Hanghang Liu: hanghangliu

Swetha Reddy Nathala: swethareddy

Andy Yuan: aayuan

Final Project for CSE 427S:

Sentiment Analysis

Development Using Small Data:

Motivation:

Sentiment analysis is an excellent source of information and can provide insights that can determine a marketing strategy, improve customer service, generate predictions, and much more. Our motivation for this project is to create a sentiment prediction program that can analyze a given homework review to determine if the review was positive, or negative. If this program is done properly, we should be able to accurately predict what label is correct for a review. Our hope is that our implementation of the sentiment analysis program achieves higher than 70% accuracy. The plan is to use the text processing features of HIVE to investigate the sentiment of the homework review data. The results of our sentiment analysis can tell a reader which homework assignments were more positively viewed and vice versa.

Documentation:

Our first approach was to use bag-of-words with one word, however accuracy was only 58%. Therefore, we switched from one word to two-word analysis. The approach for sentiment analysis we chose was n-gram, specifically bigram. We didn't use Hive's provided n-gram function, but our owned user defined function. The function still is a bag-of-words algorithm, but we analyze the words by pairs.

To further increase our accuracy, we added or removed the following words from our lists:

	Added	Removed
Positive Words	liked, learn, learned, understood, understand, understanding, simple, educational, remedied, simplicity, satisfaction, easily, detailed, popularity, luckily	like, work, variety, tough, tougher, toughest, recommendation, recommendations, ready
Negative Words	hardest, harder, unfair, vaguely, average, wacky, hijinks, awkwardly, typo, typos, random, busy, ambiguously, longer, pitfall, pitfalls, overhaul,	pig, manipulate, problem, break, cloud

	incompatibilities, verbose, consuming, tough, tougher, toughest	
Stop Words	Pig	liked, better, unfortunately, useful, appropriate
Special Words	ain't, aren't, behind, can't, cannot, couldn't, despite, didn’t, doesn't, don't, except, hadn't, hasn't, haven't, instead, isn't, least, never, not, nothing, shouldn’t, wasn't, weren’t, won't, wouldn't, yet	

Most words were removed from positive and negative lists since they were too ambiguous in meaning or were in the wrong list entirely (positive: tough => negative: tough). For example, the negative word ‘problem’ could mean both problem number or that they had a problem (trouble) on the homework assignment. Also, some cases of words were not included in the original list (original: hard, add: harder, hardest) so they were added to the list. For stop words list, the word ‘pig’ was added since it was referring to a programing tool not a negative word. On the other hand, ‘liked’ was removed from the list since it had a more positive connotation. Also, the words ‘better’, ‘useful’, ‘appropriate’, and ‘unfortunately’ were removed because they were already in the positive word list. Special words are words that were listed as stop words but we chose to evaluate them differently. These words were like a negation of the words that follow them (not good, wasn’t hard, etc.). If the following word is positive, it became negative in meaning; if the following word is negative, it became positive in meaning. For instance, we were given the sentence: “However, it is annoying not having a nice IDE to detect syntax errors and the error messages are not helpful”. Since we have the special word not, all words following not would be given not_word (not_having, not_a, not_nice, etc.) until we reach a punctuation mark, ‘and’, ‘or’, or ‘because’. In this case, we stop after reaching ‘and’ for the first not and ‘.’ for the second not. As you can tell, this generates a lot of unnecessary words with underscore, but when we evaluate them, they are treated as neutral so we can ignore them. The reason we extend until we reach a punctuation mark, ‘and’, ‘or’, or ‘because’ is that we feel the negation word negates the entire part of the sentence not just the word that follows it. Without extension, the ‘not’ negation would never reach the positive word ‘nice’.

To decide if a review was positive, negative, or neutral, we used counters for positive and negative for each n-gram or pair of words. Each word is only evaluated one time. First, we find the label for both words in the n-grams. If both words are neutral or the first word in the n-gram is neutral, the label for this n-gram is neutral. If only the second word is neutral, then the n-gram is labeled whatever label the first word of the n-gram has. If the first word is positive and the second word is positive, then n-gram is labeled as positive. On the other hand, if the second word is negative, then the n-gram is labeled as negative. If the first word is negative and

the second word is positive, then the n-gram is labeled positive. All other cases will be labeled as negative. If one of the words have an underscore, then they are analyzed differently. If one of the two words in the n-gram have an underscore, we don't analyze the word that doesn't have an underscore (set it to neutral). We then splitting the word with the underscore into two words. While ignoring the first word (since it is a word only for negation), if the second word is positive, then the n-gram is labeled as negative; if it is negative, then it is labeled as positive. After the n-gram is labeled, if the n-gram is positive or negative they are added to their respective counters as mentioned before. After all the n-grams have been labeled and added to the counters, if the counters for negative and positive n-grams were equal, the review was labeled as neutral. If the counters for negative n-grams was higher, then the review was labeled as negative. If the counters for positive n-grams was higher, then the review was labeled positive. In conclusion, our algorithm analyzes pairs of words, counts the number of positive and negative labeled pairs, and finally compares these counts to determine what label each individual review should receive.

Results and Discussions:

Step 1: Basic Analysis

a) Homework assignments 2 and 3 were the most positive with a difference of only 1 review. Homework assignment 7 was the most negative (most likely because it was assigned before spring break).

b) The five most frequent words:

Positive		Negative	
good	403	hard	315
interesting	243	difficult	258
pretty	211	long	167
easy	206	confusing	122
well	203	frustrating	109

Notice that these frequencies depended on the word lists we modified. For example, problem (843) and problems (290) had the high frequency as negative words, but we cannot tell if the it was for problem # (neutral meaning) or problem encountered (negative meaning), so they were removed from the list. They are added back to the list for amazon reviews since there is no confusion there. This table assumes we have modified our word lists. After removing words with ambiguous meanings, our results make sense.

Step 2: N-grams

a) In our method to create a hive script to find bigrams and trigrams of our reviews, we only needed to pass the value that determines whether to run bigram or trigrams. There is no need to pass the file path to the text documents, since all the files are uploaded to a Hive table due to preprocessing. This is necessary step since punctuation would throw off our n-gram. The method we chose to pass parameters to our script is to use hiveconf.

b) The three most positive n-grams:

2-gram	3-gram

(homework, good)	61.0	(homework, learned, lot)	12.0
(homework, pretty)	52.0	(learned, lot, assignment)	11.0
(enjoyed, homework)	37.0	(homework, helpful, understand)	10.0

These frequencies were calculated after we had preprocessed the data, and modify the lists. These results make sense, because our reviews are about homework assignments. Therefore, it is very likely that we would have positive words associated with the word homework or assignment quite frequently in the reviews.

c) Total Sentiment Scores ($s = \frac{positve-negative}{positive+negative}$):

Semester	Sentiment Score
FL2016	0.144348
SP2016	0.269841
SP2017	0.291803279

Based on our results all semesters were more had a more positive emotion than negative. SP2017 had the highest sentiment score, while FL2016 had the lowest sentiment score.

Step 3: What are the Favorite Topics?

a) Homework assignment in the first half (hw1-hw6) of the semester tended to be more positively reviewed.
b) Homework assignments focused on theory (hw1, hw2, and hw7) were only slightly more positively rated than assignments focused on MapReduce (hw3-6, hw8), but only by a few reviews. Both had more positive emotion than the SPARK and PIG assignments (hw9 and hw10). This is more likely do to the fact that the first few homework assignments (hw1, hw2, and hw3) were easier since they were introducing the students to the concepts. These three homework assignments had a lot more positive reviews compared to other homework assignments. Also, it is important to note that only the fall 2016 semester had any reviews for homework assignment 10.

Step 4: How good are your Sentiment Predictions

Our raw accuracy of positive and negative predictions was 72%. If we include neutral reviews the accuracy drops to 56%, however it is important to note that most incorrectly labeled reviews were in the neutral category. For example, the review, hw2_review_B95CA9, content was:

"This was a really fun assignment! Maybe it's only because it was much shorter and easier than expected, but I enjoyed it, and I can see this being useful practice for the upcoming exams. I'm also very glad that we get stubs to start our code off with; it made coding really easy (almost a matter of copy-paste from the wordcount example). I'm sure I would have a much harder time otherwise.".

However, this review was labeled as neutral despite it's positive sentiment. Other errors included duplicates of the file contents for different assignments, adding a list of adjectives to reach 50 words, reviews that didn't talk about reviewing the homework at all, and reviews that were predicted wrongly do to misspelled words. Our accuracies for the following table were calculated after removing most faulty reviews.

Semester	Label	Accuracy
FL2016	Positive	0.928571429
	Negative	0.909090909
	Neutral	0.8
SP2016	Positive	0.88
	Negative	0.8
	Neutral	0.6
SP2017	Positive	0.757281553
	Negatives	0.786516854
	Neutral	0.574468085
Total average accuracy:		0.78176987

Based on the results, our accuracy (after removing the erroneous reviews) was 78%. So, our program was relative good at predicting the original moods.

Step 5: Does your System agree with your own Emotions

Using our sentiment prediction program on our reviews, we had an average accuracy of 68%. Majority of the results agree with the moods we had ordinally assigned our reviews. Most of the loss of accuracy was from neutral review sentiment predictions.

Cloud Execution Using Big Data:

Documentation of Big Data Problem:

The Amazon EMR dataset that we executed our sentiment analysis prediction program on was about product reviews. There were 82.83 million reviews, but for the sake of testing we only used around 1.3 million reviews (82.83 million reviews would have taken too long to upload the files). There is a possibility of skew, but we can ignore this due to the size of our sample. Inside even this much smaller sample dataset is a variety of products which have no common theme. The dataset has the Big Data properties of large amount of records, structured format (json), and labels. These three properties qualify the dataset as a Big Data. The only features we were interested in were the star ratings and the content of the review itself. Unfortunately, all the reviews data is in a json format, so we had to preprocess to extract the fields we wanted (ratings, and reviewText).

Documentation of Cloud Execution Process:

To setup the cluster we had do several things. Firstly, create a cluster in AWS EMR, pairing with a pem key pair (amazon's private key). In EC2 dashboard, Network & Security tile, Security Group, choose the corresponding EMR master instance, and choose the inbound property. Add a SSH rule. Then, go to EMR dashboard, choose the cluster we created, follow the instruction to connect to the master node using SSH. Finally, create a bucket in AWS S3. We will upload files into this folder. With this, our cloud execution is setup.

Our goal was to predict what star rating a review should have using the algorithm discussed previously, so we needed to modify our code. Instead of having positive, negative, and neutral as possibilities, we had 1 to 5 stars. If the number of positive n-grams and number of negative n-grams were the same, then the review was given three stars. If the number of positive n-grams was two n-grams higher than negative n-grams, then it was given five stars. If the number of positive n-grams was only one n-gram more than the negative n-grams, then it was given four stars. If the number of negative

n-grams was one n-gram more than the positive n-grams, then it was given two stars. If the number of negative n-grams was two n-grams more than the positive n-grams, then it was given one star. Using these threshold, we could predict what stars to the review should get.

Results and Discussions:

Our accuracy was 42.4% on predicting the exact same star rating as the amazon review with an average deviation of 0.91. The total runtime for processing 1.3 million reviews was 21 minutes and 50 seconds. This was quite faster than what we had expected. Our results were not as accurate as our homework review predictions, since our lists of words were not exhaustive enough. Our current list is perfect for homework reviews. However, the amazon reviews are on a multitude of products, so it is hard to create a 'complete' list of possible words that need to be scored. Remember our approach is still a bag of words. To improve the accuracy, we should to go through a dictionary and add more words to the positive, negative, special, and stop words lists. Also notice that we ignore quite a few neutral words. If we switch to a polarity system, we should be able to increase our accuracy.

All code is in Swetha's repository