Programming for NLP
PA 3: Concordancer

ก่อนเริ่ม

PA นี้จุดประสงค์เพื่อฝึกใช้ data structure ที่ซับซ้อนขึ้น ฝึกการเขียน class และใช้ NLP Tools พื้นฐานในการประมวลผลข้อมูลภาษา สิ่งที่จะทำให้ทุกคนงงที่สุดคือ Object-Oriented Programming ถ้ายังไม่ค่อยเข้าใจให้กลับไปดูวิดีโอ หรือสอบถามพี่ ๆ ทีเอ

ทุกคนกำลังจะเขียนโปรแกรมที่ใช้วิเคราะห์ข้อมูลภาษาเบื้องต้น โดยการแสดงสิ่งที่เรียกว่า concordance ซึ่งเป็นการไล่หาประโยคทั้งหมดในคลังข้อมูลที่มีคำที่ต้องการหาอยู่ จุดประสงค์เพื่อดูว่าคำๆ นั้นใช้ในบริบทใดบ้าง เพื่อที่จะได้เข้าใจความสัมพันธ์ทางความหมายมากขึ้น Concordance เป็นเทคนิคมาตรฐานของ Corpus linguistics ซึ่งนำข้อมูลทางภาษาจริง ๆ มาเป็นข้อมูลที่จะสนับสนุนสมมติฐานต่าง ๆ เกี่ยวกับภาษาศาสตร์ Concordance มีลักษณะดังตัวอย่างข้างล่างนี้

opportunity for reflection. # I found the article	**excellent**	, and I really appreciate the deep re
to have found two honest and otherwise	**excellent**	mechanics in my city (one who specialize
eir own knowledge. Yes, there are some	**excellent**	shadetree mechanics. There are also a lot
rk by Tsasuyeda, you will find nearly 50	**excellent**	videos from which you may learn a great d
ne bag). Darwin Barney made an	**excellent**	diving catch and kept his foot on the bag to force
on an infield single. Starlin Castro made an	**excellent**	stop on a ball ticketed for center but h
BUILDING # Peace Queen! This was an	**excellent**	Article! What I appreciate most is how you c
Cosmetic LOTRO, and Sig at the	**excellent**	new (ish) Warsteeds site to name a few. I won't go
soon. # This is an	**excellent**	post!!! I'm a Certified Breastfeeding Counselor and Labor Doula a
st 10 years has been the polar opposite:	**excellent**	service wherever I go, cheapest rates in m
eitenstein for no gain, setting the offense up with	**excellent**	field position. Five plays later, Sha

คำว่า excellent เรียกว่า query word คือคำที่เราต้องการสืบค้นการใช้ในบริบท ในที่นี่บริบทหมายถึง คำที่ปรากฏอยู่รอบ ๆ query word ทำให้เราทราบถึงวิธีการใช้คำ query word ตามที่ปรากฏอยู่ในคลังข้อมูลภาษา

starter code มีอยู่สองไฟล์

- concordancer.py -- ไฟล์ที่คลาสที่จะต้อง implement ห้ามเปลี่ยน function header เด็ดขาด (บรรทัดที่ขึ้นต้นด้วยคำว่า class หรือ def)
- jane.txt -- text file ที่สามารถเอามาลองใช้ทดสอบ

Part 1: Constructor และ `read_tokens(file_name)`

token คือ หน่วยภาษาหน่วยหนึ่งที่เราต้องการใช้วิเคราะห์ภาษา ที่เราเรียกว่า token แต่ไม่เรียกว่า คำ เพราะว่านิยามของคำนั้นไม่ค่อยแน่นอน ขึ้นอยู่กับเกณฑ์ในการตัดสินว่าคำคืออะไร token คือหน่วยเราพอใจใช้ในการวิเคราะห์ซึ่งเป็นอะไรก็ได้ แต่สำหรับกรณีของเราซึ่งเป็นภาษาอังกฤษ เราจะจัดว่า token คือคำที่มีช่องว่างหรือเครื่องหมาย punctuation รายรอบอยู่

Method read_tokens จะอ่านไฟล์ที่จัดเรียงมาแบบหนึ่งประโยคต่อหนึ่งบรรทัด นำไป tokenize (ตัดให้เป็น token ๆ) ทีละบรรทัด จากนั้นนำมาเก็บใส่ list of ประโยค ซึ่งแต่ละประโยคจะเป็น list of token strings เพื่อจัดเตรียมไว้รอถูกเรียกใช้หา concordance ต่อไปโดยไม่ต้องมาอ่านข้อมูลจากไฟล์อีก

- เริ่มด้วยการสร้าง object attribute ใน constructor (__init__) เตรียมไว้สำหรับการเก็บข้อมูลประโยค
- ส่วนใน read_tokens ให้เปิดไฟล์อ่านทีละบรรทัด และ tokenize โดยใช้ library ที่ชื่อว่า nltk (ต้อง install ให้เรียบร้อยก่อน) และใช้ function nltk.wordpunct_tokenize

Part 2: `find_concordance`

Method นี้จะใช้เมื่อเราโหลดข้อมูลเข้าไปใน object แล้ว

```
cc = Concordancer()
cc.read_tokens('jane.txt')
cc.find_concordance("good", 7)
```

Print concordance ออกมา (จริงๆ ต้องมีอีก ให้ดูแค่บรรทัดแรกๆ)

```
observation , that I was endeavouring in good earnest to acquire a more sociable and
            , when she chanced to be in good humour ; and when , having brought
          What we tell you is for your good , " added Bessie , in no
                                  The good apothecary appeared a little puzzled .
          he had a hard - featured yet good - natured looking face .
           himself ; " nerves not in a good state ."
           think , have been a girl of good natural capacity , for she was smart
       eyes , very nice features , and good , clear complexion ; but she had
             Jane Eyre , and are you a good child ?"
```

```
objectionable : " I must keep in good health , and not die ."
```

method find_concordance มี parameter ดังนี้

- รับ query มาซึ่งเป็น string ที่มีคำที่เราต้องการหา
- num_words เป็น parameter ที่กำหนดว่าอยากได้ context ทางด้านข้างกี่คำ ข้างละ num_words เท่าๆ กัน

สิ่งที่ method นี้แสดงผลออกมาจะต้องเป็นไปตาม requirement ดังนี้

- context word (คำที่ปรากฏรอบ ๆ query word) ที่ print ออกมาจะต้องถูกคั่นด้วยช่องว่าง เช่น

✅	`eyes , very nice features , and good , clear complexion ; but she had`
❌	`eyes, very nice features, and good, clear complexion ; but she had`

- query word จะต้องอยู่ตรงกลางเสมอ ความกว้างของ context ด้านซ้าย และด้านขวาของ query จะต้องเท่ากับ left_context_length และ right_context_length ตามลำดับ **ถ้าหาก context มีขนาดเกินต้องตัดออก และมีขนาดสั้นเกินต้องเติมช่องว่างเข้าไป** (Hint: 'a' * 5 จะรีเทิร์น 'aaaaa') สมมติว่าถ้าตั้งให้ left_context_length = 10 และ right_context_length = 10

✅	`ures , and good , clear co` `s not in a good state ."`
❌	`eyes , very nice features , and good , clear complexion ; but she had` `himself ; " nerves not in a good state ."`

- query word อาจจะปรากฏหลายๆ ครั้งในหนึ่งประโยค ให้ print ผลออกมาทุกครั้งที่เจอ query word ในประโยคนั้น
- ถ้าไม่เจอคำ query นั้นให้ print ว่า "Query not found..."
- Method นี้ไม่ต้อง return อะไร print อย่างเดียว

Part 3: `compute_bigram_stats`

การวิเคราะห์ในระดับคำบางทีไม่เพียงพอต่อการศึกษาบริบทที่คำนั้นถูกใช้ในประโยคจริง เราจึงใช้วิธีดูสองคำที่อยู่ติดกัน ซึ่งเรียกว่า bigram เช่น ถ้าเราอยากทราบว่าคำว่า good ถูกตามหลังด้วยคำว่าอะไรบ้าง

```
cc = Concordancer()
cc.read_tokens('jane.txt')
cc.compute_bigram_stats('good', 'good_bigram.txt')
```

รันเสร็จแล้วจะได้ไฟล์ good_bigram.txt

```
good and 9
good deal 9
good man 8
good as 8
good to 7
good in 6
good lady 5
good or 4
good fire 4
. . .
```

Method นี้มี requirement ดังนี้

- รับ parameter สองตัว query word และ ชื่อไฟล์ output
- ไฟล์ที่ได้ออกมาจะต้องเรียงลำดับตามความถี่ที่เจอ bigram นั้น
    - ระหว่างไบแกรมและจำนวนครั้งที่เจอให้คั่นด้วยช่องว่างหนึ่งช่อง
    - ต้องหาให้ครบทุกไบแกรม
    - จะต้องไม่มีไบแกรมที่มี punctuation อยู่
    - ถ้าไม่เจอ bigram เลยให้เขียน file เปล่าๆ ออกมา

Part 4 (extra credit): `find_concordance_ngram`

ข้อนี้ไม่ต้องทำก็ได้แต่ใครทำจะได้คะแนนพิเศษถ้าผ่าน test case ของ Part 1-3 ทั้งหมด

Method นี้หา concordance ของ ngram query แทนที่จะเป็นแค่คำโดด ๆ เพราะฉะนั้นเราจะต้อง tokenize ngram ที่ได้มาก่อนแล้วค่อยมาไล่หา concordance ตัวอย่าง

```
cc = Concordancer()
cc.read_tokens('jane.txt')
```

```
cc.find_concordance_ngram("good lady", 7)
```

Print concordance ออกมา

```
am a little deaf , " returned good lady , approaching her ear to my mo
       " I wish , " continued good lady , " you would ask her a
either in persons or things : good lady evidently belonged to this cla
               " Yes , " said good lady , who now knew what ground we
  must want your tea , " said good lady , as I joined her ; "
```

Requirement ต่างๆ เหมือนกับ find_concordance ปกติ

การส่งงาน

ส่ง concordancer.py ไฟล์เดียวทาง MyCourseVille **โดยห้ามเปลี่ยนชื่อไฟล์และ zip โดยเด็ดขาด!** หากพบจะถูกหักคะแนน

เครื่องจะตรวจตาม output เป๊ะ ๆ เพราะฉะนั้น อ่านโจทย์และดูตัวอย่างให้ละเอียด (ลองเอาเมาส์คลุมที่ตัวอย่างได้)

FAQs

Q: find_concordancer() ใส่ query แล้วเอาตัวที่มีตัวพิมพ์ใหญ่ไหม เช่น *good* เอา *Good* morning ไหม

A: ไม่ต้อง ทุกข้อเป็นแบบ case sensitive

Q: message output ของ find_concordancer() คงความตัวพิมพ์เล็ก-ใหญ่ไว้ไหม เช่น I-i

A: ให้คงตัวพิมพ์เล็ก-ใหญ่เอาไว้เหมือนเดิม

Q: ตัวอย่าง find_concordancer_ngram() มีการตัดคำก่อนหน้า query ออก ต้องตัดออกด้วยไหม

A: ไม่ต้องตัดออก ให้คง text ไว้เหมือนตอนทำ find_concordancer() ตามปกติ

วิธี install nltk for tokenization

🔷 1. ติดตั้ง Library

ก่อนอื่นต้องติดตั้ง NLTK ก่อน (หากยังไม่ได้ติดตั้ง):

```bash
pip install nltk
```

🔷 2. นำเข้าและใช้งานฟังก์ชัน `wordpunct_tokenize`

ฟังก์ชันนี้จะตัดคำโดยอิงจาก **ช่องว่าง (space)** และ **เครื่องหมายวรรคตอน (punctuation)**
เช่นจะแยก `"Hello, world!"` ออกเป็น `['Hello', ',', 'world', '!']`

```python
import nltk
from nltk.tokenize import wordpunct_tokenize

# ตัวอย่างข้อความ
text = "Hello, world! I'm learning NLP using NLTK."

# ทำการ tokenize
tokens = wordpunct_tokenize(text)

print(tokens)
```

ผลลัพธ์ที่ได้:

```python
['Hello', ',', 'world', '!', 'I', "'", 'm', 'learning', 'NLP', 'using', 'NLTK', '.']
```