramount theater.
S: Saving Private Ryan is not playing at the Paramount theater, but it's playing at the Madison theater at 3:00, 5:30, 8:00, and 10:30.

Textual Entailment

- เมื่อไม่กี่ปีนี้ มี “shared task” ที่เรียกว่า Recognizing Textual Entailment (RTE) ซึ่งเป็นการให้หาว่าหลักฐานที่ให้มาสนับสนุนสมมุติฐานหรือเปล่า เช่น
  - สมมุติฐาน: Sandra Goudie was defeated by Max Purnell.
  - หลักฐาน: Sandra Goudie was first elected to Parliament in the 2002 elections, narrowly winning the seat of Coromandel by defeating Labour candidate Max Purnell and pushing incumbent Green MP Jeanette Fitzsimons into third place.

Textual Entailment

- ในบ้างกรณีเราไม่สามารถสรุปโดยตรงได้เช่น
  - สมมุติฐาน: Golinkin has written 18 books
  - หลักฐาน: David Golinkin is the *editor or author* of 18 books, and over 150 responsa, articles, sermons and books

Limitation of NLP

- ในปัจจุบันโปรแกรมด้าน NLP ยังไม่สามารถทำงานได้อย่างสมบูรณ์ ที่ถูกไปใช้งานในระดับ world knowledge ซึ่งเราก็ยังต้องรอให้หลายๆ ปัญหาในด้าน NLP ถูกแก้ไขอย่างสมบูรณ์ก่อน

- เราต้องยอมรับถึงข้อจำกัดของโปรแกรมด้าน NLP ที่พอจะทำงานได้ดีเฉพาะในขอบเขตที่จำกัดเท่านั้น

- การเรียนวิชานี้ก็เพื่อสามารถนำความรู้ไปใช้สร้างระบบ NLP (ที่พอใช้งานได้) และ เพื่อพัฒนาความรู้ทางด้าน NLP ให้ก้าวหน้าขึ้นไปในอนาคต