Programming Assignment #3
Hadoop *N*-Gram
Due Tue, Feb 19, 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 40GB. 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.

---

[1] 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

Contents of `/usr/class/cs149/assignments/pa3`:

- `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.
- `query1.txt` and `query2.txt` – Example query documents.
- `chunk_aa` – 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.
- `local-hadoop` – A set of scripts for running Hadoop locally on Leland machines. See below for usage instructions.

Leland machines only:

- `/usr/class/cs149/hadoop-1.1.1` – A Hadoop installation you can use when testing on Leland machines. See below for usage instructions.

Amazon EC2 only:

- `/wikipedia` – The `all` subdirectory contains the entire text of English Wikipedia, broken up into 64MB pieces. The `16gb`, `8gb`, `4gb`, `2gb` subdirectories contain various subsets of the whole dataset. Note that these files are stored in HDFS, not the local filesystem.

Your tasks

Start with one of the Hadoop word-count examples:

- `http://hadoop.apache.org/docs/r1.1.1/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 DOCUMENTATION FOR VERSION 1.1.1 AND NOT A DIFFERENT VERSION! Hadoop has not historically been API-compatible across versions.

Running on Leland machines

Before testing on Amazon EC2, you should test your code on Leland machines using the installation and configuration scripts we have provided. This installation of Hadoop is configured to run on a single node, whereas Amazon EC2 gives you access to 16 nodes at a time. To start, copy the `pa3` files to your home directory and run the following:

```
local-hadoop/start-local-hadoop.py
```

These commands create a directory called `conf` with a configuration for running Hadoop locally, and start a local Hadoop server, respectively. Since these commands spawn daemons that continue to run after you log out, you need to explicitly stop them when you are done:

```
local-hadoop/stop-local-hadoop.py
```

Next, you need to set some environment variables. Choose one of the commands below, depending on which shell you are running. (Check your shell with `echo $SHELL` .)

```
source local-hadoop/env-local-hadoop.bash
```

or

```
source local-hadoop/env-local-hadoop.tcsh
```

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 command should be a single line; the backslashes are continuations.)

```
javac -cp ${HADOOP_HOME}/hadoop-core-1.1.1.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 Leland machines or on Amazon EC2. To run a job on the Leland machines, use the following command:

```
hadoop --config conf jar ngram.jar Ngram 4 query1.txt input output
```

This will run your `Ngram` code with ngrams of size 4 using the file `query1.txt` as input. Note that the `--config conf` parameter is only necessary on Leland machines, because Amazon EC2's installation of Hadoop is configured for you.

Hadoop maintains its own filesystem called HDFS. Before running the above command, you'll need to copy files from the local filesystem into HDFS. The following commands create a new directory named `input` and copy the file `chunk_aa` from the local filesystem into HDFS.

```
hadoop --config conf fs -mkdir input
hadoop --config conf fs -put chunk_aa input
```

Similarly, you can either view files in the `output` directory directly from HDFS, or copy them back to the local filesystem:

```
hadoop --config conf fs -cat output/*
hadoop --config conf fs -get output
```

For more information on the `hadoop` command, see the documentation online:

- `http://hadoop.apache.org/docs/r1.1.1/commands_manual.html`

Running on Amazon EC2

Once you've got your code working for small datasets on the Leland machines, you can move your code to the Amazon EC2 cluster to run on a multi-node installation. As with the last assignment, we will send an email to your `@stanford.edu` address with login credentials for the Amazon EC2 machines for this assignment.

The Amazon EC2 machines are configured with a head node running Torque, which provides exclusive access to multiple clusters, each of which runs a Hadoop installation with 16 nodes. To schedule time on one of the clusters, use the `qsub` command:

```
qsub script.pbs
```

or

```
qsub -I
```

You'll have to write your own script for running Torque jobs on this assignment. As a starting point, you can copy `pa2.pbs` from the last assignment and modify it to suit your needs.

Run both `query1.txt` and `query2.txt` against different subsets of Wikipedia with *n*=4. The command for running on Amazon EC2 on the entire Wikipedia dataset is:

```
hadoop jar ngram.jar Ngram 4 query1.txt /wikipedia/all output
```

The Wikipedia files only exist in HDFS, because the local filesystem on Amazon EC2 is not large enough to hold them. You can reference them directly as shown above. Replacing `/all` with `/2gb`, `/4gb`, `/8gb`, or `/16gb` will get you a corresponding chunk of Wikipedia.

Hints and catches

File paths in Hadoop refer to HDFS rather than the local filesystem. You will need to manually copy files to and from the local filesystem in order to access them in Hadoop.

```
hadoop fs -put local_filename hdfs_filename
hadoop fs -get hdfs_filename local_filename
```

Note that some single-node Hadoop tutorials tell you to use `file://` URLs to access files on the local filesystem. We recommend that you NOT do this, because students in the past have found that this has caused issues for them when moving from Leland machines to Amazon EC2.

The Hadoop codebase reuses object instances by mutating them, in an attempt to reduce the amount of work required by the GC. The result of this 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()`.

Running jobs on a distributed file systems can often result in failed reads or 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 course staff. 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 `/wikipedia/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 $O(q+n/P+\log P)$ 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) with your name, SUNet ID, and an explanation of how your code works and why it is correct;
- Screen grabs of the INFO messages that result from a Hadoop run with n=4 against `/wikipedia/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, log into a Leland machine such as `corn.stanford.edu` and 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.