

BoW (Bag-of-words)

ในการวิเคราะห์เอกสาร นอกจากการนับจำนวนตัวอักษร จำนวนคำ และจำนวนบรรทัด ของข้อความในเอกสารแล้ว เรายังสามารถหาลักษณะเฉพาะ (feature) ของเอกสารจากสถิติบางอย่างของข้อความในเอกสารได้ ลักษณะเฉพาะหนึ่งที่ใช้กันแพร่หลาย คือ Bag-of-words ซึ่งใช้วิเคราะห์หรือจำแนกประเภทเอกสารได้ เช่น นำไปใช้ตรวจจับ spam mails (ข้อความใน spam mails มักปรากฏคำบางคำ) หรือวิเคราะห์อารมณ์ (sentiment analysis) จากข้อความวิจารณ์ภาพยนตร์ ข้อความวิจารณ์หนังสือ ว่าเป็นเชิงบวก หรือเชิงลบ เป็นต้น (https://en.wikipedia.org/wiki/Bag-of-words_model)

Bag-of-words (BoW) คือ รายการของคำ (word) และความถี่ (word frequency หรือ word counts) ที่พบในเอกสาร โดยตัดคำที่ไม่น่าสนใจ (stop words) และปรับทุกตัวอักษรเป็นตัวพิมพ์เล็ก เช่น

ข้อความ: **Shane likes football; he is a big fan of Arsenal football team.**

แปลงเป็นตัวพิมพ์เล็ก ตัดอักขระที่ไม่ใช่ตัวอักษรอังกฤษและตัวเลข และตัด stop words ออก จะได้

shane likes football big fan arsenal football team

ได้ BoW = [['shane', 1], ['likes', 1], ['football', 2],
['big', 1], ['fan', 1], ['arsenal', 1], ['team', 1]]

หมายเหตุ: ลำดับของคำใน BoW ไม่สำคัญ จะเรียงอย่างไรก็ได้

ในกรณีที่เอกสารมีขนาดใหญ่ อาจได้ BoW ที่มีขนาดใหญ่ตาม จึงมีผู้เสนอกลวิธีหนึ่งในการลดขนาด BoW ที่เรียกว่า feature hashing ซึ่งแปลงคำต่าง ๆ ในเอกสารเป็นเลขจำนวนเต็มแทน วิธีแปลงมีหลายแบบ แต่ในการบ้านนี้ จะใช้ฟังก์ชัน $fhash(w, M)$ ข้างล่างนี้ เพื่อแปลงคำ w ให้เป็นจำนวนเต็มในช่วง 0 ถึง $M - 1$ (ระบบไม่มีฟังก์ชัน $fhash$ ให้ใช้ ต้องเขียนเอง)

ให้ w คือคำที่ประกอบด้วยอักขระ $c_0 c_1 c_2 \dots c_{n-1}$

$$fhash(w, M) = fhash(c_0 c_1 c_2 \dots c_{n-1}, M) = (\text{ord}(c_0) + \text{ord}(c_1)G^1 + \text{ord}(c_2)G^2 + \dots + \text{ord}(c_{n-1})G^{n-1}) \% M$$

- โดยให้ $G = 37$ และ M เป็นค่าที่ผู้ใช้กำหนด
- Python มีฟังก์ชัน `ord` ให้ใช้แล้ว ฟังก์ชัน `ord(c)` คืนจำนวนเต็มที่แทนค่าของอักขระในตัวแปร c
เช่น `ord('b')` ได้ 98, `ord('i')` ได้ 105, `ord('g')` ได้ 103
- ตัวอย่าง: $fhash('big', 4)$ มีค่าเป็น $(98 + 105 \times 37 + 103 \times 37^2) \% 4 = 2$
- ผลจากฟังก์ชัน $fhash(w, M)$ เป็นจำนวนเต็มที่ใช้แทนคำ w ผลที่ได้นี้มีค่าตั้งแต่ 0 ถึง $M - 1$
ค่า M จึงเป็นตัวกำหนดจำนวนค่ามากที่สุดของ BoW

ข้อความ: **Shane likes football; he is a big fan of Arsenal football team.**

แปลงเป็นตัวพิมพ์เล็ก ตัดอักขระที่ไม่ใช่ตัวอักษรอังกฤษและตัวเลข และตัด stop words ออก จะได้

shane likes football big fan arsenal football team

แปลงแต่ละคำเป็นจำนวนเต็มด้วยฟังก์ชัน $fhash$ ที่นำเสนอข้างบนนี้ (สมมติว่า $M = 4$) จะได้

$fhash('shane', 4)$ ได้ 3, $fhash('likes', 4)$ ได้ 0, $fhash('football', 4)$ ได้ 3, $fhash('big', 4)$ ได้ 2,
 $fhash('fan', 4)$ ได้ 1, $fhash('arsenal', 4)$ ได้ 2, $fhash('football', 4)$ ได้ 3, $fhash('team', 4)$ ได้ 3

ได้ BoW = [[0, 1], [1, 1], [2, 2], [3, 4]]

หมายเหตุ: อาจรู้สึกวุ่นวาย ไม่เห็นจะดีเลย เพราะหลายคำมีค่า $fhash$ เหมือนกัน เช่น **shane, football** กับ **team** แปลงได้เป็น 3 เหมือนกัน อันนี้เป็นสิ่งที่ต้องสูญเสียเพื่อแลกกับเนื้อที่เก็บขนาดเล็กลง ค่า M น้อยก็เกิดเหตุการณ์นี้มาก ค่า M มากก็เปลืองเนื้อที่ (อย่างไรก็ตาม ประเด็นนี้ไม่เกี่ยวข้องอะไรกับการบ้านนี้)

สิ่งที่ต้องทำ

เขียนโปรแกรมที่ทำงานตามขั้นตอนดังนี้

1. รับชื่อแฟ้มเอกสารที่จะอ่านมาวิเคราะห์ข้อมูลจากแป้นพิมพ์ เก็บในตัวแปร **file_name**
2. รับตัวอักษรระบุทางเลือกว่า จะหา BoW แบบมี feature hashing หรือไม่
 - ผู้ใช้ป้อนตัวอักษร y, Y, n หรือ N
 - ถ้าป้อน y หรือ Y แปลว่า ต้องการ feature hashing
 - ถ้าป้อน n หรือ N แปลว่า ไม่ต้องการ feature hashing
 - ถ้าป้อนตัวอื่น ให้แสดงข้อความว่า **Try again.** แล้วรอรับใหม่
 - ถ้าผู้ใช้ป้อน y หรือ Y โปรแกรมจะรอรับจำนวนเต็มที่ไม่เกินค่า **M** สำหรับการหา feature hashing
3. อ่านข้อมูลในแฟ้มชื่อ **stopwords.txt** ภายในแฟ้มนี้เก็บข้อมูลดังนี้
 - แต่ละบรรทัดเก็บ stop words ต่าง ๆ ที่ใช้ในการตัดคำที่น่าสนใจออก ตอนหา BoW หนึ่งบรรทัดอาจมีหนึ่งคำ หลายคำ หรือไม่มีสักคำก็ได้ (ถ้ามีหลายคำ จะคั่นด้วยช่องว่าง)
 - เช่น

```
i me my myself  
we our our ourselves  
  
you  
  
you're you've
```
4. อ่านข้อความจากแฟ้มที่ชื่ออยู่ในตัวแปร **file_name** เพื่อหาและแสดงลักษณะเฉพาะ ดังต่อไปนี้
 - จำนวนอักขระ (character count)
 - จำนวนตัวอักษรอังกฤษและตัวเลขเท่านั้น
 - จำนวนคำ (word count)
นิยามให้ "คำ" คือ ตัวอักษรอังกฤษและหรือตัวเลขที่ติด ๆ กัน โดยมีอักขระอื่นเป็นตัวคั่นคำ เช่น
Abc:a18 ("Okay") มีสามคำคือ **Abc**, **a18** และ **Okay**
 - จำนวนบรรทัด (line count)
 - Bag-of-words (BoW) ของคำทั้งหลาย ที่น่าสนใจ stop words และได้เปลี่ยนเป็นตัวพิมพ์เล็กแล้ว
ในกรณีที่ต้องการ feature hashing ให้แปลงคำทั้งหลายเป็นจำนวนเต็มด้วยวิธีที่ได้นำเสนอมา กับค่า **M** ที่ได้รับ
5. แสดงผลทางจอภาพตามลำดับ และตามรูปแบบในตัวอย่าง (หน้าถัดไป)
6. แสดงข้อความโต้ตอบตามตัวอย่าง (หน้าถัดไป) ให้เหมือนกับ

ตัวอย่าง

ตัวสีแดงคือข้อมูลที่ผู้ใช้ป้อนให้เป็น input

```
>>> %Run solution.py
File name = sample.txt
Use feature hashing ? (y,Y,n,N) n
-----
char count = 81
alphanumeric count = 61
line count = 4
word count = 19
BoW = [['555', 1], ['age', 1], ['best', 1], ['times', 2], ['wisdom', 1], ['worst', 1]]

>>> %Run solution.py
File name = sample.txt
Use feature hashing ? (y,Y,n,N) Y
M = 4
-----
char count = 81
alphanumeric count = 61
line count = 4
word count = 19
BoW = [[1, 1], [2, 3], [3, 3]]

>>> %Run solution.py
File name = sample.txt
Use feature hashing ? (y,Y,n,N) OK
Try again.
Use feature hashing ? (y,Y,n,N) Krub
Try again.
Use feature hashing ? (y,Y,n,N) Yes
Try again.
Use feature hashing ? (y,Y,n,N) Y
M = 10
-----
char count = 81
alphanumeric count = 61
line count = 4
word count = 19
BoW = [[0, 2], [1, 1], [3, 2], [7, 1], [8, 1]]
```

ใส่ชื่อแฟ้ม และ ไม่ต้องการทำ
feature hashing

ใส่ชื่อแฟ้ม และ ต้องการทำ
feature hashing

ใส่ชื่อแฟ้ม และ ต้องการทำ
feature hashing
ตรงนี้ใส่ผิดหลายครั้ง

sample.txt

It was the best of times,
it was the worst of times,
it was the age of wisdom.
"555"

stopwords.txt

it they
the a an
of on in at
is am are was were

```
# Prog-08: Bag-of-words
# # 6???????21 Name ?
```

ใส่เลขประจำตัว ชื่อ นามสกุล

ห้าม import อะไรเพิ่มเติม และห้ามใช้ที่
เก็บข้อมูลจำพวก set หรือ dict หรือที่
คล้ายคลึงโดยเด็ดขาด