WEB SCRAPING

PROCESSING RAW TEXT

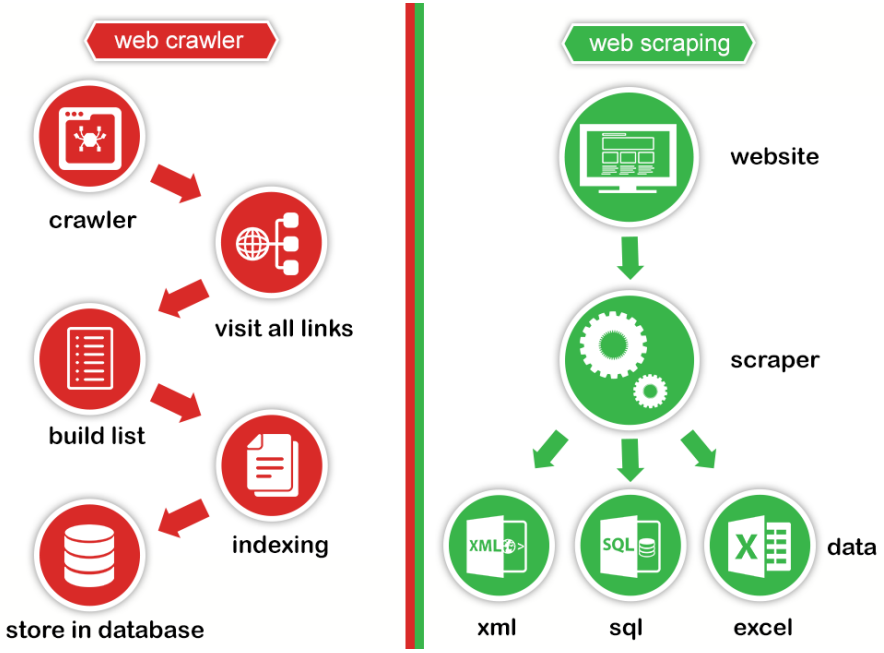
Tasanawan Soonklang

เป้าหมาย

- ทำ Web scraping ดึงข้อมูลจากเว็บมาประมวลผลได้
- เขียนโปรแกรมเพื่อเข้าถึงข้อความจากไฟล์ในเครื่อง หรือจากเว็บไซต์ต่างๆ
- เขียนโปรแกรมเพื่อสร้างผลลัพธ์ตามรูปแบบที่ต้องการ และบันทึกผลลงในไฟล์

คำศัพท์ที่เกี่ยวข้อง

- Tokenization
- Trimming
- Stemming
- Regular expression



Resource

Electronics books

Local files

Format

Text files

HTML

RSS Feeds

PDF

MSWord

Binary formats

Web Scraping

การดึงข้อมูลจากเว็บ โดยดึงเฉพาะข้อมูลที่ต้องการได้อัตโนมัติ

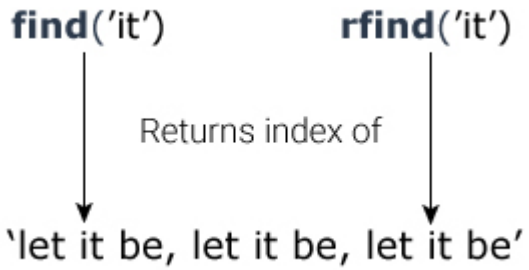
ทดลองเข้าถึงข้อมูล text files จากเว็บ

- Project Gutenberg, free online books
- เข้าถึงได้ผ่าน URL ได้เป็น ASCII text file
- คำสั่ง read() ทำการดาวน์โหลดหนังสือจาก url ที่ระบุ

```
>>> from urllib import request
>>> url = "http://www.gutenberg.org/files/2554/2554-0.txt"
>>> response = request.urlopen(url)
>>> raw = response.read().decode('utf8')
>>> type(raw)
<type 'str'>
>>> len(raw)
1176893
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n'
```

Trimming

- การตัดข้อความที่ไม่เกี่ยวข้องออก เช่น ข้อความที่เป็น header หรือ footer ของไฟล์
- find() และ rfind() – reverse find เป็น method ที่ใช้ในการหา right index เพื่อช่วยในตัดข้อความที่ไม่ต้องการออก




```
>>> raw.find("PART I")
5336
>>> raw.rfind("End of Project Gutenberg's Crime and Punishment")
1157808
>>> raw = raw[5336:1157808]
>>> raw.find("PART I")
0
```

Tokenization

- เป็นการแบ่งข้อความออกเป็น word และ punctuation

```
>>> tokens = nltk.word_tokenize(raw)
>>> type(tokens)
<type 'list'>
>>> len(tokens)
254347
>>> tokens[:10]
['The', 'Project', 'Gutenberg', 'EBook', 'of', 'Crime', 'and', 'Punishment', ',', 'by']
```

- สามารถแปลง list ที่ผ่านการ tokenization แล้วให้อยู่ในรูปของ NLTK Text ได้

```
>>> text = nltk.Text(tokens)
>>> type(text)
<type 'nltk.text.Text'>
>>> text[1019:1029]
['CHAPTER', 'I', 'On', 'an', 'exceptionally', 'hot', 'evening', 'early', 'in','July']
>>> text.collocations()
Katerina Ivanovna; Pulcheria Alexandrovna; … Project Gutenberg;  Pyotr Petrovitch; …;
Hay Market
```

Dealing with HTML

- รูปแบบของข้อความในเว็บมักจะอยู่ในรูปของไฟล์ HTML

```
>>> url = "http://news.bbc.co.uk/2/hi/health/2284783.stm"
>>> html = request.urlopen(url).read().decode('utf8')
>>> html[:60]
'<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN'
```

- เราสามารถกำจัด tag ของ HTML ได้ เพื่อดึงแต่ raw text มาทำการประมวลผลต่อไป
- ใช้ library จาก http://www.crummy.com/software/BeautifulSoup/

```
>>> from bs4 import BeautifulSoup
>>> raw = BeautifulSoup(html).get_text()
>>> tokens = nltk.word_tokenize(raw)
>>> tokens
['BBC', 'NEWS', '|', 'Health', '|', 'Blondes', "'", 'to', 'die', 'out', ...]
```

- ทดลอง trimming tokens ด้านบน เพื่อให้ได้เฉพาะ content ที่สนใจ และหา concordance ของคำว่า ‘gene’

Collocation

- การค้นคืนโดยใช้ search engines ช่วยในการหา linguistic pattern ที่เราสนใจได้
- ตัวอย่างเช่น เราต้องการหา collocation ของการใช้คำ ดังตาราง

Google hits	*adore*	*love*	*like*	*prefer*
absolutely	289,000	905,000	16,200	644
definitely	1,460	51,000	158,000	62,600
ratio	198:1	18:1	1:10	1:97

- ทดลองค้นข้อความจากเว็บต่างๆ เพื่อหาว่า ‘the of’ เป็น collocation ของภาษาอังกฤษหรือไม่

Reading Local Files

- อ่านไฟล์โดยใช้ open() และ read()

```
>>> f = open('document.txt')
>>> raw = f.read()

>>> f.read()
'Time flies like an arrow.\nFruit flies like a banana.\n‘
```

- r หมายถึงเปิดไฟล์เพื่ออ่าน

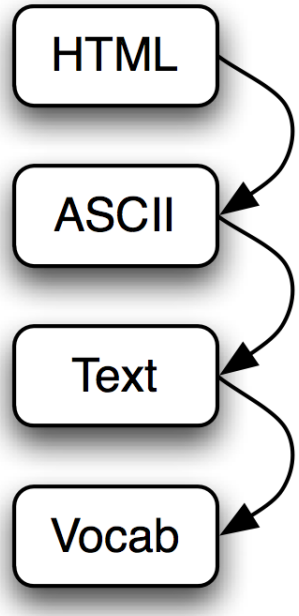
```
>>> f = open('document.txt', 'r')
>>> for line in f:
...     print line.strip()
Time flies like an arrow.
Fruit flies like a banana.

>>> path = nltk.data.find('corpora/gutenberg/melville-moby_dick.txt')
>>> raw = open(path, 'r').read()
```

The NLP Pipeline

- อ่านจากการรับข้อความเข้าทางคีย์บอร์ด

```
>>> s = input("Enter some text: ")
Enter some text: On an exceptionally hot evening early in July
>>> print("You typed", len(nltk.word_tokenize(s)), "words.“)
You typed 8 words.
```




```
html = urlopen(url).read()
raw = nltk.clean_html(html)
raw = raw[750:23506]
```

Download web page, strip HTML if necessary, trim to desired content

```
tokens = nltk.wordpunct_tokenize(raw)
tokens = tokens[20:1834]
text = nltk.Text(tokens)
```

Tokenize the text, select tokens of interest, create an NLTK text

```
words = [w.lower() for w in text]
vocab = sorted(set(words))
```

Normalize the words, build the vocabulary

The NLP Pipeline

- อ่านข้อมูลจากไฟล์ จะได้เป็นข้อความ (string)

```
>>> raw = open('document.txt').read()
>>> type(raw)
<class 'str'>
```

- แบ่งข้อความเป็นคำ จะได้ลิสต์ของคำ

```
>>> tokens = word_tokenize(raw)
>>> type(tokens)
<class 'list'>
```

- ลดคำที่ซ้ำซ้อน (normalize) และจัดเรียงคำตามลำดับตัวอักษร (sort)

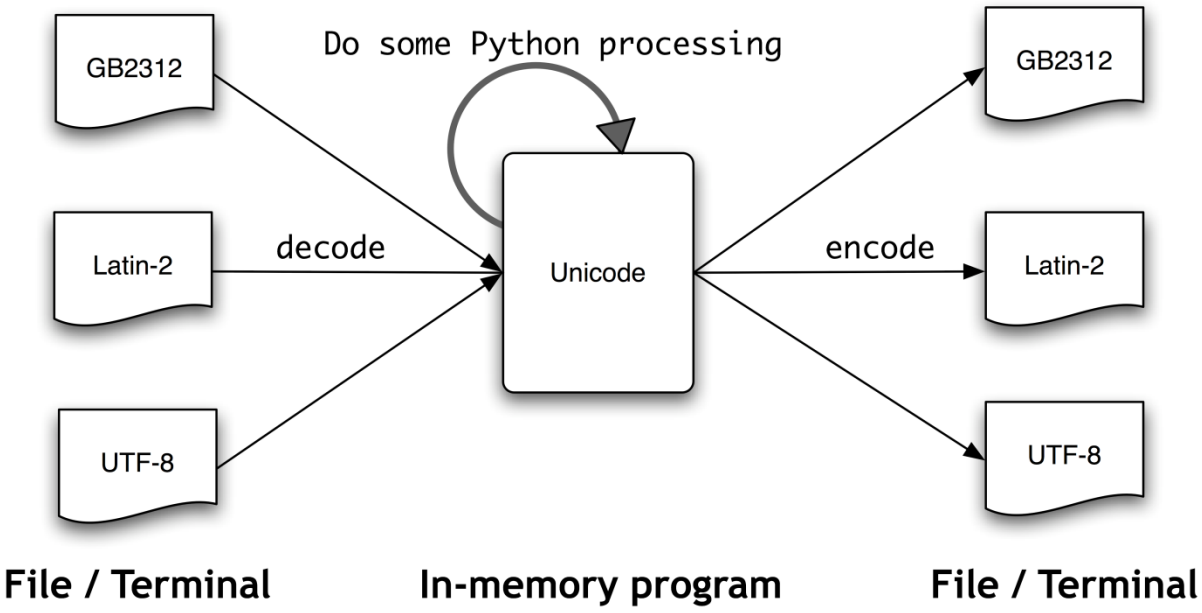
```
>>> words = [w.lower() for w in tokens]
>>> type(words)
<class 'list‘>
>>> vocab = sorted(set(words))
>>> type(vocab)
<class 'list'>
```

Useful String Methods

Method	Functionality
s.find(t)	index of first instance of string t inside s (-1 if not found)
s.rfind(t)	index of last instance of string t inside s (-1 if not found)
s.index(t)	like s.find(t) except it raises ValueError if not found
s.rindex(t)	like s.rfind(t) except it raises ValueError if not found
s.join(text)	combine the words of the text into a string using s as the glue
s.split(t)	split s into a list wherever a t is found (whitespace by default)
s.splitlines()	split s into a list of strings, one per line
s.lower()	a lowercased version of the string s
s.upper()	an uppercased version of the string s
s.title()	a titlecased version of the string s
s.strip()	a copy of s without leading or trailing whitespace
s.replace(t, u)	replace instances of t with u inside s

Text Processing with Unicode

- แต่ละตัวอักษรแทนค่าด้วยตัวเลข เรียกว่า code point เขียนอยู่ในรูป \uXXXX
- เขียนโปรแกรมจัดการ Unicode string ได้ตามปกติ แต่ในการเขียนลงไฟล์ต้อง encode ก่อน
- UTF-8 เป็นการ encoding แบบใช้หลายไบต์ต่อหนึ่ง code point ครอบคลุมตัวอักษรทุกตัว
- เมื่อต้องการอ่านไฟล์ก็ต้องทำการ decoding



Extracting Encoded Text from Files

```
>>> path = nltk.data.find('corpora/unicode_samples/polish-lat2.txt')
>>> f = open(path, encoding='latin2')
>>> for line in f:
...         line = line.strip()
...         print(line)
Pruska Biblioteka Państwowa. Jej dawne zbiory znane pod nazwą "Berlinka" to skarb kultury i
sztuki niemieckiej. Przewiezione przez Niemców pod koniec II wojny światowej na Dolny Śląsk,
zostały odnalezione po 1945 r. na terytorium Polski. Trafiły do Biblioteki Jagiellońskiej w
Krakowie, obejmują ponad 500 tys. zabytkowych archiwaliów, m.in. manuskrypty Goethego,
Mozarta, Beethovena, Bacha.

>>> url ='http://www.cl.cam.ac.uk/~mgk25/ucs/examples/UTF-8-demo.txt‘
>>> from urllib import request
>>> request.urlopen(url).read().decode('utf8')

>>> ord('ń')
324
>>> nacute=‘\u0144’
>>> nacute
'ń'
```

Extracting Encoded Text from Files

```
>>> import unicodedata
>>> lines = open(path, encoding='latin2').readlines()
>>> line = lines[2]
>>> print(line.encode('unicode_escape'))
b'Niemc\\xf3w pod koniec II wojny \\u015bwiatowej na Dolny \\u015al\\u0105sk, zosta\\u0142y\\n'
>>> word_tokenize(line)
['Niemców', 'pod', 'koniec', 'II', 'wojny', 'światowej', 'na', 'Dolny', 'Śląsk', ',', 'zostały']

>>> for c in line:
...         if ord(c) > 127:
...         print('{} U+{:04x} {}'.format(c.encode('utf8'), ord(c), unicodedata.name(c)))
b'\xc3\xb3' U+00f3 LATIN SMALL LETTER O WITH ACUTE
b'\xc5\x9b' U+015b LATIN SMALL LETTER S WITH ACUTE
b'\xc5\x9a' U+015a LATIN CAPITAL LETTER S WITH ACUTE
b'\xc4\x85' U+0105 LATIN SMALL LETTER A WITH OGONEK
b'\xc5\x82' U+0142 LATIN SMALL LETTER L WITH STROKE
```

Processing RSS Feeds

- Download library จาก Universal Feed Parser – feedparser.org
- ใช้ในการเข้าถึง content ของ blog ได้

```
>>> import feedparser
>>> llog = feedparser.parse("http://languagelog.ldc.upenn.edu/nll/?feed=atom")
>>> llog['feed']['title']
u'Language Log'
>>> len(llog.entries)
15
>>> post = llog.entries[2]
>>> post.title
u"He's My BF"
>>> content = post.content[0].value
>>> content[:70]
u'<p>Today I was chatting with three of our visiting graduate students f'
>>> nltk.word_tokenize(nltk.html_clean(content))
>>> nltk.word_tokenize(nltk.clean_html(llog.entries[2].content[0].value))
[u'Today', u'I', u'was', u'chatting', u'with', u'three', u'of', u'our', u'visiting',
u'graduate', u'students', u'from', u'the', u'PRC', u'.', u'Thinking', u'that', u'I', ...]
```

Regular Expression

- regex หรือ regexp
- สตริงที่ประกอบด้วยสัญลักษณ์พิเศษที่กำหนดขึ้น
- ใช้ในการอธิบาย pattern ที่ต้องการค้นหา
- รองรับการทำงานในหลายๆ ภาษาโปรแกรม
- Online tools, e.g. https://regexr.com/

ตัวอย่างการใช้งาน

- เช็ค validation
- ค้นหาคำ ไฟล์
- เปลี่ยนชื่อคำ หรือไฟล์ที่ค้นหา
- คัดกรองคำไม่สุภาพ

Regular Expression Meta-Characters

Operator	Behavior
.	Wildcard, matches any character
^abc	Matches some pattern *abc* at the start of a string
abc$	Matches some pattern *abc* at the end of a string
[abc]	Matches one of a set of characters
[A-Z0-9]	Matches one of a range of characters
ed|ing|s	Matches one of the specified strings (disjunction)
*	Zero or more of previous item, e.g. a*, [a-z]* (also known as *Kleene Closure*)
+	One or more of previous item, e.g. a+, [a-z]+
?	Zero or one of the previous item (i.e. optional), e.g. a?, [a-z]?
{n}	Exactly *n* repeats where n is a non-negative integer, e.g. a{5}
{n,}	At least *n* repeats
{,n}	No more than *n* repeats
{m,n}	At least *m* and no more than *n* repeats
a(b|c)+	Capturing group, parentheses that indicate the scope of the operators

Regular Expression Meta-Characters

Operator	Behavior
\w	Matches any word character (alphanumeric & underscore). Only matches low-ascii characters (no accented or non-roman characters). Equivalent to [A-Za-z0-9_]
\W	Matches any character that is not a word character (alphanumeric & underscore). Equivalent to [^A-Za-z0-9_]
\d	Matches any digit character (0-9). Equivalent to [0-9].
\D	Matches any character that is not a digit character (0-9). Equivalent to [^0-9].
\s	Matches any whitespace character (spaces, tabs, line breaks)
\S	Matches any character that is not a whitespace character (spaces, tabs, line breaks).

Regex in Python

```
import re

pattern='\d{4}'

regex = re.compile(pattern)

years = regex.findall(text)
```

Method

• compile(pattern)	create regex object
• findall(text)	return list of matched strings in text
• finditer(pattern, text)	return iterator of matched string in text
• sub(pattern1, pattern2, text)	replace pattern1 with pattern2 in text
• split(pattern, text)	remove pattern from text and then split
• search(text)	???

Normalizing Text

- การปรับเปลี่ยนข้อความเพื่อเตรียมการให้เหมาะสำหรับการประมวลผล
- ตัวอย่างเช่น
  - การเปลี่ยนให้เป็นตัวอักษรตัวเล็กทั้งหมด เพื่อไม่ให้เกิดความแตกต่างระหว่าง the, The, THE
  - การเปลี่ยนตัวอักษรย่อ เป็นคำเต็ม
  - การเปลี่ยนวันเดือนปีเป็นตัวเลขเป็นข้อความ
  - Stemming การตัด affix ออก เพื่อให้เหลือแต่รากศัพท์
  - Lemmatization การเปลี่ยนศัพท์ให้อยู่ในรูปแบบคำที่มีอยู่ในพจนานุกรม ทำงานนานกว่า stemming เพราะทำงานคล้ายกัน แต่ต้องเช็คว่ามีศัพท์นั้นมีอยู่หรือไม่

Stemmers

```
>>> raw = """DENNIS: Listen, strange women lying in ponds distributing swords
... is no basis for a system of government.  Supreme executive power derives from
... a mandate from the masses, not from some farcical aquatic ceremony."""
>>> tokens = word_tokenize(raw)

>>> porter = nltk.PorterStemmer()
>>> lancaster = nltk.LancasterStemmer()
>>> [porter.stem(t) for t in tokens]
['DENNI', ':', 'Listen', ',', 'strang', 'women', 'lie', 'in', 'pond',
'distribut', 'sword', 'is', 'no', 'basi', 'for', 'a', 'system', 'of', 'govern',
'.', 'Suprem', 'execut', 'power', 'deriv', 'from', 'a', 'mandat', 'from',
'the', 'mass', ',', 'not', 'from', 'some', 'farcic', 'aquat', 'ceremoni', '.']
>>> [lancaster.stem(t) for t in tokens]
['den', ':', 'list', ',', 'strange', 'wom', 'lying', 'in', 'pond', 'distribut',
'sword', 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'govern', '.', 'suprem',
'execut', 'pow', 'der', 'from', 'a', 'mand', 'from', 'the', 'mass', ',', 'not',
'from', 'som', 'farc', 'aqu', 'ceremony', '.']
```

Lemmatization

```
>>> wnl = nltk.WordNetLemmatizer()
>>> [wnl.lemmatize(t) for t in tokens]
['DENNIS', ':', 'Listen', ',', 'strange', 'woman', 'lying', 'in', 'pond',
'distributing', 'sword', 'is', 'no', 'basis', 'for', 'a', 'system', 'of',
'government', '.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a',
'mandate', 'from', 'the', 'mass', ',', 'not', 'from', 'some', 'farcical',
'aquatic', 'ceremony', '.']
```

Regular Expression Tokenizer

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```