Spring 4-22-2019

Understanding Homework Reviews Through Sentiment Classification

Zachary Mekus
Washington University in St. Louis

Washington University in St. Louis

McKelvey School of Engineering

Department of Computer Science and Engineering

Thesis Examination Committee:
Marion Neumann, Chair
Yevgeniy Vorobeychik
Ron Cytron

Understanding Homework Reviews Through Sentiment Classification

by

Zachary Mekus

A thesis presented to the McKelvey School of Engineering
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

May 2019
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

I want to thank Professor Neumann for advising me on this thesis and always providing great support and advise. To all my professors and classmates from five years at Wash U, I am grateful for the inspiration, help, and experiences. Finally, I want to thank my parents for always supporting me in everything I've done.

Zachary Mekus

Washington University in Saint Louis
May 2019

ABSTRACT OF THE THESIS

Understanding Homework Reviews Through Sentiment Classification

by

Zachary Mekus

Master of Science in Computer Science

Washington University in St. Louis, May 2019

Research Advisor: Marion Neumann

This thesis uses a naive bayes sentiment classifier to analyze six semesters of homework review data from CSE427S. Experiments describe the benefits of an automated classification system and explore original ways of reducing the number of features and reviews. A new algorithm is proposed that tries to take advantage of aspects of the review data that limit classification accuracy. This analysis can be used to help guide the process of automatically using short reviews to understand student sentiment.

Chapter 1

Introduction

1.1 Motivation

Student feedback plays an important part in assessing student satisfaction. However, in most cases, this feedback will be short and low quality, especially if it is asked for frequently. If they have a choice students will often leave no label, or if they are required to, will leave a label that may not accurately reflect the content of the review. So it is useful to have a system that automatically classifies reviews for later analysis, especially one that is tailored to the domain of homework reviews for a specific class. The problem of classifying student feedback by sentiment has been explored by comparing algorithms [1] and looking at long term trends [3]. This paper uses a relatively large dataset of student homework reviews from a Computer Science class at Washington University in St. Louis over several semesters. Among other topics, this paper analyzes in what cases a machine learning classifier trained on the data is useful over a classifier that doesn't rely on the data, what parts of the feature space are needed for the model to be successful, and what kinds of reviews are important to the model. Specifically, a simple Naive Bayes model with TFIDF features will be analyzed. Rather than attempting to get the best possible accuracy, this relatively simple model will be studied and used to understand qualities of the homework reviews. Hopefully readers will be able to take away a better understanding of this dataset that transfers to similar domains and informs the actions taken to utilize student feedback in the form of free-form .

1.2 Background

1.2.1 Course Review Dataset

After every homework assignment, students in CSE427S "Cloud Computing with Big Data Applications" were asked to leave a message on how they felt about the homework. The homework reviews cover the last six semesters, with the most recent one using 1-5 star review labels, and the previous five using positive, negative, and neutral. The majority of experiments and analysis is done on the five semesters of data. This includes 426 negative, 194 neutral, 788 positive, and 671 unlabeled reviews. Each review contains an average of 4.8 sentences and 424 characters.

1.2.2 Sentiment Classification

The main task considered is classifying whether a review is positive or negative. This problem has been studied extensively[10]. Current state of the art models use neural networks to exploit the complexity of natural languages and perform quite well[17]. The long runtime, need for a lot of data, and lack of interpretability of these models makes them less useful for this specific problem. This paper uses three methods described below, two out of the box algorithms of which do not use training data and one learned one which does.

1.2.3 Rules-based Classifier

One baseline classifier used VADER (Valence Aware Dictionary and sEntiment Reasoner) [6]. This is a rule-based method that classifies test reviews based on a dictionary of positive and negative words and lexical features that is specifically tuned to social media.

1.2.4 Pretrained Classifier

The next baseline is TextBlob sentiment polarity package. Like the rules-based classifier, it does not use training data, but it is the result of a pretrained Naive Bayes model on imdb movie reviews data. While it is trained similarly to how the learned model will be trained, it will not have data specific to this domain.

1.2.5 Learned Classifier

To compare against these baselines, a machine learning classifier will be built that learns to classify new reviews from training reviews and their labels.

Feature Representation A trigram bag of words model is used as feature representation. Preprocessing involved saving the top 1000 total most frequent unigrams, bigrams, and trigrams, where an n-gram is a group of n adjacent words, in the reviews and using these as features. No tokenization or stopword removal is done. Common ways of reprenting these features are prescence, which is a binary variable representing whether or not an n-gram is in the review, and count, the number of times it appears. Instead of these, TFIDF is used as the feature representation because of its general success in text classification due to its ability to make rarer words more important to the classifier by multiplying the term frequency (TF) by the inverse document frequency (IDF).

Inference Algorithm Multinomial Naive Bayes is a probabilistic classifier that is commonly used for text classification. It was also chosen because it is a fast linear classifier that can be easily analyzed. Bayes Rule is used to calculate the probability that a document is of a class and assumes that all features are independent given a class. In this case x_i is the TFIDF score of an n-gram in that review, and p_{ci} is the probability of finding that n-gram in that class based on the training data, and c is the class, positive or negative. A smoothing parameter α is incorporated into the probabilities as a pseudocount to avoid overfitting. $P(y = c)$ is calculated as the proportion of reviews belonging to that class.

$$P(y = c|X) = \frac{P(y = c)P(X|y = c)}{P(X)} \propto count(y == c) \prod_{i=1}^{d} (p_{ci}^{x_i})$$

This can be interpreted as a linear classifier in the two class case.

$$\hat{c} = argmax_c P(y = c|X) = argmax_c \log P(y = c|X) = sign(\log \frac{n_{pos}}{n_{neg}} + \sum_{i=1}^{d} x_i * \log \frac{p_{pos,i}}{p_{neg,i}})$$

The score will be defined as the log probability difference between the positive and negative class. If it is positive, the review will be classified as positive, otherwise negative. The bias term is $\log \frac{n_{pos}}{n_{neg}}$ and offers the flexibility to be adjusted. The log difference in each feature is the weight of a feature.

A more detailed introduction to Naive Bayes can be found in [9]

Chapter 2

Bias of Classifier

2.1 Initial Classification

2.1.1 Setup

A learned classifier is built in this section using the Naive Bayes classifier with TFIDF features. A smoothing parameter of $\alpha = .3$ is used. All model and parameter choices were made before testing, for their common use, simplicity, and history of performing well on similar datasets. Most of the experiments in this paper don't use the full dataset and involve random selection of data points. Unless otherwise stated, experiments were done fifty times with different random seeds to reduce the effect of randomness. Reported data is the mean result, and errorbars in plots represent a standard deviation in each direction of the collected data.

2.1.2 Results

Before looking at the success of classification decisions, an analysis is done here to see if the classifier accurately predicts the correct proportion of positive and negative reviews. While it may be appropriate in some cases to prefer one class to the other, here we assume that if the training and test set have the same proportion of positive labels, the classifier should predict that proportion on the test set. Plots in this chapter use a dotted line to show this ideal. If two classifiers predict the same number of positives, then accuracy can be used as

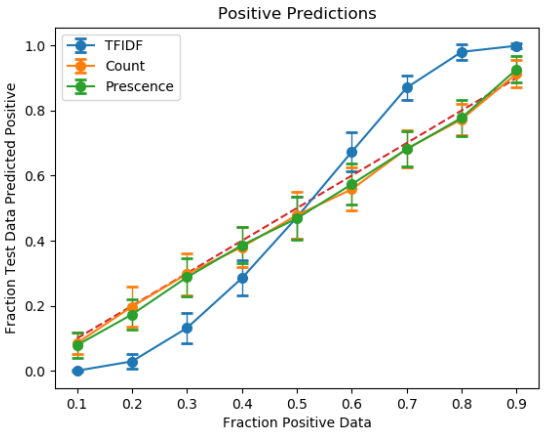

Figure 2.1: Positive predictions with Multinomial Naive Bayes on different vectorizations

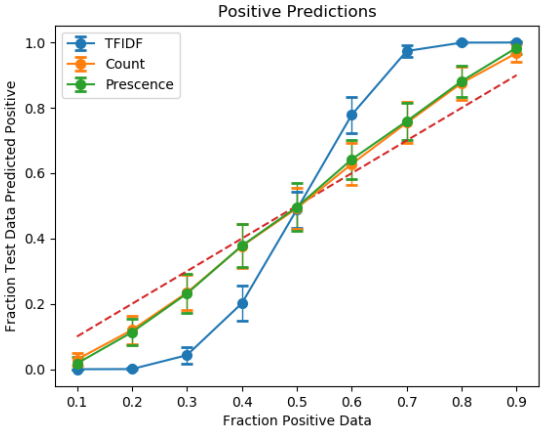

Figure 2.2: Positive predictions with Logistic Regression on different vectorizations

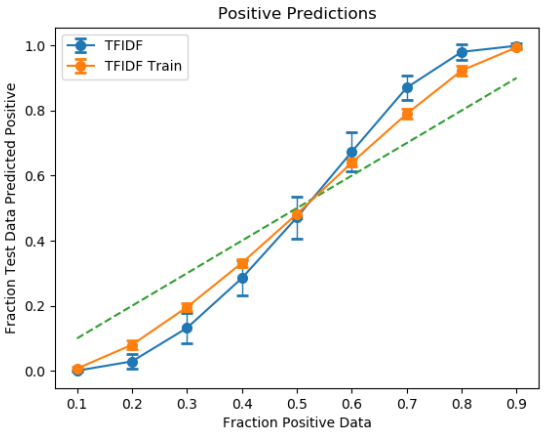

Figure 2.3: Positive predictions with Multinomial Naive Bayes on training sets

a fair measure that equally weights false positives and false negatives no matter the label balance.

The total number of reviews used for these experiments was 448, with a changing randomly selected subset of positive and negative reviews according to different ratios. 10% of the data is split for testing in a stratified way, meaning each test set has the same polarity balance as the training set. Count and prescense refer to two other feature representations described in section 1.2.2.

Experiments show two biases in the predictions using Naive Bayes and TFIDF. The first is the data balance bias. The classifier predicts much more of the label with more training data. The second bias is underprediction bias. At each ratio, the classifier the slight underprediction of positive labels at each ratio. Figure 2.1 shows that the tfidf feature representation is responsible for the data balance bias, while Figure 2.2 demonstrates that the underprediction bias is due to Naive Bayes. The classifier was also evaluated on the training set rather than a seperate test set. As seen in Figure 2.3, generalization to the test set causes some of the bias, but not all.

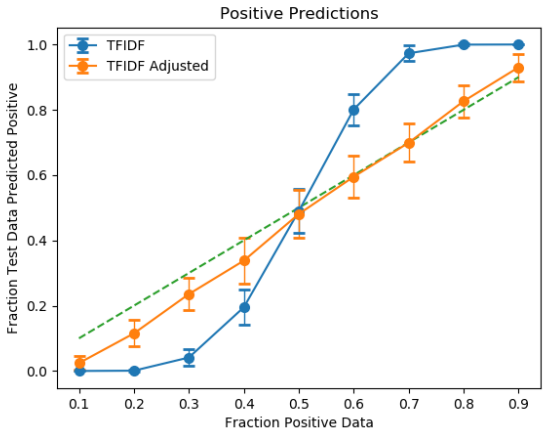

Figure 2.4: Positive predictions with Multinomial Naive Bayes after unbalanced data adjustment

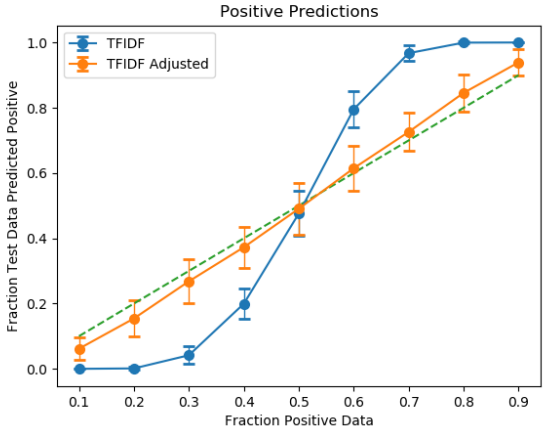

Figure 2.5: Positive predictions with Multinomial Naive Bayes after full adjuststment

2.1.3 Adjusted Naive Bayes

Although Naive Bayes results can be interpreted as a probability, in two class classification it is more useful to look at it as a linear model with a threshold to determine class. The threshold is determined as $\log \frac{n_{neg}}{n_{pos}}$ (the opposite of the bias of the linear classifier) where n_{pos} and n_{neg} are the number of positively and negatively labeled reviews respectively. When the dataset is balanced, the threshold is zero, and it is negative the more imbalanced the training set is towards a higher fraction of positive reviews, meaning a review with the same n-grams is more likely to be classified as positive. Because of a failure of the assumptions of Naive Bayes with TFIDF, the threshold does not work well on this dataset. A simple way to fix this is to adjust the threshold as a post processing step to the classifier. This is implemented here by training a classifier on some training data, sorting the scores of each training point, and choosing the threshold as the average of the points at indices n_{neg} and $n_{neg} + 1$. This adjusted threshold should be closer to the positive predictions because it uses the data instead of the theoretical model that we see is flawed in this case. Figure 2.5 shows the adjustment improves the positive predictions compared to the previous classifier, mostly fixing both biases. Figure 2.4 is the adjustment with the underprediction bias intentionally kept for comparison.

2.2 Unbalanced Bias

2.2.1 Bi-Normal Separation

Bi-Normal Separation (BNS) has been proposed as an alternative to the inverse document frequency portion of TFIDF [5] [4]. The two formulations are

$$BNS = |F^{-1}(tpr) - F^{-1}(fpr)|$$

$$IDF = log(\frac{pos + neg}{tp + fp})$$

where F^{-1} is the inverse cumulative normal distribution function, pos and neg are the term frequencies of an n-gram in a class, true positives (tp) and false positives (fp) are

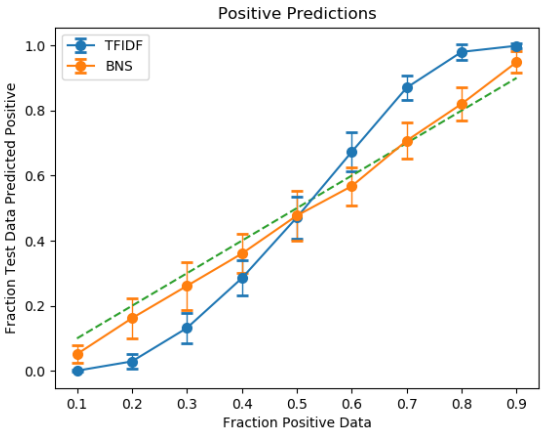

Figure 2.6: Positive predictions with Multinomial Naive Bayes after BNS

the number of times that n-gram appears in a document of that class, and true positive rate, tpr=tp/pos, and false positive rate fpr=fp/neg. Both attempt to achieve the goal of weighting more predictive features higher and less predictive features lower. While IDF highly weights features that occur infrequently in both classes, BNS highly weights features that occur often in only one class. When the classes become more unbalanced, even n-grams that are relatively frequent in a class are seen as rare compared to those of the over represented class. So the features in the overrepresented class that are important get low weights. BNS on the other hand accounts for the difference in number of documents and doesn't run into this problem BNS is compared to the version of TFIDF that is only adjusted for unbalanced data, not for underprediction.

Figure 2.6 shows the results of applying BNS instead of IDF to the term frequency. This result is very similar to 2.4. It mainly fixes the problem, but some of the unbalanced data bias still remains.

2.3 Underprediction Bias

The second bias present is the small but distinct underprediction bias. This is most clear from looking at the unadjusted results for any vectorization in the balanced data case, but it seems to extend to all balances. To try to understand why this is happening, the weights

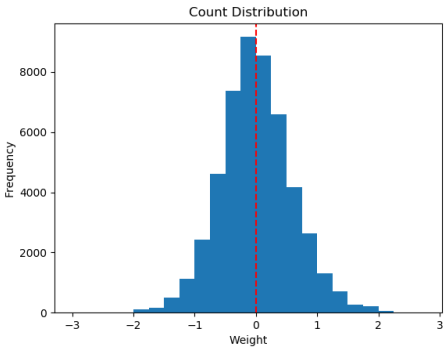

Figure 2.7: Histogram of n-grams by weight

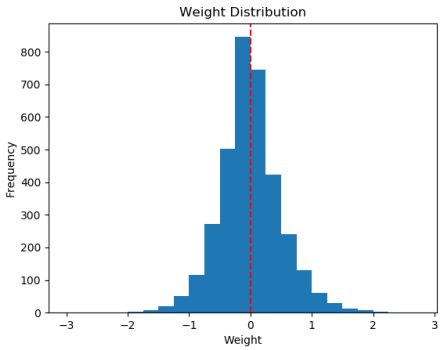

Figure 2.8: Histogram of n-grams by weight, weighted by total tfidf score

	Num Features	100	300	1000	3000
	1	.46	.48	.48	.50
Max N-Grams	2	.46	.46	.46	.51
	3	.46	.46	.46	.48

Table 2.1: Changing Positive Predictions on Different Feature Spaces

of the unadjusted classifier on balanced data are plotted in figures 2.7 (feature count) and 2.8 (weighted by frequency of features). The weight of a feature is plotted on the x-axis and the binned count and frequency of features on the y-axis. The distribution is skewed to the left, and this may explain why too many reviews are classified as positive. An assumption of Naive Bayes is that all features are independent, but that is not the case in reality. The bias seems to come from the complex dependencies between features. Table 2.1 seems to show that using more features and less max n-grams reduces this bias. A similar problem was found in [13], and in the two class case the authors used a similar adjustment. The adjustment does seem to fix this issue, with the positive predictions in the balanced case going from 48.0% to 49.9% after the adjustment.

2.4 Crossvalidated Adjustment

While the adjusted classifier (as described in section 2.1.3) does classify points to better proportions than the unadjusted, there is still a significant unbalanced bias similar to the unadjusted classifier. This is because the training data also contains part of the bias as seen

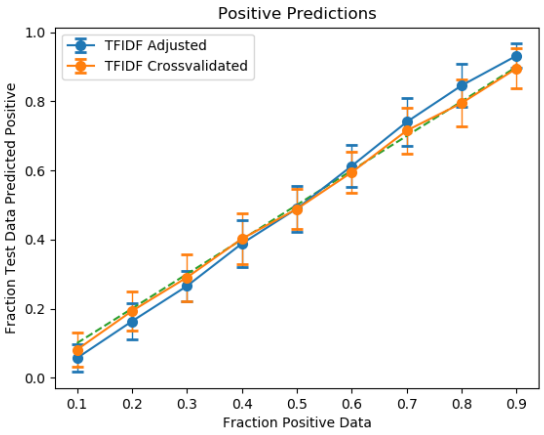

Figure 2.9: Positive Predictions on Crossvalidation Adjusted Classifier Compared to Training Set Adjusted Classifier

earlier in figure 2.3. To solve this, five-fold, stratified crossvalidation is used to determine the threshold. This might work better because the threshold is determined by using the same method on a small validation set trained on the rest of the training set that is expected to behave more similarly to the test set. Indeed, as seen in figure 2.9, this works better than adjusting the threshold just based on training results, and gives almost exactly the predictions expected. In the rest of the paper, this will be used as the default classifier.

2.5 Test Time Adjustment

If the test distribution, or at least the desired positive prediction rate, is known there is a way to always predict the same positive as are expected in the test set. Sort the test predictions by score and choose the cutoff so that the desired number of positive predictions is made. This can be a better solution if you don't want the training distribution to decide the threshold, and it will be used in certain experiments in this paper when appropriate.

Chapter 3

Adjusted Results

Now that we have adjusted the classifier to give the proportion of positive predictions that we expect, this chapter looks at the result of applying the classifier to the review data. Accuracy is used as the main performance metric here, even on unbalanced data because we are confident the classifier gives the expected number of positive predictions. Without this guarantee, using this metric would overvalue classifiers that overpredict the more common class, but if the positive predictions are the same, it is a fair measurement, and it penalizes false positives and false negatives equally.

3.1 Highest Weighted n-Grams

One advantage of the Naive Bayes classifier is its interpretability. Each feature, in this case uni-, bi-, or tri-gram, can be interpreted as a weight to a linear model calculated by $\log P(x|class)$ where $P(x|class) = \frac{\alpha + tfidf(x_i \in class)}{\alpha * count(class) + \sum_x tfidf(x_i \in class)}$. A larger choice of α will cause more smoothing, in general weighting common words more highly. Tables 3.1 and 3.2 show the list of the highest weighted positive and negative features respectively. Although larger choices of α do tend to give more common words, there isn't a large difference even when it is an order of magnitude larger. The difference between the chosen α of .3 and a very low choice and slightly larger one, there is almost no difference in this ranking. Overall, it seems that at least the order of the most important features is not sensitive to this parameter choice.

	$\alpha = 0.0001$	α **= 0.3**	α =1	α =10
1	liked	liked	liked	and
2	liked this	liked this	liked this	liked
3	liked this homework	liked this homework	liked this homework	how
4	hdfs	hdfs	good	can
5	very interesting	very interesting	hdfs	interesting
6	think about	think about	very interesting	mapreduce
7	world	world	fun	is
8	real world	real world	think about	us
9	of mapreduce	of mapreduce	and how	data
10	similar to	similar to	world	me
11	cool	cool	real world	good
12	happy	happy	of mapreduce	hadoop
13	applications	applications	similar to	hdfs
14	already	already	helps me	enjoyed
15	pig and	pig and	understanding of	understanding
16	properties	properties	cool	spark
17	was helpful	was helpful	happy	fun
18	review of	review of	pig and	helpful
19	help us	help us	works	about
20	more about	more about	help us	of mapreduce

Table 3.1: Most positively weighted n-grams with different alphas (chosen is $\alpha = .3$)

	$\alpha = 0.0001$	$\alpha = \mathbf{0.3}$	$\alpha = 1$	$\alpha = 10$
1	spring	spring	spring	spring
2	break	break	break	break
3	spring break	spring break	spring break	spring break
4	issues	issues	issues	confusing
5	frustrating	frustrating	frustrating	issues
6	unsure	unsure	unsure	frustrating
7	the homeworks	the homeworks	the homeworks	unsure
8	expected	expected	expected	expected
9	ta	ta	ta	ta
10	did not	did not	did not	the question
11	supposed	supposed	supposed	did not
12	given	given	given	supposed
13	supposed to	supposed to	supposed to	given
14	were not	were not	were not	supposed to
15	to answer	to answer	rather	rather
16	the time	the time	to answer	to answer
17	annoying	annoying	the time	annoying
18	is too	is too	annoying	spent
19	figuring	figuring	is too	is too
20	due to	due to	figuring	figuring

Table 3.2: Most negatively weighted n-grams with different alphas (chosen is $\alpha = .3$)

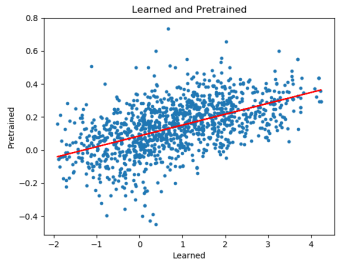

Figure 3.1: Correlations Between Learned and Pretrained

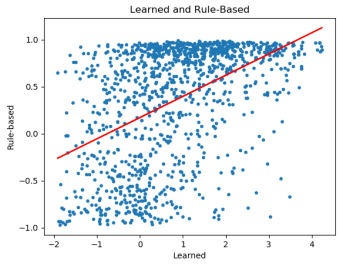

Figure 3.2: Correlations Between Learned and Rule-Based

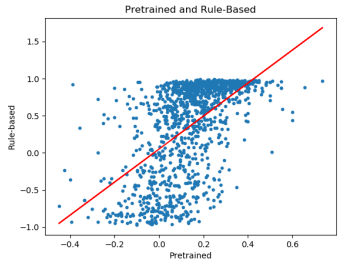

Figure 3.3: Correlations Between Pretrained and Rule-Based

3.2 Correlations Between Classifiers

The classifiers have been introduced in section 1.2.2. Two of them, the rule-based and pretrained classifiers do not rely on the data, while the learned is trained on the data. Figures 3.1, 3.2, and 3.3 show one data point for each review and the score of each classifier. There is a strong positive correlation between every pair, with positively rated reviews from one classification likely to be rated positively by another. The correlation between the pretrained and learned is very strong, likely because they were both trained with similar methods, but just different data. The rule-based tends to be more different from these two, likely because it uses a more unique method to judge the reviews.

3.2.1 Uniquely Classified Reviews

After seeing the correlations between classification methods, it would be interesting to try to understand how they differ. To see qualitative examples of differences, between the the classifier learned on this data and ones that don't, the pretrained and rule-based classification scores are averaged for each review and compared to the score from the learned classifier. The top three biggest positive and negative differences are shown in tables 3.3 and 3.4 respectively. These reviews may give us insights into the unique data in these homework reviews compared to generic sentiment classification. All of the reviews with the most positive differences mention course specific technologies that tend to appear often in positive reviews, but not in a generic sentiment classification model, like "MapReduce", "Hadoop", "combiner", and

Review	Rule-Based and Pretrained Score	Learned Score
I liked this assignment. I feel like it further gave us a really good understanding of how to implement MapReduce. It was good to get more experience building all the components of MapReduce from scratch. I also liked being able to see the pieces of the algorithm come together, as well as the different ways to execute the program. More good insight into how the program works in general. Overall happy with the assignment.	.70	4.17
This homework is very helpful for us to understand and remember the context that we learnt in the last lecture better. I got a better understanding about the map reduce, and learned more about the using and techniques of it. And I also got more familiar to the using of the Hadoop, got used to the basic commend of it.	.60	4.12
Problem 1 is mainly about using combiner. Implementing combiner is similar to Reducer, and I can even use reducer function directly. The last two question of this problem show us how to use TARN Resource Manager Web UI to monitor our MapReduce task, and use data to indicate how combiner works. And the second problem helps me understand how distributed cache and partitioner works in HDFS.	-.31	3.48

Table 3.3: Reviews that the learned classifier scored most positively compared to rule-based and pretrained

Review	Rule-Based and Pretrained Score	Learned Score
This homework took me a long time. I worked hard on it but I did not feel like I had a sufficient background in any of this information. I think spending more time making sure everyone understands the infromation before we have homeworks on it would be beneficial. This homework took me upwards of six hours to complete, and I still do not feel like I understand all of the information real well. I would prefer doing exercises rather than answering questions, but that may just be me.	.41	-1.92
This homework was okay. There were only three questions, each of which only required writing a few commands in the spark shell, so I thought it would be quick. However, I had issues overcoming bugs and interpreting errors. The main problem I had was with the join operation. It would freeze, and it took me a long time to figure out how to fix it. Besides that, most of the other parts of the problems went smoothly though. It was tedious, but now I understand spark better I think.	.39	-1.72
This assignment took a lot longer than expected. It took a while to do the set up described in Lab 6. Another tricky part was learning how all the different Pig commands worked, especially the grouping. I felt like the instructions could have been clearer in that I had some confusion on some of the steps in writing the Pig script. Overall, this assignment was fine but it could have been better if the instructions were a bit more clear	.51	-1.62

Table 3.4: Reviews that the learned classifier scored most positively compared to rule-based and pretrained

"reducer". The reviews with the most negative differences do not have obvious phrases that explain it like the positive differences do, but the language often involves things taking long and many uncertain words, both of which may be specific to negative reviews in this context. These differences show that there is potential that learning on the data will be beneficial in classifying test data.

3.3 Sensitivity to Training Size

A natural question to ask is if our classifier trained specifically on this data can outperform the rule-based and pretrained models that can classify sentiment, but weren't specically trained on these reviews. Adjustments based on the training set were used to calibrate these two models, so it does fit the data, but just to adjust the threshold, not to learn feature weights. Here randomly selected balanced datasets with different data sizes are used with 90% for training and 10% for testing. The training size is reported on the x-axis. Since the rule-based and pretrianed classifiers only need the data for calibration, it is expected that they give the same performance regardless of data size. First, figure 3.4 shows the positive predictions each classifier makes. The learned classifier struggles with only 100 datapoints, underpredicting with high variance, but besides that all of the classifiers predict very near the expected 50%. Figure 3.5 shows the accuracies. As expected, the rule-based and pretrained classifiers have a stable accuracy. The pretrained classifier performs slightly better. The learned classifier improves mean accuracy with every increase in data size with an upward trend that indicates it still has room to grow. With only 300 datapoints, the accuracy of the learned surpasses the others, performing significantly better with enough data. This shows that training a classifier on the dataset is useful for maximizing accuracy as opposed to using a generic model.

3.4 Application to Homework Reviews

The ultimate goal of sentiment classification in this case may be to get new reviews and be able to classify them automatically based on previous data. It is important to know whether

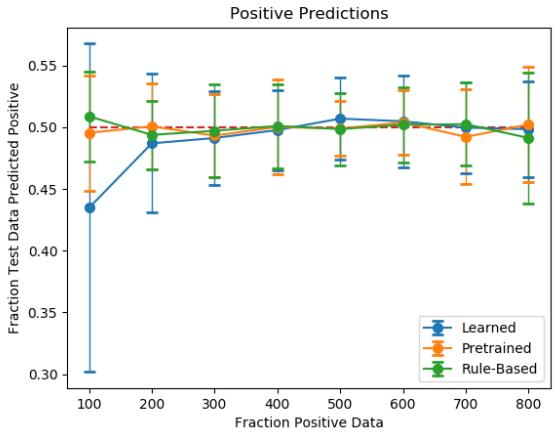

Figure 3.4: Positive Predictions with Different Size Datasets

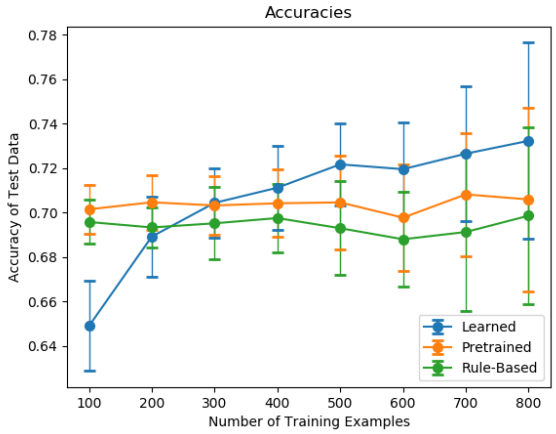

Figure 3.5: Accuracies with Different Size Datasets

	Mean Train	Std Train	Mean Test	Std Test
Positive	233	93	29	13
Negative	126	59	16	10

Table 3.5: Homework Split Statistics

Real	Mean	Std
Pos Preds	.64	.11
Accuracy	.71	.07

Table 3.6: Performance on Real Homework Split

Synthetic	Mean	Std
Pos Preds	.65	.14
Accuracy	.71	.08

Table 3.7: Performance on Synthetic Homework Split

the sentiment can be predicted well or if there is some expected loss of predictive power because the new review will use new terms specific to that homework that therefore won't yet be in the training set. For instance "spring break" is a strong indicator of a negative review, but it likely only appears in the reviews of one or two homeworks and only during the spring semester. To test this, an experiment is done on the homework reviews one semester at a time, for each homework, training on all of the other homeworks of the semester and testing on the left out homework. This is done only on the last three out of five semesters of data because they had more reviews per homework. In total, this was 27 experiments with an average test ratio of 11%, average positive ratio in training of 65%, and a mean total training size of 359. The full statistics on the mean and standard deviation of positive and negative and train and test sizes is in table 3.5. To see how impeding this split is, a synthetic data set is created from the reviews in all three semesters drawn from normal distributions of these same statistics for the same number of expirements. To remove the effect of randomness, this is done 50 times and the average mean and average standard deviation is found. This setup assumes that the number of positive and negative examples in the training and test sets are normally distributed, and each of the number of train positives, test positives, train negatives, and test negatives are independent. The results of the real homework split are in table 3.6 and the synthetic in table 3.7. Both were able to predict the correct number of positives and they have nearly the same mean accuracy. There does not seem to be any drawback to not using training examples from a homework to predict the sentiment of a review from that same homework.

Chapter 4

Feature Selection

There is a lot of work on regularization for sentiment classification[11]. Some parts of the review or certain features may be irrelevant or even harmful to the classifier. Even if regularization techniques don't work, they will reveal how important parts of the review are for classification.

4.1 Word Regularization

If the reviewer includes one strong positive or negative statement, that may be enough to classify the review on its own. The idea here is to train the classifier like normal, but when evaluating a review, only use the top N n-Grams that appear in the review with the highest absolute value weight according to the classifier. Make a classification decision using only these weights. 426 positive and 426 negative reviews were used with a random 10% test set for 50 iterations to get the results in figure 4.1. Even using just the top n-gram gives fairly good accuracy, although the accuracy slowly increases by using more and more information into the review until it appears to saturate at about 13.

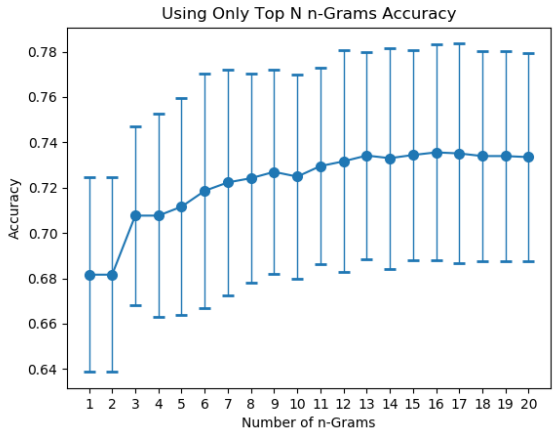

Figure 4.1: Accuracies Using Only N Most Informative n-Grams

Positive Predictions	Train All	Train First
Test All	.50	.50
Test First	.59	.50

Table 4.1: Positive Predictions Using Just First Sentence

4.2 Sentence Regularization

Sentences are natural groupings of words, and it is reasonable that only some are useful for sentiment classification, while others are noisy or misleading. It has been found that selecting for certain sentences can improve performance [14].

4.2.1 First and Last Sentence

One may think that a review's sentiment is summarized by the first and/or last sentence, while the middle tends to be more detailed information that is unneccesary or detrimental to classification. For both the first and last sentence, four classifications are done, with every combination of using the one sentence or the whole for the training and test sets. Like for word regularization, the expirement is run on 426 positive and 426 negative reviews with a random 10% test set for 50 iterations. Tables 4.1 and 4.2 show the percent positive predictions for each combination with the first and last sentences respectively. In both

Positive Predictions	Train All	Train Last
Test All	.50	.48
Test Last	.55	.51

Table 4.2: Positive Predictions Using Just Last Sentence

Accuracies	Train All	Train First
Test All	.72	.71
Test First	.65	.67

Table 4.3: Accuracies Using Just First Sentence

Accuracies	Train All	Train Last
Test All	.73	.68
Test Last	.63	.60

Table 4.4: Accuracies Using Just Last Sentence

Accuracies	Mean	Std
All Sentences	.73	.05
Most Informative Sentence	.71	.05
Random Sentence	.64	.05

Table 4.5: Accuracies Using Just Most Informative Sentence in Test

cases when training on the whole dataset and testing on just the one sentence, the positive probability is significantly higher. This seems to indicate that both the first and last sentence (but especially the first) alone are more positive than the review in general. Tables 4.3 and 4.4 show the accuracies with the just the first or last sentence. In all categories, the first sentence is more valuable information than the last. Also, testing on the whole review is more valuable than training on it. Even the worst category, training and testing on the last sentence does significantly better than randomly guessing. Surprisingly, when using just the first sentence to test, training on the first sentence only actually does better than training on everything, perhaps evidence of the power of regularization and looking for specific features used in the first sentence rather than training on irrelevant parts of the review.

4.2.2 Most Informative Sentence

Rather than assuming the first or last sentence is the most important summarizer, there may be a better way to choose a sentence from a review that can adequately describe the full

Most Informative	Mean	Std
First Sentence	.22	.05
Last Sentence	.23	.04
Random Sentence	.21	

Table 4.6: How Often Most Informative Sentence is a Particular Sentence

review. The method used in this section is to train on the whole dataset, then go through each test review sentence by sentence, score each sentence according to the classifier, and choose the sentiment by the polarity of the sentence with the absolute highest score. This one sentence may summarize the review enough to classify it correctly. Table 4.5 shows that the accuracy goes down very slightly using only one sentence, but is significantly better than testing a random sentence, which has abou the same accuracy as testing on the first or last sentence as in tables 4.3 and 4.4. The fact that it is so close does indicate that almost no information is lost from only using an important portion of the review. Another way to analyze this is to see how often the most informative sentence ends up being the first or last sentence. Table 4.6 shows those results. The first and last sentence end up being informative at almost an equal rate, and since there are an average of 4.79 sentences per review, the first and last sentence are each only slightly more likely to be the most informative as any random sentence.

Chapter 5

Review Selection

Although there has been a lot of discussion in the literature of reducing the number of features in a text classifier (as done in Chapter 4), finding ways to reduce the number of documents is less common. In these experiments, we want to reduce the number of reviews used by the classifier.

5.1 Positive and Negative Balance

Many experiments in this paper have used an equal number of positive and negative examples, but it is not clear whether that is important to get good results. It may be that either the positive or negative reviews offer more information or that the balance doesn't matter. To test this, different ratios of positive and negative randomly selected data with certain ratios are chosen, and a test set of equal positive and negative reviews are chosen. To ensure that an equal number of positive and negative predictions are made despite the makeup of the test set, the predictions are ordered and the lowest half taken as negative and highest as positive as described in section 2.5. The results are shown in figure 5.1. Generally the results seem symmetric, meaning that positive or negative reviews are not structered differently in a way that makes one class more valuable in prediction. A very unbalanced training set decreases accuracy. Although the total size is the same, the model gain in the number of examples in the majority class is less significant than the loss in examples of the minority class. The relative results would not change based on the positive ratio of the test set here because the number of positive test predictions is fixed. If the exact test size is unknown, but it is expected to be the same size as the training set, then different data sizes would

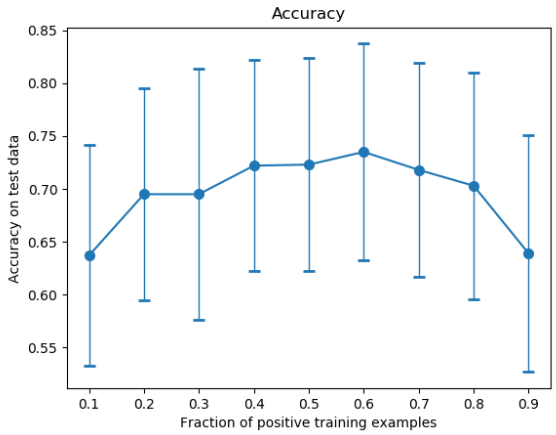

Figure 5.1: Accuracies from an equally balanced training set and from a training set with 50 negative examples and the growing only positive

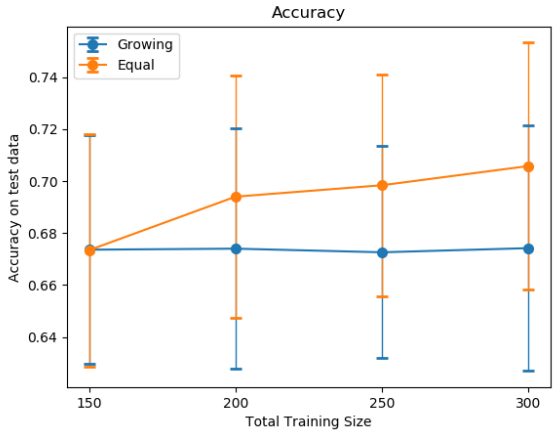

Figure 5.2: Accuracies with Different Training Sizes Selected by Lengths

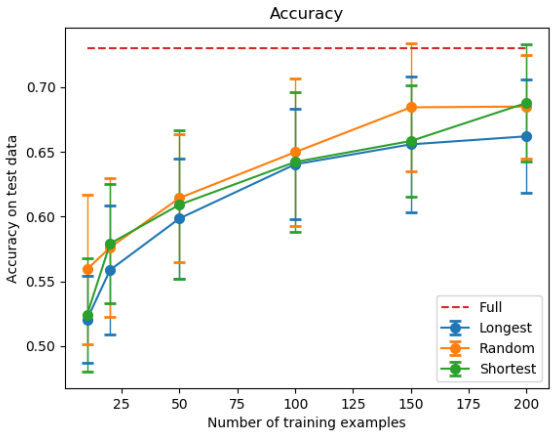

Figure 5.3: Accuracies with Different Training Sizes Selected by Lengths

give the benefit of a more accurate threshold, especially using the crossvalidation in section 2.4, but optimizing the threshold is a different problem than learning good weights from the data. So to learn good weights from limited data, a balanced data set seems better.

Figure 5.2 further enforces this argument. It shows the accuracy of a classifier with different training data sizes with an equal number of positive and negative example and one classifier with a set 50 negative examples and the rest positive. The same process is used to ensure that the positive predictions are balanced. This expirement shows that adding positive examples while keeping the number of negative the same does nothing to improve accuracy, while a growing equal training set continues to improve.

5.2 Length

There is a difference in the lengths of reviews. Since more reviews for training gives better accuracy, it seems like using longer reviews would also provide more information. To test this, the longest and shortest reviews from each class are chosen to reduce the data size. Surprisingly, figure 5.3 shows that using both the longest and shortest reviews reduces accuracy, with the longest doing the worst. This may be due to factors like the longer reviews being lower quality or more likely to be mislabeled. It may also be because the longest reviews

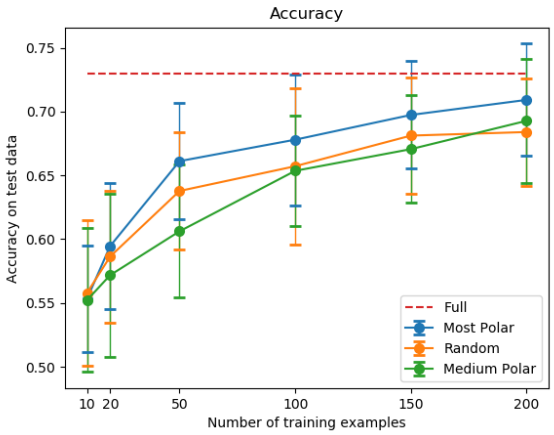

Figure 5.4: Accuracies with Different Polarity Selections

are similar to each other and this cancels out the benefit of more n-grams in each individual review.

5.3 Polarity

It is plausible that a training set with more polar sentiment would be able to generalize better to test data because there will be more of the predictive n-grams that match the class label. This is experiment tests how the polarity of selected reviews affects the classifier. A balanced dataset of different training sizes made up of different reviews is tested. A random subset of reviews along with the most and least polar reviews are selected and compared. The most polar reviews are chosen as the positive reviews classified as most positive by the pretrained classifier, divided by the number of words of a review to find those that express a strong average sentiment. The most polar negative reviews are those classified as the most negative per word. The reviews that are "medium" polar are those in the positive class with the least average positive sentiment or in the negative class with the least average negative sentiment. Figure 5.4 shows that the most polar reviews outperform random, especially with a higher training size, while the medium polar reviews usually lag slightly behind random. This demonstrates that the more polar reviews are more valuable to the classifier. The medium polar review perform surprisingly well considering it was only

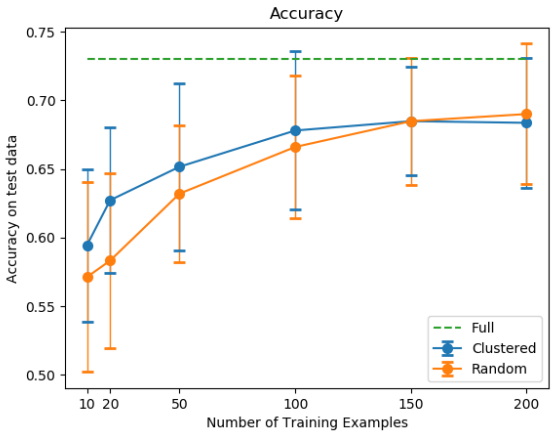

Figure 5.5: Accuracies with Different Training Sizes Selected by Clustering

using reviews that did supported the label the least. This may show that the pretrained classifier is able to identify reviews that use polar words to help the classifier, but even when pretrained disagrees with the label, there is still enough polar information in the reviews to make a decent classifier, maybe because they still contain phrases specifically important to this domain.

5.4 Clustering

The loss of generalization abilities because of less training data is because n-grams in a test review don't have as correct of weights as they would if there was more training data. The most polar reviews would likely give extra coverage to the most important weights, but it could still be choosing similar reviews, learning weights that were already learned well enough. It may be better to find a diverse set of reviews that has good coverage of the feature space. Although redundant reviews are wasteful, finding a set of the most unique reviews would give results of ones that are mislabeled or off-topic. The approach used in this section is to use k-means on the tfidf feature matrix separately on positive and negative reviews, setting k to be the number of reviews to be selected in each class. Then for each cluster, one review is chosen as the one closest to the cluster center. This gives one seperate review for each seperate grouping of positive reviews and negative reviews. Figure 5.5 shows the result

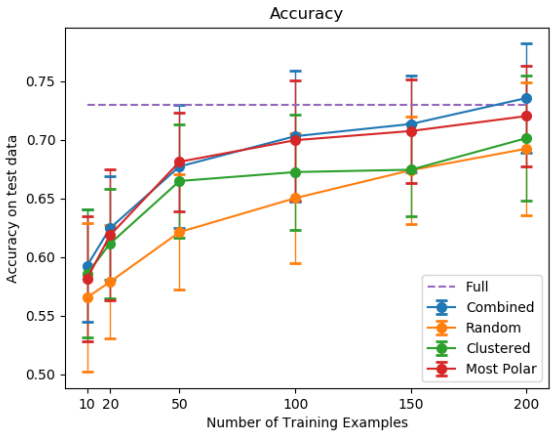

Figure 5.6: Accuracies with Different Training Sizes Selected by Combining Clustering and Most Polar

of this selection process compared to a random selection. The method gives a significant improvement when using very few reviews, but loses its benefit with a larger number of reviews. This is likely because with many reviews in a cluster, the clusters adequately represented reviews using different phrasings or on different topics, but this becomes less meaningful with very few reviews in each cluster.

5.5 Combining Methods

Independently, there is a benefit to using more polar reviews and using reviews that are cluster centers. Since the polar reviews performed best with more data, and the clustering performed best with fewer, combining them may get the benefits of both. To do that, this combined algorithm goes to every cluster, and instead of choosing the review closest to the center, chooses the most polar review in that cluster. Figure 5.6 shows the performance of this combined selection method compared to the random baseline and the two previously successful methods. The combined method does seem to outperform both of the previous methods, especially choosing cluster centers. It outperforms the most polar slightly at almost all training sizes, most notably at the smallest and largest sizes. This method does seem to retain the good properties of both.

5.5.1 Most Representative Reviews

To give a qualitative idea of how this combined selection chooses the best reviews for the classifier, it is used here to select three positive and three negative reviews from the whole dataset to train on. They are all very polar, yet don't overlap in topic. In a way these reviews can be seen as very representative of the dataset because they would be able to train a classifier much better than a random selection.

Positive Reviews:

1. *This homework was fun and entertaining.*
2. *this homework is very helpful for me to review the content of the course!*
3. *i feel homework 2 is very fair in terms of difficulty. if you follow the professor Neumann and understand the concepts about Mapreduce procusdures, you can do well in homework and it will not take you so much time. If the professor keeps asking homework questions like this, i can expect I will learn a lot from this class*

Negative Reviews:

1. *homework 7 is long and difficult. i dislike the questions don't have any hints, it is making a very difficult, i have to flip pages after pages in the book and google a lot to understand the materials so i can answer the questions. Having hints like on the previsous homework, it helps a lot i know where to go and read the materials, understand it and answer the question.*
2. *This homework is too hard for us. Although we can get some hints from ppt used in our class, we can't finish our homework without TA help. And the pyspark command line is too hard to think. For problem 2b, we feel it is too complicate to get answer. For problem 2e, we clound't run out the outcome. And we don't know why. And no TA answer this problem in piazza. We feel desperate!*

3. *This homework took me two day to finish. I used python for spark. Although I have learned python before. It still took me some time to learn how to write pyspark code. All of the outputs will be shown in the command window is sometimes annoying. It becomes a little bit hard to review the codes written before.*

Chapter 6

Combined Classifier

6.1 Motivation

The learned classifier can successfully classify reviews as positive or negative better than the rule-based and pretrained classifiers. This chapter aims to improve the performance further using just these three classifiers and the original feature set. Two opportunities for improvement are identified. 1. Some reviews may be mislabeled. 2. Although the learned classifier is the best with enough data, it may still benefit from being influenced by the nonlearned classifiers.

The algorithm described in the next section is inspired by several previous approaches. Weakly supervised learning, as described in [16] aims to train on data with unreliable labels. Labeling unlabeled data with confident labeled data is explored in [2] and confident classifiers "teaching" each other in [12]. Transfer learning can be applied here to use the rule-based and pretrained classifiers usefully as is done in [15] with a generative model that takes into account sentiment and domain importance.

6.1.1 Mislabeling

To explore the quality of labeling, 100 balanced test reviews were manually labeled. The classifier was also trained on an equal training set on the rest of the reviews. The comparison is shown in table 6.1. It seems there is some room for improvement by the classifier, but also

	Classifier	Self
Positive Predictions	.47	.53
Accuracy	.71	.79

Table 6.1: Classifier compared to self labeling

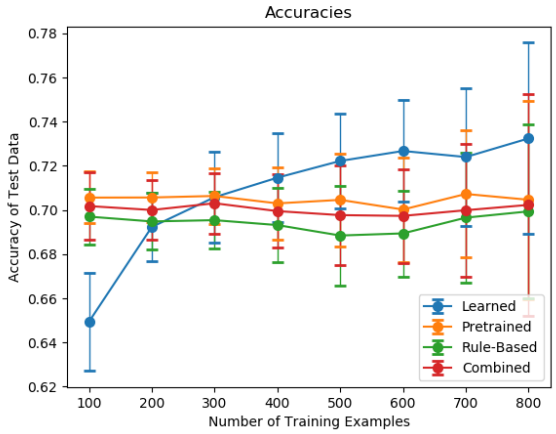

Figure 6.1: Ensemble Accuracies Compared To Each Classifier

there are many reviews that couldn't even be classified by a human annotator, indicating that there might be an upper bound on the performance of the automated sentiment classifier.

6.1.2 Ensemble

An ensemble of classifiers often outperforms each individually[8]. This experiment uses a majority vote between the three classifiers at each training size to determine the classification. Figure 6.1 shows that this strategy just gives performance between rule-based and pretrained, probably because those two often agree with each other. It doesn't improve with more training points like the learned classifier. This provides further evidence that the weights from the learned classifier are unique and important. Any successful combination of the classifiers will have to rely heavily on the learned model.

6.2 Algorithm

This section describes my algorithm. The labels, rule-based classifier, and pretrained classifiers are used to reassign labels to grow a set of reviews and labels that are more reliable. The learned classifier contributes to this labeling more and more as the size of the trusted data grows, and at the end uses this clean data to make predictions.

First, an initial seed set of size s is constructed with the same balance as the training data (here it is assumed that there are equal positive and negatives). This initial seed is chosen as the most polar positive and negative reviews according to a weighted average of their scores according to the rule-based classifier, pretrained classifier, and label, according to hyperparameter weights w_r, w_p, and w_l respectively. Now iteratively, a batch size (b) of new reviews is added to the trusted reviews pool. These are chosen as the b most polar reviews, which not already in the verified set, according to the same score used previously plus the learned classifier fit on the verified data. The weight of the learned classifier is a linear function that grows each iteration at rate g. Verified reviews are given the weight of their combined score which is often different from the original label. If there are unlabeled reviews, the same process continues without using the label in the score. The final classification is given by the learned classifier fit on the full, newly labeled data. The full pseudocode is shown in algorithm 1.

6.3 Parameter Analysis

The weight parameters for the unlearned classifier were chosen as $w_r = 1$, $w_p = 2$, and $w_l = 2$ to make the two classifiers roughly equal to each other and added together equal to the label influence, taking into account the scale of their outputs. The batch size was chosen as $b = 10$ to balance efficieny and quality. The default seed size and classifier growth were chosen as $s = 200$ and $g = 10/\text{num\_iterations}$ where $\text{num\_iterations} = (n - s)/b$. The changing parameters are compared to the normal learned classifier on the same amount of training data, which doesn't change with the x-axis.

Algorithm 1: Iterative Relabeling Classification Algorithm

unlearned_scores = w_r * rulebased_scores + w_p * pretrained_scores + w_l * numeric_labels
unlearned_pos = unlearned_scores[label==‘positive’]
unlearned_neg = unlearned_scores[label==‘negative’]
most_pos_reviews = pos_reviews[sorted(unlearned_pos)[-s/2:]]
most_neg_reviews = neg_reviews[sorted(unlearned_neg)[:s/2]]
verified_reviews = [most_neg_reviews, most_pos_reviews]
verified_labels = [b/2*‘negative’ + b/2*‘positive’]
num_verified = b
$w_l = 0$
while *num_verified* $< n$ **do**
    $w_l = w_l + g$
    learned_scores = naivebayes.fit(verified_reviews, verified_labels)
    total_pred = unlearned_scores + w_l * learned_scores
    sorted_pred = sorted(abs(total_pred))
    most_certain_reviews = reviews[sorted_pred[-b:]]
    most_certain_labels = sign(sorted_pred[-b:])
    verified_reviews = [verified_reviews, most_certain_reviews]
    verified_labels = [verified_labels, most_certain_labels]
    num_verified+ = b
end while
final_pred = naivebayes.fit(verified_reviews, verified_labels).predict(reviews_test)
return *final_pred*

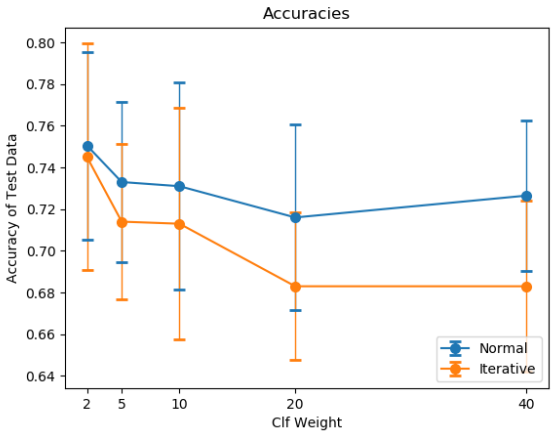

Figure 6.2: Accuracies of Iterative with Different Classifier Weights

6.3.1 Classifier Weight

As the growth weight of the learned classifier increases, more power is given to switch labels based on the classifier's results on a set of verified reviews. Figure 6.2 shows the performance with different weighting of the learned classifier. Here the x axis refers to the maximum classifier weight after all iterations. The performance deteriorates with larger classifier weights, and it never does better than the baseline.

6.3.2 Seed Size

The choice of seed size determines how much data is labeled as verified before the learned classifier gets involved and labels can be switched. A higher one would make the classifier less important in general and make this more similar to the normal method. Figure 6.3 shows that a low confidence threshold makes performance slightly worse, but the choice is not very important.

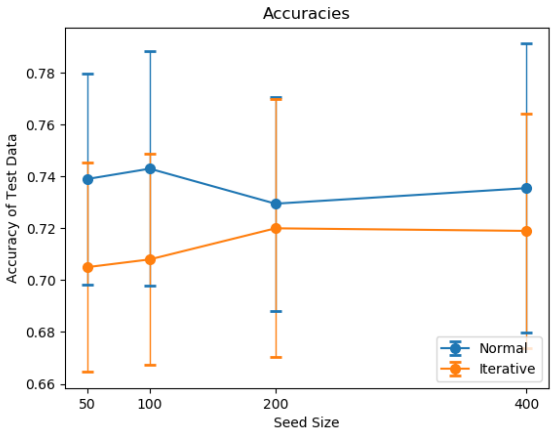

Figure 6.3: Accuracies of Iterative with Confidence Thresholds

6.4 Results in Different Settings

With the current training data, the iterative classifier fails to make improvements. Two different settings are proposed that might make this iterative classifier useful. 1. Noisy labels - It might be that there is not enough noise in the data justify flipping labels, but here noise is artificially added to the training set. 2. Unlabeled data - This tests the iterative classifier's natural ability to use unlabeled data to expand training size. In both cases, the iterative classifier is compared to a baseline with a simpler solution to the setting.

6.4.1 Noisy Labels

A noisier dataset is simulated by flipping a percentage of labels in the training set, but keeping the labels the same in the test set. The simple baseline to compare against is one that doesn't use the original labels at all, but instead labels everything with the pretrained classifier. Figure 6.4 shows that the performance of the normal classifier deteriorates with more noise, and the iterative classifier is able to correct this. However, it is consistently worse than the baseline method.

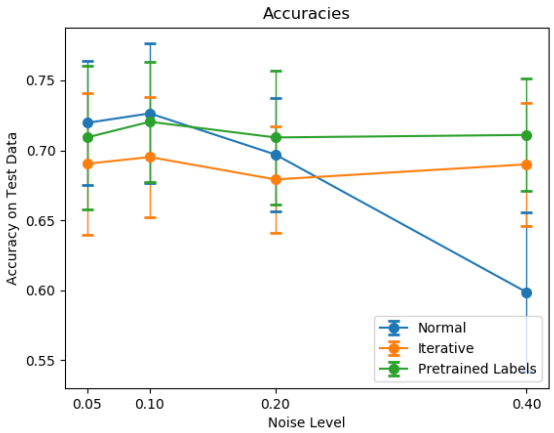

Figure 6.4: Accuracies with Different Polarity Selections

6.4.2 Unlabeled Data

Instead of using 700 training examples, only 100 are used in this experiment with a seed size of 50. Then different amounts of unlabeled data are added to training for the benefit of the iterative classifier, but not the normal one. A baseline classifier that labels all unlabeled points with the pretrained classifier and then uses the learned classifier on the full data is also used. Figure 6.5 shows that the iterative classifier uses some of the unlabeled data successfully, but after getting more becomes a worse classifier. In all cases, it was better to use pretrained labels. It is also interesting to see that none of these methods were able to use the unlabeled data as well as they would labeled data. The baseline method stopped improving after only 100 more unlabeled data examples even though it has been shown that usually more data can improve the accuracy at this number of training examples.

6.5 Discussion

This iterative approach decreases accuracy of the classifier under any tested parameter selections. Flipping the labels seems to have no effect in the best case and make them less accurate in the worst case. Rather than building a verified dataset, this method becomes unreliable and more biased towards bad data as it goes on. It seems that the two benefits

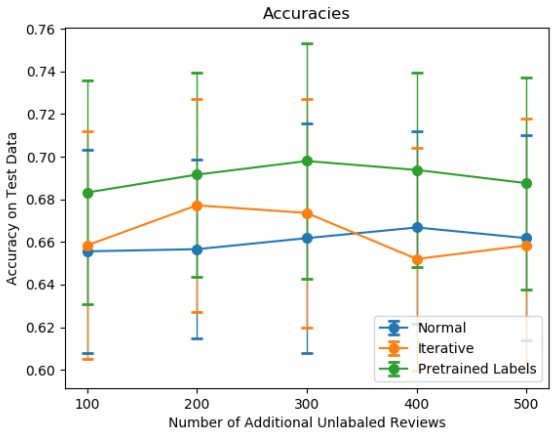

Figure 6.5: Accuracies with using Unlabeled Data

theorized about in the motivation, don't hold. The classifier already seems robust to the mislabeling in this dataset, so trying to fix this problem isn't worth it. Even with more noisy data, it is better to use the simpler method of relabeling. Possibly because of the high correlation between classifiers, the ensemble didn't improve performance, even when weighting them to favor the learned classifier with more data. Unlabeled data is not hard to deal using a pretrained classifier, and even easier after having a good learned classifier, so the iterative method also does not benefit in that case. To make this work in practice, there would have to be a more careful process of building the verified dataset, and to be useful the options available for a classifier would have to be worse. More testing could be done on different kinds of datasets to see the benefit of this iterative classifier.

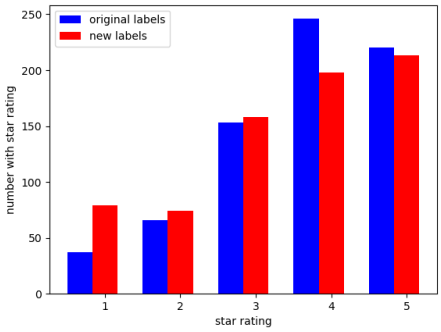

Figure 7.1: Histogram Comparing Original Ratings To New

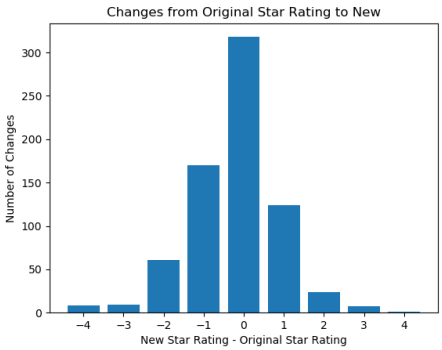

Figure 7.2: Histogram of the Changes of Ratings From Original To New

Chapter 7

Star Reviews

The Fall 2018 review data uses a five star labeling system. I relabeled each of these reviews manually and called these the "new labels" and the student self-reported ones the "original". This chapter specifically does analysis that benefits from more descriptive labels and multiple sources of labeling.

7.1 New Labels

Figure 7.1 shows a histogram of the labels for both the original student labels and the new ones. Similarly, figure 7.2 details the histogram of changes from the original labels to new.

Accuracies	5 star	4,5 star	3,4,5 star
Original	.73	.67	.65
New	.83	.76	.72

Table 7.1: Star Ratings Considered Positive and Accuracy with Original and New Labels

Overall the new labels are more negative than the original, but they estimate the original pretty well.

7.2 Train and Test on Different Thresholds

This section has two purposes - to find out if it is harder to classify positive reviews if they have a lower star rating, and to see if the new labels are easier to classify than the old. 100 training points and 20 test points, both sets evenly split between positive and negative, were used over 50 iterations to get each mean accuracy. The negative class always contained one and two star reviews (because there weren't enough one stars alone to do this experiment), while the positive class contained five star, four and five star, and three, four, and five star. Table 7.1 shows the results. Using lower rated reviews in the positive dataset significantly decreases performance. Also, the new labels have a much higher agreement with the classifier the original. It seems that the new labels are much closer to the classifier's beliefs and they match up closer to the text of the review. This is also observed in [7].

7.3 Identifying One Star Reviews

One practical task is to automatically identify all of the one star reviews to follow up with the student or look at the negative feedback quickly. But the one star reviews are only a small fraction of the data. In the new labeled data, 79 reviews are labeled one star and 643 higher than that. To test the practicality of identifying one star reviews, half of each class is put in training and half in testing. A classifier is learned on the data and the threshold is moved so that it predicts a certain percentage negative (one star). The recall of predicting one star reviews is assessed, that is how many of the one star reviews were correctly predicted as negative. Figure 7.3 shows the results. As the number of negative predictions increases, the

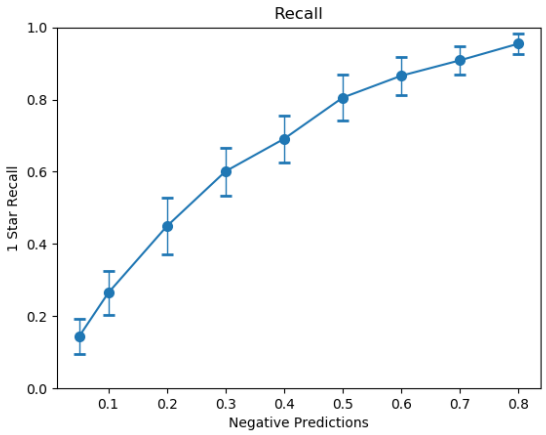

Figure 7.3: Predicting One Star Reviews

number of correctly identified one star reviews increases at a rate far higher than random. One would need to look at 25% of total reviews to see 50% of one star reviews, 50% for 80%, and 70% for 90% with this much training data available. One star reviews can be identified fairly well, but if the tolerance for missing some is low, this approach may not be efficient enough.

Chapter 8

Conclusion

This paper analyzed the sentiment classification performance of a multinomial Naive Bayes classifier on homework review data. After adjusting for biases in the classifier, it was shown that with enough data, the learned model outperforms baseline models that don't use training data. Different techniques were used to reduce the feature and review spaces. None were found to improve performance, but by seeing how much accuracy depended on each part of the model, these experiments show what is important to the model. The attempt to improve performance by relabeling reviews and using multiple classifiers at the same time never performed better than the base model.

The lack of success in coming up with a way to combine multiple classifiers or use regularization to improve the model while still using the TFIDF features, may show that the simple supervised learning method already performs quite well. It was also shown in practical scenarios that this could be used to classify positive and negative reviews if there is enough data, and if there isn't enough, models that aren't trained on the data still work well.

There are many areas in this paper that could be explored further. In particular, more analysis could be done to find reviews that are useless or detrimental to the classifier. Other parts of the dataset that could be useful, but weren't analyzed here are the knowledge of what student wrote each review, what each students' grade was on each homework assignment, and the use of neutral and unlabeled reviews.

References

[1] Nabeela Altrabsheh, Mihaela Cocea, and Sanaz Fallahkhair. Sentiment analysis: Towards a tool for analysing real-time students feedback. 11 2014.

[2] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[3] Ida Camacho and Ashok Goel. Longitudinal trends in sentiment polarity and readability of an online masters of computer science course. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, L@S '18, pages 21:1–21:4, New York, NY, USA, 2018. ACM.

[4] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003.

[5] George Forman. Bns feature scaling: An improved representation over tf-idf for svm text classification. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 263–270, New York, NY, USA, 2008. ACM.

[6] Clayton J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, 2014.

[7] Isa Maks and Piek T. J. M. Vossen. Sentiment analysis of reviews: Should we analyze writer intentions or reader perceptions? In *RANLP*, 2013.

[8] Isidoros Perikos and Ioannis Hatzilygeroudis. Recognizing emotions in text using ensemble of classifiers. *Eng. Appl. of AI*, 51:191–201, 2016.

[9] Sebastian Raschka. Naive bayes and text classification I - introduction and theory. *CoRR*, abs/1410.5329, 2014.

[10] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *CoRR*, cs.LG/0212032, 2002.

[11] Andrea Vanzo, Danilo Croce, and Roberto Basili. A context-based model for sentiment analysis in twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2345–2354, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[12] Wei Wang and Zhi-Hua Zhou. Theoretical foundation of co-training and disagreement-based algorithms. *CoRR*, abs/1708.04403, 2017.

[13] Geoffrey I. Webb and Michael J. Pazzani. Adjusted probability naive bayesian induction. In *Australian Joint Conference on Artificial Intelligence*, volume 1502 of *Lecture Notes in Computer Science*, pages 285–295. Springer, 1998.

[14] Dani Yogatama and Noah Smith. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 656–664, Bejing, China, 22–24 Jun 2014. PMLR.

[15] Yasuhisa Yoshida, Tsutomu Hirao, Tomoharu Iwata, Masaaki Nagata, and Yuji Matsumoto. Transfer learning for multiple-domain sentiment analysis - identifying domain dependent/independent word polarity. In *AAAI*, 2011.

[16] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5, 08 2017.

[17] David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Trans. Manage. Inf. Syst.*, 9(2):5:1–5:29, August 2018.

Sentiment Analysis of HW Reviews, Mekus, M.S. 2019