Large-scale Sentiment Analysis with HIVE: Discovering the Complexity of Language

Hannah Mehrle, Tony Bumatay, Jacob Pepe

**Abstract**

In this project we used Hive, a batch processing programming language to do large scale data analysis. The data that we used consisted of the homework reviews from students in Nuemann’s cloud computing class from Spring 2016 and Fall 2016. We pre-processed student reviews to get them into a format that Hive could interpret. We then used a list of positive and negative words to figure out how positive and how negative each review was. We compared our results to how students actually said they felt about the homework, at the time that they wrote the review and our algorithm was accurate 55% of the time. To improve algorithm we filtered out words from the positive and negative lists that should not be there in the context of what we were doing. For example ‘problem’ was the most used negative word, but in context of homework problems it should not have been counted as negative. We then ran our algorithm again with the filtered lists of reviews from Spring 2016, and our accuracy improved to 60.2%. Next we ran the updated algorithm on our own reviews, finding that it was correct 51.6% of the time. Lastly we edited our algorithm a little (changed file paths and the number system we used to see if the ratings were correct) and ran it on automobile data from Amazon on the cloud. This time our algorithm was correct only 27% of the time, but we felt this was because we could not effectively filter the positive and negative word files for the new data because of how many reviews that there were. Although these results were not what we were aiming for (70%), we feel that it was hard to do much better as the labels on the reviews were solely based on human emotion, and we feel that overall the algorithm and the text processing in Hive were successful.

**1. Introduction**

*Motivation*

Large-scale text processing and sentiment analysis are in their exciting adolescence with extremely pertinent applications to computer science research and business analytics. It has become increasingly higher and higher in demand because it allows researchers to gain insights into how people feel -- what their opinions, emotions, and moods are at a large scale. Previously, this was only possible by conducting focus groups and getting people to explicitly talk about their sentiments regarding a certain event, person, product, etc. in a face to face setting. In research, the larger and more diverse the sample size, the more accurately your research results will reflect some inherent truth about the entire population at large.

The applications of sentiment analysis are seemingly endless, but a few examples include analyzing the sentiment of: aggregated news articles and blog posts about a particular person (i.e. US President) or event (i.e. Donald Trump wins the US election), reviews of a hotel company on different travel websites (i.e. reviews of Days Inn on TripAdvisor), consumer product reviews on online marketplaces (i.e. reviews of the XBOX 360 on Amazon). In a lot of cases, huge tech companies like Amazon and Twitter that are uniquely positioned with access to gigantic pools of public data, provide the raw data that enables this scale and diversity of research and commercial application. One of the most notable commercial applications is made by a company named Medallia that focuses on helping other companies capture feedback and reviews, analyze the data, and turn it into action items to improve the company’s sales, brand image, pricing, etc. Essentially, they’ve built a $1.25 billion company off of the idea of helping other companies understand how customers feel about their products and/or services and where they can improve.

*Drawbacks*

The potential benefits of an accurate sentiment analysis algorithm are huge, but there are certainly a number of drawbacks; after all, if it was so easy everyone would have been utilizing this technique already. The difficulty of accessing data, processing structure-less data, and designing an accurate algorithm are just a few of the most major obstacles that analysts face.

*Purpose*

This project is broken into two main parts. Part 1: process and analyze student written text reviews of each homework assignments over the course of the semester (For Marion Neumann’s CSE 427 course at Washington University in St. Louis - Spring 2016 and Fall 2016 semesters) to determine whether each homework assignment was perceived to be positive or negative. Part 2: repurpose our preprocessing functionality and our algorithm to analyze Amazon’s text reviews for automotive related products. Then, based on the sentiment score of each review, we would predict a 1-5 star rating and see how well our sentiment analysis translate to the objective. Throughout the remainder of the paper we will refer to these sections as Part 1 and Part 2.

*Outline*

The remainder of the paper is outlined in the following way: Our Methodology is described in section 2, the findings of our experiment are located in section 3, our discussion of these findings makes up section 4, and our conclusion wraps up the paper in section 5.

**2. Methods**

*Workspace Environment*

Part 1: We ran HIVE running on top of a hadoop pseudo cluster on a virtualbox instance.

Part 2: We ran HIVE on an Amazon Elastic MapReduce cluster using data stored in an Amazon S3 bucket, run using an Amazon EC2 instance.

*Data Collection*

Part 1: The total possible subject pool was the set of students who completed Professor Neumann’s Cloud Computing with Big Data Applications at Wash U in the Spring 2016 and Fall 2016 semesters. At the end of each of the eleven homework assignments throughout the semester each student had the opportunity to write a homework review in exchange for 5% extra credit on the given assignment (with a maximum total score of 100% on the assignment). The goal of the homework review was for students to write a short reflection on how they felt about the assignment, knowing that it would be used for sentiment analysis at the end of the semester. These reviews were graded simply on completion, and Professor Neumann did not look at the content so the students would, in theory, be open and honest with their feelings without fear of upsetting or offending the professor.

At the end of the semester, each student completed a file classifying each assignment as simply “positive”, “neutral”, or “negative” which served as the ground truth for our comparisons with the sentiment analysis. The individual reviews were aggregated by the professor and grouped by their respective labels (positive, neutral, negative). Refer to the discussion section to read more about why we think this was a faulty way of determining a supposed “ground truth” for student sentiments.

Part 2: We downloaded the 5-core JSON data for Amazon purchases in the Automotive category between May 1996 and July 2014 from the following website: <http://jmcauley.ucsd.edu/data/amazon/>.

*Data Preprocessing*

Part 1:In this step we started off by writing a python script to get the homework reviews in a format that Hive could process. We filtered all special characters out of the review files, and we formatted the reviews so that the homework number (a string) was the first field followed by a tab then the homework label (a string), another tab, and then the review (a string). We loaded the data into hive, and wrote a UDF (user defined function) that filtered the stop words and made the review lowercase. A UDF is a function written and compiled in java that can be used on Hive tables.

Part 2:The data for this step was in json format. We used Amazon EMR’s json parser to parse the data, and then we ran it through the same UDF as in part 1.

*Sentiment Analysis Algorithm*

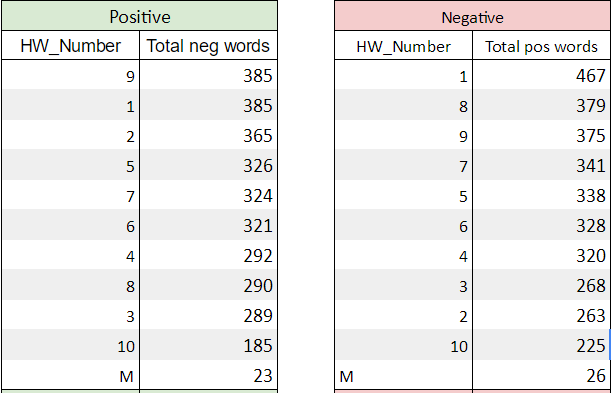
Part 1: In steps 1-3 of the analysis (up until we tested our algorithm on the labeled reviews) we ran our algorithm twice, once for positive words and once for negative words because we felt that a homework could be both negative and positive at the same time. Therefore we have both negative and positive results for this step. Starting in step 4 we combined the two tables.

We loaded the positive and the negative word tables into HIVE we gave negative words a value of -1 and positive words a value of +1. We unioned these tables together to get one table. Next we added the UDF as a jar and created a temporary function in Hive to filter the words. We created a table with the pre processed reviews. Next we edited the table so that there was a fourth column that contained the row number. We did this so that each review was given a unique number that we could group by later. Next we exploded the reviews so that each word of the review was its own row. Next we did a join on the review table and the positive and negative word table. We did the join such that the only rows that were kept were ones that either had a positive or negative word in them, and in the new table that was created there was a new column added that was either a 1 if the word was positive or a -1 if the word was negative. Next we used the row number generated earlier to join all of the exploded reviews back together so that all of the words in one row came from the same original review, in this table we also summed the total count of positive and negative words from each reviewer. Lastly we created a table we called accuracy that kept only the reviews that correctly matched their label. We said that a sum of less than zero was negative, 0 was neutral, and greater than 0 was positive. We then counted the number of rows in each of our last two tables to get the accuracy.

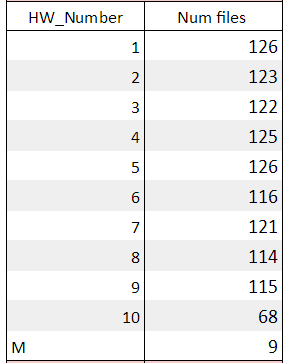
Part 2: There were two main differences when we did this on the cloud. First was where the data was located so we had to change our paths to load the data and jars. Second we were doing a rating out of five as opposed to positive, negative, and neutral in part 1. We said that if count was greater than 2 then the review should be 5 stars, if count was 1 it should be 4 stars, if it was 0 it should be 3 stars if it was -1 if should be 2 stars and if it was -2 or lower it should be one star.

**3. Results**

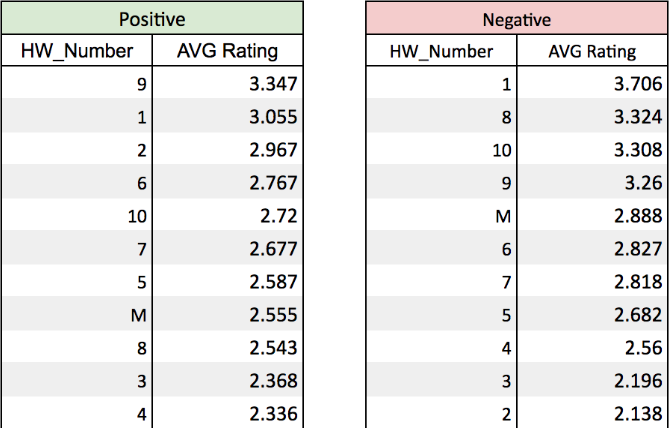
First, we found the total number of positive and negative words over all reviews:



To give these results more context we found the total number of reviews submitted for each homework

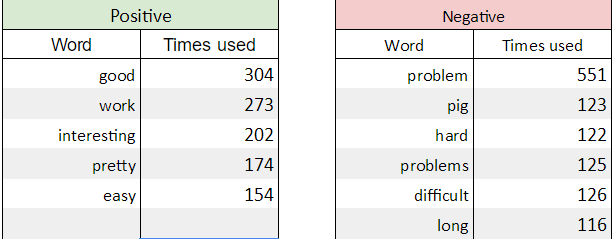


Next we found the average number of positive words and negative words per individual review, and averaged all of the people together to account for the fact that not every homework assignment was reviewed by every student. The following are the results that we used for this part:



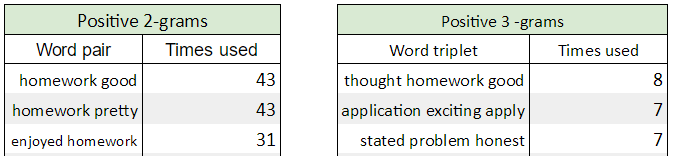
Based on these results the most positive homework was homework 9, and the most negative homework was homework 1. By our interpretation, the “most positive” assignment was the one with the highest average number of positive words per review. The same interpretation holds for our “most negative” calculation.

**Top 5 positive and top 5 negative words**



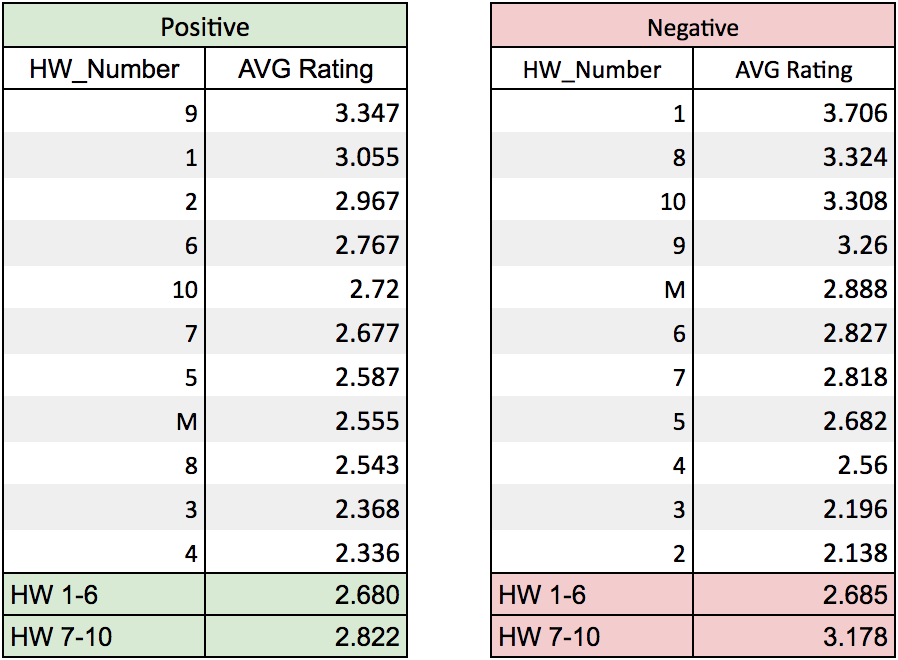
We calculated the top six negative words because problem and problems have the same meaning.

**N-grams**



**Favorite topics**

In this step, we grouped the homeworks by first half of the semester and second half of the semester



Next we looked at whether there was a difference in different topics.

MapReduce (HW3, HW4, HW5, HW6, HW8)

Avg positive - 2.5202

Avg negative - 2.718

Theory(HW1, HW2, HW7)

Avg positive - 2.90

Avg negative - 2.887

Spark and Pig(HW8, HW10)

Avg positive - 2.632

Avg negative - 3.316

These results show that Spark and Pig were liked the least, and theory was liked the most. Spark and Pig had the highest average negative sentiment by a significant margin, while the Theory assignments were regarded most positively by a slimmer margin, but was the only topic in which the positive average was higher than the negative average, indicating an overall positive sentiment score.

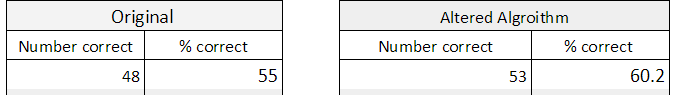
**Accuracy**

After developing our algorithm we ran the pre processed homework reviews from the spring 2016 class through the algorithm to see how accurate we were.

Two improvements:

1. Editing the positive and negative word list to remove things like 'pretty' , 'pig', and 'problem', which in the context of these reviews aren't actually expressing negative or positive emotion.
2. Use 2-grams to ignore (or even flip the ratings of) the word grams that contain the word 'not' as the first word and a positive or negative word as the second. This way our ratings aren't skewed by people saying 'not bad' or 'not good'.

We altered our algorithm to account for improvement #1 and round the following results:



Next we ran our own reviews through our altered algorithm, to see how well our algorithm agrees with our own emotions.



**Running on the cloud**

Running the data from automotive 5-core, we got 5176 reviews correct out of 19155 = 27%

Total runtime from logs below: 549.195 second = 9 minutes 9.195 seconds See exhibit 1 for detailed runtime

**4. Discussion**

*Shortcomings of our Process*

Our results were promising to some degree, but largely we found our sentiment analysis process to be inadequate to accurately analyze or predict user sentiments about a topic. The following are some of our hypothesis as to why we encountered low accuracy levels:

Data collection: what is considered ground truth for the hw sentiments was most likely not accurate in a lot of cases. The professor asked students to rank each hw with the one word label of “positive”, “neutral”, or “negative” at the end of the semester long after most students had forgotten which topic each assignment even covered. The primary issue was the high degree of ambiguity and subjectivity allowed for these terms. There were not definitions or criteria provided for what constitutes which label. Additionally, feelings and sentiments are very fickle and ephemeral, meaning it may be very difficult to accurately recall exactly how one felt about a homework assignment from three months ago. Finally, there was no real incentive for students to be invested in accurately determining their true feelings, so there is no way to ensure the accuracy of the “ground truth” data in this experiment. Accuracy would have been greatly improved if each student recorded their one word label feeling at the time of submitting the assignment.

Homework as a subject: Beyond the shortcomings of our data collection, we think studying sentiment analysis for homework assignments is not the best way to set yourself up for success. It is very unlikely that people love homework assignments. Often there is a much more complex relationship with homework than a simple positive or negative binary. Often assignments that are most difficult, take the largest time investment, and require the largest toil also produce the largest understanding of the given material. Is that a positive or a negative situation? It’s highly subjective and depends on the criteria used to define “positive” and “negative”. In this case, no criteria was given, making it very subjective and inconsistent from student to student.

Sentiment Score Algorithm: We used a simple count for each occurrence of a positive or negative word in a review. This algorithm could be more robust by giving each word a positivity score from -5 to +5. This is because, for example, “good” does not convey the same sentiment as “fantastic”.

Negativity Bias: One challenge with sentiment analysis is overcoming negativity bias. Negativity bias is the phenomena that people are likely to react or register negative interactions, feelings, events, etc. more negatively than they will register a positive event of the same magnitude. This would likely skew a large majority of ratings in the negative direction.

*Language is Complex and Still Very Difficult to Understand Computationally*

Language is inherently contextual. If reviews were one word long we could take them at face value, but otherwise there is much more complexity involved.

* “I was not impressed.”: This should be a negative sentiment because not is an inverter.
* “This product allowed me to avoid an unnecessary tragedy”: Classified as 2 negative sentiments when in reality, it’s very positive.
* Despite a very similar sentiment score the following sentences have dramatically different meanings:
  + “This is my favorite sweater of all time”, and
  + “This isn’t my favorite sweater, but it I’ll wear it when the rest of my clothes are dirty”

*Amazon Star Ratings are Often Uncorrelated with Sentiment Score*

Take the review: “Great product.”, for example. It’s difficult to register a high sentiment score for a review like that algorithmically, but as a human, I can guess that the user likely rated the product 4 or 5 stars. On the other hand, a very long winded review that contains lots of positive words but then ultimately ends with a negative idea is likely to be predicted as a highly rated product. Leading research in the field have spent years and years designing far superior and accurate algorithms than we were able to in a matter of weeks.

**5. Conclusion**

Despite the rapid acceleration of computing power, techniques, and algorithms in the past few decades, analyzing the emotion, feeling, and mood behind language is a very difficult computational problem to tackle. We found that conducting a sentiment analysis by simply counting positive and negative words at face value proved to be inaccurate in both our analysis of homework assignment reviews and reviews for automotive products on Amazon. Feelings and emotions are very fickle and impulsive, and change significantly based on complex contexts. Given this variability we were able to capture the correct sentiment of a homework review 60.2% of the time with our best algorithm which we were very proud of. However, when we analyzed Amazon review and rating data for Automotive products we were only able to predict a rating (1-5 stars) correctly 27% of the time. Upon manual human analysis of a sample size of the reviews and rating pairs, we determined that there was very little correlation with the overall rating and the likely sentiment score of the review. An interesting future study might be the interesting psychological phenomenon of the negativity bias and the correlation ratings and reviews. We were inaccurate, but we hypothesize that our error was not evenly distributed, but rather the vast majority of users rated products with lower star grades than their words might indicate they felt about the product.

**6. Exhibits**

Exhibit 1: Detailed running time on AmazonEMR of Hive script.

Runtime =

Stuff for runtime:

OK

Time taken: 1.382 seconds

OK

Time taken: 0.313 seconds

OK

Time taken: 0.105 seconds

OK

Time taken: 0.188 seconds

OK

Time taken: 0.129 seconds

OK

Time taken: 0.196 seconds

Total MapReduce CPU Time Spent: 1 seconds 370 msec

OK

Time taken: 22.357 seconds

OK

Time taken: 0.055 seconds

OK

Time taken: 0.035 seconds

OK

Time taken: 0.078 seconds

OK

Time taken: 0.16 seconds

Stage-Stage-1: Map: 1 Cumulative CPU: 1.37 sec HDFS Read: 47766 HDFS Write: 59146 SUCCESS

Total MapReduce CPU Time Spent: 1 seconds 370 msec

OK

Time taken: 22.811 seconds

OK

Time taken: 0.061 seconds

Total MapReduce CPU Time Spent: 4 seconds 550 msec

OK

Time taken: 52.042 seconds

OK

Time taken: 0.018 seconds

OK

Time taken: 0.071 seconds

OK

Time taken: 0.123 seconds

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.87 sec HDFS Read: 5665491 HDFS Write: 5768741 SUCCESS

Total MapReduce CPU Time Spent: 6 seconds 870 msec

OK

Time taken: 35.324 seconds

OK

Time taken: 0.118 seconds

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0

2016-12-12 14:28:33,447 Stage-1 map = 0%, reduce = 0%

2016-12-12 14:28:43,100 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.27 sec

MapReduce Total cumulative CPU time: 3 seconds 270 msec

Stage-Stage-1: Map: 1 Cumulative CPU: 3.27 sec HDFS Read: 5773216 HDFS Write: 24531044 SUCCESS

Total MapReduce CPU Time Spent: 3 seconds 270 msec

OK

Time taken: 141.155 seconds

MapReduce Jobs Launched:

Stage-Stage-4: Map: 1 Cumulative CPU: 4.04 sec HDFS Read: 24537439 HDFS Write: 2938812 SUCCESS

Total MapReduce CPU Time Spent: 4 seconds 40 msec

OK

Time taken: 30.599 seconds

OK

Time taken: 0.072 seconds

HDFS Write: 505653 SUCCESS

Total MapReduce CPU Time Spent: 6 seconds 30 msec

OK

Time taken: 35.226 seconds

OK

Time taken: 0.101 seconds

Stage-Stage-1: Map: 1 Cumulative CPU: 2.84 sec HDFS Read: 510074 HDFS Write: 125165 SUCCESS

Total MapReduce CPU Time Spent: 2 seconds 840 msec

OK

Time taken: 23.821 seconds

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 2.42 sec HDFS Read: 131655 HDFS Write: 5 SUCCESS

Total MapReduce CPU Time Spent: 2 seconds 420 msec

OK

5176

Time taken: 154.392 seconds, Fetched: 1 row(s)

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 2.71 sec HDFS Read: 512389 HDFS Write: 6 SUCCESS

Total MapReduce CPU Time Spent: 2 seconds 710 msec

OK

19155

Time taken: 31.436 seconds, Fetched: 1 row(s)