CSC 343: Big Data Programming and Management
Lab #4: MapReduce

For this lab, you will develop several Map and Reduce scripts in Python and use Hadoop streaming to process data stored on HDFS. You will turn in hard copies of your map and reduce scripts, and will push an image for this lab to Docker Hub. The name of the image must be in the format *gdancik/cloudera:lab4* and must contain the files and folders indicated below. Note that images can be renamed using a command like

```
docker tag gdancik/cloudera gdancik/cloudera:lab4
```

Note: It is recommended that you periodically save your work, by using *docker commit* to create a new image from a current container, after data is added or after map reduce jobs are run. In the event that a MapReduce job 'freezes', or a connection to HDFS is lost, creating a new container from a most recently saved image is recommended.

1. **Lab Setup.** Create a new container from the gdancik/cloudera image, that is running Hadoop, and copy the following directories to HDFS:
    a. Copy the WizardOfOz directory to hdfs://user/cloudera/WizardOfOz
    b. Copy the customers directory to hdfs://user/cloudera/customers

2. **Word Length**. Create MapReduce python scripts that will organize words by length. In order to do this, write a mapper script that emits *length, word* pairs for every word in its input. The reducer then accumulates sets of words for each length. The final output should be in the form *length*: *word set*, where *length* is the length of a word and *word set* is a comma separated list of unique words (see piazza post for more information about Python sets).

    Note #1: To accurately organize words, convert all words to lowercase and remove all punctuation. See the piazza post for how to do this correctly.

    Note #2: When running Map Reduce, the default 'comparator' used for sorting assumes character data (e.g., '11' will be before '2', since this is true alphabetically). To sort numerically, add the following two options to the Hadoop streaming command, which specifies to sort numerically in *reverse* order (so the longest words will be at the top of the results):

    ```
    -jobconf mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator
    -jobconf mapred.text.key.comparator.options=-r
    ```

    Note that for testing locally, add the *-h* option to the linux *sort* command so that a *human-numeric-sort* is used, e.g. the full command for testing will be similar to

    ```
    cat files | python3.4 mapper.py | sort -h | python3.4 reducer.py
    ```

Store your mapper and reducer files in the directory /home/cloudera/wordlength, use Hadoop streaming to run a Map Reduce job on the Wizard Of Oz data set, and store your output in /user/cloudera/output/wordlength.

For the remaining questions, you will process a *customers* data set. Each file is a tab-delimited file with columns corresponding to the following:

- the *id* of each customer
- the *first name* of the customer
- the *last name* of the customer
- the customer's e-mail address (hidden)
- the customer's password (hidden)
- the customer's *street*
- the customer's *city*
- the customer's *state*
- the customer's *zip code*

3. **Look-up customers by state.** Use Hadoop streaming to output all customers from CT, sorted alphabetically by last name. Your mapper and reducer code should be stored in */home/cloudera/customers_CT* and your output should be stored in */user/cloudera/output/customers_CT*.

4. **Count customers by state**. Use Hadoop streaming to find the number of customers from each state. Your final output should be in the form of *state: number_of_customers*. Your mapper and reducer code should be stored in */home/cloudera/count_by_state* and your output should be stored in */user/cloudera/output/count_by_state*.

Note: for parts (3) and (4), you are implementing standard database query operations using MapReduce. For those familiar with SQL, you are implementing the following queries:

```
# query for (3)
select * from customers WHERE state = "CT" ORDER BY lname;

# query for (4)
select state, count(*) from customers GROUP BY state;
```