

Homework 3 - PageRank on the Wikipedia Corpus

This assignment is based heavily on Lab 3 of The University of Washington's "Problem Solving on Large Scale Clusters"

***** Please direct questions to 15-505-instructors@kamalnigam.com *****

Deliverables – **Due October 16, 2007**

1. Writeup as outlined at the end of this document
2. Your .java files and documentation detailing how to run your code (arguments, expected input, etc)

The Goal --

Implement PageRank, turn Wikipedia into a giant graph, run PageRank on said graph, run it several more times (ideally until the values converge), return (in a humanly parseable sort of way) the PageRank of all the articles.

The Algorithm –

<http://infolab.stanford.edu/~backrub/google.html> (section 2.1.1)

With examples: <http://www.ams.org/featurecolumn/archive/pagerank.html>

The Corpus--

1. Wikipedia offers a compressed XML file for download that contains every article on the site, including redirect and disambiguation pages.
2. We've reformatted the data set into 194 ~64MB files. Each 64MB file contains several thousand Wikipedia articles, formatted one per line.
3. The characters '<' and '>' have been replaced by '<' and '>' because the input format we recommend strips XML tags.
4. That input format is `org.apache.hadoop.mapred.TextInputFormat.class`
5. The title of each article is marked by "`<title>Name of the article</title>`" in which the brackets have been reformatted (as mentioned above) so the title is now marked by "`<title>Name of the article</title>`".
6. Links to other wikipedia articles are of the form "`[[Name of other article]]`". A full explanation of link formatting can be found by searching for wikitext on wikipedia.
7. Find the data files on dfs at `/shared/wikipedia/`
8. You can use a subset of the data by using `addInputPath` instead of `setInputPath`
9. If you'd like, get your own copy of the compressed XML at download.wikimedia.org/enwiki or the HTML version at static.wikipedia.org (several languages are available in HTML!)

One Possible Outline (each of these is a MapReduce) --

1. `graphBuilder` - Takes the Wikipedia data and constructs a link graph in a `SequenceFileFormat` file.
2. `pageRankIter` - Iteratively updates the page rank value of a graph in `sequenceFileFormat`
3. `pageRankViewer` - Converts `sequenceFileFormat` to text format and sorts nodes

by their page rank score. It's helpful here to negate the pagerank scores so that sites with the highest values appear first.

Complications --

This isn't super efficient (sorting all of the Wikipedia articles in the reduce step of PageRank is slow!) so keep three things in mind:

1. Don't save this for last minute.
2. Set the number of reduce tasks using `setNumReduceTasks` (five is good, more is probably better).
3. Get extra use out of your PageRank reduce method by also setting it as your combiner class.

Write-up --

1. How long do you think this project will take you to finish? (Record this before you start)
2. How much time did you actually spend on the project?
3. Acknowledge any assistance you received from anyone except assigned course readings and the course staff.
4. Explain why it's ok to use a combiner class with the PageRank algorithm.
5. What was your stop criterion for the `pageRankIter` phase?
6. What are the ten pages with highest rank in the provided Wikipedia corpus?

Some extensions to think about, but not required--

1. We used the `TextInputFormat` class because we didn't have the time to figure out how to MapReduce over an XML file. It's probably trivial... how would you do it?
2. If you tweak your `graphBuilder` so that it works with the Wikipedia HTML input you could run PageRank on some other languages. Then you could make a couple sweeping conclusions about cultural differences evidenced by differences in highly ranked pages.
3. Implement variably weighted links. What's your heuristic? How would you convince yourself that it works?
4. $(\text{InvertedIndex} + \text{Pagerank}) * \text{Wikipedia} = \text{rudimentary search engine}$. 😊