15-719 / 18-847B, Fall 2013, Project 1: Analyzing the popularity of Wikipedia articles

Assigned: September 23rd, 2013

Due: October 11h, 2013, 5:00PM

For this project, you will develop a MapReduce application for analyzing the popularity of Wikipedia articles. Specifically, you will build an infrastructure that ranks English Wikipedia pages according to total page views and popularity trend. Your end result will have similar functionality to the Trending Topics web site.

Your application should use the AWS Elastic MapReduce service. It should not use Hadoop streaming; rather, your application should be written within the MapReduce framework and in Java. AWS charges can accumulate rapidly, so please **shut down** your AWS instances when you're not using them. You should aim to spend no more than $30 of your AWS credit for this project.

1 Collaboration & cheating policy

This project should be completed individually. Teamwork is prohibited. Do not share code or pseudo-code with others.

Cheating will not be tolerated. You are prohibited from using any solutions or code you find online, especially from the Trending Topics website or their Github repository. Please check the official 15-719 cheating policy for additional details.

2 Dataset

The Wikimedia dataset repository contains statistics about all articles published by Wikimedia, including Wikipedia, Wiktionary, and so on. Details about this repository can be found at `http://dumps.wikimedia.org/other/pagecounts-raw/`. We have uploaded days 1-15 of the June 2013 dataset to *s3://15719f13wikitraffic/*. You can use the AWS client interface or the S3 management console to browse this location.

The dataset contains one file for each hour of the 15-day-long period. Each line of each file contains four fields: `projectcode`, `pagename`, `pageviews`, and `bytes`. Many items in the `pagename` field are percent-encoded.

3 Step 1: Filter, transform, & aggregate the input dataset

You should write a MapReduce job to perform this step. Your MapReduce job should filter and transform the input dataset according to the specifications listed below. It should then aggregate the filtered/transformed data by date and article name. Each line of the output should look like:

```
Barack_Obama}20130601 143
```

Where `Barack_Obama` is the article title, `}` is a separator, `20130601` is the date, and `143` is the number of page views.

3.1 Specifications for filtering and transforming the input dataset

1. Transform `pagename` entries to article names by replacing any `%22`s with underscores.
2. Exclude pages outside of English Wikipedia by filtering out any items whose `project` field does not begin with `en`. `Project` fields that begin with `en` should be included only if they have no suffix.
3. Exclude pages that do not need to be considered when finding trending topics. Exclude any pages whose title starts with `Media`, `Special`, `Talk`, `User`, `User_talk`, `Project`, `Project_talk`, `File`,`File_talk`, `MediaWiki`, `MediaWiki_talk`, `Template`, `Template_talk`, `Help`, `Help_talk`, `Category`, `Category_talk`, `Portal`, `Wikipedia`, or `Wikipedia_talk`.
4. Wikipedia policy states that all English articles must start with an uppercase character. So, filter out articles that start with lowercase English characters. You may notice that some articles have non-English titles. You should not filter out these articles.
5. Exclude any article that ends with an image or text-file extension (`jpg`, `gif`, `.png`, `.JPG`, `.GIF`, `.PNG`, `ico`, and `txt`).
6. Exclude boilerplate pages (`404_error`, `Main_Page`, `Hypertext_Transfer_Protocol`, `Favicon.ico`, and `Search`).

4 Step 2: Create a time series of article names & page views

Write a second MapReduce job that takes the output of the first MapReduce as input and, for each article, outputs a time series of page views, total page views, and popularity trend. Your MapReduce job should exclude any article that has less than 10 total page views from the results. Here's an example of what one line of the output might look like:

```
Barack_Obama\t[20130601,20130609]\t[143,129]\t272\t-14
```

In this example, `Barack_Obama` is the article name. The dates in the first set of brackets are dates when the corresponding page was accessed. The dates in the second set of brackets are the page views for each date. The next number is the total sum of page views. The final number is the article's popularity

trend, which is calculated as the sum of page views during days 8-15 of the month minus the sum of page views during days 1-7 of the month.

5 Step 3: Import the data into Hive

Load the time series data from the previous step into Hive so that it can be easily queried. The choice of schema is up to you.

6 Testing your code

During development, please test your code using micro instances or small instances. To limit processing time, test your code during development using "toy" datasets made up of a few hours of the provided dataset.

When you believe your code works and are ready to test on the entire 15-day period (June 1st-June 15th, 2013), use five c1 medium instances for the first MapReduce job and three c1 medum instances for the second. You should try to minimize the number of times you run using this configuration.

7 Deliverables & grading

There are three graded deliverables for this project. Each is worth a different number of points. All of your deliverables should be obtained by running your project code on days 1-15 of the June 2013 dataset provided in *s3://15719f13wikitraffic*. When creating your deliverables, please adhere EXACTLY to the formats specified below. The first two deliverables will be graded on a binary basis. For the third deliverable, 15 points will be awarded for each search term for which your code generates correct results.

When you are ready to hand in your graded deliverables, please create a tarball of them (the extension must be .tar.gz) and hand them in to *s3://15-719-F13_project_1_handin*. Please name your tarball: `<your_andrew_user_id>_p1.tar.gz`.

Please also hand in a tarball of your source code to *s3://15-719-F13_project_1_code_handin*. Please name your tarball: `<your_andrew_user_id>_p1_source.tar.gz`. To identify instances of cheating, we will use an automated program to check your code for excessive similarities with other students' code.

7.1 Graded deliverables

20 points Turn in a text file with a ranked list of the 100 most popular Wikipedia articles as determined by number of total page views. Items should be ranked in descending order. Each line of the text file should contain two fields—the article name and number of total page views—in the following format:

```
<article_name>\t<page_views>\n
```

For example, one line of your results might read: `Barack_Obama\t272\n`
Please name your file as per the following format:

`<your_andrew_userid>_top_100_pageviews.txt`

20 points Repeat the above exercise for the 100 highest-ranked Wikipedia articles as determined by popularity trend. For this part, one line of your results might read:

`Barack_Obama\t-19\n`

Please name your file as per the following format:

`<your_andrew_userid>_top_100_trending.txt>`

60 points We will send you a list of four search terms (either via S3 or e-mail). For each search term, hand in a text file that contains a ranked list of articles that contain the search term. Articles in the list should be ranked by number of page views. Each line of the list should contain five fields: the article name, dates on which the article was accessed, numbero f page views on those days, the total number of page views, and the popularity trend score. Use the following format for each line of the output:

`<article_name>\t<time_series_data>\t<page_views>\t<popularity_trend>\n`

For example, one line of your results might read:

`Barack_Obama\t[20130601,20130609]\t[143,129]\t272\t-14\n`

Each of the four text files you will hand in for this deliverable should be named as per the following format:

`<your_andrew_userid>_<search_term>._pageviews.txt`

Acknowledgements

The steps for this project were obtained from a tutorial on how to create the Trending Topics website.