Programming Assignment 3

Estimating the Importance of Wikipedia Articles using the PageRank Algorithm

Due: 4/30/2016

(1) Overview

In this assignment you will design and implement a distributed version of the PageRank algorithm based on the Azure platform, where you will calculate the importance of Wikipedia articles using their references of each other. As we mentioned in class, the PageRank algorithm was used by early versions of the Google search engine to rank web pages. The underlying assumption is that more important web pages are likely to receive more links from other web pages.

Your implementation needs to be scalable, where the goal is to estimate PageRank values over the Wikipedia dataset. These results can be used to rank search results within Wikipedia. As most of the Wikipedia articles contain links to other articles, we are going to take advantage of such inter-article links to measure the importance of each page. Note that we do not consider links to outside webpages. As the current version of Wikipedia contains more than 5 million articles, we are going to use the Simple English version of Wikipedia as the dataset (its dump is available at https://dumps.wikimedia.org/simplewiki/). Your algorithm should be able to parse this dataset and generate ranking results.

(2) Requirements

Your implementation should be based on the Azure platform, but you are free to choose whatever programming method (e.g., Hadoop, Spark, etc) and data model that fit your needs for this project. You only need to rank pages that have meaningful titles. To do this, you may develop a preprocessing phase of your implementation, and document your design choices in the submission. Hence, there is no single standard solution. In the deliverables of this assignment you should provide:

A. A sorted list of Simple English Wikipedia pages based on their PageRank values in descending order. Each row should contain the title of the article and its PageRank value. This computation should be performed using at least 5 virtual machines in Azure. You should present results from at least 25 iterations.

B. A step-by-step experiment report with screenshots to show your results, including how you preprocess data, deploy your running code, and obtain results. You should also include the running time for your code.

(3) PageRank background

The term PageRank comes from Larry Page, a founder of Google. PageRank is a function that assigns a real number to each page in the Web (in our case Wikipedia articles). A page with a higher PageRank value will be considered as a more important page. To use this algorithm, we choose the dataset of the English-language Wikipedia corpus. Each page of the Simple English Wikipedia is represented in XML as follows:

<page>
<title> Page_Name </title>
(other fields we do not care about)
<revision>
<text> (Page body goes here)
</text>
</revision>
</page>

Note that the <text> may contain other fields such as “xml:space="preserve", among others. You should ignore such fields in your preprocessing of the data.

If we use the version 20160305 of the dump as an example, you will find that the decompressed XML file is about 520MB in size. Links are specified in “[[ ]]”. Only consider terms that have been as titles. You can also download a file containing the titles only as “simplewiki-20160305-all-titles.gz” in the page of https://dumps.wikimedia.org/simplewiki/20160305/. You can choose to ignore pages that are either placeholders or redirections, as such pages do not contain meaningful texts.

To use the PageRank algorithm, you may either choose to use the MapReduce as the underlying paradigm, or Spark (GraphX), according to your choice. There are plenty of resources available online on how to implement PageRank with these paradigms, such as:

Pagerank on Mapreduce:

http://michaelnielsen.org/blog/using-mapreduce-to-compute-pagerank/
https://www.cs.utah.edu/~jeffp/teaching/cs5955/L24-MR+PR.pdf

Pagerank on Spark:

https://spark.apache.org/docs/1.1.0/graphx-programming-guide.html#pagerank
https://www.airpair.com/apache-spark/posts/distributed-pagerank-using-apache-spark-and-neo4j

You should not, however, directly copy any code from online resources. Instead, you should develop your own version of data processing with the help of these

references. Your implementation should scale to multiple virtual machine instances. There is no limit on what programming language you will use.

(4) Grading

This assignment will be graded as follows:

- 40 points: Implementation of the Page Rank algorithm (you need to document your design decisions in the report).

- 40 points: Step-by-step experiments and screenshots. We are going to use your screenshots to perform grading, so providing detailed step-by-step screenshots is necessary.

- 20 points: Project report, the analysis of results and conclusions. Does your ranked list seem to make sense? Add your discussions and conclusions as needed.