Homework 2 : Page Rank

108062608 黃柏翰

Map Reduce Algorithm and Code

已知 PageRank 值求法和 Markov chain 的求法相近，加上

transition matrix 是 sparse matrix

所以參考以下的作法

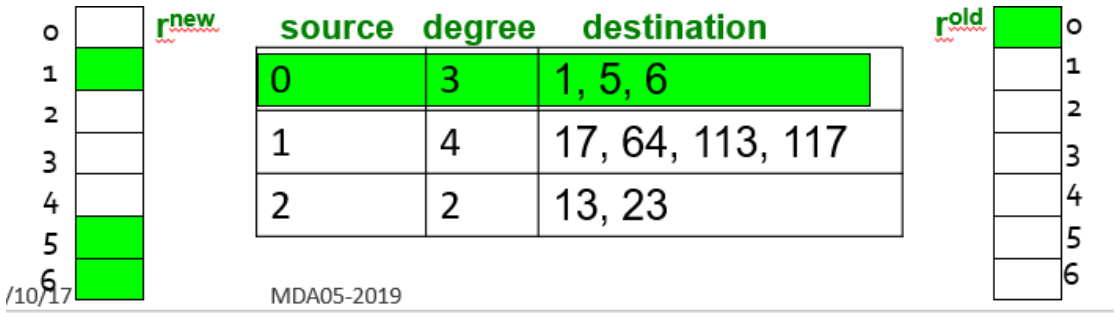
- $\forall j: r'^{new}_j = \sum_{i \to j} \beta \frac{r^{old}_i}{d_i}$

  $r'^{new}_j = 0$ if in-degree of j is 0
- Now re-insert the leaked PageRank:

  $\forall j: r^{new}_j = r'^{new}_j + \frac{1-S}{N}$ where: $S = \sum_j r'^{new}_j$
- $r^{old} = r^{new}$

a. Map

將資料讀入之後，可知資料的形式可以轉換成以下表格。



經過轉換希望能轉成跟 linked list 很相近的形式。

e.g.從 1 2, 1 3 -> (1,(2, 1/2)), (1,(3, 1/2))，

表示說 node1 會到 node2 的機率為 1/2 以及 node3 的機率 1/2

程式碼

```
def Mapper(self, process):
    data = process.groupByKey().map(lambda element: ((element[0], 1/len(element[1])), element[1]))
    # Reverse the groupby & to (1,(2, 1/2)), (1,(3, 1/2))
    transition = data.flatMapValues(lambda x : x).map(lambda x :  (x[0][0], (x[1],x[0][1])))
    return transition
```

結果顯示

```
[(10874, (10876, 0.1)), (10874, (10875, 0.1)), (10874, (9711, 0.1)), (10874, (5543, 0.1)),
(10874, (3251, 0.1))]
```

b. Reduce

1.將相同的 key 值的 pair 做 join，表示都是 node 1 要做處理

2. (1, ((2, 0.5), 0.2) -> 表示 node 1 PageRank 值是 0.2，跑到 node 2 的機率是 0.5

3. (2, 0.5*0.2) 表示 node 2 再這一次的轉換中累積到 0.5*0.2 的 PageRank 值

4. ReduceByKey()就能得到全部 node 的最新 PageRank 值，最後要考慮是否 deadend(也就是 sum all PageRank 值是否 1)

5. 沒有的話要加上

$$\forall j\text{: } r_j^{new} = r'^{new}_j + \frac{1-S}{N} \qquad \text{where: } S = \sum_j r'^{new}_j$$

程式碼

```
def Reducer(self):
    self.transition = self.Mapper(self.preprocess()).cache()
        #transition
    beta = self.beta
    N = self.N
    for i in range(self.times):
        # 先將node相同的join起來，to distribute the probability of node to others throgh transition matrix
        result = self.transition.join(self.state)
        # (1, ((2, 0.5), 0.2) -> represent node 1 have 0.2 point,
        # there is 50% chance for node 1 to node 2
        # after the next step, tuple will become (2, 0.5*0.2)
        # indicate that in this part, node 2 get 0.5*0.2 point, then reduceByKey.
        # And we can find the point for each node

        self.state = result.map(lambda x : (x[1][0][0], beta * x[1][0][1]*x[1][1])).reduceByKey(lambda x, y: x + y)
        sumofvalue = self.state.values().sum()
        # need to re-insert the leaked pagerank
        self.state = self.state.mapValues(lambda x : x + (1-sumofvalue)/N)
```

結果顯示

```
Final State :  [(1054, 0.0006296331377412288), (1056, 0.0006294743450160488), (1536,
0.0005250179229647834), (171, 0.0005125976925255169), (453, 0.0004961220101900608), (407,
0.0004856436417166822), (263, 0.0004803611430116l4), (4664, 0.0004717169212589019 6), (261,
0.0004638206024674985), (410, 0.0004618880576802773 6)]
```

以下是寫檔案的程式碼

寫檔程式碼

先取前 10 大 PageRank 的值，再寫入檔案

```
def filewriter(self):
    data = self.state.top(10, key = lambda x : x[1])
    with open ('Outputfile.txt', 'w') as f:
        for i,towrite in enumerate(data):
            f.write(str(towrite)+'\n')
```