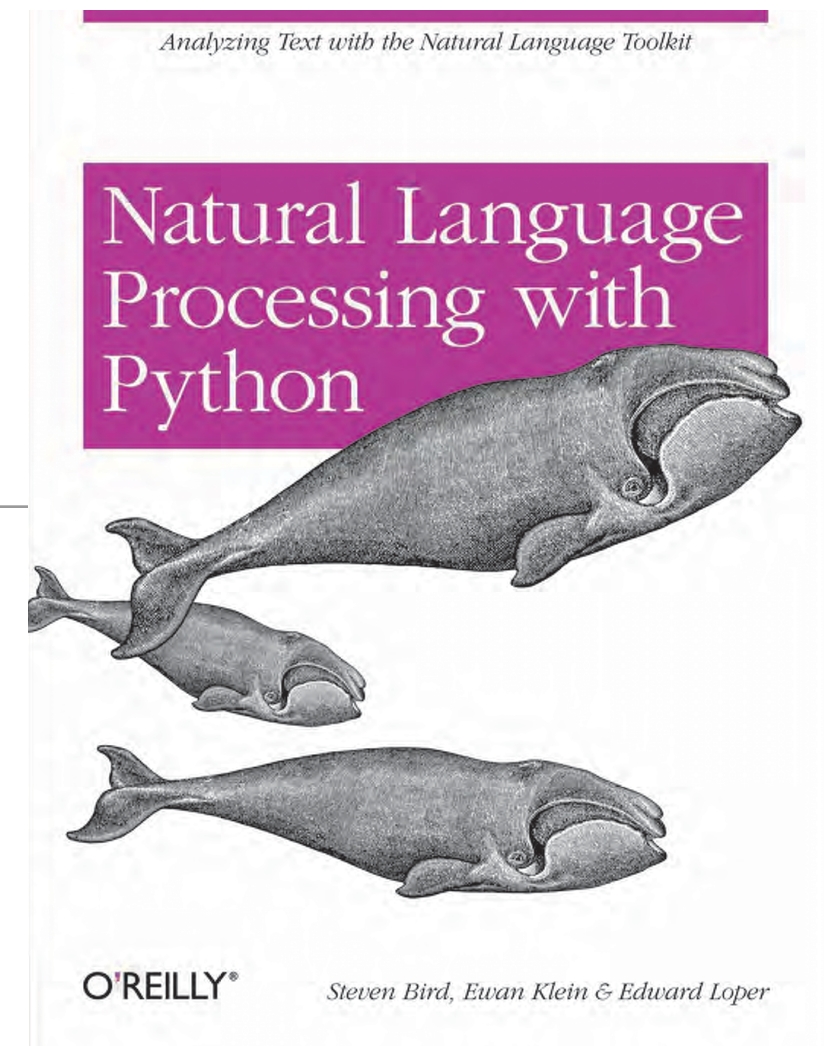


Language Processing and Python

Sutee Sudprasert

เนื้อหาจากบทที่ 1 ของหนังสือ

Natural Language Processing with Python



คำถามก่อนเรียน

- เราจะเขียนโปรแกรมอย่างไรในการจัดการกับเอกสารข้อความขนาดใหญ่?
- เราจะสกัดคำและวลีที่บ่งบอกถึงรูปแบบและเนื้อหาสำคัญของเอกสารโดยอัตโนมัติได้อย่างไร?
- เราจะใช้ภาษา Python เพื่อแก้ไขปัญหาก็กล่าวมาได้อย่างไร?
- 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
```

Searching Text

- มีวิธีการหลายวิธีในการตรวจหาส่วนที่เราต้องการจากเอกสารทั้งหมด ซึ่ง concordance เป็นวิธีหนึ่งที่ใช้ในการแสดงบริบทรอบข้างคำๆ หนึ่งที่เราสสนใจ

- 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

Searching Text

- ทดสอบสร้างข้อความแบบอัตโนมัติ

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

- ทดลองหลายๆ ครั้ง แล้วจะเห็นว่าโปรแกรมสร้างข้อความที่ไม่เหมือนกัน

Counting Vocabulary

- ฟังก์ชัน `len()` ใช้ในการนับ token ทั้งหมดในข้อความ

```
>>> len(text3)
```
- token หมายถึงกลุ่มของตัวอักษรที่ถูกแบ่งด้วย white-spaces
- ฟังก์ชัน `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

- เราสามารถเขียนฟังก์ชัน `lexical_diversity` และ `percentage` ขึ้นมาเพื่อที่จะได้ไม่ต้องพิมพ์สูตรการคำนวณซ้ำหลายๆ ครั้ง

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

- เมื่อประการฟังก์ชันแล้ว เราสามารถเรียกฟังก์ชันได้ดังนี้

```
>>> 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

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

- หากเราต้องการหาคำใดในเอกสาร ควรจะเป็นคำที่สื่อถึงหัวข้อและหมวดหมู่ของเอกสารนั้น เราอาจทดลองนับคำที่มีความถี่มากที่สุด 50 อันดับแรก
- ถ้าเรานับด้วยมือ เราอาจจะได้ตารางดังนี้

Word Tally

the	
been	
message	
persevere	
nation	

- ตารางนี้เรียกว่า frequency distribution ซึ่งบอกถึงการกระจายตัวของ tokens ไปบนคำศัพท์ทั้งหมด

Frequency Distributions

- NLTK มีเครื่องมือสำหรับการหา frequency distributions ที่ใช้ในการประมวลผลภาษา โดยเฉพาะซึ่งคือ FreqDist

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

- วาดกราฟเพื่อแสดงความถี่สะสมของคำที่พบมากที่สุด 50 อันดับแรก

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

Frequency Distributions

- จากผลลัพธ์ที่ได้จะเห็นว่า 50 คำที่มีความมากที่สุด ไม่สามารถใช้เป็นตัวแทนเอกสารได้ (มีคำว่า whale คำเดียวที่ใช้ได้)
- หากทดลองดูคำที่มีความถี่น้อยที่สุดแทน

```
>>> fdist1.hapaxes()
```
- ซึ่งจะเห็นว่าผลลัพธ์ที่ได้ก็ไม่สามารถใช้เป็นตัวแทนเอกสารได้เหมือนกัน

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

- [illegible]

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

Collocation and Bigrams

- Collocation คือการที่ลำดับของคำหนึ่งเกิดขึ้นพร้อมกันบ่อยๆ ในลักษณะที่มีความหมายพิเศษแตกต่างไปจากเดิม เช่น red wine หรือ white wine

- เราสามารถหา collocation อย่างง่าย โดยการเริ่มต้นจากการหา bigrams

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

- สิ่งที่ต้องทำต่อไปคือการหาคู่ของคำที่มีความถี่สูง ซึ่งขั้นตอนทั้งหมดสามารถทำได้โดยใช้ฟังก์ชัน collocations

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

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

Some word comparison operators

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

Automatic Natural Language Understanding

- หลังจากเราได้เริ่มเรียนรู้การประมวลข้อความในระดับพื้นฐาน ตอนนี้เราจะเข้าไปในระดับขั้นสุด 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
- บางที่เราคิดว่าไม่น่าจะเป็นเรื่องยากที่จะตอบคำถามนี้ แต่ในบางกรณีไม่เป็นเช่นนั้น
 - 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

- ในกรณีที่เรารู้สำหรับวิเคราะห์และเข้าใจภาษาได้แล้ว 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

- เมื่อไม่กี่ปีนี้มี “shared task” ที่เรียกว่า Recognizing Textual Entailment (RTE) ซึ่งเป็นการให้หาว่าหลักฐานที่ให้มาสนับสนุนสมมติฐานหรือเปล่า เช่น
 - สมมติฐาน: 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

- ในบางกรณีเราไม่สามารถสรุปโดยตรงได้เช่น
 - 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 ซึ่งเราก็คงต้องรอให้หลายๆ ปัญหาในด้าน NLP ถูกแก้ไขอย่างสมบูรณ์ก่อน
- เราต้องยอมรับถึงข้อจำกัดของโปรแกรมด้าน NLP ที่พอจะทำงานได้ดีเฉพาะในขอบเขตที่จำกัดเท่านั้น
- การเรียนวิชานี้ก็เพื่อสามารถนำความรู้ไปใช้สร้างระบบ NLP (ที่พอใช้งานได้) และ เพื่อพัฒนาความรู้ทางด้าน NLP ให้ก้าวหน้าขึ้นไปในอนาคต