DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF GEORGIA

CSCI6900 Assignment 2: Naïve Bayes on Hadoop

DUE: Friday, September 18 by 11:59:59pm

Out September 4, 2015

1 IMPORTANT NOTES

You are expected to use the Microsoft Azure elastic compute platform (HDInsight) for this assignment. If you have any questions, Microsoft has significant documentation to assist you: `https://azure.microsoft.com/en-us/documentation`.

If you get stuck, email the list. **Don't go looking for existing code.** Believe me, I know where to find it and what it looks like. I won't hesitate to zero out assignments that look similar.

2 OVERVIEW

Your assignment is to implement the Naïve Bayes classification algorithm again, this time within the Hadoop framework, and execute it on a scalable platform. If you designed your first assignment well, you should be able to reuse significant chunks of your code in the requisite mappers and reducers you'll be coding.

For an overview of the algorithm itself, please refer to your first homework assignment. Nothing has changed in terms of the mechanics of the algorithm itself. As before:

- Scan through the document data, parsing the documents and generating "messages".
- Reduce the messages into word counts. Also make sure to tabulate counters for total number of unique labels, total number of unique words, number of words under a label, and number of times word w appears under a label.

- Use the model (saved to HDFS) to perform classification of a test corpus. Like last time, you only need to get 1 label correct to consider the whole document correctly classified.

3 HADOOP

3.1 LOCAL MACHINE

Installing Hadoop on your machine entails two modes of operation: "standalone" mode, where you effectively download the Hadoop .jars and dependencies and run it immediately; and "pseudo-distributed" mode, where Hadoop behaves as though it was running in a distributed environment complete with HDFS and the supporting Hadoop job daemons. The former is an excellent way to make sure your code runs correctly without NullPointerExceptions; the latter is useful for a first-pass at testing how your program will behave in a distributed environment.

See the Hadoop documentation for excellent instructions on setting up local installs of both standalone and pseudo-distributed Hadoop: `http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html`. The Stanford MMDS course also has a great tutorial for setting up Hadoop locally: `http://web.stanford.edu/class/cs246/homeworks/tutorial.pdf`. It also has excellent instructions for installing a preconfigured Hadoop virtual environment (Cloudera), if you prefer that route.

3.2 MICROSOFT AZURE

Each registered student received Azure credits. If you have not received your code, please let me know. For this assignment, you'll be deploying your Hadoop code on the Microsoft Azure elastic compute platform.

1. Go to `http://www.microsoftazurepass.com/`. Enter your the redemption code you received by email.
2. You'll be asked to log in or create a Microsoft account. Feel free to use whatever account you wish (remember your MyIDs are tied to Microsoft accounts, so you can use this if you would prefer).
3. After filling everything out, activate your account–this can take some time.

Some things to keep in mind about your codes:

- You get $100/month in Azure services.
- If you go over that amount, the default behavior is to halt all your services.
- You can change this default by attaching e.g. a credit card to your account, so any overages are billed automatically. This is up to you; just be aware of your settings.

To start up a Hadoop cluster, go to the "HDInsight" tab on your main control panel. That has all the options you'll need. Alternatively, if you are running Linux or OS X, you can install the Azure CLI (`https://azure.microsoft.com/en-us/documentation/articles/xplat-cli/`) for easy access to all Azure resources. For this assignment, we'll only need two: Storage (essentially HDFS) and HDInsight (Hadoop clusters). Therefore, any commands you issue through the Azure CLI will be either of the form "`azure storage`" or "`azure hdinsight`". Whether you use the Azure Portal (GUI) or Azure CLI (terminal) is entirely up to you.

4 DATA

For this assignment, we are using the Reuters Corpus, which is a set of news stories split into a hierarchy of categories. There are multiple class labels per document. This means that there is more than one correct answer to the question "What kind of news article is this?" For this assignment, we will ignore all class labels except for those ending in `*CAT`. This way, we'll just be classifying into the top-level nodes of the hierarchy:

- CCAT: Corporate/Industrial
- ECAT: Economics
- GCAT: Government/Social
- MCAT: Markets

There are some documents with more than one CAT label. Treat those documents as if you observed the same document once for each CAT label (that is, add to the counters for all labels ending in CAT). If you're interested, a description of the class hierarchy can be found at `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a02-orig-topics-hierarchy/rcv1.topics.hier.orig`.

The data for this assignment is at: `https://ugammd.blob.core.windows.net/rcv1/`. This data is set to be publicly readable. You'll still need to set up your own Storage container as a prerequisite for spinning up an HDInsight cluster, but you can use the `ugammd` URL as the input path for your map-reduce job. You can view the data through your browser by appending the following filenames to the URL:

```
RCV1.full_train.txt
RCV1.full_test.txt
RCV1.small_train.txt
RCV1.small_test.txt
RCV1.very_small_train.txt
RCV1.very_small_test.txt
```

[1]

These are the same datasets as before.

5 DELIVERABLES

Create a folder in your repository named `assignment2`. **Keep all your code for this assignment there; I will be using autograding scripts to check your progress, so make sure your repository is named correctly and *precisely*!**

I will reference the code in that folder for partial credit. This needs to have a commit timestamp *before* the deadline to be considered on time. You should implement the algorithm by yourself instead of using any existing machine learning toolkit.

As with the previous assignment, the only output your program should generate is in the testing phase. After training your model, scan through the testing documents, classifying each one and printing out the label you predicted along with its log probability:

```
Best Class<tab>LogProbability
```

Here, the Best Class is the class with the maximum log probability:

$$\ln(Pr(Y = y)) + \sum_{w_i} \ln(Pr(W = w_i | Y = y)) \tag{5.1}$$

Note that we are using the natural logarithm. Here's an example of the output format:

```
CCAT    -1042.8524
CCAT    -4784.8523
...
Percent correct: 9/10 = 90.0%
```
[2]

You are also required to submit a README file with answers to the following questions:

1. For parallel computing, the optimal speedup gained through parallelization is linear with respect to the number of jobs running in parallel. For example, with 5 reducers, ideally we would expect parallel computing to take 1/5 wall clock time of single machine run. However, this optimal speedup is usually not achievable. In this question, set the number of reducers (`job.setNumReduceTasks()`) in your Hadoop run to 2, 4, 6, 8, 10, and record the wall clock time. Plot a curve, where the horizontal axis is the number of reducers, and the vertical axis is the wall time. Is the wall time linear with respect to the number of reducers? Explain what you observed.

2. Compare the performance of your program when you include a Combiner during the training phase (to reduce the counts for each word before they reach the network) to the program when you leave the Combiner out (simply comment out the line in the driver file that sets the Combiner class). Is there any change in the runtime of your program? Why or why not?

6 MARKING BREAKDOWN

- Code correctness (commit messages, program output, and the code itself) **[70 points]**
- Question 1 **[15 points]**
- Question 2 **[15 points]**

7 OTHER STUFF

START EARLY. I cannot emphasize this enough. There is not a lot of actual code required (a few hundred lines total), but it can take much longer than you think to debug and fix problems you encounter. Also, HDInsight clusters on Azure take **10-15 minutes** to spin up. Essentially, everything takes awhile, so make sure you plan for it.

Most of your time will likely be spent figuring out how to use Microsoft Azure. Check out their documentation, send emails to the list, and if you're on Twitter, their support account is extremely responsive (@AzureSupport).

Your code will involve chaining together more than one map-reduce step. For example, you'll need a map step to read the training data and a reduce step to sum the word counters, but you'll need at least 1-2 more map-reduce steps (depending on your join strategy) to join your trained model with the testing data. Whether you configure and chain these jobs together in a single .java file or submit them as distinct jobs that create intermediate output files on your Storage container, is entirely up to you.

If you need a development strategy, check out William Cohen's words of advice for his CMU big data course: `http://curtis.ml.cmu.edu/w/courses/index.php/Guide_for_Happy_Hadoop_Hacking`. You can also send messages to the mailing list if you're stuck; chances are, someone else is having the same issue.