Introduction to Computational Linguistics 2025  
PA1 - Logistic Regression Model

Due 7 Feb 2025

# คำชี้แจง

* ทำเดี่ยว คุยกันได้ ปรึกษาทีเอได้ แต่ต้องเขียนโค้ดด้วยตัวเองทั้งหมด
* เสร็จแล้วส่ง logistic\_regression.py บน mycourseville (ไฟล์เดียวเท่านั้น)

# Starter code

เราจะ implement class TextClassifier ซึ่งจะโหลด parameter matrix สำหรับ bag-of-word model features กล่าวคือทุกคำที่พบใน training set ถูกนำมาใช้เป็น features ในโมเดลนี้เราจะไม่ใช้ ค่า bias หรือ intercept ในการคำนวณเพื่อความง่าย

constructor จะโหลด parameter matrix ที่ถูกเก็บอยู่รูปของ csv ที่แต่ละแถวแสดงคำ และค่าน้ำหนักของคำนั้นและ label ต่างๆ เราต้อง assume ว่าคอลัมน์ซ้ายสุดจะเป็นคำและมีชือว่า word ส่วนคอลัมน์ที่เหลือเป็นค่าน้ำหนัก และชื่อ column คือชื่อ label ซึ่งขึ้นอยู่กับ csv ที่เราได้มา (จะไม่เป็น A B C เหมือนในตัวอย่างเสมอไป และจะไม่ได้มีแค่ 3 label เสมอไป) เช่น

| word | A | B | C |
| --- | --- | --- | --- |
| เงิน | 0.2 | -0.5 | 0.3 |
| เศรษฐกิจ | 0.4 | 0.3 | 0.2 |
| ฝุ่น | -0.2 | -0.5 | 0.1 |
| เกลียด | 0.1 | 0.1 | 0.1 |
| ม็อบ | -0.3 | 0.6 | -0.2 |

จากตาราง parameter ข้างต้น weight ค่าน้ำหนัก ของคำว่า ฝุ่น สำหรับ label A, B, C คือ -0.2, -0.5, 0.1 ตามลำดับ

ในโค้ดที่เตรียมมาจะสั่งให้ constructor จะทำการอ่าน csv file และเก็บไว้ใน attribute ที่ชื่อว่า self.model\_parameters โดยที่ตั้งค่าให้ชื่อเรียกแต่ละแถว (column index) เป็นคำที่อยู่ในคอลัมน์ word ซึ่งทำให้เราเลือกแถวได้อย่างง่าย ๆ แนะนำให้ลองทดสอบโค้ดโดยการเรียกดู model parameter ด้วยคำสั่ง self.model\_parameters.loc['เงิน']

# ฟังก์ชันที่ต้อง Implement

## 1. get\_all\_possible\_features

return list of strings ของ feature ทั้งหมดที่โมเดลนี้มี เช่น  
model.get\_all\_possible\_features() → ['เงิน', 'เศรษฐกิจ', 'ฝุ่น', 'เกลียด', 'ม็อบ']

Hint: แต่ละแถวมีชื่อ Index เป็นคำที่อยู่ใน self.model\_parameters.index

## 2. get\_all\_possible\_labels

return list of strings ของ label ที่โมเดลนี้มี เช่น  
model.get\_all\_possible\_labels() → ['A', 'B', 'C']

## 3. compute\_probability

ให้ตัด string ภาษาไทยให้เป็นคำโดยใช้ pythainlp.word\_tokenize โดยไม่ต้องเปลี่ยนค่า engine เป็นอย่างอื่น จากนั้นให้ใช้ parameter matrix ในการ classify เป็นหนึ่งใน label string ที่แสดงอยู่ใน csv ตอนแรก จากนั้นให้คำนวณและคืนค่าความน่าจะเป็นของแต่ละ label ในรูปแบบของดิกชันนารีที่คีย์เป็นชื่อ label และแวลูเป็นค่าความน่าจะเป็นของ label นั้น ตัวอย่างเช่น

ฉันเกลียดฝุ่น → ฉัน|เกลียด|ฝุ่น (tokenization)

คะแนนของ Label A = 0.1 - 0.2 = -0.1

คะแนนของ Label B = 0.1 - 0.5 = -0.4

คะแนนของ Label C = 0.1 + 0.1 = 0.2

ข้อสังเกตุอย่างหนึ่ง คือ เราไม่ได้ใช้คำว่า​ *ฉัน* มาคำนวณด้วยเพราะว่าไม่ได้อยู่ใน parameter matrix ของ จึงไม่ได้จัดเป็น feature

ความน่าจะเป็นของ Label A = exp(-0.1) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.32  
 ความน่าจะเป็นของ Label B = exp(-0.4) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.23  
 ความน่าจะเป็นของ Label C = exp(0.2) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.43

ค่าจะได้คืนมาจากฟังก์ชันนี้คือ

{'A': 0.32, 'B': 0.23, 'C': 0.43}

Hint 1: เปิดหาค่าที่ต้องใช้ในการคำนวณจาก parameter matrix ที่เราโหลดเก็บไว้ใน model\_parameters โดยตรง ไม่มีความจำเป็นต้องสร้าง feature vector ขึ้นมา เนื่องจากค่าส่วนใหญ่จะออกมาเป็น 0   
 Hint 2: เราอาจจะใช้ pseudocode ในการคำนวณคะแนนของ Label

for label in labels:  
 for feature in token\_list:  
 d.loc[feature][label] # weight ของ feature, label คู่นี้  
 update scores

## 4. classify

ใช้โมเดลในการ predict/infer ว่า label ที่ถูกต้องคืออะไร ตัวอย่าง เช่น ตามค่าความน่าจะเป็นที่เราคำนวณข้างบน เราต้อง return 'C' เพราะว่า C ได้ความน่าจะเป็นสูงสุด