MapReduce on Amazon Web Services*

Due: Wednesday, November 16, 2011
Turn in at the start of class.

Overview

In this assignment you will learn to some basics of using MapReduce, and you will get some hands-on experience using Amazon's Web Services. This assignment will be done in groups of 1 or 2.

The assignment will be based on a report. All that is needed in the report is the answer to a set of questions below. For each question provide the (short) answer, followed by a brief explanation of how you accomplished this. I do not need elaborate essays on the existential nature of MapReduce or AWS.

There will also be an opportunity for a *bonus challenge*. There is an open ended question that I don't know the answer to. The higher your answer scores (or the more interesting your documented techniques) the more bonus you get.

1 Set Up

First arrange your account.

- Sign-up for an account at Amazon Web Services : `http://aws.amazon.com/` . Register for at least S3, EC2, and ElasticMapReduce.
- Contact the TA (Bigyan) if you have not done so to get a certificate for AWS credits. Go to `http://aws.amazon.com/awscredits` to redeem the credits.

To begin with Amazon Web Services, set up a bucket in S3. This is the storage application. You can actually store lots of data here if you want.

- Create a bucket, choose a name carefully. Keep in mind the top-level bucket needs to be unique, kind of like a username.
- Upload some document into the bucket. This is where you can store your input code and output data. Choose your naming of folders carefully as it is a bit tedious to change or reorganize stuff.

Next, make sure you can use EC2. This is how you can manually launch instances, but more importantly, how you can monitor and terminate your instances. Whenever you use AWS and are not letting a big job run, check to make sure you don't have any extra instances running when you log off.

- Launch an Amazon EC2 instance.
- Log into it with ssh. Do a little dance.
- Terminate the instance.

Now, use Elastic MapReduce. This is going to be the key application you will be running for this project.

- Launch an Elastic MapReduce instance. Follow the instructions to run the *sample application* word count.

- Note that more information can be found here: `http://aws.amazon.com/articles/code/Elastic-MapReduce/2273`
- This will take a while to get started (maybe several minutes), and then will dump some output in your main bucket.

Note that the output will not be a single file, but rather split into several files. This is actually convenient for MapReduce storage and should not pose much difficulty for MapReduce inputs (the input can be a folder with several files, and this works the same as one file)

2 Word Count on NSF Abstract Data

The first task is to run Elastic MapReduce, using:

- Mapper : `s3://elasticmapreduce/samples/wordcount/wordSplitter.py`
- Reducer : aggregate
- Input : `s3://cs7960/NSFAbstracts/*`

This is a much larger job than the sample input (it contains $129,000$ files, each with about 250 words) and you are recommended to use more than 2 instances. Run this experiment a few times and see how the total *time* and *cost* vary based on the number of instances.

Question 1: How many instances did you find had the best time/cost tradeoff?

Hold onto this output as it will serve as the baseline for what comes next.

3 Anomaly-Based Clustering on NSF Abstracts

We are now going to build a model for how *likely* a particular abstract is (or a set of abstracts), using the full set as a baseline.

Notation. Let $\mathcal{A}$ be the set of all abstracts, and we will use $a \in \mathcal{A}$ to denote a particular abstract. Also $A \subset \mathcal{A}$ will refer to a subset of abstracts.

After running word count, we have established a corpus of words $\mathcal{W}$ that appear in the abstracts. Let each $w \in \mathcal{W}$ represent some word that is in at least one abstract in $\mathcal{A}$.

Let $f_w(\mathcal{A})$ be the output value of running word count (as above) associated with the word $w \in \mathcal{W}$. It measures the number of times that the word w is in the full corpus $\mathcal{A}$. For simplicity, we often use f_w (without the implicit argument $\mathcal{A}$) to refer to the count on the full set.

Similarly, let $f_w(a)$ refer to the count of word w in the abstract a. Finally, given a set $A \subset \mathcal{A}$ (a subset of all abstracts), let $f_w(A) = \sum_{a \in A} f_w(a)$ represent the number of instances of word w over all abstracts in A.

Let $m = \sum_{w \in \mathcal{W}} f_w$ represent the total number of words in the entire corpus, and let $m(A) = \sum_{w \in \mathcal{W}} f_w(A)$ represent the number of words in the set of abstracts $A \subset \mathcal{A}$. We let $\mathcal{W}(A)$ be the set of all words that appear in the word count on just the subset of abstracts A; thus we can equivalently define $m(A) = \sum_{w \in \mathcal{W}(A)} f_w(A)$.

Anomaly Index. Now we will define the *anomaly index* of an abstract or set of abstracts, this assigns a score ρ to each set of abstracts. We define $\rho(A)$ as follows:

$$\rho(A) = \sum_{w \in \mathcal{W}(A)} \left(\frac{f_w(A)}{m(A)} \right) \ln \left(\frac{f_w(A)/m(A)}{f_w/m} \right).$$

Note that this also applies to a single abstract a when $A = \{a\}$ is a set of size 1. This definition is based on the Poisson log-likelihood ratio of each word in the subset of abstracts under consideration.

3.1 Tasks Using the Anomaly Index

Single abstract anomaly index. Use MapReduce to compute the anomaly index for all individual abstracts. That is, for each $a \in \mathcal{A}$, compute $\rho(\{a\})$. You should use the output of your initial word count as part of your input here; although, you may wish to modify it first so it is easier to use as input to a different Mapper function that you will need to write.

Question 2: What are the 5 abstracts with the top anomaly index (have largest $\rho(\{a\})$ value)?

You may find it useful to write a simple top-k MapReduce function that takes your output of key-value pairs and outputs the top k key-value pairs sorted by the value.

Double abstract anomaly index. Use MapReduce to compute the anomaly index of all pairs of abstracts. That is, for any two different abstracts $a_i, a_j \in \mathcal{A}$ let $A_{i,j} = \{a_i, a_j\}$. Then for each $A_{i,j}$ compute $\rho(A_{i,j})$.

Question 3: What are the 5 pairs of abstracts with the top anomaly index (have largest $\rho(A_{i,j})$ value)?

Full credit will be given to computing the largest $\rho(A_{i,j})$ values for all pairs of abstracts in folder `cs7960/NSFAbstracts/awd_1990_00/`, but if you can find largest $\rho(A_{i,j})$ values for all pairs of abstracts, you will be given bonus points.

Arbitrary subset anomaly index. You will now write a MapReduce program that can compute the anomaly index for any subset $A \subset \mathcal{A}$. Your program will take as input a file with key-value pairs with the key as the name of each file corresponding to an abstract in A, the value can be ignored and will always be 1. Given such an input corresponding to an arbitrary subset $A \subset \mathcal{A}$, you must computer $\rho(A)$.

Question 4: What is the value $\rho(A)$ for input : `cs7960/input` ?

3.2 Bonus: Finding an Anomalous Subset

Finding subsets which score high on the anomaly index is a challenging optimization problem. It corresponds to clusters of abstracts that, based on word count, are all different from an "average abstract," and are all different in the same way. Findings clusters (or related groupings of documents) is a challenging problem that many companies face when mining their data. Your bonus task is to try to find a subset $A \subset \mathcal{A}$ of abstracts with the largest anomaly index $\rho(A)$ as possible.

This is not required for the assignment. But you can earn bonus points for attempting this challenge. If you did poorly on an early assignment, you can make it up with bonus points here.

To participate in this challenge, create an output folder that contains your large-anomaly-index cluster. The format should be the same as the input for question 4. Send the subset to the TA, he will verify the score, and maintain a leaderboard. Your team may send no more than one subset per 24 hour period.

To gain credit for this bonus activity, make sure you describe your process for finding your cluster in the report.

4 Hints

Here are a few hints or tips for navigating AWS and MapReduce:

- You may find this tutorial on debugging MapReduce jobs on AWS helpful:
  `http://aws.amazon.com/articles/code/3938`
- To transfer files to and from AWS, you can store them in S3. If for some reason you need to move a large number of individual files, it may be tedious (I was unable to use their Enhanced Uploader (BETA)). I found some success with Bucket Explorer which you can download and use as a free trial version for 30 days.

Please also monitor the discussions on Canvas. If you post information on there that will appear in the hints for next time I offer this class, you will receive bonus points.