In this assignment, you’ll gain more experience with running Hadoop jobs and writing scripts in Pig to run on the Flux Hadoop cluster to process a Yelp dataset.   **This homework is worth a maximum 10% bonus credit to your course grade (or, something between a lab and regular homework).**

This assignment assumes you have successfully set up and logged into your flux account. Accounts on Flux can be applied for at [http://arc-ts.umich.edu/hpcform (Links to an external site)](http://arc-ts.umich.edu/hpcform). Fill in your name under "Name", select "Information" as the department, put my name (Teplovs) as the Advisor's name, cteplovs@umich.edu as the Advisor's email address, and leave the description box empty.  
  
You will need to read agree to the terms of usage (click the "Yes" box), and click submit.  
  
All the Hadoop and related tool commands are available on flux-login.  
Documentation for Hadoop is provided at: [http://arc-ts.umich.edu/hadoop-user-guide/ (Links to an external site.)](http://arc-ts.umich.edu/hadoop-user-guide/)  
  
For the purposes of SI 601, the queue name is 'si601f16'.

For review, a tutorial on pig can be found here: [http://pig.apache.org/docs/r0.7.0/tutorial.html (Links to an external site.)](http://pig.apache.org/docs/r0.7.0/tutorial.html)

**The dataset for this assignment comes from the Yelp Dataset Challenge (**[http://www.yelp.com/dataset\_challenge/ (Links to an external site.)](http://www.yelp.com/dataset_challenge/)).   (Check out their fun video on the site.)

**The input file you’ll use consists of reviews on businesses: it’s been placed in the Hadoop file system at: /user/cteplovs/yelp\_academic\_dataset\_review.json  (You may not copy or otherwise redistribute the data from this file without signing the Yelp Academic data usage agreement on the dataset challenge page.)**

**For example, you should be able to see the file's size and other metadata by listing my directory in HDFS:**

**hdfs dfs -ls /user/cteplovs**

**and dump the first few reviews in this file using the command:**

**hdfs dfs -cat /user/cteplovs/yelp\_academic\_dataset\_review.json | head**

To run a pig script, the Hadoop queue name for the class is "si601f16", and e.g. if your script name is "yelp.pig" then the command is:

**pig -Dmapreduce.job.queuename=si601f16 -f yelp.pig**

**\*\*Please note\*\***: To save yourself a lot of time, and also to avoid wasting the valuable shared resources on the flux cluster, you should \*always\* test your pig script locally on a small test file, such as the first few lines of the full json file.

pig –x local yourscript.pig

If MapReduceLauncher returns Success, and you’re certain you’re getting the output you expect, you can submit the full job to the cluster using the full datafile and the command:

**pig -Dmapreduce.job.queuename=si601f16 -f yourscript.pig**

The goal of this assignment is to extract two sets of words from the text of the reviews in this file:  words highly associated with positive reviews, and words highly associated with negative reviews.   This builds on the ‘word counting’ example pig script shown in class.  A very helpful resource is my yelp sentiment analysis script available from [https://github.com/cteplovs/mortar-examples/blob/master/pigscripts/yelp\_sentiment.pig, (Links to an external site.)](https://github.com/cteplovs/mortar-examples/blob/master/pigscripts/yelp_sentiment.pig,) which is forked from the MortarData repo at [https://github.com/mortardata/mortar-examples (Links to an external site.)](https://github.com/mortardata/mortar-examples).  That demo script takes a different approach to sentiment analysis, but you will find it helpful nonetheless.

**Computing word positivity/negativity scores**

To measure how much a word is associated with positive reviews (say), we will compute the following quantity for each word *w*.

Positivity(*w*) = log P(*w* in PositiveReviews) – log P(*w* in AllReviews)

where P(·) denotes probability and log(*x*) is the natural logarithm of *x*.  The term P(*w*in PositiveReviews) means the probability that a word occurs, given we are looking at the set of positive reviews.  The term P(*w*in AllReviews) means the probability of a word occurring, given that we are looking at all reviews.

Looking at the formula, you can see that words with a high Positivity score are words that occur with much higher probability in positive reviews than all reviews in general.  The “all reviews” probability distribution over words is sometimes called the “background model”, and “positive reviews” probability distribution over words is called the “foreground model”.   (The same formulas apply respectively for negative reviews, which you will need to construct also.)

**Examples.**To compute the probability P(*w*in T) of the word *w* occurring in a set of text T, one basic method is to simply count up how many times the word occurs in T, divided by the total number of words in T.  (There are more sophisticated methods that give better estimates if there are rare or missing word observations, but we don’t need these for now.)  Suppose a word *w*occurs 10 times in T, and there are a total of 100,000 words in T, then

P(*w*in T) = 10/100000 = 0.0001  and log P(*w*in T) = log(0.0001) = -9.21.

If the word ‘awesome’ appears 10 times in positive reviews, and all positive reviews together contain 10000 words, then P(awesome in PositiveReviews) = 10/10000 = 0.001.  If ‘awesome’ appears 20 times in all reviews, and there are 100000 words in all reviews, then P(awesome in AllReviews) = 20/100000 = 0.0002.  So the positivity score of ‘awesome’ turns out to be:

Positivity(awesome) = log P(awesome in PositiveReviews) – log P(awesome in AllReviews)

= log(0.001) – log(0.0002) = -6.9 – (-8.51)  = 1.61

Words that are more neutral, like ‘the’, that have similar probability given a positive review compared to any review, should have a positivity score close to zero.

In this assignment please follow these definitions:

**The Positivity score will be computed based on the set of reviews with >= 5 stars.**

**The Negativity score will be computed based on the set of reviews with <= 2 stars.**

**Step 1. (20 points)**

**Use the LOAD command in pig to read in the reviews file using the provided JsonLoader syntax**

R = LOAD 'yelp\_academic\_dataset\_review.json' USING JsonLoader('votes:map[], user\_id:chararray, review\_id:chararray, stars:int, date:chararray, text:chararray, type:chararray, business\_id:chararray');

Then, extract the ‘text’ field of each review, along with the number of stars for the review, using the flatten and tokenize commands in pig in preparation for step 2.

**Step 2. (30 points)**

Process the stream of text you created in step 1 by dividing it into three new streams:

1.     Text from all reviews

2.     Text from reviews with >= 5 stars  (the positive set)

3.     Text from reviews with <= 2 stars  (the negative set)

**Output.**For each of these three streams, compute the counts of all words appearing in those streams, and output three intermediate files called ‘output-step-1a’, ‘output-step-1b’ and ‘output-step-1c’ corresponding to the streams 1-3 above.  Like in the class lab, these are word count files with the two columns word, word\_count.

See the HDFS files /user/kevynct/output-step-\*  for the desired output at each step.

**Step 3. (15 points)**

**Filter out (i.e. do not consider) any words occurring less than or equal to 1000 times in the “all reviews” stream.**

**Join the word counts of the remaining (high frequency) words in stream 1 (all reviews) with the word counts stream 2 (positive) on the ‘word’ field to produce an output file with these columns:**

**word  countAll  word countPositive**

**Similarity, join stream 1 (all reviews) with stream 3 (negative).**

**Output.  Write your outputs to ‘output-step-2a’ and ‘output-step-2b’ respectively.**

**Step 4. (35 points)**

**Using the output from Step 3, compute the positivity and negativity scores for all remaining words using the formula given above.  Note that to compute the probabilities, as an intermediate step for all three streams, you will need to compute the total word count for each stream.**

**Output.  Write your positive and negative word lists using STORE commands to create two output folders: one called ‘output-positive’ that holds words associated with the ‘positive’ reviews, and ‘output-negative’ for words associated with the ‘negative’ reviews.  Hadoop will write the final word list into a file *in the Hadoop File System* called**

**/user/youruniqname/output-positive/part-r-00000**  
**(similarly for output-negative folder).**

**We have provided samples of the desired output at each stage. The first lines of your two files should match those of the corresponding samples in /user/kevynct/output-positive and /user/kevynct/output-negative .  These files have the first 1000 words.  You must provide your complete output, not just the first 1000 words.  These files are tab-delimited and they have these columns:**

**Column 0: word**

**Column 1: Positivity(word)  for output-positive folder,  and Negativity(word) for the output-negative folder.**

**Example output:**

**Most negative words, with negativity scores:**

unprofessional  1.3159149308302283

refund  1.306756309681317

rudely  1.2881802173912842

scam    1.2799716208525265

incompetent     1.2721048119019134

poisoning       1.252900906957354

worst   1.2433239510735445

disgusted       1.2411775408163255

refused 1.2257483115214054

apology 1.2218778051575985

**Most positive words, with positivity scores:**

passionate      0.8547682798288889

dentist 0.8330084244283551

lashes  0.8258323748340857

dental  0.8236724797897406

knowledgable    0.8158597340891962

flawless        0.8130409950229627

caring  0.8130026152762984

dr      0.8109967738599817

exceeded        0.8001766481099573

kelly   0.7969865646834453

gentle  0.7933084991821833

thorough        0.788555110907426

compliments     0.7728702044989522

professional    0.7680758353567754

highly  0.766959409183789

incredible      0.7645182575295664

impeccable      0.7633607009651371

remembers       0.756887657362947

skilled 0.754704019683377

jeff    0.7512733233655506

**What to submit**

As usual put your code (pig script) in a single file named si-601-hw5\_***youruniquename***.pig. Submit a zip file named si-601-hw5\_***youruniquename***.zip containing your pig source code file and all output files from the steps above (steps 1a,b,c;  2a,b; and positive, negative).

**Helpful links**

You can find documentation on useful Pig classes like JsonLoader here:

http://pig.apache.org/docs/r0.11.1/func.html