Fourth Homework Assignment (15% of grade)

Due on Friday, May 27 at 5 pm.

Deliver the following in folder /home/public/assignment (you should have write permissions; after you copy every file, execute: chmod 400 file_name) for the exercise:

1. Your pig source code file: name the file lastname_x.pig
2. Your UDF source code file: name the files lastname_x.java
3. A sample of the output file from a reducer: name the file lastname_x.txt

Here 'x' is 'jobs' and 'tweets.'

The assignment needs to be done in pig. You are allowed to use UDFs (and you have to).

Job Descriptions

The data comes from a current project of mine (only a tiny subset of the data is provided). There are several files containing job descriptions (from a job advertising web company). There is one job description per line with '\n' as the delimiter and it is included in the second field. The first field includes a unique job id. All necessary files are in folder /home/public/course/pig/jobs.

The goal is to provide files with a single row per job ad (thus in total the same number of records as the input data files). The order of the words in the final file for a given ad doesn't matter.

For each row the following tasks must be performed in the same order:

1. Tokenize
2. Remove all stop words (these are words such as 'a,the,in,of,…'). All stop words are provided in the file stopwords-en.txt
3. To all remaining words apply stemming (change 'badly' to 'bad,' 'going' to 'go,' …). An implementation is provided in the java file Porter.java.
4. Correct the misspelled words. The list of all English root words is available in dictionary.txt. Given a word, if it is available in the dictionary, it is spelled correctly. If it is not available, then compute the Levenshtein distance with all the words in the dictionary and select the one with the lowest distance. An implementation is provided in the java file Levenshtein.java.

Hint: There are two ways to do this: (1) record by record which will require a lot of java, or (2) an alternative way that creates many more "key-value" pairs. The first option requires the use of the distributed cache with pig. You are free to select either way or to come up with your own way.

Optional Exercise 2: Tweets (worth additional 20 points)

In folder /home/public/course/pig/tweets you will find tweets related to major US retailers. This task to use pig to access sentiment of every tweet. Your final output must be the number of tweets with positive sentiment and the number of tweets with negative sentiment.

There are two other files: good.txt and bad.txt which contain the set of all good, bad words, respectively. The sentiment score of a tweet is defined as: the number of good words in the tweet – the number of bad words in the tweet. A tweet has a positive sentiment if the sentiment score is great than 0 and negative if it is less than 0.