2018 Introduction to Massive Data Analysis Assignment 2 -Finding Similar Articles

107062558 賴怡惠

Design of mapper and reducer

**Flowchart**

shingling

Minhashing

Merge two minhashing arrays

LSH

Merge LSH of same pair in different order

50 files

<shingling id> <shingling> <file1>,<file2>…

<iteration> <minhashing array>

<iteration> <minhashing array>

**100次**

<article no.> <article no.>

<article no.> <article no.>

**Design**

1. Shingling

Mapper: 文章內容先濾掉特殊、標點符號，因為我認為「有意義」的內容對於identical pair的辨識比較精準。接著根據空白切割成一個word，再從file path讀filename得知此3-shingling屬於哪個檔案，即可記錄下：<shingling> <file1><file2>…。

Reducer:因為下一步的minhashing需要利用shingling id做hashing，所以在此設定只能有一個reducer，如此一來每個shingling的id都是unique，因為只有一個reducer對下一個進來的key去update shingling id。output： <shingling id> <3-shingling> <file1><file2>…

1. Minhashing

此步驟因spec要求需要有100個minhashing function，所以需要做100次！

Mapper: 為了對shingling做從新排列，我們可以把這個問題想成是對shingling id做某種運算，讓新的shingling id仍在原來的範圍，但因id改變，而我們的排序是利用id的小到大排列，所以排列也改變了！為了達成以上的目的，我把shingling id做以下運算

(shingling id)%prime%N

，即可得到新排列的hashing id，output: <hashing id> <file1>,<file2>…

Reducer: 為了下一步merge minhashing array到同一個檔案的「方便」起見，這一步驟只能有一個reducer!

此外，此步驟還需要創建minhashing array，創建方法是一開始先建立一個大小為50的array，因為我們最一開始的input是50個檔案，而minhashing array的意義即利用hashing id代表每個檔案！而現在有50個檔案，所以minhashing array大小為50。

透過mapper的files資訊，我們將含有此file的hashing id assign給minhashing array對應的檔案，但若minhashing array對應該檔案的element已有被assign過，我們需要判別哪個hashing id比較小，擇小assign!

此步驟的output: <iteration> <minhashing array>，但我是在main function呼叫100次這組mapper、reducer然後把output存在不同iteration的directory下，所以這100個minhashing array會分散在不同目錄下。

1. Merge two minhashing arrays

因為上個步驟的100個minhashing array會分散在不同目錄下，但下個步驟的LSH需要把兩個minhashing array一起看，然後map到bucket，所以在此步驟我先把兩個minhashing array合在一起！

Mapper: 利用iteration mod 50作為reducer key，確保一個reducer一次合併「兩個」minhashing array。

Reducer: output key仍是mapper送進的key，作為strip的編號，value則是合併兩個minhashing array，我是利用“＠＠＠”作為下次切個的tag。

1. Lsh

Mapper: 因為minhashing array有50篇文章，我們需要把strip切割成一篇一篇的再送給bucket。送給bucket的行為我們利用reducer當成一種hashing，所以只需要把minhashing array的pair值當作key讓mapreduce的架構幫我們hashing！而LSH對於不同文章同個strip放到同個bucket才算是identical pair，所以我們也需要把第幾篇文章、哪個strip的資訊當作value傳給reducer。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | … | 50 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 50 |

Reducer:透過mapreduce架構做hashing後，雖然同個bucket，但我們仍需判斷是不是在同一組strip！若在同一組strip即output兩篇文章編號。

1. Merge LSH of same pair in different order

因為上個步驟不同reducer之間無法得知哪些pair已經output過，所以可能會出現同一組identical pair output太多次的情況，故在此做簡化，並把reducer限定一個，方便之後只需要開一個有整理過的檔案算jaccard similarity。

Mapper: 對pair做排序，小編號的文章排前面，大編號的文章排後面，也就是把小編號文章當key，大編號文章當value。

Reducer: 對於每個key，先把其value存到set，使的同樣的pair出現一次！然後再把不重複的value和key作為整理過後的output！

1. Calculate Jaccard Similarity

此步驟不寫成mapreduce，因為測試結果發現main內改變global static value結果在mapper或reducer裡面讀出得值和global初始值一樣，並不會隨著main改變而改變，所以無法打開第一個步驟的shingling檔案對identical pair找連集和交集，因為mapper、reducer讀不到identical pair！所以此步驟就在main裡面用sequential的方法！先把similarity當treemap的key，pair當成值，然後利用treemap會default排序的特性，descendingMap即可得到由高到低排序的相似度！

Final output

