MDA HW4 Report 108062533 洪翎恩

語言 python / 環境 pyspark / 執行 python3 LSH.py

步驟

1. 讀 txt files 並作前處理

```
#
# read file
#
document=[]
document_num=0
def read_file():
    global document,document_num
    for file in sorted(os.listdir("athletics/")):
        if file.endswith(".txt"):
            filename = os.path.join("athletics", file)
            txt = open(filename, 'r')
            txt = txt.read().replace("\n", " ").replace(","," ").replace("."," ").replace('"'," ").split()
            words=sc.parallelize(txt)
            document.append(words)
            document_num += 1

conf = SparkConf().setMaster("local").setAppName("MDA_HW4")
sc = SparkContext(conf=conf)
print("reading files...")
read_file()
```

每個文件讀進來後先去除文章內的符號並用 split()以 word 為單位
執行後 document 為 rdd type，內容為['word','word',...]，document_num=50 表示讀了 001.txt~050.txt (公告說資料改為前 50 個檔案，所以在 athletics 資料夾下只有該 50 個檔案)

2. Shingling (k_shingle=3)

```
#
#  SHINGLING
#
print("Shingling...")
k_shingles=3
shingle_dict=dict()
matrix=[]
shingle_num=0
for id in range(0,document_num):
    temp=[]
    tempdoc=document[id].collect()
    for i in range(len(tempdoc)-k_shingles+1):
        shingle=tempdoc[i:i+k_shingles]
        shingle=' '.join(shingle)
        temp.append(shingle)
        if shingle not in shingle_dict:
            shingle_dict[shingle]=shingle_num
            shingle_num+=1
    matrix.append(temp)
```

執行後輸出的 2D matrix 每一個 matrix[i] 表示 doc i+1 shingle 後的內容
Ex: matrix[i]=['ABC','BCD','CDE'...]
設一個 dictionary 去存 shingle 出現的順序。

3. Min-hash(hash_num=100)

```
#
#   MIN-HASH
#
print("Min-hash...")
num_hashes=100
a_hash = [randrange(0,shingle_num) for a in range(0, num_hashes)]
b_hash = [randrange(0,shingle_num) for b in range(0, num_hashes)]

def min_hash_function(a, b, sig):
    hashes = [((a * x) + b) % shingle_num for x in sig]
    return min(hashes)

def get_min_hash_row(sig):
    hashes = [min_hash_function(a, b, sig) for a, b in zip(a_hash, b_hash)]
    return hashes

signatures=[]
for id in range(0,document_num):
    each_doc=sc.parallelize(matrix[id])
    each_doc_signature=each_doc.map(lambda x: shingle_dict.get(x))
    min_hash_row = get_min_hash_row(each_doc_signature.collect())
    signatures.append(min_hash_row)
```

a_hash,b_hash 分別為長度 100 的亂數 list 這為 100 個 minhash 的參數
針對 matrix[i] 去查 dictionary 並回傳值 表示在 shingle * doc 中為 1 的位置
再把每個 document 經由 100 個 minhash 做出其 signature

4. LSH (band=50, row=2)

```
#
#   LSH
#
print("LSH...")
row=2
buckets_num=10
bucket=[]
for i in range(0,buckets_num):
    bucket.append([])

random_band=randrange(0,100,row)
for id in range(0,document_num):
    hash2bucket=((a_hash[0]*(signatures[id][random_band]+signatures[id][random_band+1]))+b_hash[0])%buckets_num
    bucket[hash2bucket].append(id)
```

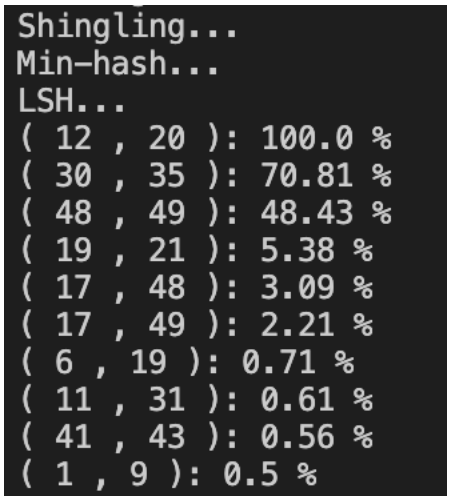
隨機取一組 band 將每個 document 對應到此 band 的部分做 hash 到 25 個 bucket 中

5. 選擇 25 個 bucket 中裡有 2 個以上的 documents 選為 candidate pair 並計算此 pair 的 Jaccard similarity 後放進 result list 並 sort 輸出在這些 candidate pairs 前 10 的 similarity

```
result=[]
for index in range(0,buckets_num):
    if int(len(bucket[index]))>=2:
        for i,j in list(combinations(bucket[index], 2)):
            a=set()
            b=set()
            for k in range(len(matrix[i])):
                a.add(matrix[i][k])
            for k in range(len(matrix[j])):
                b.add(matrix[j][k])
            score=(len(a&b)/len(a|b))*100
            result.append(list([i+1,j+1,score]))
result=sc.parallelize(result).sortBy(lambda x: x[2],ascending= False).collect()

for i in range(0,10):
    print("(",result[i][0],",",result[i][1],"):",round(result[i][2],2),"%")
```

Result:



每次執行會不一樣唯一不變的是第一名的 (12,20) 因為這兩個 document 內容一模一樣不論 hash functions 怎麼變 或是選到不同的 band 接會被 hash 到同一個 bucket 裡