2

คลังคำและข้อความ

2.1 การเข้าถึงและใช้งานคลังข้อความ

2.2 การเข้าถึงและใช้งานคลังคำ

2.3 พจนานุกรม WordNet

วัตถุประสงค์

- ☐ รู้จักความหมายคลังคำและคลังข้อความ
- ☐ สามารถเข้าถึงคลังคำและคลังข้อความ และเลือกใช้ได้อย่างเหมาะสม
- ☐ ประยุกต์คลังคำและคลังข้อความ เพื่อประมวลผลหาสถิติที่น่าสนใจได้
- ☐ สามารถใช้คำสั่งใน NLTK เพื่อใช้งานคลังคำและคลังข้อความใน NLTK ได้
- ☐ รู้จักโครงสร้างของพจนานุกรม WordNet และสามารถเขียนคำสั่งเพื่อเรียกใช้งานได้

คลังข้อความ (text corpora) คือ ข้อความปริมาณมากที่จัดเก็บไว้ด้วยกัน ใน NLTK มีหลายคลังข้อความให้ใช้งาน แต่ละคลังข้อความจะได้จากการรวบรวมเอกสารจำนวนมากไว้ด้วยกัน ลักษณะของเอกสารในแต่ละคลังก็แตกต่างกันไป บางคลังอาจจะมีการจัดแบ่งเป็นหมวดหมู่ย่อยไว้อยู่ภายในด้วย เช่น คลังข้อความข่าว แบ่งเป็นหมวดการเมือง กีฬา บันเทิง และอาชญากรรม ในบทที่ 1 เราได้ทดลองใช้คำสั่งเบื้องต้นกับ text1, text2, ..., text9 จาก nltk.book ซึ่งข้อความเหล่านี้ก็เป็นตัวอย่างหนึ่งของคลังข้อความที่รวบรวมจากหนังสือหรือเอกสารจำนวนมากไว้ให้ใช้ใน NLTK

คลังคำศัพท์ (lexical resources) การรวบรวมคำศัพท์ที่เกี่ยวข้องกันเข้าไว้กัน เช่น คลังคำหยุด (stop words) เป็นการรวบรวมคำหยุด ซึ่งเป็นคำที่พบได้ทั่วไป ไม่มีความหมายพิเศษ จึงเป็นกลุ่มคำที่ไม่นำมาประมวลผลการในงานค้นคืนและงานประมวลผลภาษาธรรมชาติ หรืออาจจะการรวบรวมคำศัพท์และข้อมูลที่เกี่ยวข้องเข้าไว้ด้วยกัน เช่น พจนานุกรมต่างๆ ที่เก็บคำศัพท์ พร้อมทั้งหน้าที่ของคำ และความหมายไว้ด้วยกัน

ในบทนี้เราจะได้รู้จักคลังคำและข้อความต่างๆ ที่มีให้ใช้ใน NLTK และเรียนรู้วิธีการใช้งานคลังเหล่านั้น โดยยกตัวอย่างการใช้งานบางคลัง เพื่อให้เข้าใจแนวคิดในการเข้าถึงและเรียกใช้ สามารถสร้างคลังข้อความขึ้นเอง และเรียกใช้ได้ รวมถึงสามารถนำแนวคิดนี้ไปใช้กับคลังอื่นๆ ได้

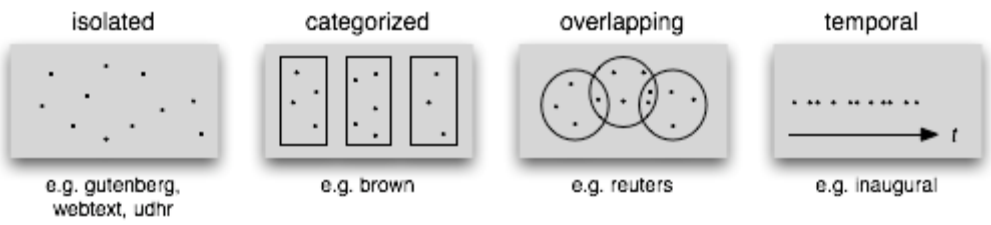
2.1 การเข้าถึงและใช้งานคลังข้อความ

ใน NLTK มีหลายคลังข้อความให้ใช้งาน ในส่วนนี้จะแสดงตัวอย่างการเข้าถึงและใช้งานบางคลังข้อความดังต่อไปนี้

- Gutenberg corpus คลังหนังสืออิเล็กทรอนิกส์ขนาดใหญ่ ส่วนใหญ่จะเป็นวรรณกรรม
- Web and Chat text คลังที่รวบรวมภาษาที่ปรากฏในเว็บหรือบทสนทนา
- Brown corpus คลังที่รวบรวมคำจากหนังสือหรือวารสารต่างๆ โดยมีการจัดแบ่งเป็นหมวดหมู่
- Inaugural address คลังที่รวบรวมคำปราศรัยของประธานาธิบดีของสหรัฐอเมริกาในวันเข้ารับตำแหน่งอย่างเป็นทางการ โดยจัดเก็บเป็นไฟล์ตามปี ค.ศ.

โครงสร้างคลังข้อความ

คลังข้อความแต่ละคลังใน NLTK มีโครงสร้างที่หลากหลายรูปแบบดังแสดงในรูป 2.1 โครงสร้างแบบง่ายที่สุด คือ แบบที่ไม่มีโครงสร้างใดๆ เป็นการรวบรวมข้อความเข้าด้วยกันเท่านั้น โครงสร้างส่วนใหญ่จะเป็นการจัดกลุ่มแยกตามประเภท แหล่งที่มา ผู้แต่ง หรือภาษา แต่ก็มีบางคลังที่มีโครงสร้างที่มีการจัดกลุ่มที่ทับซ้อน นั่นคือ ไฟล์หนึ่งๆ อาจจะจัดอยู่ได้หลายกลุ่ม และบางคลังมีโครงสร้างในลักษณะแยกตามช่วงเวลา



รูปที่ 2.1 โครงสร้างของคลังข้อความและตัวอย่างของคลังข้อความแบ่งตามโครงสร้าง

การเข้าถึงข้อความในคลังต่างๆ จะมี nltk.corpus.reader ซึ่งเป็นโมดูลที่รวบรวมชุดคำสั่งที่ใช้สำหรับคลังข้อความ ซึ่งสามารถใช้กับคลังข้อความที่สร้างขึ้นใหม่ได้ ตัวอย่างฟังก์ชันที่ใช้งานแสดงในตารางที่ 2.1 (ดูรายละเอียดเพิ่มเติมโดยใช้คำสั่ง help(nltk.corpus.reader) หรืออ่านเพิ่มเติมที่ *www.nltk.org/howto/corpus.html*)

ตาราง 2.1 ตัวอย่างฟังก์ชันสำหรับใช้งานคลังข้อความใน NLTK

ตัวอย่างฟังก์ชัน	**คำอธิบาย**
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 fields
words(categories=[c1,c2])	the words of the specified categories
sents()	the sentences of the whole corpus
sents(fileids=[f1,f2,f3])	the sentences of the specified fields
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

ตัวอย่างการใช้งานฟังก์ชันอ่านข้อความในคลัง และผลที่ได้

```
>>> from nltk.corpus import *
>>> raw = gutenberg.raw("burgess-busterbrown.txt")
>>> raw[1:20]
'The Adventures of B'
>>> words = gutenberg.words("burgess-busterbrown.txt")
>>> words[1:20]
['The', 'Adventures', 'of', 'Buster', 'Bear', 'by', 'Thornton', 'W', '.', 'Burgess', '1920', ']', 'I', 'BUSTER',
'BEAR', 'GOES', 'FISHING', 'Buster', 'Bear']
>>> sents = gutenberg.sents("burgess-busterbrown.txt")
>>> sents[1:20]
[['I'], ['BUSTER', 'BEAR', 'GOES', 'FISHING'], ['Buster', 'Bear', 'yawned', 'as', 'he', 'lay', 'on', 'his',
'comfortable', 'bed', 'of', 'leaves', 'and', 'watched', 'the', 'first', 'early', 'morning', 'sunbeams', 'creeping',
'through', ...], ...]
```

จากตัวอย่างจะเห็นข้อแตกต่างของผลที่ได้จาก 3 ฟังก์ชัน ดังนี้

- raw() ใช้ดึงข้อมูลดิบจากไฟล์ข้อความในคลังโดยไม่มีการประมวลผลใดๆ
- words() แบ่งข้อความในคลังออกเป็นประโยคจัดเก็บในรูปลิสต์ของคำ
- sents() แบ่งข้อความออกเป็นประโยค จัดเก็บอยู่ในรูปแบบลิสต์ของประโยค โดยในลิสต์ของแต่ละประโยคประกอบด้วยลิสต์ของคำในประโยคนั้น

Gutenberg Corpus

เป็นคลังข้อความที่รวบรวมข้อความบางส่วนจากหนังสืออิเล็กทรอนิกส์ในโครงการ Gutenberg ซึ่งเป็นแหล่งรวมไฟล์หนังสืออิเล็กทรอนิกส์ขนาดใหญ่กว่า 25,000 เล่มที่แจกให้ใช้ฟรี เนื่องจากหนังสือเหล่านั้นหมดลิขสิทธิ์และกลายเป็นสมบัติสาธารณะ โดยส่วนใหญ่จะเป็นหนังสือภาษาอังกฤษ และรวบรวมให้อยู่ในรูปไฟล์ข้อความที่ไม่มีรูปแบบ (plain text) ดูเพิ่มเติมได้ที่ *http://www.gutenberg.org/* ถ้าต้องการทราบว่าในคลังนี้จะประกอบด้วยไฟล์ข้อความของหนังสือใดบ้าง จะใช้คำสั่งดังตัวอย่างด้านล่างนี้

```
>>>import nltk
>>>nltk.corpus.gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt','blake-poems.txt',
'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-
brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-
paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt','shakespeare-macbeth.txt',
'whitman-leaves.txt']
```

ถ้าต้องการทราบว่าในหนังสือนั้นมีจำนวนคำเท่าไหร่ ก็สามารถใช้ฟังก์ชัน words() เพื่อดูได้ ดังตัวอย่างด้านล่างนี้ที่เก็บลิสต์ของคำในหนังสือ Emma ของ Jane Austen ไว้ในตัวแปรชื่อ emma

```
>>>emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>>len(emma)
192427
```

ถ้าไม่ต้องการอ้างถึงชื่อ corpus package ทุกครั้ง เราสามารถใช้คำสั่ง import เพื่อเรียกใช้คำสั่งให้สั้นลงได้

```
>>>from nltk.corpus import gutenberg
>>>gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', ...]
>>>emma = gutenberg.words('austen-emma.txt')
```

ในบทที่ 1 เราสามารถใช้คำสั่ง text1.concordance() เพื่อหาบริบทของคำที่สนใจได้ แต่เนื่องจากคำสั่งนี้ใช้ได้กับ book package ของ nltk เท่านั้น โดยต้องมีเรียกคำสั่ง from nltk.book import * ไว้ก่อน เพื่อให้เราสามารถใช้คำสั่งใน book package ได้กับ nltk.corpus จะต้องเรียกใช้คำสั่งดังตัวอย่างต่อไปนี้ เพื่อแปลงจากคำใน nltk.corpus เป็น Text เสียก่อน

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

ตัวอย่างต่อไปนี้ เป็นการใช้คำสั่งวนซ้ำร่วมกับฟังก์ชันต่างๆ ของ gutenberg object เพื่อแสดงข้อมูลและสถิติของไฟล์ต่างๆ ในคลัง Gutenberg เช่น ความยาวเฉลี่ยของคำ ความยาวเฉลี่ยของประโยค จำนวนครั้งเฉลี่ยของการใช้คำศัพท์

```
>>>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(round(num_chars/num_words), round(num_words/num_sents),
round(num_words/num_vocab), fileid)
...
```

```
5 25 26 austen-emma.txt
5 26 17 austen-persuasion.txt
5 28 22 austen-sense.txt
4 34 79 bible-kjv.txt
5 19 5 blake-poems.txt
4 19 14 bryant-stories.txt
4 18 12 burgess-busterbrown.txt
4 20 13 carroll-alice.txt
5 20 12 chesterton-ball.txt
5 23 11 chesterton-brown.txt
5 18 11 chesterton-thursday.txt
4 21 25 edgeworth-parents.txt
5 26 15 melville-moby_dick.txt
5 52 11 milton-paradise.txt
4 12 9 shakespeare-caesar.txt
4 12 8 shakespeare-hamlet.txt
4 12 7 shakespeare-macbeth.txt
5 36 12 whitman-leaves.txt
```

สังเกตว่าความยาวเฉลี่ยของคำในภาษาอังกฤษจะยาวประมาณ 4 ตัวอักษร แต่ความยาวเฉลี่ยของประโยคและความถี่ในการใช้คำศัพท์ซ้ำจะขึ้นอยู่กับนักเขียนแต่ละคน ตัวอย่างการหาประโยคที่มีความยาวสูงสุด

```
>>>macbeth_sentences = gutenberg.sents('shakespeare-macbeth.txt')
>>>macbeth_sentences
[['[', 'The', 'Tragedie', 'of', 'Macbeth', 'by', 'William', 'Shakespeare','1603', ']'], ['Actus', 'Primus', '.'], ...]
>>>macbeth_sentences[1116]
['Double', ',', 'double', ',', 'toile', 'and', 'trouble', ';', 'Fire', 'burne', ',', 'and', 'Cauldron', 'bubble']
>>>longest_len = max(len(s) for s in macbeth_sentences)
>>>[s for s in macbeth_sentences if len(s) == longest_len]
[['Doubtfull', 'it', 'stood', ',', 'As', 'two', 'spent', 'Swimmers', ',', 'that', 'doe', 'cling', 'together', ',', 'And',
'choake', 'their', 'Art', ':', 'The', 'mercilesse', 'Macdonwald', ...]]
```

Web and Chat Text

Webtext เป็นคลังข้อความที่รวบรวมข้อความจากเว็บ ซึ่งเป็นแหล่งที่แสดงถึงการใช้ภาษาอย่างไม่เป็นทางการ ต่างจากคลัง Gutenberg ที่รวบรวมจากหนังสือหรือวรรณกรรม ซึ่งเป็นการใช้ภาษาเขียนอย่างเป็นทางการ โดยข้อความที่รวบรวมจากเว็บได้จาก 6 แหล่งต่อไปนี้

- forum ที่อภิปรายเกี่ยวกับโปรแกรม Firefox
- บทสนทนาที่ได้ยินมาเกี่ยวกับเมืองนิวยอร์ค
- บทภาพยนตร์เรื่อง Pirates of the Carribean
- ประกาศโฆษณาของบุคคล (คล้ายมาลัยไทยรัฐ)
- รีวิวเกี่ยวกับไวน์

```
>>>from nltk.corpus import webtext
>>>gutenberg.fileids()
...     print fileid, webtext.raw(fileid)[:65], '...'
firefox.txt Cookie Manager: "Don't allow sites that set removed cookies to se...
grail.txt SCENE 1: [wind] [clop clop clop] KING ARTHUR: Whoa there!  [clop...
overheard.txt White guy: So, do you have any plans for this evening? Asian girl...
pirates.txt PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terr...
singles.txt 25 SEXY MALE, seeks attrac older single lady, for discreet encoun...
wine.txt Lovely delicate, fragrant Rhone wine. Polished leather and strawb...
```

นอกจากนี้ยังมีคลังข้อความ nps_chat ที่รวบรวมบทสนทนาจากโปรแกรมรับส่งข้อความผ่านอินเทอร์เน็ต (instant messaging) มีจำนวนมากกว่า 100,000 โพสต์ โดยมีการปกปิดชื่อหรือข้อมูลที่สามารถระบุตัวบุคคล และจัดกลุ่มเป็น 15 ไฟล์ตามกลุ่มอายุ (วัยรุ่น, 20, 30, 40, และกลุ่มผู้ใหญ่ทั่วไป) และวันที่ดึงข้อมูลมา รูปแบบไฟล์จะเป็น xml ชื่อไฟล์จะประกอบด้วยวันที่ กลุ่มอายุ และจำนวนโพสต์ทั้งหมด เช่น 10-19-20s_706posts.xml เป็นไฟล์ที่รวบรวมเอาโพสต์ในวันที่ 19 เดือน 10 ปี ค.ศ.2006 ของผู้ใช้กลุ่มอายุ 20 ปี มีจำนวนทั้งหมด 706 โพสต์

```
>>> from nltk.corpus import nps_chat
>>> chatroom = nps_chat.posts('10-19-20s_706posts.xml')
>>> chatroom[123]
['i', 'do', "n't", 'want', 'hot', 'pics', 'of', 'a', 'female', ',', 'I', 'can', 'look', 'in', 'a', 'mirror', '.']
```

Brown Corpus

เป็นคลังข้อความแรกที่รวบรวมข้อความขนาดใหญ่เป็นหลักล้านคำ จัดทำโดยมหาวิทยาลัย Brown ในปี ค.ศ.1961 ซึ่งรวบรวมจาก 500 แหล่ง โดยจัดกลุ่มเป็นหมวดหมู่ตามประเภทของเอกสารทั้งหมด 15 ประเภท เช่น ข่าว บทบรรณาธิการ รีวิว นิยาย ดังแสดงในตาราง 2.2 (ดูรายละเอียดเพิ่มเติมที่ *icame.uib.no/brown/bcm-los.html*)

ตาราง 2.2 เอกสารในคลังข้อความ Brown แบ่งตามประเภทของเอกสาร

รหัส	**ไฟล์**	**ประเภท**	**คำอธิบาย**
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*

ถ้าต้องการทราบจำนวนคำ หรือจำนวนประโยคของเอกสารประเภทใดในคลัง สามารถทำได้โดยระบุประเภทที่ต้องการเป็นพารามิเตอร์ หรือหากต้องการทราบในระดับไฟล์ก็ระบุชื่อไฟล์ที่ต้องการเป็นพารามิเตอร์

```
>>> from nltk.corpus import brown
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore',
'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']
>>> brown.words(categories='news')
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

```
>>> brown.words(fileids=['cg22'])
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
>>> brown.sents(categories=['news', 'editorial', 'reviews'])
[['The', 'Fulton', 'County'...], ['The', 'jury', 'further'...], ...]
```

คลัง Brown ยังเหมาะสำหรับการศึกษาเกี่ยวกับสำนวนหรือรูปแบบภาษาระหว่างเอกสารแต่ละประเภท ซึ่งในทางภาษาศาสตร์เรียกว่า วัจนลีลาศาสตร์ (Stylistics linguistics) เช่น การเปรียบเทียบการใช้กิริยาช่วย (modal verb) ของเอกสาร ดังตัวอย่างการนับกิริยาช่วยที่สนใจที่ปรากฏในเอกสารประเภทข่าว ในโค้ดต่อไปนี้

```
>>> 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],
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

Inaugural Address Corpus

เป็นคลังข้อความที่รวบรวมข้อความจากคำปราศรัยในวันเข้ารับตำแหน่งอย่างเป็นทางการของประธานาธิบดีสหรัฐอเมริกาโดยจัดเก็บเป็นไฟล์ตามปี ซึ่งต่างจาก text4 ในบทที่ 1 ที่มีการจัดเก็บรวมกันเป็นไฟล์เดียว ชื่อไฟล์จะระบุปี ค.ศ.และชื่อประธานาธิบดี ดังตัวอย่างการใช้งานต่อไปนี้ที่ดึงเฉพาะชื่อปี ค.ศ. มาแสดง

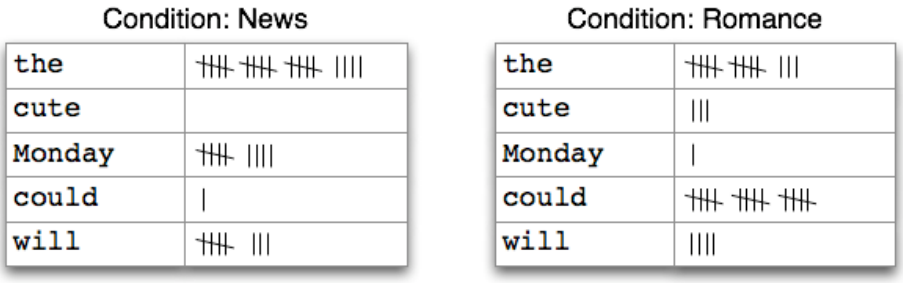
```
>>> from nltk.corpus import inaugural
>>> inaugural.fileids()
['1789-Washington.txt', '1793-Washington.txt', '1797-Adams.txt', ...]
>>> [fileid[:4] for fileid in inaugural.fileids()]
['1789', '1793', '1797', '1801', '1805', '1809', '1813', '1817', '1821', ...]
```

การแจกแจงความถี่แบบมีเงื่อนไข

ในบทที่แล้ว เราสามารถแจกแจงความถี่ซึ่งเป็นการนับสิ่งที่สนใจ (event) ในลิสต์และแสดงในรูปตารางหรือกราฟได้ โดยใช้ class FreqDist ใน NLTK แต่หากต้องการนับความถี่สิ่งที่สนใจโดยมีเงื่อนไข (condition) จะต้องใช้ class ConditionalFreqDist โดยจะนับข้อมูลที่อยู่ในรูปแบบ tuple ที่เป็นคู่ลำดับของเงื่อนไขและเหตุการณ์ในรูปแบบ (condition, event) ดังตัวอย่างการใช้งานต่อไปนี้

ตัวอย่างที่ 1 นับคำศัพท์โดยจำแนกตามหมวดหมู่ของเอกสารใน corpus Brown เราจะได้ตัวอย่างตารางการแจกแจงความถี่ดังรูปที่ 2.2 โดยเงื่อนไขคือหมวดหมู่ของเอกสาร และสิ่งที่สนใจคือคำศัพท์ สามารถเขียนเป็นโค้ดได้ดังนี้

```
>>> from nltk.corpus import brown
# สร้างคู่ลำดับของ (condition, event) เก็บไว้ในลิสต์
>>> genre_word = [(genre, word)
...               for genre in ['news', 'romance']
...               for word in brown.words(categories=genre)]
>>> genre_word[:4]   # ตัวอย่างคู่ลำดับของ (‘news’, คำศัพท์) ที่อยู่ต้นลิสต์
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
>>> genre_word[-4:]  # ตัวอย่างคู่ลำดับของ (‘romance’, คำศัพท์) ที่อยู่ท้ายลิสต์
[('romance', 'afraid'), ('romance', 'not'), ('romance', """), ('romance', '.')]
```



Condition: News

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

Condition: Romance

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

รูปที่ 2.2 การแจกแจงความถี่ของคำศัพท์แบบมีเงื่อนไขตามประเภทของเอกสารใน brown corpus

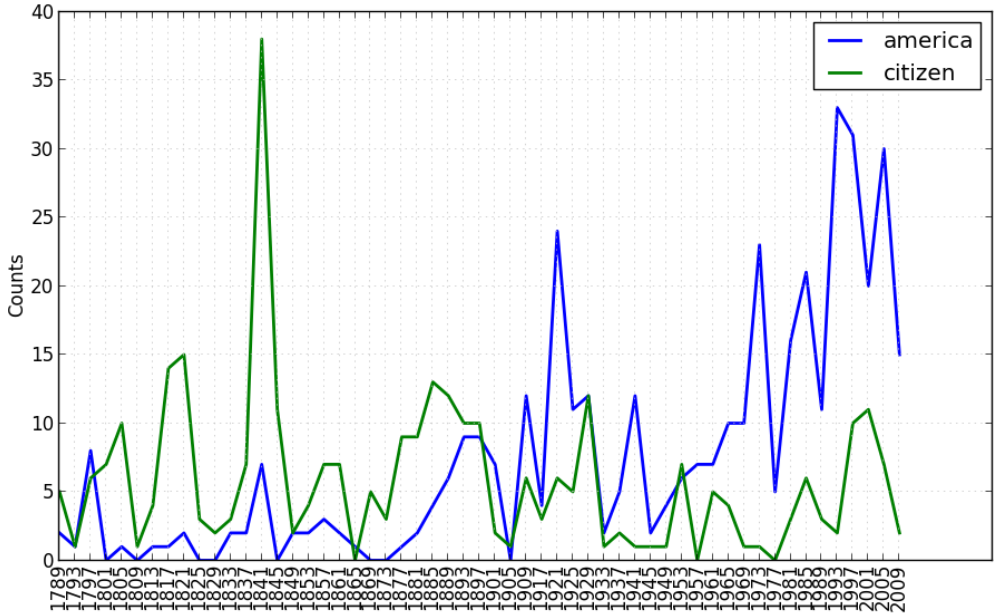
```
# นับความถี่โดยจำแนกตามเงื่อนไข
>>> cfd = nltk.ConditionalFreqDist(genre_word)
>>> cfd.conditions()
['news', 'romance']

# ตัวอย่างการใช้ cfd
>>> print(cfd['news'])
<FreqDist with 14394 samples and 100554 outcomes>
>>> print(cfd['romance'])
<FreqDist with 8452 samples and 70022 outcomes>
>>> cfd['romance'].most_common(20)
[(',', 3899), ('.', 3736), ('the', 2758), ('and', 1776), ('to', 1502),
('a', 1335), ('of', 1186), ('``', 1045), ("''", 1044), ('was', 993),
('I', 951), ('in', 875), ('he', 702), ('had', 692), ('?', 690),
('her', 651), ('that', 583), ('it', 573), ('his', 559), ('she', 496)]
>>> cfd['romance']['could']
193
```

ตัวอย่างที่ 2 การนับกิริยาช่วยที่สนใจที่ปรากฏในเอกสาร 5 ประเภทที่ระบุในลิสต์ genres

```
>>> 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)
                  can could  may might must will
          news 93   86    66   38    50   389
       religion 82   59    78   12    54    71
       hobbies 268   58   131   22    83   264
science_fiction  16   49     4   12     8    16
       romance  74  193    11   51    45    43
         humor  16   30     8    8     9    13
```

ตัวอย่างที่ 3 พิจารณาการใช้คำว่า America และ citizen ในแต่ละปี สามารถทำได้โดยใช้การแจกแจงความถี่แบบมีเงื่อนไขของคำในลิสต์ target โดยควรจะใช้ฟังก์ชัน startswith() และ w.lower() เพื่อค้นหาทุกคำที่ขึ้นต้นด้วยสองคำนี้ไม่ว่าจะเป็นตัวอักษรตัวเล็กหรือใหญ่ และแสดงกราฟในรูปที่ 2.3

```
>>> 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)) ❶
>>> cfd.plot()
```



รูปที่ 2.3 กราฟแสดงความถี่ของกลุ่มคำ ["citizen", "america"] ในคลัง inaugural

ตัวอย่างฟังก์ชันที่ใช้งาน ConditionalFreqDist แสดงในตารางที่ 2.3 (ดูรายละเอียดเพิ่มเติมโดยใช้คำสั่ง help(nltk.ConditionalFreqDist) หรืออ่านเพิ่มเติมที่ *http://www.inf.ed.ac.uk/teaching/courses/icl/nltk/probability.pdf*)

ตาราง 2.3 ตัวอย่างฟังก์ชันสำหรับใช้งาน class ConditionalFreqDist

ตัวอย่าง	คำอธิบาย
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()	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

2.2 การเข้าถึงและใช้งานคลังคำ

NLTK มีคลังคำที่จำเป็นให้ใช้งาน ในส่วนนี้จะแสดงตัวอย่างการเข้าถึงและใช้งานคลังคำบางส่วน ดังต่อไปนี้

- Wordlist
- Stopwords
- CMUDict
- Names
- WordNet

นอกจากนี้ เราสามารถหาคำที่คล้ายคลึงกันได้ โดยหาจากคำที่มีบริบทรอบข้างเหมือนกัน ดังตัวอย่าง

```
>>>text1.similar("silently")
mildly round steadfastly stood through

>>>text9.similar("silently")
close extraordinary indisputable slight steadily without
```

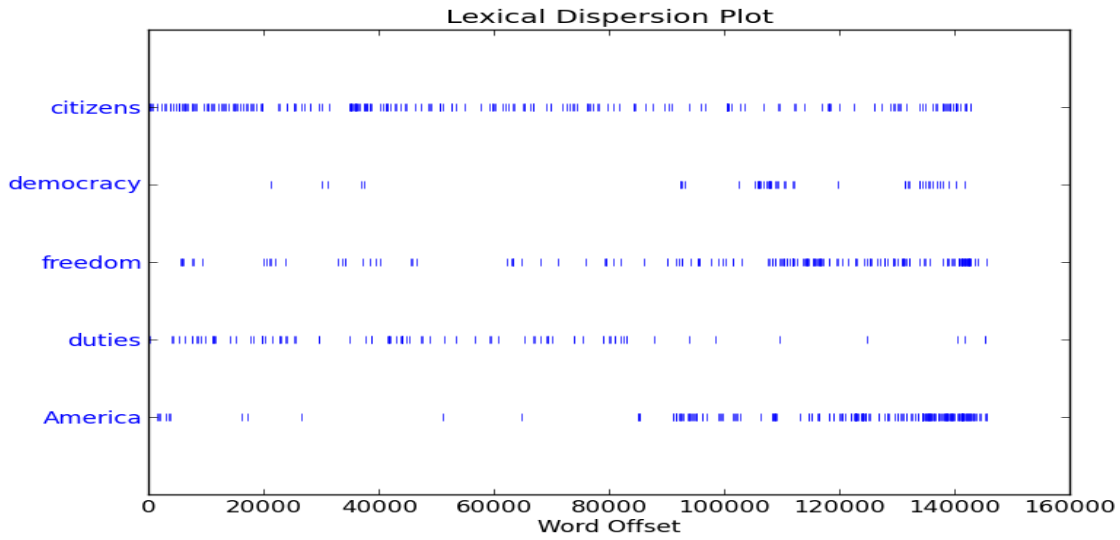
ตัวอย่างด้านล่างนี้ เป็นการค้นหาบริบทที่เหมือนกันของคำทั้งสามคำในลิสต์

```
>>>text1.common_contexts(['silently','mildly','steadfastly'])
and_eyeing
```

นอกจากนี้เรายังสามารถหาตำแหน่งของคำในเอกสารโดยนำมาแสดงผลในรูปของกราฟ เพื่อดูลักษณะการกระจายตัวในเอกสาร ดังตัวอย่างคำสั่งต่อไปนี้

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

จากคำสั่งข้างต้น จะได้ผลลัพธ์ดังรูปที่ 1.1 ทำให้เราสามารถวิเคราะห์รูปแบบการใช้คำทั้ง 5 ในลิสต์ตามแต่ละยุคสมัยได้



รูปที่ 1.1 กราฟแสดงตำแหน่งของกลุ่มคำ ["citizens", "democracy", "freedom", "duties", "America"] ในเอกสาร text4 ซึ่งเป็นคลังที่รวบรวมสุนทรพจน์ของประธานาธิบดีของสหรัฐอเมริกาในวันเข้ารับตำแหน่ง

เมื่อเราต้องการนับจำนวนครั้งที่โทเค็นใดๆ ปรากฏในเอกสาร จะใช้คำสั่ง countนอกจากนี้ เรายังสามารถหาเปอร์เซ็นของการใช้โทเค็นใดๆ เมื่อเทียบกับจำนวนโทเค็นทั้งหมดในเอกสารได้ ดังตัวอย่าง

```
>>>text3.count("smote")
5
>>>100 * text4.count('a') / len(text4)
1.4643016433938312
```

เราสามารถเขียนชุดคำสั่งที่ยกตัวอย่างมาแล้วข้างต้นเป็นฟังก์ชันเพื่อสะดวกในการเรียกใช้งานในครั้งถัดไปได้ ดังตัวอย่างต่อไปนี้

```
>>>deflexical_diversity(text):
...     return len(set(text)) / len(text)
...
>>>defpercentage(count, total):
...     return 100 * count / total
...
>>>lexical_diversity(text3)
0.06230453042623537
>>>lexical_diversity(text5)
0.13477005109975562
>>>percentage(4, 5)
80.0
>>>percentage(text4.count('a'), len(text4))
1.4643016433938312
>>>
```

1.3 สถิติกับการประมวลผลทางภาษา

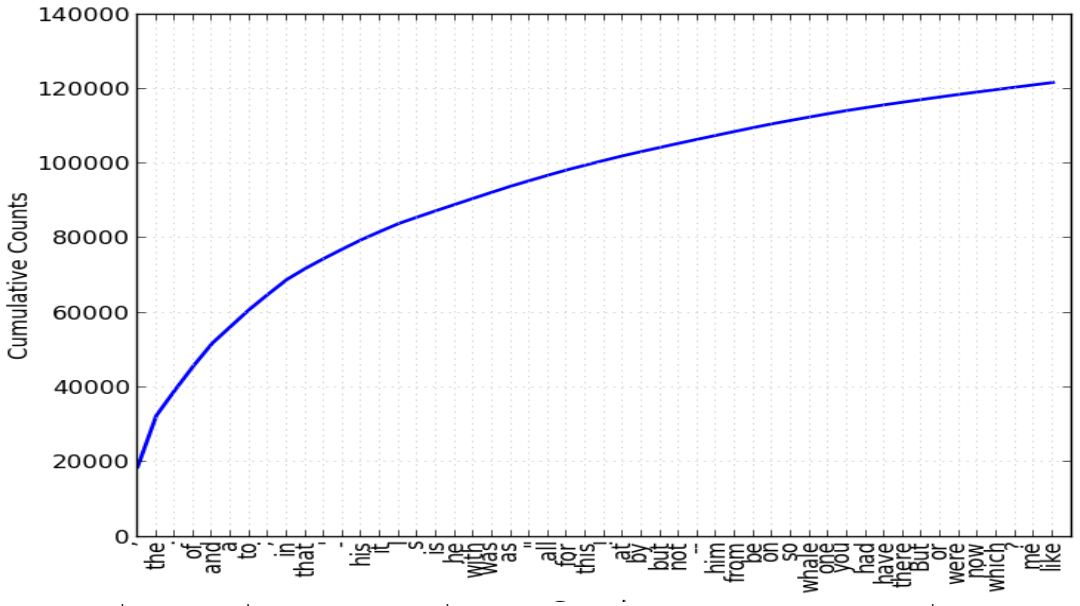
การประมวลผลข้อความมักจะเกี่ยวข้องกับการคำนวณหาค่าทางสถิติต่างๆ สถิติที่มักจะใช้บ่อยได้แก่การแจกแจงความถี่ (frequency distribution) เพื่อดูลักษณะการกระจายตัวของโทเค็นบนคำศัพท์ทั้งหมดในเอกสาร ใน NLTK เราสามารถใช้ชุดคำสั่ง FreqDist เพื่อหาการแจกแจงความถี่ของคำศัพท์ในเอกสารได้ ดังตัวอย่าง

```
>>>fdist = FreqDist(text1)
>>>print(fdist)
<FreqDist with 19317 samples and 260819 outcomes>
>>>fdist.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024),('a', 4569), ('to', 4542), (';', 4072),
('in', 3916), ('that', 2982),("'", 2684), ('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124),('s', 1739),
('is', 1695), ('he', 1661), ('with', 1659), ('was', 1632),('as', 1620), ('"', 1478), ('all', 1462),
('for', 1414), ('this', 1280),('!', 1269), ('at', 1231), ('by', 1137), ('but', 1113), ('not', 1103),
('--', 1070), ('him', 1058), ('from', 1052), ('be', 1030), ('on', 1005),('so', 918), ('whale', 906),
('one', 889), ('you', 841), ('had', 767),('have', 760), ('there', 715), ('But', 705), ('or', 697),
('were', 680),('now', 646), ('which', 640), ('?', 637), ('me', 627), ('like', 624)]
>>>fdist['whale']
906
>>>fdist.plot(50, cumulative=True)
```

จากตัวอย่าง เป็นการใช้ชุดคำสั่ง FreqDist เพื่อแจกแจงความถี่ของเอกสาร text1แล้วเก็บค่าไว้ในตัวแปร fdist เมื่อต้องการหาความถี่ของคำศัพท์ใดๆ สามารถอ้างถึงได้โดยใช้ตัวแปรfdist1 แล้วระบุคำที่ต้องการทราบความถี่ในรูปดัชนี ส่วนการหาความถี่ของคำศัพท์ที่พบบ่อยๆ จำนวน n ตัวแรก จะใช้คำสั่ง most_common(n) คำสั่งสุดท้ายในตัวอย่างจะแสดงกราฟความถี่สะสมของคำศัพท์ที่พบบ่อย 50 คำแรก ดังรูปที่ 1.2 ถ้าเราสนใจคำศัพท์ที่ปรากฏเพียงครั้งเดียวในเอกสาร จะใช้คำสั่ง hapaxes() ส่วนตัวอย่างคำสั่งอื่นๆ ใน FreqDist แสดงในตารางที่ 1.2

การสกัดคำหรือวลีสำคัญ (keyword extraction)เป็นงานประมวลผลข้อความที่สามารถใช้การแจกแจงความถี่ เพื่อหาลักษณะของคำสำคัญที่สามารถใช้เป็นตัวแทนของเอกสารได้ จากตัวอย่างจะเห็นได้ว่า คำศัพท์ที่พบได้บ่อยที่สุดในเอกสารไม่ใช่คำที่เหมาะสมสำหรับเป็นตัวแทนของเอกสารได้ และจากกราฟในรูป 1.2 จะพบว่า ความถี่

รวมของคำศัพท์ที่พบบ่อย 50 แรก มีจำนวนเกือบครึ่งหนึ่งของจำนวนโทเค็นทั้งหมดในเอกสาร ถ้าทดลองใช้คำสั่ง hapaxes จะพบว่ามีคำจำนวนมาก (ประมาณ 9000 คำ) ที่ปรากฏเพียงแค่ครั้งเดียวในเอกสาร ซึ่งก็ไม่เหมาะที่จะนำมาเป็นคำสำคัญเช่นกัน



รูปที่ 1.2 ความถี่สะสมของคำศัพท์ที่พบบ่อย 50 คำแรกในเอกสาร text1 (หนังสือชื่อ Moby Dick)

ตารางที่ 1.2 ตัวอย่างฟังก์ชันใน FreqDist

ตัวอย่าง	อธิบายการใช้
fdist = FreqDist(samples)	สร้างตารางแจกแจงความถี่ของเอกสาร samples
fdist['monstrous'] += 1	เพิ่มความถี่ให้กับคำศัพท์
fdist['monstrous']	ความถี่ของคำศัพท์
fdist.freq('monstrous')	ค่าความหลากหลายของคำศัพท์
fdist.N()	จำนวนโทเค็นทั้งหมดใน samples
fdist.most_common(n)	แสดงคำและความถี่ ที่มีความถี่สูงสุด n อันดับแรก
for sample in fdist:	วนลูปเท่ากับจำนวนคำศัพท์ใน samples
fdist.max()	คำศัพท์ที่มีความถี่สูงสุด
fdist.tabulate()	แสดงคำศัพท์เรียงตามความถี่
fdist.plot()	แสดงการแจกแจงความถี่ในรูปกราฟ
fdist.plot(cumulative=True)	แสดงการแจกแจงความถี่สะสมในรูปกราฟ
fdist1 |= fdist2	ปรับปรุงค่าความถี่ในตัวแปร fdist1 โดยรวมกับความถี่ในตัวแปร fdist2

หากเราตั้งสมมติฐานว่า คำที่มีความยาวมากน่าจะเป็นคำสำคัญ เช่น ความยาวของคำมากกว่า 15 ตัวอักษร สามารถเขียนอยู่ในรูปของเซตได้ว่า $\{w \mid w \in V \& P(w)\}$ เมื่อ $P(w)$ คือคุณสมบัติของคำ w ที่จะเป็นจริง ถ้าคำมีความยาวมากกว่า 15 ตัวอักษรส่วน V คือเซตของคำศัพท์ทั้งหมด เมื่อแปลงเป็นนิพจน์ในภาษาไพธอนจะเขียนอยู่ในรูปของการสร้างลิสต์จากข้อมูลในลิสต์อื่น (list comprehension) ได้ดังนี้ [w for w in V if p(w)] และสามารถแปลงเป็นคำสั่งได้ตัวอย่างต่อไปนี้

```
>>V = set(text1)
>>>long_words = [w for w in V if len(w) > 15]
>>>sorted(long_words)
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', 'cannibalistically',
'characteristically', 'circumnavigating', 'circumnavigation', 'circumnavigations',
'comprehensiveness', 'hermaphroditical', 'indiscriminately', 'indispensableness',
'irresistibleness', 'physiognomically', 'preternaturalness', 'responsibilities',
'simultaneousness', 'subterraneousness', 'supernaturalness', 'superstitiousness',
'uncomfortableness', 'uncompromisedness', 'undiscriminating', 'uninterpenetratingly']
```

จะเห็นว่าคำที่มีความยาวมากส่วนใหญ่จะเป็นคำวิเศษณ์ และคำนามที่มีรากศัพท์มาจากคำคุณศัพท์ ซึ่งคำทั้งสองประเภทนี้ ไม่ใช่ลักษณะของคำสำคัญอีกเช่นกัน แต่ด้วยวิธีการนี้ทำให้เราสามารถตัดคำสั้นที่มีความถี่มาก รวมถึงตัดคำยาวที่มีความถี่น้อยออกไปได้ ดังนั้น หากพิจารณาคุณสมบัติทั้งในแง่ความยาวของคำศัพท์ร่วมกับความถี่ก็อาจจะให้ผลที่ดีขึ้นได้ ดังตัวอย่างคำสั่งต่อไปนี้ที่กำหนดเงื่อนไขว่าเลือกคำที่มีความยาวมากกว่า 7 ตัวอักษร และมีความถี่มากกว่า 7 ครั้ง

```
>>>fdist5 = FreqDist(text5)
>>>sorted(w for w in set(text5) if len(w) > 7 and fdist5[w] > 7)
['#14-19teens', '#talkcity_adults', '((((((((((', '.....', 'Question',
'actually', 'anything', 'computer', 'cute.-ass', 'everyone', 'football',
'innocent', 'listening', 'remember', 'seriously', 'something', 'together',
'tomorrow', 'watching']
```

จากวิธีการข้างต้น จะเห็นว่าเราสามารถเปลี่ยนข้อมูลดิบที่ประกอบด้วยคำจำนวนมากเป็นสารสนเทศที่มีโอกาสเป็นคำสำคัญที่ใช้เป็นตัวแทนของเอกสารได้ โดยใช้คำสั่งในการประมวลผลไม่กี่คำสั่งแม้ว่าขั้นตอนนี้จะยังไม่สามารถสกัดคำสำคัญออกมาได้ แต่ก็เพียงพอที่จะนำไปใช้ในขั้นตอนต่อไปได้ไม่ยาก

นอกจากนี้ เราสามารถประยุกต์ชุดคำสั่งการแจกแจงความถี่ ในการวิเคราะห์ค่าทางสถิติต่างๆ ได้ตัวอย่างเช่น การนับความถี่ของคำที่มีความยาวต่างกันเพื่อพิจารณาการกระจายตัวของความยาวของคำได้ โดยใช้ FreqDist กับลิสต์ของความยาวของคำทั้งหมดในเอกสาร ดังตัวอย่างต่อไปนี้

```
>>>len_list = [len(w) for w in text1] ❶
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1, 1, 8, 2, 1, 4, 11, 5, 2, 1, 7, 6, 1, 3, 4, 5, 2, ...]
>>>fdist = FreqDist(len_list) ❷
>>>print(fdist) ❸
<FreqDist with 19 samples and 260819 outcomes>
>>>fdist
FreqDist({3: 50223, 1: 47933, 4: 42345, 2: 38513, 5: 26597, 6: 17111, 7: 14399,  8: 9966, 9: 6428,
10: 3528, ...})
>>>fdist.most_common()
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), (5, 26597), (6, 17111), (7, 14399),
(8, 9966), (9, 6428), (10, 3528), (11, 1873), (12, 1053), (13, 567), (14, 177),
(15, 70), (16, 22), (17, 12), (18, 1), (20, 1)]
```

```
>>>fdist.max()
3
>>>fdist[3]
50223
>>>fdist.freq(3)
0.19255882431878046
```

จากตัวอย่าง คำสั่งที่ 1 เป็นการสร้างลิสต์ของความยาวของคำในเอกสารจากนั้นจึงนำมาแจกแจงความถี่โดยใช้ FreqDist ในคำสั่งที่ 2 เมื่อสั่งพิมพ์ค่า fdist จะพบว่าเอกสาร text1 มีคำที่มี 19 ความยาวที่แตกต่างกัน โดยความยาวที่พบมากสุด คือ ขนาด 3 ตัวอักษร ซึ่งมีจำนวนทั้งหมด 50223 คำ และความหลากหลายของคำศัพท์ที่มีความยาว 3 ตัวอักษรมีค่า 0.19 (50223/260819)

หลักการทางสถิติอีกวิธีการหนึ่งที่เกี่ยวข้องกับงานการประมวลผลข้อความ คือ การหาความน่าจะเป็นของคำที่ปรากฏร่วมกัน n คำ เรียกว่า n-gram เช่น ถ้า n=2 จะเรียกว่า 2-gram หรือ bigram จะเป็นการหาค่าความน่าจะเป็นของคำใดๆ โดยดูจากคำก่อนหน้าเพียงหนึ่งคำ เรานิยมเขียนค่าความน่าจะเป็นนี้ในรูปของ P(w2|w1) ซึ่งหมายถึงความน่าจะเป็นที่คำ w2 จะเกิดตามหลังคำ w1

ในภาษาไพธอนเราสามารถสร้าง bigram จากลิสต์ได้โดยใช้คำสั่ง bigrams() ดังตัวอย่าง

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

เมื่อเราได้ลิสต์ของ bigram แล้ว เราสามารถนำมาหาคำที่เกิดขึ้นพร้อมกันบ่อยๆ ที่เรียกว่า collocation ได้โดยดูจากค่าความน่าจะเป็นของ bigram ที่มีความถี่สูง ซึ่งขั้นตอนทั้งหมดเราสามารถคำสั่งcollocations() ใน NLTK ที่ใช้ในการหาคำประเภทนี้ได้เลย ดังตัวอย่าง

```
>>>text4.collocations()
United States; fellow citizens; four years; years ago; FederalGovernment; General Government;
American people; Vice President; OldWorld; Almighty God; Fellow citizens; Chief Magistrate;
Chief Justice;God bless; every citizen; Indian tribes; public debt; one another;foreign nations;
political parties
>>>text8.collocations()
would like; medium build; social drinker; quiet nights; non smoker;long term; age open; Would like;
easy going; financially secure; funtimes; similar interests; Age open; weekends away; poss rship;
wellpresented; never married; single mum; permanent relationship; slimbuild
```

collocation มักจะเป็นคำที่จำเป็นต้องใช้ร่วมกันในประโยค และให้ความหมายพิเศษต่างไปจากเดิม คำที่ใช้ร่วมกันนี้ จะใช้คำอื่นที่มีความหมายทำนองเดียวกันแทนที่ไม่ได้ เช่น fast color หมายถึงสีไม่ตก ถ้าเรานำคำที่มีความหมายทำนองเดียวกันมาแทนที่ เช่น stable color หรือ speed color จะกลายเป็นคำที่มีความหมายผิดไปหรือเป็นคำคู่ที่ไม่ใช้ในภาษานั้นเพื่อให้ความหมายแบบเดียวกัน ดังนั้น fast color จึงถือเป็น collocationถ้าเป็นคำที่ปรากฏร่วมกันแต่ไม่มีความหมายพิเศษก็จะไม่ถือว่าเป็น collocation เช่น the color หรือ red color