คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

ข้อสอบกลางภาค ภาคการศึกษาปลาย ปีการศึกษา 2558

ข้อสอบวิชา 517 432 การประมวลผลภาษาธรรมชาติ

517 661 การประมวลผลภาษาธรรมชาติ

สอบวันอาทิตย์ที่ 6 มีนาคม 2559 เวลา 13.00-16.00 น. ห้อง 1638 ว.1

คำสั่ง 1. ข้อสอบมีทั้งหมด 3 หน้า 12 ข้อ 70 คะแนน (35 %)

2. ทำข้อสอบทุกข้อในกระดาษคำตอบ ถ้าที่ว่างในกระดาษคำตอบไม่พอ สามารถขอกระดาษเปล่าเพิ่มได้

3. ส่งเฉพาะกระดาษคำตอบ สามารถนำข้อสอบกลับไปทำต่อได้

# 4. ข้อที่มีเครื่องหมาย \* ต่อท้ายเลขข้อ คือ ข้อที่สามารถนำไปเขียนโปรแกรมส่งมาทางอีเมล์เพื่อเพิ่ม คะแนนได้ ถ้าพบว่าลอกกันมาจะได้ 0 คะแนน

# 4.1 ส่งอีเมล์มาที่ [soonklang\_t@silpakorn.edu](mailto:soonklang_t@silpakorn.edu)

# 4.2 กำหนด subject ชื่อ [517432Mid] รหัสนักศึกษา

# 4.3 ตั้งชื่อไฟล์ รหัสนักศึกษา\_mid.py โดยรวมทุกข้อไว้ในไฟล์เดียวกัน แต่ให้เขียน comment

# ระบุว่าเป็นข้อใดไว้บรรทัดก่อนหน้าโค้ดนั้น เช่น #exam1a

4.4 กำหนดส่งภายในเที่ยงคืนวันที่ 6 มีนาคม 2559

# 5. ข้อความ (text) ที่กล่าวถึงในข้อสอบนี้หมายถึง ข้อความในภาษาอังกฤษ

# 6. อนุญาตให้นำเอกสารเข้าห้องสอบได้

1. อธิบายความหมายของคำต่อไปนี้โดยสังเขป และยกตัวอย่างประกอบ ไม่มี ต.ย.ไม่ได้คะแนน (14 คะแนน)

1. Corpus

คลังคำที่เก็บข้อมูลคำหรือประโยคสำหรับใช้ประมวลผลภาษาธรรมชาติ   
เช่น brown, gutenberg เป็นหนึ่งใน corpus ที่ให้ติดมากับแพ็คเกจ nltk.corpus

1. Token

ใช้เรียกชุดของข้อความ เพราะบางครั้งการทำ tokenization อาจตัดมาเป็นชุดของข้อความไม่ได้ออกเป็นคำเสมอไป

ตัวอย่างเช่น “Hello”, “ice-cream”

1. Concordance

แสดง token ที่เหมือนกันในชุดของคลังคำ เพื่อให้หาความหมายได้ง่ายขึ้น

ตัวอย่างเช่น concordance ของคำว่า monstrous ใน text1

จะได้ one was of a most **monstrous** size . และ Touching that **monstrous** bulk of the whale เป็นต้น

1. Collocation

คือ คำประสม เป็นคำที่สร้างจากคำตั้งแต่สองคำขึ้นไป เกิดเป็นคำใหม่

ตัวอย่างเช่น

แม่น้ำ เกิดจาก คำว่าแม่ และ คำว่าน้ำ

พ่อบ้าน เกิดจาก คำว่าพ่อ และ คำว่าบ้าน

เป็นต้น

1. Lexeme

คือ คำที่เป็นตัวแทนของคำนั้นในพจนานุกรม

ตัวอย่างเช่น **กิน** (น.) การนำอาหารเข้าทางปาก

จะเรียกส่วนที่เน้นสีดำเข้มว่า Lexeme

1. Phoneme

คือ ตัวอักษรวิธีการออกเสียง ทำให้ผู้ที่รู้ Phoneme สามารถออกเสียงได้ถูกต้องแม้ไม่รู้ภาษานั้นก็ตาม

ตัวอย่างเช่น **/ˈfoʊniːm/** คือ Phoneme ของคำว่า Phoneme

1. Stopword

คำที่มีความถี่สูงพบได้บ่อย แต่เมื่อตัดออกไม่มีผลกับใจความสำคัญ

ตัวอย่างเช่น นี้,มี,ยัง

1. Part-of-speech (POS)

ใช้บอกหน้าที่ของคำในประโยชน์

ตัวอย่างเช่น นาม (น.) , กริยา (ก.) ,วิเศษณ์ (ว.)

1. Hyponyms

Y เป็น Hyponyms ของ X คือ Y เป็นประเภทย่อยของ X ใน wordnet

ตัวอย่างเช่น cat เป็น hyponym ของ feline

1. Hypernyms

Y เป็น Hypernyms ของ X คือ X เป็นประเภทย่อยของ Y ใน wordnet

ตัวอย่างเช่น feline เป็น hypernyms ของ cat

1. Tokenization

การตัดแบ่งข้อความยาวๆ ออกเป็น token

ตัวอย่างเช่น “ฉันกินข้าว” เมื่อทำ tokenization จะได้ [“ฉัน”,”กิน”,”ข้าว”]

1. Stemming

การหารากศัพท์ของคำ โดยการตัว Prefix และ suffix ออก

ตัวอย่างเช่น คำว่า preprocessing เมื่อทำการ stemming จะได้คำว่า process

1. Lemmatization

การทำให้ข้อความกลับเป็นรูปแบบดั้งเดิมของขำ

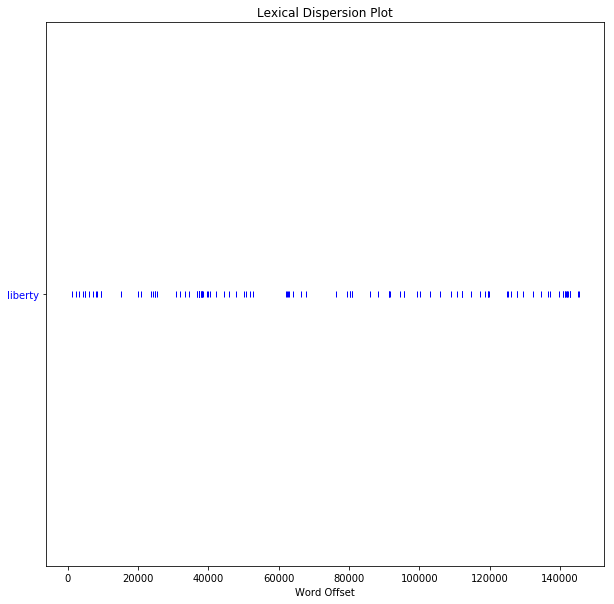
ตัวอย่างเช่น Better เมื่อทำ lemmatization จะได้คำว่า good

1. Bigram

พิจารณา token 2 ตัวที่อยู่ติดกัน โดยเมื่อมี n โทเคนจะสร้าง bigram ได้ n-1 ตัว

ตัวอย่างเช่น [“ฉัน”,”กิน”,”ข้าว”] จะสร้าง bigram ได้เป็น [(“ฉัน”,”กิน”),(“กิน”,”ข้าว”)]

2.\* สร้างกราฟการกระจายตัวของคำว่า liberty และ constitution ใน text4 จาก corpus book แล้ววิเคราะห์ผลที่ได้ว่ามีลักษณะเช่นไร (3 คะแนน)



พบกว่าคลังคำศัพท์นี้มีคำว่า liberty มากในช่วง 60,000 และช่วง 140,000

3. \*เขียนโปรแกรมเพื่อคำตอบต่อไปนี้ใน brown corpus เลือกมา 6 categories

1. คำที่มีความยาวเท่าใดมีความถี่สูงสุด เหมือนกันในทุก categories หรือไม่ (3 คะแนน)

ตอบ คำที่มีความถี่สูงสุด เหมือนกันใน 6 categories

1. คำที่มีความยาวมากที่สุดมีความยาวเท่าใด (3 คะแนน)

|  |  |  |
| --- | --- | --- |
| Categories | a) ความยาวของคำที่พบมากสุด | b) ความยาวของคำที่ยาวที่สุด |
| adventure | 3 | 17 |
| belles\_lettres | 3 | 19 |
| editorial | 3 | 19 |
| fiction | 3 | 16 |
| government | 3 | 18 |
| hobbies | 3 | 18 |

โค้ด

from nltk.corpus import brown

from nltk import FreqDist,ConditionalFreqDist

most\_common\_list = []

longest\_list = []

categories = brown.categories()[0:6]

for catogory in categories:

words\_len = [len(w) for w in brown.words(categories=catogory) if w.isalpha()]

freq = FreqDist(words\_len)

longest\_list.append(max(freq.keys()))

most\_common\_list.append(freq.most\_common(1)[0][0])

if(len(set(most\_common\_list)) == 1):

print("คำที่มีความถี่สูงสุด เหมือนกันในทุก categories ")

else:

print("คำที่มีความถี่สูงสุด ไม่เหมือนกัน สำหรับแต่ละ categories ")

print("Categories | a) ความยาวของคำที่พบมากสุด | b) ความยาวของคำที่ยาวที่สุด")

for i in range(len(categories)):

print(categories[i]," | ",most\_common\_list[i]," | ",longest\_list[i])

4.\* สร้างฟังก์ชันชื่อ longest\_vocab(text) ที่รับพารามิเตอร์เป็น list ของคำ และทำ return ค่าเป็นคำศัพท์

ทั้งหมดที่มีความยาวมากที่สุดใน list และทดลองเรียกใช้กับ corpus ใดก็ได้ (4 คะแนน)

ตอบ

def longest\_vocab(text):

max\_length = max([len(w) for w in text])

return [w for w in text if len(w) == max\_length]

ตอบ

from nltk.corpus import brown

print(longest\_vocab(brown.words(categories='adventure')))

5.\* ปรับปรุงฟังก์ชัน unusual\_word(text) ให้เป็นฟังก์ชัน unknown( ) โดยรับ URL เป็นพารามิเตอร์ เพื่อ

return list ของคำศัพท์ที่ไม่รู้จัก (unknown word) คือ คำที่ไม่พบในพจนานุกรม เช่น ชื่อเฉพาะ คำที่ไม่

รู้จัก คำศัพท์ใหม่ๆ ในการหาคำศัพท์ที่ไม่รู้จัก โดยใช้ lexical resources ที่ nltk มีให้ เช่น wordnet หรือ

ใช้เทคนิคอื่นๆ ที่เรียนมาทำได้ ใน main program ทำการเรียกใช้ฟังก์ชันและส่งพารามิเตอร์เป็น

[news.bbc.co.uk/2/hi/health/2284783.stm](http://news.bbc.co.uk/2/hi/health/2284783.stm) (10 คะแนน)

คำแนะนำ: ควรลบ tag html ออกจากนั้นสร้าง list ของคำที่ขึ้นต้นด้วยตัวอักษร แล้วทำการลบคำใดๆ ใน

list ที่ปรากฏอยู่ใน Words Corpus (nltk.corpus.words) และ WordNet lemma ใช้ stemmer หรือ

lemmatizer เข้ามาช่วย

ตอบ

from nltk import word\_tokenize

from urllib import request

from bs4 import BeautifulSoup

def unknown(url):

response = request.urlopen(url)

html = response.read().decode('utf8')

text = BeautifulSoup(html,'html.parser').get\_text()

tokens = word\_tokenize(text)

text\_vocab = set(w.lower() for w in tokens if w.isalpha())

english\_vocab = set(w.lower() for w in nltk.corpus.words.words())

unusual = text\_vocab - english\_vocab

return sorted(unusual)

6. อธิบายว่า string ลักษณะใดที่จะตรงกับ regular expression ในข้อต่อไปนี้ พร้อมยกตัวอย่าง string

ประกอบ ข้อละ 1 ตัวอย่าง (6 คะแนน ข้อละ 1 คะแนน)

1. [a-zA-Z]+

ตัวอักษรที่เป็นภาษาอังกฤษทั้งตัวพิมพ์ใหญ่หรือพิมพ์เล็ก ตัวอย่างเช่น HelloWorld

1. [A-Z][a-z]\*

ตัวอักษรภาษาอังกฤษที่ตัวหน้าเป็นตัวพิมพ์ใหญ่และหากมีตัวต่อมาเป็นตัวพิมพ์เล็กต้องเป็นพิมพ์เล็กทั้งหมด ตัวอย่างเช่น Hello

1. p[aeiou]{,2}t

คำที่มีตัว p ตามด้วยสระ 0 ถึง 2 ตัว ตามด้วยตัว t ตัวอย่างเช่น put

1. \d+(\.\d+)?

ตัวเลขที่อาจมีทศนิยามตามท้าย ตัวอย่างเช่น 0.999

1. ([^aeiou][aeiou][^aeiou])\*

คำที่ตัวแรกเป็นพยัญชนะ ตัวที่สองเป็นสระ และตัวสุดท้ายเป็นพยัญชนะ เช่น bed

1. \w+|[^\w\s]+

คำนั้นเป็นข้อความและตัวเลข หรือเป็นอักขระพิเศษเท่านั้น เช่น !@#$%^&\*()

6. เขียน regular expression เพื่อใช้ในการค้นหา string ที่มีลักษณะต่อไปนี้ (6 คะแนน)

1. คำนำหน้าคำนาม ซึ่งประกอบด้วยคำว่า a, an, the (3 คะแนน)

\s?(a|an|the)\s\w+

1. นิพจน์ทางคณิตศาสตร์ที่มีการใช้เลขจำนวนเต็ม การบวก และการคูณ เช่น 2\*3+8 (3 คะแนน)

\d+(\+|\\*)\d+(\+|\\*)?

7.\* เปลี่ยนคำสั่งบรรทัดที่ 1-4 เปลี่ยนเป็น list comprehension (3 คะแนน)

1. result = [ ]
2. for word in sent:
3. word\_len = (word, len(word))
4. result.append(word\_len)
5. print result
6. # output is [('The', 3), ('dog', 3), ('gave', 4), ('John', 4), ('the', 3), ('newspaper', 9)]

ตอบ result = [(word,len(word)) for word in sent]

9.\* จงเขียนคำสั่งเพื่อรับ list ของคำ (เช่น ตัวแปร words ในตัวอย่าง) แล้วสร้างเป็น list ของสระทั้งหมดที่

ปรากฏอยู่ในคำเหล่านั้น โดยเรียงตามลำดับที่ปรากฏ (เช่น ตัวแปร vowels) (5 คะแนน)

input:

words = ['attribution', 'confabulation', 'elocution', 'sequoia', 'tenacious', 'unidirectional']

output:

vowels= ['aiuio', 'eaiou', 'eouio', 'euoia', 'oauaio', 'uiieioa']

จากตัวอย่าง คำแรกคือ **a**ttr**i**b**u**t**io**n ประกอบด้วยสระคือ aiuio จึงนำมาสร้างเป็นสมาชิกตัวแรกใน vowels

ตอบ

words = ['attribution', 'confabulation', 'elocution', 'sequoia', 'tenacious', 'unidirectional']

vowels = ["".join(re.findall('[aeiou]',word)) for word in words]

10. อธิบายปัญหาทาง NLP ที่มีชื่อว่า Recognizing Textual Entailment ว่ามีลักษณะอย่างไร พร้อมทั้ง

เขียนตัวอย่าง input และ output ของปัญหาประกอบการอธิบายด้วย (4 คะแนน)

ตอบ

Recognizing Textual Entailment คือการหาประโยคที่มีใจความสัมพันธ์กับประโยคก่อนหน้า

ตัวอย่างเช่น

Input: ถ้าให้ทานแล้วจะได้บุญ

Output: ต้องบริจาคเงินให้ผู้ยากไร้

11.\* เขียนฟังก์ชัน bigram(words\_list) เพื่อหาว่า bigram ที่มีความถี่มากที่สุด 5 อันดับแรกในคำที่ส่งเข้า

มาเป็นพารามิเตอร์ โดยไม่นับ bigram ที่มี stopwords หรือสัญลักษณ์ต่างๆ ที่ไม่ใช่คำศัพท์ จากนั้นให้

ทดลองเรียกใช้ฟังก์ชันโดยส่งคำในไฟล์ 1797-Adams.txt ใน inaugural corpus ไปเป็นพารามิเตอร์

Bigram ที่มีความถี่สูงสุด 5 อันดับแรกในไฟล์ 1797-Adams.txt ได้แก่ (6 คะแนน)

1. ('foreign', 'nations') ปรากฏ 5 ครั้ง
2. ('people', 'America') ปรากฏ 3 ครั้ง
3. ('Constitution', 'United')
4. ('United', 'States')
5. ('fellow', 'citizens')

โค้ด

from nltk.corpus import stopwords, inaugural

from nltk import FreqDist,bigrams

def bigram(wordlist):

stopword = stopwords.words('english')

words = [w for w in wordlist if w.isalpha() and w not in stopword]

word\_bigram = bigrams(words)

freq = FreqDist(word\_bigram)

return freq.most\_common(5)

print(bigram(inaugural.words(fileids = '1797-Adams.txt')))

12.\* เลือกทำข้อใดข้อหนึ่งต่อไปนี้ เพียงข้อเดียว โดยใช้ข้อมูลจาก brown corpus หมวด news (7 คะแนน)

1. เขียนโปรแกรมเพื่อหาว่าคำใดเป็นที่คำที่กำกวมมากสำหรับการ ติด POS tag โดยพิมพ์คำที่พบว่าติด tag มากกว่า 5 ประเภท ไม่ต้องใส่ option tagset
2. เขียนโปรแกรมเพื่อหาว่า tag ใดที่ปรากฏอยู่หน้าคำ verb มากที่สุด 3 ลำดับแรก ใช้ option tagset เป็น Universal
3. เขียนโปรแกรมเพื่อหาคำ 3 คำที่มีรูปแบบคำและการ tag เป็น ‘in’ + DET +NOUN ใช้ option tagset เป็น Universal เช่น คำว่า in the lab (เขียนคำตอบแค่ 3 ตัวอย่าง)

เลือกทำข้อ 2

ตอบ คำ verb มากที่สุด 3 ลำดับแรกได้แก่ NOUN ปรากฏ 4180 ครั้ง, VERB ปรากฏ 2858 ครั้ง และ PRON ปรากฏ 1863 ครั้ง

โค้ด

from nltk.corpus import brown

from nltk import pos\_tag,bigrams,FreqDist

wordlist = brown.words(categories='news')

tags = pos\_tag(wordlist,tagset='universal')

bigram\_list = list(bigrams([tag for word,tag in tags]))

freq = FreqDist([b[0] for b in bigram\_list if b[1] == 'VERB'])

print(freq.most\_common(3))