[redacted]
Danny Weiner: *dnweiner*
Professor [redacted]
CSE 427S: Cloud Computing with Big Data Applications

Final Project #2 Report: Large-scale Text Processing and Sentiment Analysis in Hive

ABSTRACT / PROJECT GOAL

Using data collected throughout the course of both the Spring 2016 (SP16) and Fall 2016 (FL16) semesters, we sought to perform sentiment analysis on homework reviews for *CSE 427S: Cloud Computing with Big Data Applications*. Our work required emotional reviews of each homework assignment (1-10 and M); we aimed to take advantage of text (pre-)processing features in *HIVE* to assess whether the encapsulated sentiments were positive, negative, or neutral, as well as how different course concepts were perceived by the students. Finally, we shall predict emotion per SP16 assignment and compare our results to the actual overall sentiment of the reviews, followed by the same prediction/comparison approach on our personal reviews.

In this report we document our sentiment analysis approach, our implementations, and the obtained results. The *SVN* repository used for submission is [redacted]. We divide this information chronologically, problem by problem, highlighting key points and decisions.

PROBLEM 1: EXPLORE *HIVE* AND CUSTOMER REVIEW DATA

This problem defines *Milestone1* and part of *Milestone2*, which familiarize us with *HIVE* commands and prepares the data we need in further problems.

Step 1: Data Management with HIVE

This step prepares the data for Step 2: we are given instructions to create and load a *HIVE*-managed table. This tale of ratings (sourced from Dualcore, 2012 and 2013) will be used solely for Problem 1. Herein, the practice of creating and populating a *HIVE*-managed table serves as foundational knowledge for the ensuing problems in the overarching project.

Step 2: Gaining Insight with Sentiment Analysis

In this step, we follow the instructions to explore the text processing features in *HIVE* to analyze customers' comments and product ratings. *HIVE* provides extensive support for text processing, which we utilize as customer comments are typically free-form text. Steps involved analyzing numeric product ratings as well as rating comments. We have our *HIVE* scripts in `.hql` files submitted to our *SVN* repository.

- rated_lowest.hql
    - Given: a prescribed query for finding the product with the highest average rating amongst those with at least 50 ratings.
    - Accomplished: rewrote the given query to instead display the product with the lowest average rating amongst those with at least 50 ratings.
- trigrams.hql
    - Given: a prescribed query for normalizing product comments to lower case, tokenizing, and finding the five most common bigrams.
    - Accomplished: rewrote the query to find the five most common *tri*grams.
- messages_red.hql
    - Given: a prescribed query for finding product comments matching a given phrase.
    - Accomplished: rewrote the query to instead find product comments matching a different phrase.
- find_red_product.hql
    - Given: implication that the product with ID# 1274673 is overpriced relative to similar products.
    - Accomplished: wrote a query to display the full record for product ID# 1274673.

PROBLEM 2: DATA PREPARATION

In this problem we prepare the homework review datasets as part of *milestone 2*. The homework review data is given to us in two *hw_reviews* compressed files and each is organized into four subfolders: *positive*, *negative*, *neutral*, and *not_labeled*; the data in each folder is as the folder name suggests. Each individual review is stored in a `.txt` file without author attribution: *hw*[homework number (1,...10, M)]*_review_*[unique, random six-character review ID].*txt*. The two *hw_reviews* folders contain Fall 2016 and Spring 2016 course semester data. Although the homework numbers across the two semesters do not refer to the exact same homework assignments, the themes of the addressed content at various points in the semester are consistent.

Step 1: Filter out Invalid Reviews

As per the instructions, we prepare the homework review datasets for our sentiment analysis as follows, writing a *python* script (*filter_invalid_review.py*):

1. We filter out and delete all empty reviews and all reviews shorter than 50 words.
2. We remove all non-word characters (including newline and tab characters) from all files remaining, using regular expressions.
3. We add the homework number and label at the beginning of each file before the actual review text, separating these three columns using the tab character. For the Fall 2016 reviews, which do not have labels, we simply insert a blank in place.
4. We put a newline character ('\n') at the very end of each review.

Step 2: Text Pre-Processing

In this step, we continue to follow the instructions, doing the following in *HIVE*:

1. We insert the data (now "prepared" in Step 1) using three fields: *hw_number* as *INT*, *label* as *STRING* (we chose to do this as *STRING* and not as *INT* because we simply inserted a blank in place of the label for Fall 2016 rather than a number), and *review* as *STRING*. There are both *reviews_FL2016* and *reviews_SP2016* tables present.
2. We pre-process our text data into all lowercase letters.
3. We remove the stop words—words that occur frequently in natural language text, but without adding sentiment, or meaning, to the text—that are listed in *english.stop* from our dataset. This is done via a second python script (*stop.py*) that converts the words in each file to lowercase, reads them in, and checks if they are members of a set of stop words. That set was constructed by opening *english.stop* and appending its contents.

In completion of this problem, we submit our python scripts—*filter_invalid_review.py*, *stop.py*—and one pre-processed example review—*prep_review.txt*—to the *SVN* repository.

Problem 3: Sentiment Analysis on the Homework Review Data

Step 1: Basic Analysis

A:

We executed this task using the labeled FL16 data. We did this by constructing a table that associated hw_number and an array version of the review string, which was then adapted into another table that associated hw_number and individual words. That table was joined first with our table of positive words (constructed from *pos-words.txt*), then with negative words (constructed from *neg-words.txt*), resulting in two new tables: one of featured positive words and one of featured negative words. We then grouped by hw_number and label and ordered by the count for each group. In other words, the most positively viewed assignment is the (hw_number, 'positive') group associated with the longest list of reviews, and the most negatively viewed assignment is the (hw_number, 'negative') associated with the longest list of reviews.

- **Most positively viewed assignment:** hw9
- **Most negatively viewed assignment:** hw1

Scripts included in *SVN* repository as:

- *create_pos_neg_word_tables.hql*
- *pos_neg_join.hql*
- *hw_freq.hql*

B:

Five most frequently used positive words:		**Five most frequently used negative words:**	
Word	**Frequency**	**Word**	**Frequency**
good	188	problem	323
work	152	difficult	124
pretty	110	hard	120
interesting	92	problems	103
easy	84	pig	103

Our results mostly make sense, because the majority of the found terms are, indeed, appropriate to their category (i.e., the five positive words listed below are positive and the five negative words are negative). Moreover, these frequently used terms are familiar words in the vernacular of a Washington University student; videlicet, their presence here does not seem out of place, as though the found words are at too high or low of a level of diction.

There are, however, three questionable word assignments:

- "work"
  - "Work" as a noun is neutral, though positive in the case of "it works!", although "a lot of work" is negatively connotated.
- "pretty"
  - "Pretty" is not something we would expect to describe a homework assignment devoid of aesthetics, although it is a valid modifier "pretty easy"/"pretty hard".
- "pig"
  - "Pig" is going to be used here to describe the big data tool, not the animal or its form as a pejorative descriptor.

We did this by grouping the tables from A by word instead of hw_number, aggregating, and sorting in descending order (limited to 5). Script included in *SVN* repository as: *word_freq.hql*

Step 2: N-grams

A:

We mirrored the structure of our ngrams exercises from *Milestone1*, albeit with the added feature of parameterization. Table creation was managed in a manner very similar to Step 1. The path to the text files and the n in ngrams are entered on the command line using `-hiveconf`, and n is converted to an int. Script included in *SVN* repository as *step2_ngrams.hql*.

B:

Three most positive bigrams:		**Three most positive trigrams:**	
Bigram	**Frequency**	**Trigram**	**Frequency**
("pretty", "easy")	13.0	("work, "good", "comprehensive")	2.0
("pretty", "cool")	9.0	("work", "pretty", "cool")	1.0
("pretty", "fun")	7.0	("fairly", "straightforward", "easy")	1.0

We were tasked to find the three most positive bigrams and trigrams. To do this, we expanded on the script written for A, seeking now to identify the overall rating for an ngram. That is to say, a bigram with two positive words is the most positive possible, just as a trigram with three positive words is the most positive possible. We then used the ngrams' `estfrequency` to determine their frequency. For the same reasoning as with *Step 1b*, our results here make sense. In point of fact, ngrams have proven themselves more reliable, as expected, because they incorporate modifiers like "pretty" into collective positivity, rather than standalone.

Step 3: What are the Favorite Topics?

A:

There does appear to be a difference between the homework assignments in the first half of the semester (1-5) and those of the second half (7-10). We examine our ranked results for each homework assignment (from Step 1) in this categorized manner, subtract the negative counts from the positive counts, and then divide by the number of assignments in each half (to normalize). The breakdown is as follows:

1. First half (1-6)
    a. 1035 - 1069 = -34 / 6 = -5.667 net favorability
2. Second half (7-10)
    a. 743 - 810 = -67 / 4 = -16.750 net favorability

We can see that the former half of the class was generally viewed more favorably. Script included in *SVN* repository as *favorite_topics.hql.*

B:

There does appear to be a difference between the homework assignments on Theory, MapReduce, and Pig and Spark. We examine our ranked results for each homework assignment (from Step 1) in this categorized manner, subtract the negative counts from the positive counts, and then divide by the number of assignments in each category (to normaliz). The breakdown is as follows:

1. Theory assignments (1, 2, 7)
    a. 578 - 562 = 16 / 3 = +5.333 net favorability
2. MapReduce assignments (3-6, 8)
    a. 780 - 867 = -87 / 5 = -17.400 net favorability
3. Pig and Spark assignments (9-10)
    a. 420 - 450 = -30 / 2 = -15.000 net favorability

We can see that Theory was generally viewed the most favorably, followed by Pig and Spark, and then MapReduce. Script included in *SVN* repository as *favorite_topics.hql*

Step 4: How good are your Sentiment Predictions?

A: Approximate average accuracy: 49.315%

We did this by using our program and process from *Step 1*, this time applying it to the review data for SP2016. The only major difference was the added comparison of findings to ground truth label, in order to determine the above accuracy percentage. To achieve this, we needed to add a unique id column to our table, in order to associate analyzed review strings with their ground truth labels. We then joined with the positive and negative word tables, as before, and subtracted the counts to generate the net favorability. Depending on this value, we could assign an experimental label to a review entry, that label being the one we sought to compare to the ground truth. Script included in *SVN* repository as *step4.hql*.

B:

Based on the above accuracy, our results match the ground truth labels at an above-average rate, though there is room to improve. Possible improvements include:

1. More thorough lists of positive and negative words (or simply lists of positive and negative ngrams), with the understanding that human communication relies far more on context than individual morpheme definition.
2. Additionally, we require ground truth labels on the contents of the *not_labeled* directory within the SP2016 reviews. Since we lack these results, our average accuracy is not indicative of the full catalogue of reviews.
3. As an additional point of improvement, developers with broader polylingual capabilities could implement this system for languages beyond English. Washington University is a community that brings in individuals from an array of backgrounds and heritages, and disregarding this can be insensitive. For the scope of this project, it is understandable, though one notes that a "perfect world" redo would account for this much-needed inclusivity.

Step 5: Does your System agree with your own Emotions?

A: Data collected into *review_labels.txt*.

B:

With 85% accuracy, our results appear to align with our actual moods, which were a mix of positive and negative feedback (albeit skewed towards the latter). This was accomplished in a manner very similar to the procedure we developed for the bulk data, complete with table creation, pre-processing, etc. We then used a similar execution to that of *Step 1* and *Step 4*, although modified to account for the difference in labeling and the manual comparison to *review_labels.txt*. Script included in *SVN* repository as *step5.hql*.

Step 6: Documentation of Approach and Results

Completion of this report entails completion of this step. Report included in SVN repository as *project_report.pdf.*

CONCLUSION

Through rigorous testing, debugging, and implementation, we have successfully performed sentiment analysis on the homework review data for both SP16 and FL16 semesters of *CSE 427S: Cloud Computing with Big Data Applications*. Our various findings are documented throughout the relevant sections of this report. In summary, homework reviews were primarily negative, especially in the latter half of the class, though the MapReduce homework assignments were the most widely disliked. Our system performs with approximately 49.315% accuracy, and could be improved with updated word lists to cross-reference, as well as some accounting for the fluidity and context-dependence of English communication. This accuracy extended to our own reviews as well, which reflected the general trends of the course in their negative slant.

REFERENCES

- http://www.cse.wustl.edu/~m.neumann/fl2016/cse427/Project2_text.pdf
- http://jmcauley.ucsd.edu/data/amazon/
- http://bigdatabloggin.blogspot.com/2012/08/trending-topics-in-hive-ngrams.html