Programming Assignment #3
Hadoop *N*-Gram
Due Tue, Feb 21, 11:59PM

In this programming assignment you will use Hadoop's implementation of MapReduce to search Wikipedia. This is not a course in search, so the algorithm focuses on simplicity rather than search quality or performance.

N-Grams

The *n*-grams of a sequence of symbols are all of the subsequences of length *n*. If each symbol is a word, for example, then the 3-grams of "now is the time for all good men" are "now is the", "is the time", "the time for", "time for all", "for all good", and "all good men." *n*-grams have many uses in natural language processing.

In this assignment you will use *n*-grams to compute a (toy) similarity metric between a query document and the entire text of the English Wikipedia, about 31GB. The score for a document consists of the number of *n*-grams from its text that also occur in the query document[1]. Your program will find the Wikipedia page with the highest score.

Specification details

- Words are any consecutive sequence of alphanumeric characters.
- The punctuation marks '.', '!' and '?', and consecutive sequences of those marks, are considered to be a special word "#". (This gives the algorithm access to sentence boundaries.)
- All non-alphanumeric characters that are not punctuation marks are considered to be whitespace.
- Pages are concatenated together in the input document. To detect a new page, look for (and parse) a line containing

  "<title>" + new-title + "</title>"
- The line with the title is not included in the *n*-grams for a page.
- No *n*-gram should include symbols from more than one page.
- If two pages have the same score, prefer the title that is lexicographically largest.
- You are not required to compute an answer if all pages have a score of 0.
- Don't worry about getting the first or last pages in a file correct.

Note that this metric is not symmetric. The score for document *A* against query document *B* is not the same as the score for *B* when *A* is the query document.

What we give you

- `Tokenizer.java` – A Java class that performs the tokenization procedure detailed above. Note that once you pass a line to Tokenizer it will remove all punctuation so you will have to check for a title line (as defined above) prior to passing it a line.
- `/tmp/enwiki/*` (Amazon EC2 only) – The all subdirectory contains the entire text of English Wikipedia, broken up into 64MB pieces. The half, quarter, eighth, and sixteenth subdirectories contain non-overlapping fractions of all.
- `query1.txt` and `query2.txt` – Example query documents.
- `chunk0` - The first 64 MB of Wikipedia. Do NOT try to parse it. You should only be using the Tokenizer to pull words (strings of alpha-numeric characters) out of it.
- `/usr/class/cs149/hadoop-0.21.0` – A Hadoop installation that can be used to run Hadoop jobs. On the `corn` machines this is a single-node installation that you should use to do your primary development. Once you have something working you can migrate your code over to `Amazon EC2`. The Hadoop installation on `Amazon EC2` has access to 16 nodes, but you are sharing it with the entire class so please limit yourself to only one job at a time.
- `wiki_hadoop` (Amazon EC2 only) – A script for running Hadoop on the Amazon EC2 machine. You should copy this script to your local directory on Amazon EC2. It will create an output directory called output in your local directory and place the results of your Hadoop job in this directory. Note that it will OVERWRITE results from one run to the next, so copy any files you want to keep out of the output directory between runs.

`Tokenizer.java, chunk0 (64MB), wiki_hadoop` and the two query files are in `/usr/class/cs149/assignments/pa3` .

Your tasks

Starting with one of the Hadoop word-count examples such as http://hadoop.apache.org/mapreduce/docs/current/mapred_tutorial.html implement a map-reduce program that finds the title of the page with the highest matching metric. Your program should take as input parameters *n*, the name of the query file, a directory that contains the input files, and the name of a directory to create to store the output. When looking at Hadoop examples or Hadoop documentation MAKE SURE YOU ARE LOOKING AT A TUTORIAL OR DOCUMENTATION FOR VERSION 0.21.0 AND NOT AN EARLIER VERSION!

Before running a Hadoop job you must first set up your environment. This only has to be done once per shell on a given machine. You might consider adding these instructions to your login script so you don't have to enter these commands every time you login to a machine.

➢ export HADOOP_HOME=/usr/class/cs149/hadoop-0.21.0

➢ export PATH=$PATH:${HADOOP_HOME}/bin

Compiling a Hadoop job must be done in two steps. First you must compile your Java code and then you must link it into a jar file. Let's say your map-reduce job is in Ngram.java and class_dir is a directory you have created for storing class files. The following commands will compile your code into a jar file ngram.jar. (Note the first line should be continuous, ignore the newlines)

- javac –cp ${HADOOP_HOME}/hadoop-common-0.21.0.jar:${HADOOP_HOME}/hadoop-hdfs-0.21.0.jar:${HADOOP_HOME}/hadoop-mapred-0.21.0.jar –d class_dir Tokenizer.java Ngram.java
- jar –cvf ngram.jar –C class_dir/ .

You now have a jar file that can be used for running jobs on either the corn machines or on Amazon EC2. To run a job on the corn machines, use the following command:

- hadoop jar *ngram.jar Ngram 4 query1.txt* \
  file://$PWD/input file://$PWD/output

This will run your `Ngram` code with ngrams of size 4 using the file `query1.txt` as input. Note that for the input and output directories you always must specify the global path and prefix it with the `file://` qualifier. Hadoop assumes that all input and output directories are references to the Hadoop file system (HDFS). The `file://` qualifier tells Hadoop to use files from the local file system instead. You should create your own test inputs and test them on the `corn` machines first.

Once you've got it working for small data sets on the local machine, you can move your code to the Amazon EC2 cluster to run on multi-node installation. Although small by Hadoop cluster standards, 16 nodes should be sufficient to tackle the entire enwiki input set. Run both query1.txt and query2.txt against `/tmp/`enwiki/all with *n*=4. The command for running on `Amazon EC2` is:

- wiki_hadoop jar *ngram.jar Ngram 4 query1.txt* /tmp/enwiki/all

To ssh into the Amazon EC2 instance,

ssh –i <SUNetID>.pem <SUNetID>@ec2-50-18-28-188.us-west-1.compute.amazonaws.com

Note that is a different Amazon EC2 instance from the last assignment. A private key for this instance will be sent to you via your Stanford email.

Note that we have already moved all of the Wikipedia files into HDFS for you. You can reference them directly as shown above. Replacing `/all` with `/half /quarter /eighth` or `/sixteenth` will get you a corresponding chunk of Wikipedia. You do not need to specify a target directory. The `wiki_hadoop` script will create an `output` directory in the directory in which it is run. It will OVERWRITE this directory from one run to the next. Make sure to move any files you wish to save out of the `output` directory between runs.

Hints and catches

By default, file paths in Hadoop refer to HDFS. To refer to paths in the normal file system, prefix the absolute path with "file://".

Hadoop processes (due to their distributed nature) often run with different permissions than the user that starts them. As a result you sometimes might encounter problems creating output directories. If this occurs try using the `chmod` command to give write permissions to all directories along the path you wish to write.

The Hadoop codebase reuses object instances by mutating them, in an attempt to reduce the amount of work required by the GC. (This is premature optimization at its worst) The effect is that your reduce function must **copy** the object returned from the values Iterable if it wishes to keep it after a call to next().

When running on the cluster, if you would like to check how busy the Hadoop server is, you can run the following command: `hadoop job -list` to see who else is running Hadoop jobs.

Running jobs on a distributed file systems can often result in failed reads/writes of files. Hadoop will report these errors as exceptions. In almost all cases Hadoop will recover from errors by itself. If you receive these messages, scrutinize them carefully and make sure you understand them as well as whether Hadoop handled them correctly before contacting the TAs. If your `output` directory contains a `_SUCCESS` file then Hadoop successfully recovered.

Correctness (80%)

There are several opportunities for performance results to be affected by contention, both on a per-node basis and on the cluster interconnect, so we are not going to include a specific performance target in the assignment. The bulk of the grade will be on the correctness of your implementation. We will evaluate this by looking at your code and by looking at the answers for querying the two example query files against the enwiki/all data set with n=4.

Scalability (20%)

We're not worried about the exact constants (map-reduce is a truck, not a sports car), but we care about the expected runtime of your algorithm in the limit. Let q be the size of the query document, n the size of the input pages within which to search, and P the number of processors. Your algorithm should complete in time $O(q+n/P+\ln P)$ or better, and it should require only a single map-reduce pass.

Extra credit (15%)

Modify your code to compute the 20 best matches and their scores in a single map-reduce pass, rather than just the single best. Include all of your results in answers.txt. Your extra-credit should still make only a single map-reduce pass over the data.

Submission instructions

You should submit

- The complete source code for your working solution;
- A brief text file named readme.txt (maximum 1 page) explaining how it works and why it is correct;
- Screen grabs of the INFO messages that result from a Hadoop run with n=4 against `/tmp/enwiki/all` for query1.txt and query2.txt.
- Your answers from the full runs, in a file answers.txt.

To submit the contents of the current directory and all subdirectories, run

```
/usr/class/cs149/bin/submit pa3
```

This will copy a snapshot of the current directory into the submission area along with a timestamp. We will take your last submission before the midnight deadline. Please send email to the staff list if you encounter a problem. You may run the submission script on a Leland machine.