Final Project

Recommendation System

109062640 莊沐庭

Project 摘要：

採 item-item 之 collaborative filtering，similarity 的部分使用 cosine similarity 的方法，最後再為所有 user 推薦一部電影

這次作業程式碼分為 3 個區塊

分別為 (**preprocess、cosine similarity、recommendation**)

Step1. Preprocess

(1) 包含讀 csv 檔，加上找總共有多少不同的電影，最後把 input 放到 pdata 這個 RDD 架構中。

```
def Preprocess() :
    global data, num_movie, check_movie_id
    data = pd.read_csv("test.csv", encoding="utf-8", names = ["User", "Movie", "Rating"])
    check_movie_id = sorted(list(set(data['Movie'].tolist())))
    num_movie = len(check_movie_id)
    for value in range(num_movie):
        movie_list.update({check_movie_id[value] :value})
    l = []
    with open( "test.csv", encoding="utf-8", newline = '' ) as file : # read file
        rows = csv.reader(file)
        for row in rows :
            l.append(row)
    pdata = sc.parallelize(l)
    return pdata
```

Step2. Cosine Similarity

(1) 做 cosine sim 之前置動作(整理資料、算長度等等)，把資料整理成[(user1, point1), (user2, point2) ...] ，再做卡氏積方便求 cosine sim，再求算 cosine sim 之間若發生除以 0 的話，則會多加一個常數避免發生 overflow，最後使用 dictionary 紀錄 user 評分電影狀況

```
def Cosine(plist) :
    global cos, movie_dict, sim_dict
    pdata = plist

    # [ ( user1, point1 ), ( user2, point2 ) ... ]
    plist = plist.map(change1).reduceByKey( lambda x, y : x + y ).mapValues(update_list1)

    # dictionary of user's point  ex : { user1 : 2 5 3 0, user2 : 3 0 2 5 ... }
    pmovie = pdata.map(change2).reduceByKey( lambda x, y : x + y ).mapValues(update_list2)

    # 計算每個user評分之length、計算每個評分扣掉mean ( 避免之後會divide zero狀況，所以算到0的話會一常數，本例是+0.3 )
    prate = plist.mapValues(Cal)

    # ( ( user1, user2 ), ( len1, ( point1 ) , len2, ( point2 ) ) ) 做user彼此之卡式積
    pcar = prate.cartesian(prate).map(lambda x: (tuple((x[0][0], x[1][0])), x[0][1], x[1][1]))

    # ( user1, ( ( user2, sim(1,2) ) , ( user3, sim(1,3) ) ... ) ) (算出彼此之sim，如果sim是0一樣加常數 (+0.1))
    cos = pcar.map(Cos).reduceByKey(lambda a, b : a + b)

    movie_dict = pmovie.collectAsMap()
    sim_dict = cos.collectAsMap()
    return plist
```

Step3. Recommendation

(1)針對該 user 沒有評分之電影做推薦評分，計算的方法是先前算出來各 user 之間的 sim 和各 user 對該電影評分做乘積後除以 sim 之和

```python
def Recommendation(plist) :
    user = plist[0]
    l = plist[1] # 原本user評分的list
    res = []
    for i in range(len(l)) :
        rate = l[i]
        if rate == 0 :
            s = 0
            total_sim = 0
            point = movie_dict[check_movie_id[i]] #電影對所有user評分之狀況
            for j in range(len(point)) :
                sim = sim_dict[user] # user之間的sim
                s = s + ( point[j] * sim[j][1] )
                if point[j] != 0 : # 有其他user評過分才要加上他們的sim
                    total_sim = total_sim + sim[j][1]
            temp = s / total_sim
            tup = ( user, check_movie_id[i], temp )
            res.append(tup)
    return res
```

最後是寫檔

```python
def writefile(ans) :
    with open("final_output.txt", 'w') as file :
        for i in ans :
            User = i[0]
            movie = i[1]
            msg = 'Recommend User {} : Movie {}'.format(User, movie)
            file.writelines( msg + "\n" )
```

結果：(只擷取一部分)

```
Recommend User 1 : Movie 76293
Recommend User 2 : Movie 1465
Recommend User 3 : Movie 7215
Recommend User 4 : Movie 93287
Recommend User 5 : Movie 6616
Recommend User 6 : Movie 140711
Recommend User 7 : Movie 3325
Recommend User 8 : Movie 2492
Recommend User 9 : Movie 30850
Recommend User 10 : Movie 100810
Recommend User 11 : Movie 140247
Recommend User 12 : Movie 1190
Recommend User 13 : Movie 32892
Recommend User 14 : Movie 3194
Recommend User 15 : Movie 164200
Recommend User 16 : Movie 3639
Recommend User 17 : Movie 26052
Recommend User 18 : Movie 1881
Recommend User 19 : Movie 2633
Recommend User 20 : Movie 667
```