BigML Assignment 4: Phrases on Hadoop MapReduce

TAs: Abhinav Maurya, Rahul Goutam  
Autolab Admin: Yun Ni

Due: Saturday, March 07, 2015 23:59 EST via Autolab  
Late submission with 50% credit: Monday, March 09, 2015 23:59 EST via Autolab

Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld’s previous version of 10601 from 2013. The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone, and the student should be ready to reproduce their solution upon request. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment. Specifically, each assignment solution must start by answering the following questions in the report:

- Did you receive any help whatsoever from anyone in solving this assignment? Yes / No. If you answered ‘yes’, give full details: ______________ (e.g. “Jane explained to me what is asked in Question 3.4”)

- Did you give any help whatsoever to anyone in solving this assignment? Yes / No. If you answered ‘yes’, give full details: ______________ (e.g. “I pointed Joe to section 2.3 to help him with Question 2”.

Collaboration without full disclosure will be handled severely, in compliance with CMU’s Policy on Cheating and Plagiarism. As a related point, some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be (or may have been) available online, or from other people. It is explicitly
forbidden to use any such sources, or to consult people who have solved these problems
before. You must solve the homework assignments completely on your own. I will mostly
rely on your wisdom and honor to follow this rule, but if a violation is detected it will be
dealt with harshly. Collaboration with other students who are currently taking the class
is allowed, but only under the conditions stated below.

1 Important Note

As usual, you are expected to use Java for this assignment.

This assignment is worth 100 points. In this assignment, you will have to port your
Phrases code from assignment 2 to the real Hadoop environment using Hadoop APIs.

Abhinav Maurya (ahmaurya@cmu.edu) and Rahul Goutam (rgoutam@cs.cmu.edu) are
the contact TAs for this assignment. Please post clarification questions to the Piazza, and
the instructors can be reached at the following email address: 10605-Instructors@cs.cmu.edu.

2 Introduction

In this part of the assignment, you need to re-implement Phrases assignment for the Hadoop
MapReduce framework. For a description of the method, please see section 2 of assignment
2 that you submitted earlier.

2.1 Using AWS and elastic MapReduce (EMR)

We have already distributed the AWS gift code to every registered student. If you have
not got one, let us know. Here are a few hints for running the real Hadoop jobs on AWS:

2.1.1 Submitting a Jar job

Important: When setting up your job on EMR, make sure you select “Custom Jar” in
the add step option, so that the EMR will run your job in the Hadoop API mode.

2.1.2 Viewing job progress

Tutorial for viewing the jobtracker on your local machine (via proxy)[1]

UsingtheHadoopUserInterface.html
2.2 Debugging with the CMU Hadoop cluster

To help you debugging your Hadoop code, you may debug on the hadoop cluster at CMU. See the course webpage for details: [http://curtis.ml.cmu.edu/w/courses/index.php/Hadoop_cluster_information](http://curtis.ml.cmu.edu/w/courses/index.php/Hadoop_cluster_information). Another option would be setting up the Hadoop environment on your local machine and simulate the large jobs by running a single thread with a small file locally.

2.3 Additional Hadoop Tutorial

In case you want to study extra tutorials about Hadoop, your honorary TA Malcolm Greaves has kindly put together a wiki page here: [Guide for Happy Hadoop Hacking](#)

3 The Data

We are using the google books corpus. We have done some preprocessing of the data, and created two sets of data files. This is unsupervised learning, so there is no test set. The data has the following format:

```
<text>\t<decade>\t<count>
```

where text is either a bigram or a unigram, count is the number of times that text occurred in a book in the given decade. These are separated by ASCII tabspace. The smaller subsampled dataset consists of files `bigram_apple.txt` containing all bigrams that contain the word apple, and `unigram_apple.txt` containing all unigrams appearing in those bigrams. You should ignore the bigrams with a stop word inside since we are interested in the informative phrases.

Please use the bigrams/unigrams from the 1960s as your foreground corpus, and those from 1970s to 1990s as the background corpus. You will need to remove the following stopwords (unigrams or bigrams that contain any of the stopwords): String stopwords = "i,the,to,and,a,an,of,it,you,that,in,my,is,was,for";

The data appears at /afs/cs.cmu.edu/project/bigML/phrases/ as well as on S3:

s3://bigmldatasets/phrases/full/unigram
s3://bigmldatasets/phrases/full/bigram
s3://bigmldatasets/phrases/apple/unigram
s3://bigmldatasets/phrases/apple/bigram

The input data is already sharded. You do not need to upload the dataset to S3.
4 Deliverables

4.1 Steps

What you need to do in this assignment can be summarized in the following steps:

- Port the phrase finding code into Hadoop using Hadoops MapReduce API.
- Run the Hadoop API MapReduce job on AWS with the full dataset with elastic MapReduce using the Custom Jar option. It is recommended (not mandatory) that you use m3.medium instance type.
- Download the controller and syslog text files, and submit via Autolab together with the report and your source code in a tar ball.

The controller and syslog files can be downloaded from the AWS console. Simply go to the Elastic Mapreduce tab. Select your job in the list, click View details and expand Steps to see jobs and log files.

4.2 Report

Submit your implementations via AutoLab. You should implement the algorithm by yourself instead of using any existing machine learning toolkit. You should upload your code (including all your function files) along with a report, which should solve the following questions:

1. What are the top 20 phrases (sorted by total score) from the full data set only (again, with the 1960s as the foreground corpus this time). (5 points)
2. Write down the test statistic for phraseness and informativeness based on Binomial Likelihood Ratio Test (BLRT). (5 points)
3. Using the test statistic from previous question, write down the hypothesis test (i.e. a condition involving the test statistic) for determining phraseness and informativeness. (5 points)
4. Provide a toy example where BLRT-based informativeness performs much worse than pointwise KL-divergence based informativeness. (5 points)
5. An improvement to Rocchio feedback is proposed in the paper Model-based feedback in the language modeling approach to information retrieval by Zhai and Lafferty. In particular, consider the KL-divergence based estimation of the query model described in section 5:

\[ D_e(\theta, \mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{i=1}^{n} D(\theta \| \hat{\theta}_{d_i}) \]
Here, $D_e(\theta, \mathcal{F})$ is the empirical KL-divergence between query model $\theta$ and the set of feedback documents $\mathcal{F} = \{d_1, d_2, ..., d_n\}$ (a small set of relevant documents specified by a human or retrieved automatically as in the case of pseudo-relevance feedback). For simplicity, consider the query model $\theta$ to be a multinomial distribution over words. $\hat{\theta}_{d_i}$ is the empirical word distribution of document $d_i$. The minimizer $\hat{\theta} = \arg\min \theta D_e(\theta, \mathcal{F})$ is given by:

$$p(w|\hat{\theta}) \propto \exp\left\{ \frac{1}{|\mathcal{F}|} \sum_i \log p(w|\hat{\theta}_{d_i}) \right\}$$

where $p(w|\psi)$ denotes the probability of drawing word $w$ from a multinomial distribution $\psi$. Explain how you would calculate $p(w|\theta)$ from $p(w|\hat{\theta}_{d_i})$ using only map and reduce steps. (5 points)

6. Since the above task can be performed using only map and reduce steps, do you think it is a good candidate for a Hadoop implementation? Why or why not? (5 points)

7. Under what precise conditions on a reduce function can it also be used as a combiner? (5 points)

8. Explain how you can use org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl class to run multiple Hadoop MapReduce jobs with dependencies between them. Here dependency between a pair of jobs means one job cannot start until the other has finished. As an example for your explanation, consider MapReduce-based Naive Bayes training and testing to be the pair of dependent jobs. (5 points)

4.3 Autolab Implementation Details

You must have a main Hadoop function named run_hadoop_phrase.java. To let Autolab run your job, we specify the following function signature:

```
java -cp ../bin/hadoop/hadoop-core-1.0.1.jar:../bin/hadoop/lib/*:. \nrun_hadoop_phrase unigram.txt bigram.txt output/aggregated/ \noutput/sizecount output/combined output/final
```

Here, the first argument is the provided unigram input file, the second argument is the provided bigram input file, the third argument is the output path for your aggregation MapReduce job, the fourth argument is the output path for your size counts, the fifth argument is the output path for the combined messages and aggregated counts, and the final argument is the final output path that generates the phrase finding results. Note that your main function should expect the above arguments (not throwing exceptions and your MapReduce jobs should proceed smoothly), even if you choose a different pipeline and not using some of the above arguments. Here is the recommended pipeline that includes four MapReduce jobs for this assignment:
• Aggregate.java: this is the class where you take the given unigram and bigram counts and run a MapReduce job to aggregate the counts for the foreground and background corpora.

• CountSize.java: this is the class where you take the aggregated results to run a MapReduce job to derive the various denominators for you to use in the final stage KL calculation.

• MessageUnigram.java: this is the class where you take the aggregated results to run a MapReduce job to combine unigrams, bigrams, and messages, before you send to the final job to compute the KL divergence results.

• Compute.java: this is the final step where you take the output from last step as input, and calculate the KL divergence results to generate the final phrase and scores table.

The final output must follow the tab-separated format defined in homework 2. However, instead of showing just the top-20 score-sorted phrase list with their scores, you need to output all the keys and their three scores. To make your life easier, you do not need to write another MapReduce program to sort the Hadoop key-sorted results into a score-sorted output. You should tar the following items into hw4.tar and submit via Autolab:

• run_hadoop_phrase.java

• Aggregate.java

• CountSize.java

• MessageUnigram.java

• Compute.java

• controller.txt

• syslog.txt

• any other auxiliary functions you have written

• report.pdf

Tar the files directly using "tar -cvf hw4.tar *.java *.txt report.pdf". Do NOT put the above files in a folder and then tar the folder. You do not need to upload the saved temporary files.
5 Grading

If you are able to successfully run the job on full dataset with AWS EMR and included your controller and syslog files in the submission, you will receive 20 points. We will test your Hadoop code on Autolab in real time, and check the correctness of your output (40 points). The report will be graded manually (40 points).