CS123 Wikipedia Project: Writeup

Julian McClellan, Bobby Adusumilli, Andy Zhu

May 30, 2016

Contents

Data Files

wikidata/wikistats (650G Compressed | 2.5TB Uncompressed)

Contains hourly Wikipedia article traffic statistics dataset covering 16 month period from October 01 2008 to February 6, 2010, from raw anonymous logs provided by Domas Mituzas at http://dammit.lt/2007/12/10/wikipedia-page-counters/

Each log file is named with the date and time of collection and there is one file per hour:

pagecounts-20090430-230000.gz

Each line has 4 fields: projectcode, pagename, pageviews, bytes

```
en Barack_Obama 997 123091092
en Barack_Obama%27s_first_100_days 8 850127
en Barack_Obama,_Jr 1 144103
en Barack_Obama,_Sr. 37 938821
en Barack_Obama_%22HOPE%22_poster 4 81005
en Barack_Obama_%22Hope%22_poster 5 102081
```

wikidata/wikilinks (1.1G)

Contains a wikipedia linkgraph dataset provided by Henry Haselgrove. These files contain all links between *proper english language Wikipedia pages*, that is pages in "namespace 0". This includes disambiguation pages and redirect pages.

In links-simple-sorted.txt, there is one line for each page that has links from it. The format of the lines is:

```
from1: to11 to12 to13 ...
from2: to21 to22 to23 ...
...
```

where from1 is an integer labeling a page that has links from it, and to11 to12 to13 ... are integers labeling all the pages that the page links to. To find the page title that corresponds to integer n, just look up the n-th line in the file *titles-sorted.txt*.

File summary:

pagecounts-.gz
* Contains n lines with 4 fields: projectcode (we want "en" only!), pagename, pageviews, and bytes
* Dates range from 10/01/2008 to 02/06/2010

links-simple-sorted.txt
* Contains all links between proper English language wikipedia pages.

titles-sorted.txt
* Contains the names of all English Language Wikipedia pages.

How we found our Data.

We searched through the public data sets available on Amazon in search of a dataset whose information was contained in a relatively clean format. Each line of the pagecounts data is relatively easy to interpret and along with the data on the links all of our files *appeared* relatively easy to parse in a systematic way.

Initial Hypothesis

We will attempt to quantify the relationship between pages and inlinks to these pages.

Imagine the simplest case. A page only has one inlink, IL. We propose a linear model for the traffic of that page as:

$traf\_page_t = \beta_0 + \beta_1 traf\_IL_t + \beta_2 bratio\_IL_t + \beta_3 bytes\_page_t + \epsilon_t$

Where $bratio\_IL_t = \frac{bytes\_IL_t}{bytes\_page_t}$. That is, the bytes ratio is the ratio of the page size in bytes, of the inlink IL to the page.

We would be most interested in the coefficients β_1 and β_2. Respectively, these can be interpreted as the change in page traffic when the inlink traffic increases by one unit holding the bytes ratio constant (β_1) and vice versa (β_2).

However, we do not simply believe that the traffic of a page is a function of the attributes related to the link (or links) to that page. Given the low dimensions of our data set we use the predictor bytes_page, the size of the page in bytes as an "internal" cause of the traffic to that page. Bytes can be thought of as a direct way to measure the amount of content the page contains.

Once the actual data has been more thoroughly explored, we might be inclined to introduce an interaction term between the bytes ratio and inlink traffic. Doing so complicates a simple explanation of what the regression coefficients mean, but suffice to say, a statistically significant interaction term means that the effect on page traffic that the traffic to the inlink IL has depends on the value of the bytes ratio.

Now, to generalize the model, say a page has i inlinks ($2 \leq i < \infty$), then the model generalizes to:

$traf\ page_t = \beta_0 + \beta_1 traf\_1 + ... + \beta_i(traf\_i) + \beta_{i+1} bratio\_1 + ... + \beta_{2i} bratio\_i + \beta_{2i+1} bytes\_page + \epsilon_t$

In this case, we would be interested in all β's with subscripts greater than 0. With this linear regression we can test a number of hypotheses:

Does the traffic from each inlink affect the page traffic in the same way?

Null | $\beta_1 = \beta_2 = ... = \beta_i$

Do the bytes ratios have any effect on page traffic?

Null | $\beta_{i+1} = \beta_{i+2} = ... = \beta_{2i} = 0$

Do the bytes ratios all have the same effect?

Null | $\beta_{i+1} = \beta_{i+2} = ... = \beta_{2i}$

etc.

Procedure

Initial Setup and Data Filtering

Before the data could even be used, it had to be moved from the read only volume it originated from on Amazon.

Over the course of a week or so, the pagecounts data was copied to a new volume with more permissions and then decompressed and uploaded (in one step) to an S3 bucket to facilitate mrjob use. This

However, within the 2.5TB of uncompressed data in the S3 bucket there was a lot of unnecessary information. The data contained within wikidata/wikilinks, which was necessary to test our hypothesis, pertained only to English language Wikipedia pages, so the first step was to scrub the pagecounts files clean of anything else. This process, which took about a day, involved creating 20 instances, each utilizing the s3fs library to mount the S3 bucket onto these instances and running a sed script on each machine to clean data pertaining to a certain hour of the data, and then mopping up the remaining hours with 4 of the 20 instances. This semi-automated process reduced the uncompressed size of the data from 2.5TB to 660GB uncompressed.

Our data spanned 16 months, but given our time constraints level of proficiency with methods like mrjob we opted to focus on a smaller subset of data: one out of the 16 months provided to us

This initial setup and exploration of the data was done in an ad-hoc fashion while other group members focused on gaining a familiarity with mrjob and refining our initial rough hypothesis.

Reconfiguration, Exploration, and Further Filtering of the Data

Even with our data free of all non-English entries, there were still a number of entries within the data concerning Wikipedia pages we had no interest in. For example, Wikipedia images do not link to other pages, so we decided to remove these entries like these from our raw data. Through our mapreduce_step1.py ("Step 1") file, we remove all pagenames that cannot have inlinks associated with them. Removing these entries helped to decrease our raw English-only data by roughly 25%.

Additionally, some of the pagenames in the raw data contain percent encoding, meaning we had to remove this percent encoding in "Step 1". Finally, since a pagename could theoretically have an entry for every hour in the dataset, we decided to include the datetime in each line of the data, in order to make further processing of the data easier.

Thus, "Step 1" would take a line of data of the form "language pagename pageviews bytes" and yield a line of data of the form "pagename datetime pageviews bytes". To run "Step 1" for all of the raw data in October 2008, it took roughly 12 hours for 14 instances, resulting in 42.15G of data.

When we have a specific pagename or group of pagenames and associated inlinks of which we want the corresponding hourly information over the monthly dataset, we use this "Step 1" output in our mapreduce_step2_no_json.py ("Step 2") to yield only those lines pertaining to the pages of interest and their corresponding inlinks.

Producing Data to Test Our Hypothesis

Ideally, if we wanted to test our hypothesis on "Page A" over a period of time, we would want to be able to record a single observation for every hour, with each observation containing the traffic and bytes information for Page A *and* it's inlinks. Unfortunately our data was only a sample of traffic, so we weren't guaranteed to have information about Page A for any particular hour. (Indeed, in the initial filtering of the non-English entries there were entire files containing no English entries whatsoever!) Given this fact and our one month data constraint (744 hours), testing our hypothesis on any single page with a large number of inlinks was not feasible, as the higher the number of inlinks a page had, the fewer "complete" observations we would be able to skim from the data (probabilistically speaking), and combined with the higher number of predictor variables that comes with more inlinks, this would have decreased the chances that we could run a valid regression. Accordingly, we chose to focus on pages with anywhere from 1-5 inlinks.

Still, according to our link data, there were still ~700,000 or so pages with 1-5 inlinks. We didn't exactly have any criteria for selecting any subsets of pages out the 700,000 available, so we opted to select small numbers of pages with the same number of inlinks. E.g., we would select anywhere from 5-20 pages, each with the same number of corresponding inlinks and then run our "Step 2" mrjob to parse our 744 hours of "Step 1" data and output information pertaining to these pages and their inlinks. This information then was passed through a Python script that created CSV files that were read into dataframes for manipulation in R. As mentioned previously, it was impossible for us to gather a full 744 observations (an observation being the combined individual observations of a page and all of its inlinks), but most of the time we were able to gather enough observations to run a linear regression.

Our initial hypothesis placed a lot of emphasis on comparing the effects of traffic and bytes ratios for all the inlinks of a page. However, upon running a full regression of a page's traffic on all of the predictor variables, it turned out (not unexpectedly) that a lot of our coefficient estimates for our predictor variables were not significant. We decided then to use an iterative AIC (Akaike information criterion) method in order to find the best parsimonious model. This iterative work was done automatically using a function in R, but the general idea behind the iterative AIC method is too continuously remove predictor variables from the model if they lowered the AIC of the model (i.e. they made the model "better") until the AIC could not be lowered any further.

The resulting models were much more likely to have significant predictor variables, and indeed, one could interpret the remaining predictor variables as the inlinks to a page that had the strongest effects on the hourly traffic of the page.

Let's take a look at the regression models concerning a specific page and its inlinks, starting with the full model, then a reduced model obtained with an AIC method, and ending with some discussion on the results of the reduced model.

Note that the following analysis could be reproduced for any page someone could be interested in.

Results: Case Study of the page on the compilation rap album *Nigga Deep* (1998)

Our page had 5 inlinks, so we begin with a full model that has a total of 11 predictor variables, 5 (bytes ratios for inlinks) + 5 (traffic for inlinks) + 1 (bytes of our main page).

The inlinks correspond either to Brotha Lynch, SicX (The two artists featured on the album), as well as Dead $ Life, If These Walls Could Talk, and Loaded, other rap albums by either Brotha Lynch Hug or Sicx.

```
# Let's view our starting model
summary(nigga.lm)
```

```
##
## Call:
## lm(formula = X.traf_Nigga_Deep. ~ ., data = nigga)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.627 -0.502 -0.184  0.417  3.760
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                          5.77e-01   2.48e-01    2.33
## X.bytes_Nigga_Deep.                  9.80e-05   8.31e-06   11.79
## bratio_Brotha_Lynch_Hung            -5.82e-05   5.41e-03   -0.01
## bratio_Dead_4_Life                  -2.63e-01   1.11e-01   -2.38
## bratio_If_These_Walls_Could_Talk_.album.  1.33e-01   1.45e-01    0.92
## bratio_Loaded_.Brotha_Lynch_Hung_album.   6.81e-02   1.28e-01    0.53
## bratio_Sicx                          8.63e-03   2.64e-02    0.33
## traf_Brotha_Lynch_Hung              -7.45e-03   8.02e-03   -0.93
## traf_Dead_4_Life                     2.15e-01   8.72e-02    2.46
## traf_If_These_Walls_Could_Talk_.album.   -6.17e-02   1.00e-01   -0.62
## traf_Loaded_.Brotha_Lynch_Hung_album.     1.01e-01   6.36e-02    1.58
## traf_Sicx                            1.47e-03   2.84e-02    0.05
##                                     Pr(>|t|)
## (Intercept)                            0.022
## X.bytes_Nigga_Deep.                   <2e-16
## bratio_Brotha_Lynch_Hung               0.991
## bratio_Dead_4_Life                     0.019
## bratio_If_These_Walls_Could_Talk_.album.    0.361
## bratio_Loaded_.Brotha_Lynch_Hung_album.     0.595
## bratio_Sicx                            0.744
## traf_Brotha_Lynch_Hung                 0.355
## traf_Dead_4_Life                       0.015
```

```
## traf_If_These_Walls_Could_Talk_.album.      0.539
## traf_Loaded_.Brotha_Lynch_Hung_album.       0.116
## traf_Sicx                                   0.959
##
## Residual standard error: 0.859 on 120 degrees of freedom
## Multiple R-squared:  0.638,  Adjusted R-squared:  0.605
## F-statistic: 19.2 on 11 and 120 DF,  p-value: <2e-16
```

As we can see from the p-values column, most of the predictors are not significant, at least at the 5% level. We do note that our *most* significant predictor is X.byes Nigga Deep, which is the hourly measure of the size of the page in bytes. This isn't too surprising, the size of the page only increases if users add content to the page, and the more content a page has, the more enticing it is for traffic or(?) the more representative it is of how many people would like to visit the page.

Now let's try and get a simpler model with fewer, and hopefully more significant predictor variables.

```
# View the reduced model (created using stepAIC(nigga.lm))
summary(nigga.lm.aic)
```

```
##
## Call:
## lm(formula = X.traf_Nigga_Deep. ~ X.bytes_Nigga_Deep. + bratio_Dead_4_Life +
##     bratio_Loaded_.Brotha_Lynch_Hung_album. + traf_Dead_4_Life +
##     traf_Loaded_.Brotha_Lynch_Hung_album., data = nigga)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.664 -0.449 -0.215  0.433  3.759
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                          4.41e-01   1.94e-01    2.28
## X.bytes_Nigga_Deep.                  9.72e-05   8.10e-06   12.00
## bratio_Dead_4_Life                  -1.32e-01   6.54e-02   -2.02
## bratio_Loaded_.Brotha_Lynch_Hung_album.  8.83e-02   5.65e-02    1.56
## traf_Dead_4_Life                     1.44e-01   7.09e-02    2.04
## traf_Loaded_.Brotha_Lynch_Hung_album.    8.42e-02   4.71e-02    1.79
##                                     Pr(>|t|)
## (Intercept)                            0.024
## X.bytes_Nigga_Deep.                   <2e-16
## bratio_Dead_4_Life                     0.045
## bratio_Loaded_.Brotha_Lynch_Hung_album.    0.120
## traf_Dead_4_Life                       0.044
## traf_Loaded_.Brotha_Lynch_Hung_album.      0.076
##
## Residual standard error: 0.848 on 126 degrees of freedom
## Multiple R-squared:  0.629,  Adjusted R-squared:  0.614
## F-statistic: 42.7 on 5 and 126 DF,  p-value: <2e-16
```

Firstly, we notice that the number of predictor variables remaining in the model has been reduced to 5 from our original 11 predictors. The remaining predictor variables are the bytes of our original page, and the

traffic and bytes ratio variables for Dead 4 Life, an album by Sicx, and Loaded, by Brotha Lynch Hung. The p-values for Dead 4 Life are about as significant as they were in the full model, the p-value for Loaded's bytes ratio has improved from ~.6 to ~.12, and the p-value for Loaded's traffic has improved from ~.12 to ~.08.

Let's take a look at the coefficients. We expect the size of the page in bytes to move positively with traffic, so the first coefficient's sign comes as no surprise. Additionally, looking at the traffic variables, we see that their coefficient values are also positive, which also makes sense intuitively. As an inlink to the page receives more traffic, we'd expect some of that traffic to bleed into our page of interest as people have some chance of clicking the link to Nigga Deep.

Generally, we can interpret the bytes ratio coefficients as the change in the traffic for Nigga Deep as the ratio of bytes between the other album and Nigga Deep increases by 1. E.g., if the page for the album Dead 4 Life became twice as big as the page for Nigga Deep, we would expect to see a change in the traffic of Nigga Deep equal to the coefficient.

The coefficients for the bytes ratios for the two other albums appear a bit confusing. We see that the coefficients for the bytes ratios are either positive and negative (although the positive one's p-value is not significant at the 5% level). While trying to come up with a hypotheses to explain a positive or negative coefficient, we figured it might not have been prudent to utilize bytes ratio as a predictor variable, since any predictor variable that has to do with an inlink is an attempt to explain why someone on the inlink page might click the link to navigate to our page of interest. However, the bytes ratio predictor variable includes information about our page of interest (its size in bytes) that someone one the inlink page would not know, so it's not fully representative of that situation.

Ideally then, in place of bytes ratios, it would probably have been good to utilize bytes instead, along with hourly information number of links each inlink page has, in order to give a rough idea of how big the inlink page is, and how likely one might click on our page of interest given a number of other links to click on. However, we only had access to the hourly bytes observation of the page.

Closing Thoughts

Julian McClellan

They say to live life with no regrets, but time and time again while doing this project I found that after I had done something, I then thought of a way to accomplish the same thing in a way that would have saved me much time and effort. For example, the transferring of the data from the read-only volume to (eventually) the S3 bucket could have been done in a much more efficient fashion. I initially transferred the data from a read-only volume to a volume that I could control, and then unzipped it to the S3 bucket, all with only one instance. I learned later to utilize more instances in an ad-hoc approach (i.e. 20 instances running sed across the S3 Bucket).

The general model for our linear regressions could have been improved upon, but this isn't a statistics class, so I'll just leave it at that.

Additionally, throughout the project, I relied too much on utilizing mrjob and AWS with their command line interfaces or web interface (for AWS). I.e., I utilized these things mostly in the manner we learned in labs. Consequently, the flow of our work was less cohesive than it could have been. While I was creating my 20 instances to scrub the data I did get integrate some AWS features with Python (although I did this by calling command line arguments in a Python function rather than utilizing a package like boto3). There were definitely areas where we could have utilized mrjob as a nicely integrated aspect in a Python function. Of course, this would have involved some overhead in reading documentation and gaining experiences with these different methods, but in hindsight, it may have been worth the investment.

Anyways, moving forward, thanks to plenty of things I've realized about doing this project in hindsight, I think I'm in a good spot to further improve working with AWS, mrjob, and big data and experimental design

in general. I still have ~10 months left in my free AWS account and I'll spend some of my summer refining some of the procedures that my group and I used in this project.

Bobby Adusumilli

Working primarily on the MRJob functions, I went through many different iterations of both the MRJob "Step 1" and MRJob "Step 2" files. Particularly with the MRJob "Step 1" file, I went from an attempt to gunzip a file within the function (couldn't do it, but took a while to figure that out) to the current version of removing percent encoding and all pagenames that cannot have inlinks.

For the MRJob "Step 2" file, we went through several iterations (primarily successful, yet each doing something slightly different) before we decided on the optimal solution of yielding all of the entries of the pages of interest and the corresponding inlinks, and having other functions to convert this data to perform data analysis in R. This is probably the step that I learned the most from, because I tried so many different ways of getting the data to output a certain way.

Reading through the MRJob documentation towards the end of the project, I realize how much time I could have saved by reading that documentation before I started the project. But at least I learned a lot from these mistakes (even if I lost a good amount of time). Overall, I enjoyed working on this project. I also realized that Julian and Andy are vastly superior programmers than me. I got lucky with my group.