PageRank with Hadoop

TELECOM PARISTECH

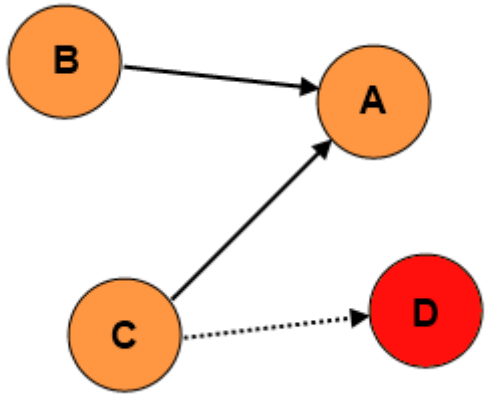
16/05/2017

PageRank with Hadoop

Page Ranking

Larry Page came up with the algorithm to determine the page ranking and build a search engine around it in 1996 and named it Google.

In this example we will use 4 pages: A, B, C and D an non-existing page. This is a page that has not been created yet, but is being links to from C. In Wikipedia you recognize those pages as red and underlined.



The links between the pages are as follows:

Rank of A is highest, because it will get points from B and C.

PageRank of page A = 'share' of the PageRank of the pages linking to A.

The formula of calculating the points is as following:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

The formula can be simplified to this:

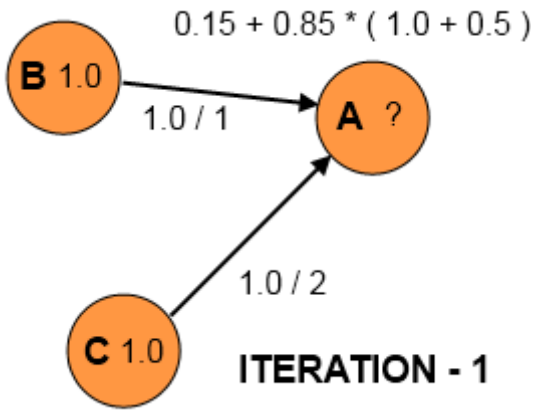
PR(A) = (1-d) + d(PR(B) / Cout(B) + ... + PR(C) / Cout(C))

The **d** in the formula is the damping factor to simulate 'a random surfer' and is usually set to 0.85.

If you apply the formula to our example:

PageRank of A = 0.15 + 0.85 * (PageRank(B)/outgoing links(B) + PageRank(...)/outgoing link(...))

Calculation of A with initial ranking 1.0 per page:



If we use the initial rank value 1.0 for A, B and C we would have the following output:

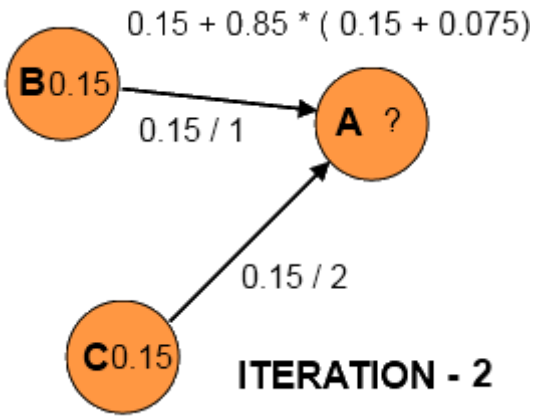
I have skipped page D in the result, because it is not an existing page.

A: 1.425

B: 0.15

C: 0.15

Calculation of A with ranking from ITERATION-1:



If we use these ranks as input and calculate it again:

A: 0.34125

B: 0.15

C: 0.15

We see that the page rank of page A is reduced. The PageRank is based on previous calculations and will get more accurate after more runs. You can add new pages, new links in the future and calculate the new rankings. This is one of the tools which search engines use to create there index.

The Plan

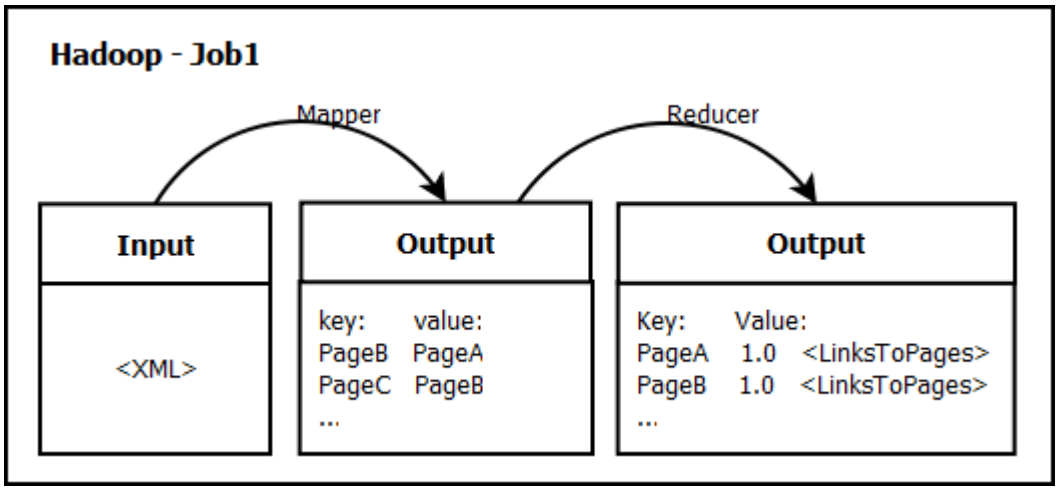
We will split the work in three different Hadoop jobs: parsing, calculating and ordering.

Hadoop Job 1 will parse the wiki pages xml into articles

In the Hadoop mapping phase, get the article's name and its outgoing links.

In the Hadoop reduce phase, get for each Wikipage the links to other pages.

Store the page, initial rank and outgoing links (remove redundant).



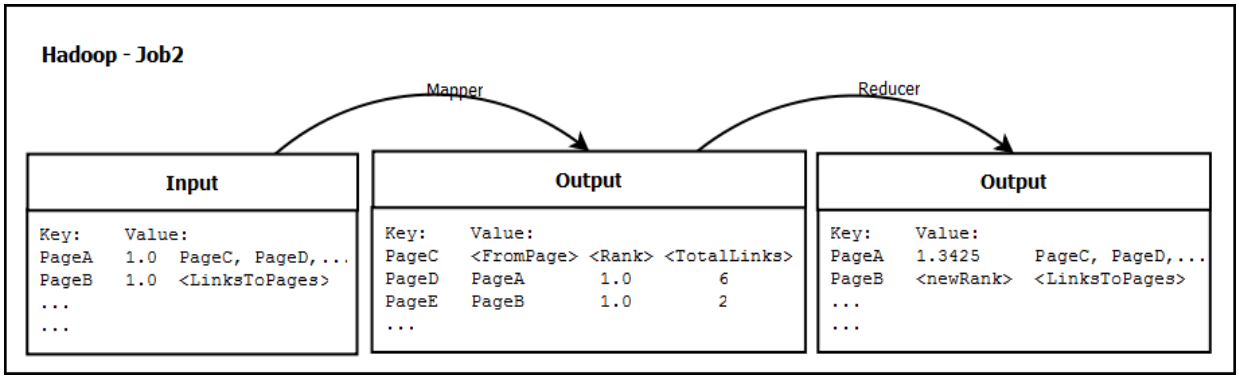
Hadoop Job 2 will calculate the new pageRank

In the mapping phase, map each outgoing link to the page with its rank and total outgoing links.

In the reduce phase calculate the new page rank for the pages.

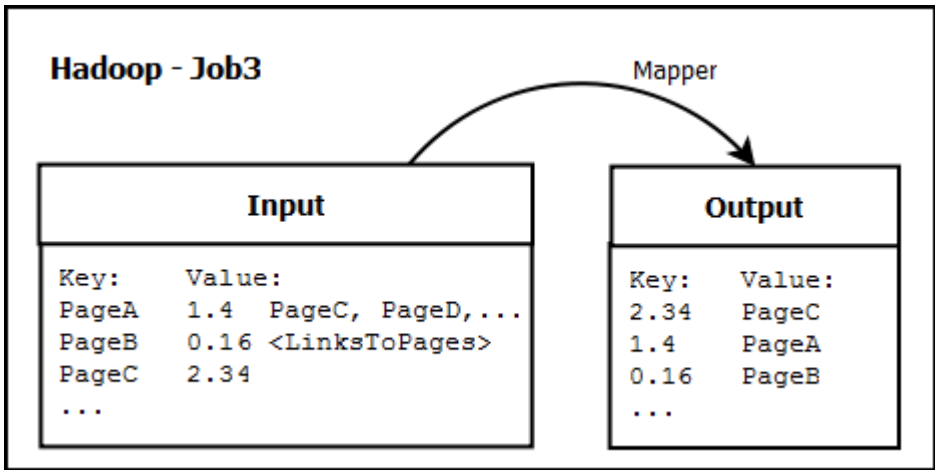
Store the page, new rank and outgoing links.

Repeat these steps for more accurate results.



Hadoop Job 3 will map the rank and page

Store the rank and page (descending ordering on rank)



Resources & Evaluation

1. Download the baseline code source from;
    a. http://tiresias.enst.fr/submitServer/ (using your credentials received in a separate email)
    b. Click on "**INF344 (2016-2017): Web Data (INF344) 2017**"
    c. Download starter files
2. Try to fill the "not yet implemented" methods (tagged with TODO)
3. Test your code with the **PublicTestPageRank.java**
4. Connect to this platform http://tiresias.enst.fr/submitServer/ and click on "**Submit**"
5. Upload the following java classes to the platform and submit your code (your code will be run against the public unit tests, if succeeded it will be run against the secret ones)
    a. Job1Reducer.java
    b. Job2Mapper.java
    c. Job2Reducer.java
    d. Job3Mapper.java
    e. Job3SortingComparator.java
6. Scoring
    a. Job1 reduce (2pts)
    b. Job1 map reduce (2pts)
    c. Job2 map (4pts)
    d. Job2 reduce (4pts)

e. Job2 map reduce (2 pts)

f. Job3 map (4pts)

g. Job3 map reduce (2pts)

Notes

- On time deadline: 20/05/2017 at 11:00 PM
- Late deadline: 21/05/2017 at 11:00 PM
- In order to release your code, you've to succeed the public unit tests
- You can release your code twice and we will consider the highest scoring submission
- Only the above mentioned classes are considered during the evaluation, so adding other files will not be taken into consideration
- The late penalty is 2pts