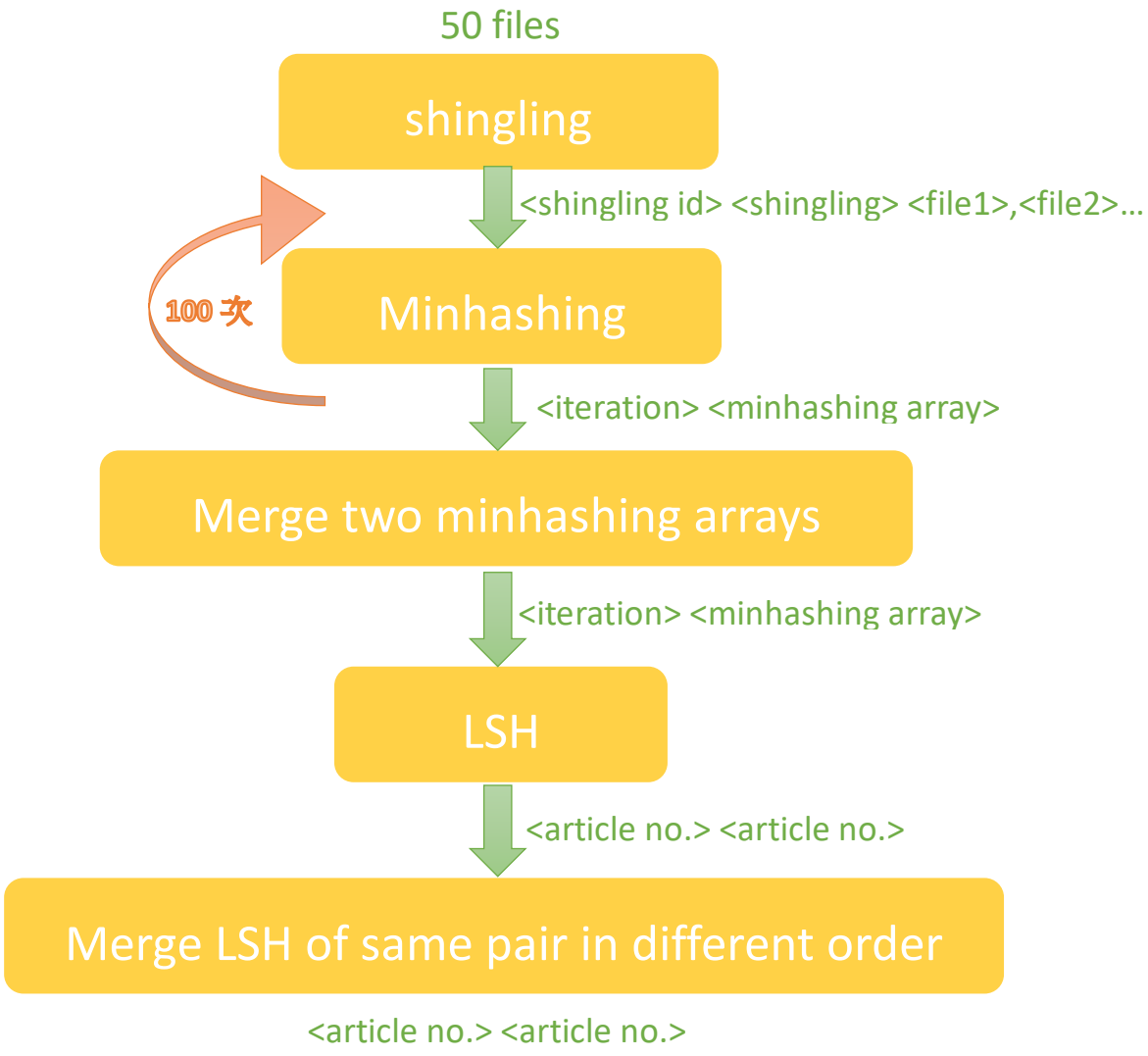
2018 Introduction to Massive Data Analysis Assignment 2 -

Finding Similar Articles

107062558 賴怡惠

Design of mapper and reducer

Flowchart



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>…

2. 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× $\text{random\_}constant1 + random\_constant2$)%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。

3. 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！

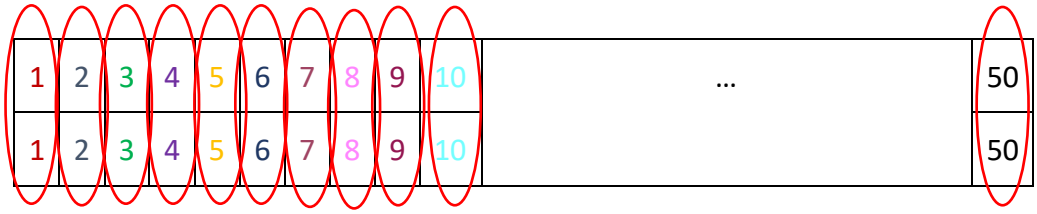
Mapper: 利用 iteration mod 50 作為 reducer key，確保一個 reducer 一次合併「兩個」minhashing array。

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。

4. 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。



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。

5. 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！

6. 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