Programming Assignment 3 Report and Usage Guide

Ellias Palcu and Caleb Williamson

Abstract

This document outlines the implementation and usage of a Hadoop MapReduce program that implements the PageRank algorithm. The simple English version of Wikipedia is used as input to this program, and a ranking of most relevant pages is its output.

I. Introduction

Programming Assignment 3 explains that in order to predict surfing habits of users, it is necessary to be able to rank page relevancy. One way to produce this ranking was developed by Larry Page for Google, and is known as the PageRank algorithm. Our implementation performs the page ranking similar to the initial 1998 implementation by Google; however, it extends the approach slightly. Whereas the initial implementation considered a page's rank based upon the number of its in-links, our implementation takes it a step further by iterating across each in-link and determining the ranking based upon the multitude of each in-link's in-link.

II. Background

The approach taken to solve the given problem can be split into three parts: preprocessing, MapReduce, and finally postprocessing. Preprocessing is necessary in this case to read and list the graph created by links found to other articles within the simple English version of Wikipedia. The Map portion of the implemented MapReduce program handles building the graph based on the preprocessed data and assigns initial PageRank values. The Reduce step iterates through the graph created and calculates a page's true rank value based on the links to it, the links to its links, and so on for 25 iterations. Finally, the postprocessing phase takes the page number of links output by the Reduce step and orders them, ranking from highest to lowest results.

Some assumptions were made when implementing the Map and Reduce steps that may have lead to specific behavior of the programs. These assumptions and behaviors include:

- Only valid Wikipedia article titles were considered valid links. This was accomplished by building a set of valid titles based on the all valid titles file provided by Wikipedia, namely "simplewiki-20161120-all-titles". The links were then compared to this set to ensure that only valid link titles were included in the MapReduce step.

- If a link points back to itself (i.e. to the respective Wikipedia article), it is not included as a valid link.

III. Usage

Included within the tar file is all of the source code utilized to achieve the MapReduce functionality, namely "mapper.py", "reducer.py", "parse.py", and "ranks.py". Furthermore, the results from each part are also included, labeled "adj_list.txt", "hadoop_results.txt", and "hadoop_ranks.txt". The original input, the simple English wikipedia dump from November 20th, 2016, is not included due to its size. The MapReduce portions of this submission follow a similar invocation procedure to other Hadoop based programs. The preprocessor and postprocessor are simply Python scripts. The following steps show how to run these series of programs:

1) Download necessary simple English Wikipedia dumps
2) Run preprocessing

```
$ python parse.py
```

```
Mexico  April
Chile   April
Marathon        April
Diamond April
Wales   August
Uruguay August
Painting        Art
Sculpture       Art
Aircraft        Air
Airplane        Air
Airport Air
1978    Autonomous communities of Spain
C++     Adobe Illustrator
Janus   Adobe Illustrator
Popeye  Adobe Illustrator
Simba   Adobe Illustrator
Pangaea Adobe Illustrator
Fertilizer      Farming
Weed    Farming
Breeding        Farming
Ranching        Farming
Erosion Farming
```

Fig. 1: File containing results from preprocessing step

3) Set up Hadoop environment

```
$ export HADOOP_HOME=/usr/local/hadoop-2.7.1
$ export PATH=${HADOOP_HOME}/bin:${PATH}
```

4) Create a test directory and add input files to Hadoop File System

```
$ sudo -s /usr/local/hadoop-2.7.1/bin/hadoop fs -mkdir /test
$ sudo -s /usr/local/hadoop-2.7.1/bin/hadoop fs -chmod 777 /test
$ hadoop fs -put shakespeare.txt /test/adj_list.txt
```

5) Run the MapReduce program

```
$ hadoop jar ${HADOOP_HOME}/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar
-files mapper.py,reducer.py
-mapper mapper.py -reducer reducer.py
-input /test -output /test/output
```

```
epalcu@resourcemanager:~/cosc560_PA3$ hadoop jar ${HADOOP_HOME}/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar -files mapper.py,reducer.py -mapper mapper.py -reducer reducer.py -
input /test -output /test/output
packageJobJar: [/tmp/hadoop-unjar1158957209586370863/] [] /tmp/streamjob2629014149333091302.jar tmpDir=null
17/04/29 15:16:49 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/10.10.1.2:8032
17/04/29 15:16:50 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/10.10.1.2:8032
17/04/29 15:16:51 INFO mapred.FileInputFormat: Total input paths to process : 1
17/04/29 15:16:52 INFO mapreduce.JobSubmitter: number of splits:2
17/04/29 15:16:52 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1493310469146_0002
17/04/29 15:16:53 INFO impl.YarnClientImpl: Submitted application application_1493310469146_0002
17/04/29 15:16:53 INFO mapreduce.Job: The url to track the job: http://resourcemanager.epalcu-qv25257.educationproject-pg0:8088/proxy/application_1493310469146_0002/
17/04/29 15:16:53 INFO mapreduce.Job: Running job: job_1493310469146_0002
17/04/29 15:17:13 INFO mapreduce.Job: Job job_1493310469146_0002 running in uber mode : false
```

Fig. 2: Spinning up the MapReduce jobs

```
17/04/29 15:18:25 INFO mapreduce.Job:  map 0% reduce 0%
17/04/29 15:18:59 INFO mapreduce.Job:  map 52% reduce 0%
17/04/29 15:19:04 INFO mapreduce.Job:  map 100% reduce 0%
17/04/29 15:19:33 INFO mapreduce.Job:  map 100% reduce 17%
17/04/29 15:19:36 INFO mapreduce.Job:  map 100% reduce 71%
17/04/29 15:19:40 INFO mapreduce.Job:  map 100% reduce 78%
17/04/29 15:19:46 INFO mapreduce.Job:  map 100% reduce 82%
17/04/29 15:19:49 INFO mapreduce.Job:  map 100% reduce 88%
17/04/29 15:19:53 INFO mapreduce.Job:  map 100% reduce 91%
17/04/29 15:19:56 INFO mapreduce.Job:  map 100% reduce 94%
17/04/29 15:20:00 INFO mapreduce.Job:  map 100% reduce 96%
17/04/29 15:20:03 INFO mapreduce.Job:  map 100% reduce 98%
17/04/29 15:20:06 INFO mapreduce.Job:  map 100% reduce 99%
17/04/29 15:20:09 INFO mapreduce.Job:  map 100% reduce 100%
```

Fig. 3: MapReduce progress

6) Copy the result from Hadoop File System to the main file system

```
$ hadoop fs -get /test/output/part-00000 hadoop_results.txt
```

7) Run postprocessing

```
$ python ranks.py
```

```
elliass-MacBook-Pro:cosc560_PA3 epalcu$ python ranks.py
Rank 1: Feyenoord
Rank 2: Eredivisie
Rank 3: Recovery
Rank 4: Eminem
Rank 5: Jay-Z
Rank 6: Ne-Yo
Rank 7: Rihanna
Rank 8: Bangladesh
Rank 9: Dhaka
Rank 10: Chittagong
```

Fig. 4: Ranking results from postprocessing

IV. Results and Conclusions

The MapReduce process took approximately 39 minutes. It is worth noting that a majority of that processing time came from the final gathering of the results, not from the computation, which took only 1.75 minutes, of the results themselves. Even so, this is still more efficient than running an unparallelized version of the PageRank algorithm, as preliminary tests showed.

The article ranking results shown in /reffig:ranks represent the top 10 most highly ranked results according to the PageRank algorithm. An inspection of these results shows that many of them are references to societal or cultural entities, such as musicians, soccer clubs and leagues, and nations and cities. These are logical results, as each of them would contain many in-links from other important articles. As these entities are central to a variety of

current and developing cultural interests, it makes sense that they would top the list. It is also logical to think that these entities will continue to rank highly for several years to come, as new links to these articles are likely generated on a regular basis.