10605 BigML Assignment 4(c): Phrase Finding with Hadoop API

Due: Friday, Mar. 7, 2014 23:59 EST via Autolab

Late submission with 50% credit: Sunday, Mar. 9, 2014 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 part of the assignment is worth 100 points. Similar to part (b), in this assignment, you will have to port your phrase finding code to the real Hadoop environment using Hadoop APIs.

Siddharth Varia (varias@cs.cmu.edu) is the contact TA 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 phrase finding using the Hadoop MapReduce framework.

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: unlike homework 4a, in this part c, 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]. (You can also ssh into the machine using the command line interface, and then use the lynx commands in the login preamble to view the job tracker.)

[1] `http://docs.amazonwebservices.com/ElasticMapReduce/latest/DeveloperGuide/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`. 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 and AWS Tutorial

In case you want to study extra tutorials about Hadoop, your honorary TA Malcolm Greaves has kindly put together a wiki page here: `http://curtis.ml.cmu.edu/w/courses/index.php/Guide_for_Happy_Hadoop_Hacking` In addition to this, your honorary TA has also given a recitation on AWS/EC2, and can be downloaded at the folloing URL: `/afs/cs.cmu.edu/project/bigML/ec2-streaming-demo.MTS`

3 The Data

We are using the google books corpus. We've 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, and count is the number of times that text occurred in a book in the given decade.

The dataset is the same one you use in the previous phrase finding assignment, but with the following notable differences:

- In the interest of freshness, this time, use the bigrams/unigrams from **the 1960s as your foreground corpus**, and those from **70s to 90s as the background corpus**.
- You will need to remove the following stopwords (unigrams or bigrams that contain any of the stopwords) in the **Aggregate.java** MapReduce class by yourself:

  ```
  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
```

Files with the word *apple* in the name are the subsampled datasets. 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 Hadoop's MapReduce API.
- Run the Hadoop API MapReduce job on AWS with the **full dataset** with elastic MapReduce using the Custom Jar option.
- Download the controller and syslog text files, and submit via Autolab together with the report and your source code in a tar ball.

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:

Consider the lecture of Feb 3.

1. In the lecture, we argued that

   $$VC(C) \leq M_{opt}(C) \leq \log_2(|C|)$$

   where C is any finite set of hypotheses, $M_{opt}(C)$ is the mistake bound of the optimal algorithm, VC is Vapnik-Chernonenkis dimension, and $|C|$ is cardinality of C (the number of $c \in C$).

   Let $M_A(C)$ be the mistake bound of a particular learning algorithm (say, perceptron) when the example sequence is constrained to be labeled consistently with some concept $c \in C$. Does this inequality hold?

   $$VC(C) \leq M_A(C)$$

   Why or why not?

2. Consider the analysis in lecture of Feb 3, and Theorem 3 from the paper by McDonald, Hall, and Mann discussed in the lecture of Feb 10. Assume a dataset where $||x_i||_2^2 < R^2$ labeled with a classifier of margin γ.

Let m_p be the *total* number of mistakes made, using uniform mixing, for a group of fifty perceptrons trained with PerceptronIterParamMix until convergence. Let m_s be the *total* number of mistakes made, using uniform mixing, for a single perceptrons trained until convergence.

Which of these statements must be true, according to Theorem 3?

(a) $m_s \leq m_p$.

(b) $m_p \leq m_s$.

(c) $m_s \leq 50m_p$.

(d) $m_p \leq 50m_s$.

(e) $m_p \leq 50b$, where b is an upper bound on m_s.

3. Below are some decompositions of a matrix into blocks (the X's indicate the blocks that are included in a strata). Which decompositions are "valid" (ie satisfy the requirements for the Gemulla et al DSGD algorithm), and why?

(a)

-	-	-	-	X
-	-	-	X	-
-	-	X	-	-
-	X	-	-	-
X	-	-	-	-

(b)

X	-	-	-	X
-	X	-	X	-
-	-	X	-	-
-	X	-	X	-
X	-	-	-	X

(c)

X	-	-	-	-
-	-	-	X	-
-	-	X	-	-
-	X	-	-	-
-	-	-	-	X

(d)

X	-	-	-	-
-	X	-	-	-
-	-	X	-	-
-	-	-	-	-
-	-	-	-	X

4. Modify this decomposition below by adding as *many* blocks as you can, while keeping it valid.

X	-	-	-	-
-	X	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	X

5. Answer the questions in the collaboration policy on page 1.

The controller and syslog files **from AWS** for the mapreduce job 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.

4.3 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/*:. \
run_hadoop_phrase unigram.txt bigram.txt output/aggregated/ \
output/sizecount output/unigrammessage 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 you 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[2].
- **Count_Size.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.
- **Message_Unigram.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.

[2]In this MapReduce class, don't forget to remove the stopwords listed in the Data section. Also, the definition of foreground and background corpora was different than homework 3.

- **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 3. 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 **hw4c.tar** and submit to the homework 4c assignment via Autolab:

- run_hadoop_phrase.java
- Aggregate.java
- Count_Size.java
- Message_Unigram.java
- Compute.java
- controller.txt
- syslog.txt
- and all other auxiliary functions you have written
- report.pdf

Tar the files directly using "tar -cvf hw4c.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 Submission

You must submit your homework through Autolab. In this part of homework 4, there will be a validation link.

- Homework4c-validation: You will be notified by Autolab if you can successfully finish your job on the Autolab virtual machines. Note that this is not the place you should debug or develop your **algorithm**. All development should be done on linux.andrew.cmu.edu machines for standalone Hadoop, and the GHC cluster for real Hadoop tests. This is basically a Autolab debug mode. There will be **NO** feedback on your **performance** in this mode. You have unlimited amount of submissions here.

To avoid Autolab queues on the submission day, the validation link will be closed 24 hours prior to the official deadline. If you have received a score of 1000 with no errors, this means that you code has passed the validation.

- Homework4c: This is where you should submit your tar ball. You have a total of **5 possible submissions**. Your score will be reported, and feedback will be provided immediately.

6 Grading

If you are able to successfully run the job on full dataset with AWS EMR, you will receive 20 points. The successful run of your job should be reflected in your log files. We will test your Hadoop code on Autolab in real time, and check the logs (30 points) and the correctness of your output (30 points). The report will be graded manually (20 points).