CSE427S

Sentiment Analysis of Course Reviews with Applications

12/13/2019

Authors: Sid Kurkure, Kevin Li, Christian Ralph, Howard Lan

Introduction

With the rise of the Internet of Things (IoT), big data, and means for capturing, analyzing, and utilizing big data, new possibilities have arisen to take advantage and learn from this data. One such area is Natural Language Processing (NLP) and Text Analysis. Major entities such as Amazon, Google etc. have invested significant resources in improving and perfecting their NLP abilities. Concurrently, major entities have moved away from on-premises servers and their limitations in efficiency and scalability towards more modern solutions within the cloud that allows for virtually unlimited storage, massive distributed computation power, and scalability. These companies have the ability to work with millions of records to mine insights from data that could not be discovered by traditional means.

Sentiment analysis is a key part of NLP that leverages user/customer written reviews on various topics and products to gauge opinion and satisfaction levels. Especially today, customers are encouraged to leave reviews on online products and services and give honest feedback. To maintain good business-customer relations, it is critical for businesses to accurately assess how their customers feel in order to target areas for improvement and enhance their provided products and services. With the rise of e-commerce and social media platforms, opinionated text is now more commonplace than ever, so sentiment analysis has become an increasingly important application to extract meaningful insights from data.

In this project, sentiment analysis was performed on homework reviews from previous semesters and used to predict the sentiment of homework reviews in the current semester. We started with simpler models and default parameters to establish a baseline accuracy for our models, and later iterated to more complex models with fine-tuned parameters for comparison.

Finally, we applied the models that scored highest on our success metrics to a larger dataset of Amazon product reviews. Commentary on interesting observations and thorough explanations on reasoning behind our model choices are included throughout the report.

<u>**Possible Solutions**</u>

Before we implement various techniques and learning models, we must first transform text into a format that can be usable. There are a variety of techniques and representations of text data that can be utilized. The main ones include one hot encoding, frequency encodings such as TFIDF or bag-of-words and prediction-based embeddings such as Word2Vec. We had to determine which method we wanted to explore in this project.

One hot encoding is the most straightforward data preprocessing method. It involves creating a vector for each document in our corpus in which each dimension of the vector represents a unique word. If the column that represents a word is within that document, we assign that column a value of 1 (or the number of instances it occurs depending on the implementation). While it is relatively simple, we lose contextual information in this process due to the fact that the vector is unordered and the dimensionality to represent each document is huge, therefore creating a more computationally expensive process.

Another possibility is utilizing term frequency inverse document frequency (TF-IDF). This frequency representation adds significance to words that appear frequently in a single document but not within the entire corpus of documents. This method combats some of the disadvantages of one hot encoding by representing a document by a set of TF-IDF scores instead of a high dimension vector. It also accounts for common words not removed in the initial stopword removal process. However, it is computationally expensive to calculate these TF-IDF scores and again we lose contextual information in the process. Another frequency encoding is the bag of words model; while it is superior to one hot encoding, it suffers the same issues that impact TF-IDF.

The final major method investigated is Word2Vec, a method that accounts for word context by using neural networks to map similar words closer together by using cosine similarity in Euclidean vector spaces. So words that are similar are given numerical representations that are closer to each other. While this method is the only major method discussed so far that is able to take into consideration context, it is relatively difficult to implement. Other methods have been developed that are superior to Word2Vec, such as Poincaré representations of words. This method improves on Word2Vec's latent hierarchical structure that results from utilization of euclidean space and instead maps word in hyperbolic space (in this case an n-dimensional Poincaré ball) that preserves a more dynamic hierarchical structure. However these methods are outside of the scope of this class.

There are a few other potential methods that could be utilized, such as feature hashing, but our group decided to choose between the first two representations. Ultimately TF-IDF was chosen to do our data pre-processing with a bag of words model as a comparison. TF-IDF balanced simplicity in implementation while preserving the necessary information needed. It is also significantly less expensive in terms of processing, especially when we implement the second portion of the project.

Methodology

We wanted to complete a semantic analysis of the homework reflections across the years to determine if we could utilize TF-IDF to create a rule-based model that would hold up to other common models for semantic analysis such as machine learning. As a result, we set up an experiment in which we would compare our rule-based model to a classic machine learning model. Both models would rely on the TF-IDF dataset as the basis for insights found.

Our rule-based model would use TF-IDF scores as weights and we would develop a sentiment score for each document. The range of scores are from -1 to 1 where -1 represents a negative document and 1 represents a positive document. The scores for each document would be calculated by $s = \sum_{i=1}^{n} W_i * sentiment_i$ where W_i is the TF-IDF score and sentiment_i is the sentiment score for that particular word (1 for positive, -1 for negative or 0) for the n words in that particular document.

We wanted to compare this rule-based model to a classic machine learning model used to predict sentiment. There are multiple potential models such as Deep Learning or Support Vector Machines (SVM), but we decided to use a logistic regression model from Spark's MLlib, since it is the most straightforward model of the 3 mentioned, and its results are more easily interpretable. We will do a 80-20 train test split on our dataset and use the test set to test both the logistic regression model and our rule-based model.

Data Preprocessing

Even before exploring a basic model, data preprocessing was required to make downstream analysis easier and more efficient. The provided data included 4 semester's worth of data, including Spring 2017, Fall 2017, Spring 2018, and Fall 2019, with reviews categorized as positive, neutral, negative per folder. It is important to note that the homework assignments varied year to year, with respective topics subject to change as well. This made analysis on individual homework topics difficult, and instead guided us to a more holistic sentiment analysis based on content-agnostic text models, such as the bag-of-words model.

To preprocess the data, we used PySpark to load the text into separate RDDs grouped by year and label (positive, negative, and neutral). An example is shown below for Fall 2017.

```
>>> fl17neg.first()
[(u'hdfs://quickstart.cloudera:8020/user/cloudera/fp_data/hw_re
views_fl17/negative/hw1_review_4651AC.txt', u'The homework was
more theoritical than I thought it will be. Problem 3 was
especially lot more theoritical. I was expecting it to be more
of introductory questions. Overall I was bit disappointed with
the homework this time. Hopefully it will get better going
forward. There was very less to do with the textbooks content
in this homework which was unexpected to me.\n')]
```

Figure: RDD representing the Fall 2017 negative review data

Then, for each RDD, we performed a series of transformations to convert each review to include the homework number, label, and text. The text was stripped of punctuation and converted to lower-case for downstream preprocessing steps. Additionally, we only included reviews with greater than 50 words to account for potential outliers and ensure that our sentiment metric is representative of the data. An example is shown below.

```
>>> fl17neg.first
['1__negative__the homework was more theoritical than i thought
it will be problem 3 was especially lot more theoritical i was
expecting it to be more of introductory questions overall i was
bit disappointed with the homework this time hopefully it will
get better going forward there was very less to do with the
textbooks content in this homework which was unexpected to me
\n']
```

Figure: RDD transformation for Fall 2017 negative review data

We then filtered out the stop words from the review text. For this step, we initially used the default stop-words provided to us inside the project folder, but then used word-count analysis to provide us a more relevant set of stop words, as discussed later on in the report.

```
>>> fl17neg.first()
[1__negative__the homework theoritical thought problem 3 lot
theoritical expecting introductory questions bit disappointed
homework time forward textbooks content homework unexpected
\n']
```

Figure: Removing default stop words from the review text

<u>**Preliminary Models**</u>

For the first model, we implemented a bag-of-words model using the default stop words and default sentiment word lists of positive and negative words. Later on, the default lists were replaced with more refined lists to improve the model.

To determine the sentiment score for a particular review, we used the following formula.

$$Sentiment\ Score\ =\ \frac{Number\ of\ Positive\ Words - Number\ of\ Negative\ Words}{Number\ of\ Positive\ Words + Number\ of\ Negative\ Words + 1}$$

Intuitively, the sentiment score is normalized from -1 to +1, with -1 implying that the review text contains only negative words, and +1 implying that the review text contains only positive words. A ***+1*** constant was added to the denominator to cover edge cases in which a review contained 0 sentiment words. We then grouped all the reviews into their respective true labels (positive, negative, neutral, and not-labeled), and compared the sentiment score metric to their true labels. The plots of our results are shown below.

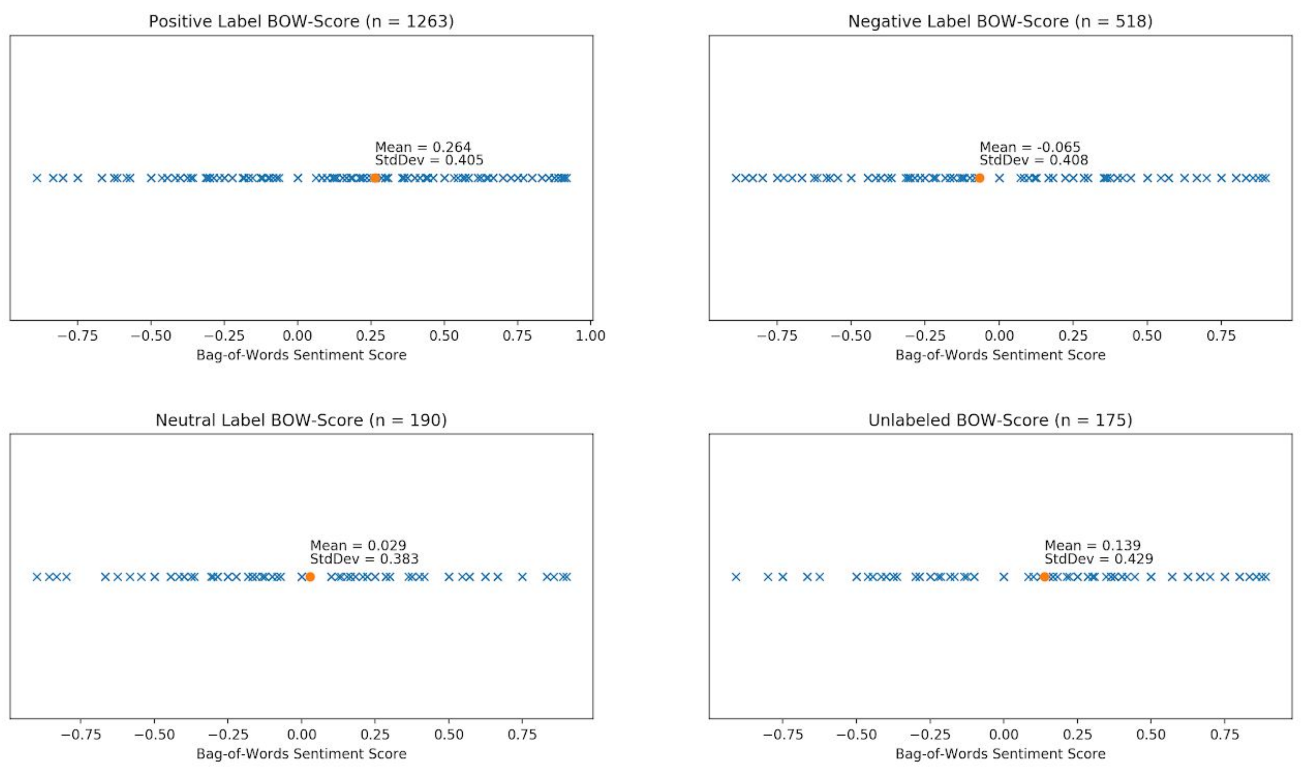

Figure: Plots of sentiment-score metric as compared to the true sentiment labels

Generally, as expected, a positive true label was associated an overall positive sentiment score, and a negative true label was associated with an overall negative sentiment score. It can be seen that the sentiment scores were generally skewed more positive, indicating that reviews with slightly negative to neutral scores were more likely to be labeled negative, while reviews with positive scores were more likely to be labeled positive. In other words, it seemed that generally people were kinder in their words than their actual labels would suggest.

Updated Bag-of-Words Model

To further refine this basic Bag-of-Words model, we looked to update the stop-words and positive and negative lists of words with information relevant to the course reviews, and not just the English vocabulary as a whole. This reprocessing step required visualizing the most frequent words, finding words that contribute no significance to the sentiment, adding those to the list of stop words, and iterating on this process many times to update the list. The following histogram shows the most frequent words across all reviews after removing the initial list of stop words.

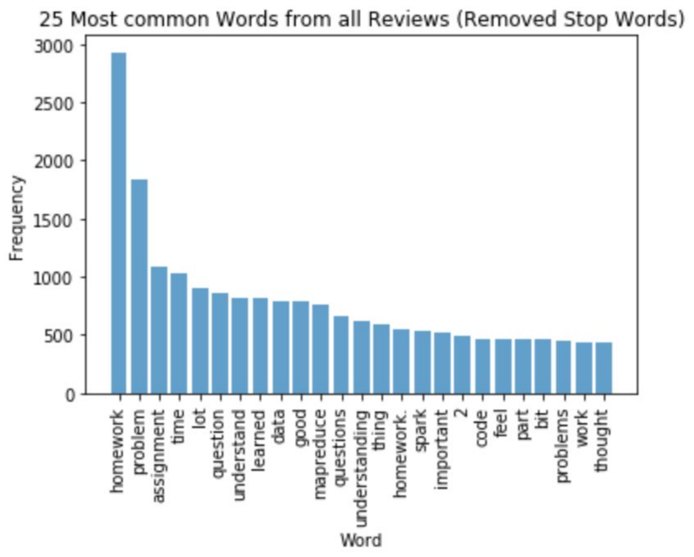

Figure: Histogram showing the most frequent words used among all reviews

Clearly, even after trimming the reviews of basic stop-words, there were even more words that can be added to the list, such as “assignment”, “question”, and “mapreduce”. We repeated this process three times, each time removing words we deemed to have little to no contribution to sentiment.

To update the list of positive and negative list of words, we cross-referenced the histograms of the overall word frequencies (shown above) with histograms showing the most frequent words of positive and negative reviews. Generally, if a word scored high frequency on both the overall histogram and the positive histogram, then that word would likely be associated with a positive sentiment in this context, and the same applies when considering the negative words list as well. Shown below are the respective histograms of the positive and negative word frequencies.

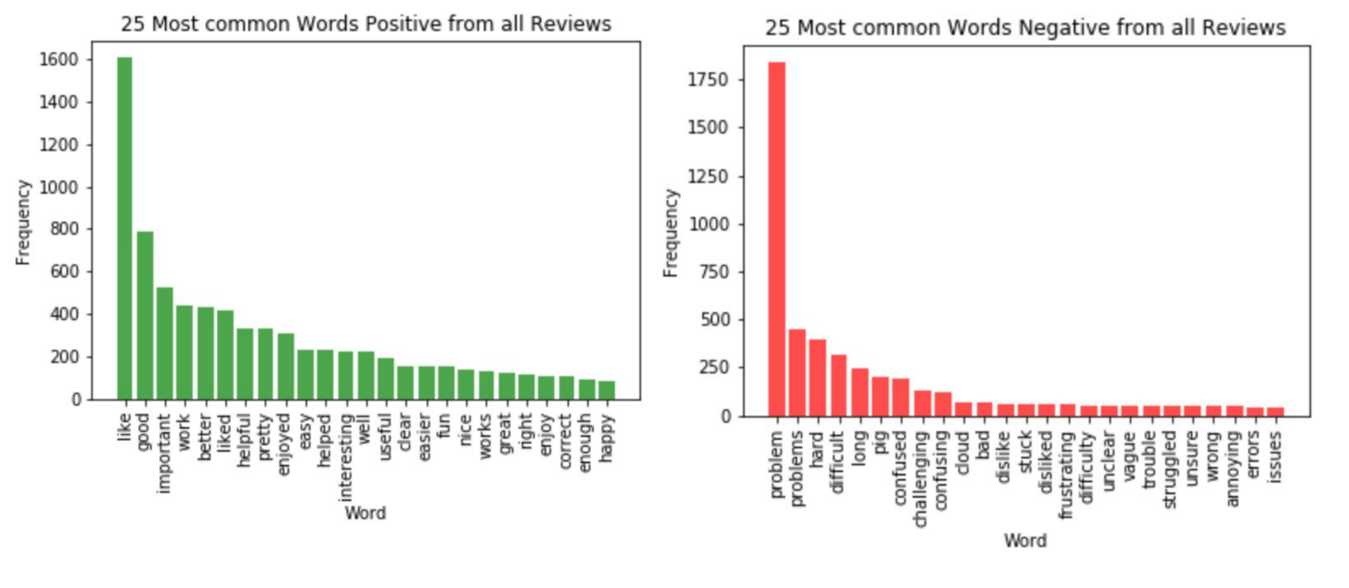

Figure: Histograms showing the most frequent words used in positive and negative reviews

We then updated the bag-of-words model to reflect these additions and parameter changes, and received the following results.

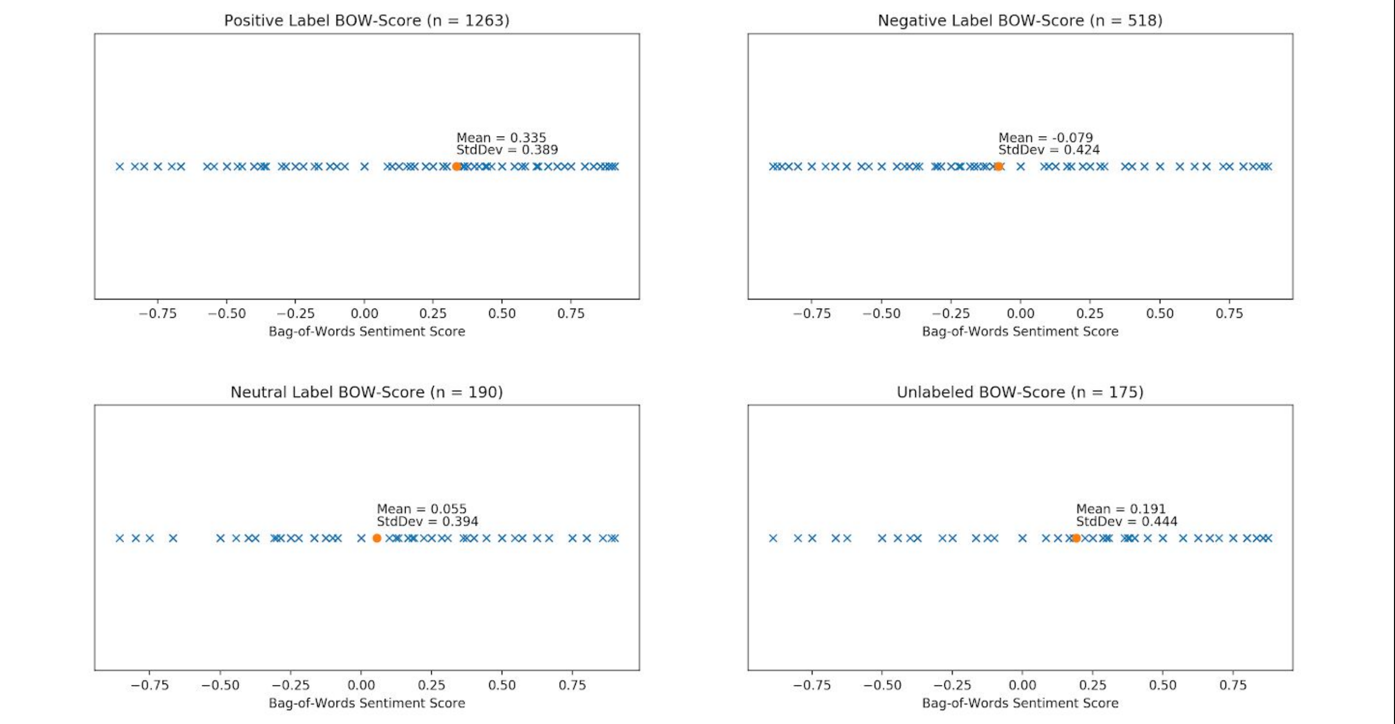

Figure: Updated bag-of-words model using the refined list of stop-words, positive, and negative words

Although there was not a significant difference between this updated model and the old model, this still represents an overall improvement from both a qualitative and quantitative perspective. It can be seen that the average positive label score is now more positive (0.335 vs. 0.264), and the average negative label score is more negative (-0.079 vs. -0.065), suggesting that there is now greater separation between the positive and negative average scores. Greater separation between groups implies greater distance between clusters of data, resulting in a model that is better able to differentiate between positive reviews and negative reviews.

Applications

After refining our models on the smaller dataset of course reviews, we decided to apply these polished models on the larger Amazon dataset publicly provided to us. First, we tested our model on a smaller Amazon dataset, the Amazon Instant Videos dataset, containing just over 37,000 reviews. Our testing procedure was as follows:

1. Hold out 20% of the dataset to test, and use the other 80% to train
2. Calculate the sentiment score for each training review using the bag-of-words approach
3. Find the average sentiment score for each label (1 star, 2 star, 3 star, 4 star, 5 star)
4. Calculate the sentiment scores for the testing dataset, and designate a predicted label based on the sentiment score's closest neighbor average (from step 3, shown below).

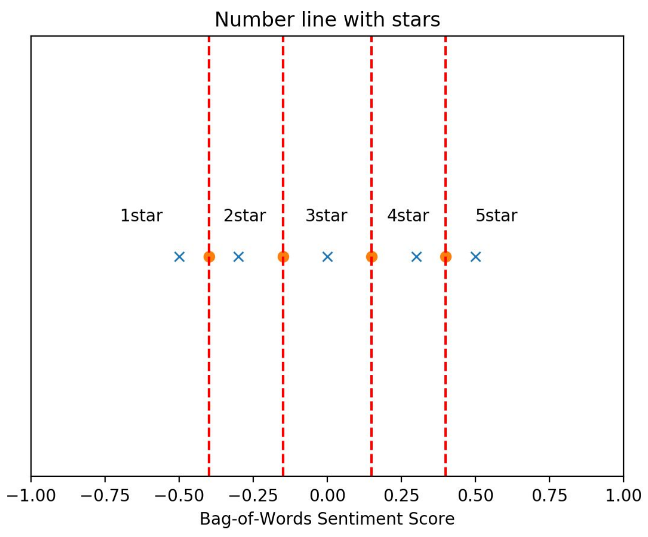

Figure: Boundaries are determined by the midpoints between the average sentiment scores for different labels. Average scores are shown as 'x' and the midpoints are shown as orange dots.

5. Calculate the error comparing predicted label and actual label for the testing dataset

The results did not appear too promising, as we received a 59% error. Comparing this to the baseline error (from random guessing) of 80%, this model did, however, seem to suggest that the bag-of-words approach provides useful information gain. We then modified the labels such that any score equal to or greater than 3 was labeled positive, and any score equal or lesser than 2 was labeled negative. We ran the model again, and received a 29% error. This result, compared with the baseline error (from random guessing) of 50%, appears to support the results the idea that a bag-of-words model, while simple, does provide some value to predicting sentiment.
We then decided to run this binary classification model on the Amazon book reviews (over 8 million reviews), and received an overall error of 28%, consistent with the results from the smaller Amazon dataset. Even for such a simple model, the process took over 2 hours. The results are shown below.

	1-star	**2-star**	**3-star**	**4-star**	**5-star**
Average Sentiment	0.036	0.079	0.181	0.304	0.382

Figure: Results of the bag-of-words model on the books dataset, including sentiment score bounds

Possible sources of error that arose from this model include the disparity in sample sizes of differently labeled reviews and equal weighting of positive and negative words. Because there was much more 5 star and 1 star reviews than other reviews, it seemed to indicate there was a large amount of selection bias in the data. With the large disparity in sample size amongst labels, our method for determining the classification bounds did not account for the large variations in standard deviations among groups, and therefore was not able to correctly predict reviews that deviated too far from their label average. Additionally, the bag-of-words model, without any TF-IDF implementation, weighted each sentiment word the same. For example, the words “bad” and “terrible” are both labeled as -1 (for negative), but should in reality have different weights, since “terrible” is a stronger word than “bad”.

From a qualitative standpoint, we picked the bag-of-words model to generalize and work across vastly different data sets, such as course reviews and Amazon reviews, so we were not surprised that as a result, the model was not particularly accurate for any one dataset (bias-variance trade-off). Features that were relevant for one dataset may not be relevant for another, and vice versa. For example, the types of stop words that should be accounted for was much different in the course review data than in the amazon book review data. A comparison of the stop-words for the two data sets is shown below.

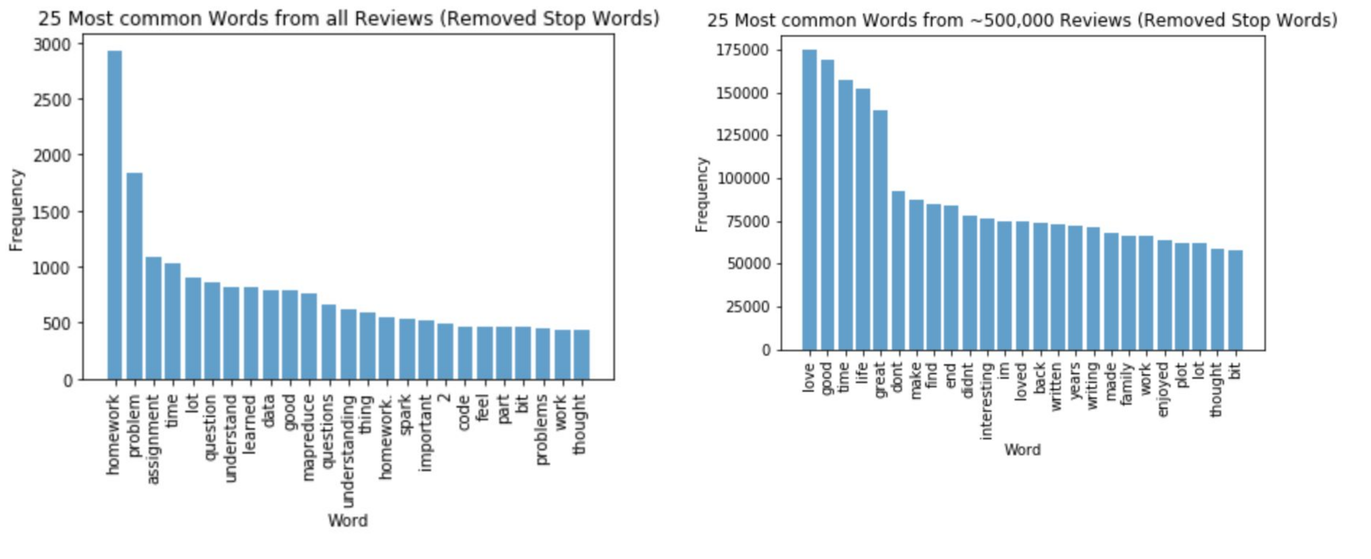

Figure: Comparison of stop-word variations between course review dataset (left) and Amazon review dataset (right) after filtering.

As shown, the Amazon review data had a much greater variety of stopwords, such that even after the filtering step we applied previously, there were still many more stop-words to be added, unlike in the case of the smaller course review dataset.

Another consideration is that sentiment analysis is difficult to objectively classify, since different people have different tolerances and ways of expression. Generally, we noticed that people are kinder in their words than in their actual rating, since reviews with noticeably high sentiment scores were often rated positive, but slightly positive/neutral scores were often rated negative. However, this is more of an observation than a rule, and varies from person to person, and dataset to dataset. Therefore, it was not entirely surprising that the models achieved roughly a 30% error, since a large part of that could be attributed to the subjectivity of sentiment itself.

TF-IDF & Machine Learning

We were interested in determining whether or not we could create a rule-based model that would provide an alternative to machine learning algorithms in certain specific use cases. To that end, we utilized term frequency-inverse document frequency (TF-IDF) to represent our data. There were a significant number of data cleaning steps that we had to implement; without completing each step, the results of our project would be negatively impacted. We relied on tokenization of our data, followed by the removal of stop words, correcting spelling errors, and finally lemmatization. We utilized Spark to calculate the TF-IDF values for our data. The TF-IDF scores were the basis for both our rule-based model and our machine learning model; before we could use them, we vectorized the scores to then apply them to our models.

First, we used TF-IDF as a backbone for our rule-based model. While the individual TF-IDF scores gave information on word importance in a specific document, we also needed the semantic information in the positive / negative text files. With both pieces of information, we were able to create a potential rule that could be used to predict the aggregated sentiment of a reflection. Our rule-based approach would calculate the sum of the product of each word's TF-IDF scores and its sentiment (-1, 1, or 0). This raw score would also be mapped to a final prediction based on inputs that would serve as the positive and negative thresholds. Scores above the positive threshold would be classified as positive and scores below the negative threshold would be classified as negative.

This TF-IDF rule-based model was compared to a classic machine learning sentiment analysis problem. Logistic Regression is a common learner utilized, so we wanted to compare our results to the accuracy of a logistic regression model. We took advantage of Apache Spark's MlLib library and its Logistic Regression with Stochastic Gradient Descent (SGD) to train and test our model. We divided our dataset into a train and test model with 20% of our data relegated to our test dataset.

Our rule-based model used the training data to determine appropriate positive and negative threshold values that would optimize our accuracy. With the rule-based model, we ended up with an accuracy of 67.8%. Our logistic regression model resulted in an accuracy of 70.9%. Both scores align with the general expectations that sentiment is a relatively difficult thing to determine and it is common for there to be disagreement present between different individuals as to the sentiment of a document.

Predicted	0	1	Total
Actuals			
0	3	102	105
1	10	233	243
Total	13	335	

Table: Confusion Matrix for Rule-Based Model

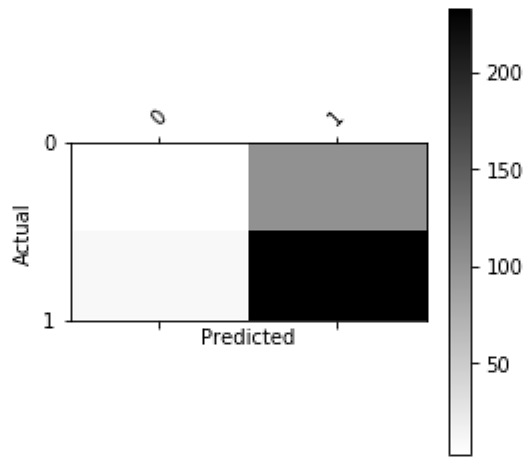

Figure: Confusion Matrix for Rule-Based Model

Our data revealed that our general idea of utilizing TF-IDF and a rule-based approach was not unfounded. With an accuracy relatively similar to that of our logistic regression model, for certain simple examples, it is a viable option. That being said, a TF-IDF rule-based approach suffers when the corpus of documents becomes more complicated. The major drawback of TF-IDF in general is the loss of context when the values are vectorized. While this may have impacted the machine learning model, it especially played a large impact on the results of the rule-based model and our ability to get accurate results. In addition, our logistic regression model would have benefitted from the utilization of certain techniques such as hyperparameter tuning, adjusting the threshold value for the classifier, and other modifications to boost accuracy.

Future Directions

Given additional time, we would have liked to run the bulk of our computations on Amazon EMR. For a lot of small datasets, we opted to test locally before running on the cloud. Even so, it was difficult to transform the Python scripts for local testing into PySpark applications that could be run on the cluster. Additionally, it would be useful to consider a more holistic view of the text data, such as bi-grams, and sentiment trends throughout a semester. Additionally, because we initially chose the bag-of-words model that generalizes well across data sets, we were not able to fine-tune parameters specific to any given dataset (to maintain generality). If we were optimizing for accuracy in the Amazon review dataset, we would consider n-grams, weighted sentiment scores, and employ a machine learning (possibly perceptron linear classifier) algorithm to determine the best classification bounds (rather than using an average label). Furthermore, we would be more strict about preprocessing and filter out misspelled words (using nltk packages) and account for acronyms amongst other anomalous phrases.

With regards to the TF-IDF based models, with more time, we could have investigated the Word2Vec encodings that would have preserved the contextual information of our documents. Such an encoding could have played a large role in increasing the accuracy of our models. In addition, we utilized a supervised classification model; in theory we could have attempted to determine sentiment via the use of exploiting semantic structures within our data by utilizing an unsupervised learning model that would have separated between positive and negative classes.

Conclusion

Sentiment analysis has the power to expose patterns in text data and reveal various layers of opinions. Any useful model must be efficient and take into account the variety of text in a dataset, as well as the differences in text across datasets. A simple bag-of-words model generalizes well among all text data, but was shown to be unsuccessful in accurately predicting sentiment across various datasets. A more robust model, such as those involving TF-IDF calculations, is much more nuanced but also much harder to implement, especially under reasonable amounts of time. Should a model successfully and consistently predict sentiment of subjective reviews, it would open new ways to optimize business, classroom, and customer experiences.

References and Data Sources:

http://jmcauley.ucsd.edu/data/amazon/links.html

CSE427S TA’s

Professor Neumann

Stack Overflow

PySpark Documentation