Laureen Lam, CS 224N, Final Project

Introduction

The goal of this project is semantic analysis of movie reviews to help predict movie ratings. Although the data set used for this project is labeled with a specific rating, there are texts, such as blogs, on the Internet that have no explicit rating label, but that could potentially help in determining the quality of something of interest, whether it be a movie, a restaurant, or an everyday consumer product. Additionally, some texts might have the wrong rating labels (i.e. be mis-labelled) given the preferences of a particular user; for instance, someone who dislikes vegetarian restaurants might give one a 0-star rating, and that would make any non-0 ratings be mislabelled ratings for that restaurant, given the preferences of that particular user. Accurately predicting star ratings could help determine the correct labels for items given a particular user. In this project, I investigate using a Maximum Entropy classifier with several different features to help predict movie ratings based on movie review text.

Linguistic Assumptions

The linguistic assumptions I made were that movie reviews consisted of words separated with spaces, and that word N-grams are an important part of what constitutes the semantic meaning of a piece of text. So, for instance, the bigram "I liked" might have a positive connotation, and the bigram "didn't like" might indicate a negative one. Taking N-grams of higher order N's might help differentiate between phrases such as "I didn't like" versus "they didn't like", and improve the accuracy of the classifier, since the former would likely affect the final rating that the author assigns while the latter may not.

Additionally, I assume that the constituents of a word might have significance in the semantic meaning of the text. For example, the first 3 letters of the words "dislike" and the first 2 letters of the words "uninspired" and "unmotivating" indicate a negation of some sort. The more often people use words with negation-associated letter N-grams like these, the more they tend to dislike the object are speaking about; in the case of movie reviews, I assume that this would consequently cause them to give the movie a low or negative rating.

Data

For this project, I used the movie review data sets provided by Bo Pang and Lilian Lee on their website at http://www.cs.cornell.edu/people/pabo/movie-review-data/. I used two data sets: the Polarity Dataset v2.0 and the Scaled Dataset v1.0. The Polarity Dataset has

movie reviews classified using 2 labels: either positive sentiment or negative sentiment. The Scaled Dataset includes movie reviews that are labeled based on the rating that the author gave to the movie, using a 3-class scale of 0, 1, or 2 stars. The Polarity Dataset contains 1000 positive reviews and 1000 negative reviews, and the Scaled Dataset contains 27,886 labeled movie reviews. I downloaded both data sets; they can be found in the /data subdirectory under /polarity and /scaled in the submitted final project files.

Maximum Entropy Classifier

For this project, I used the Maximum Entropy Classifier that I implemented in PA3, and modified it to read in data from the two different movie review data sets. I then investigated improving the accuracy of the classifier using several different features, detailed below.

Performance on the Polarity Dataset

I divided the Polarity Data Set into a training set and test set using a standard 80/20 split, applied to both the 1000 positive and 1000 negative reviews. This means that the training data contained the same number of positive reviews as negative reviews, and likewise for the test data.

Baseline Performance

Without any features, the MaxEnt classifier achieved a 50% accuracy, as expected, which is equivalent to randomly guessing the label of each review in the 50% positive / 50% negative test set.

I use this as the baseline performance. All the features presented below are investigated by themselves, in isolation from other features, unless otherwise stated.

Numbers as Features

The first feature I tried was whether a review contained any numbers in it. This feature was not very useful for biomedical data, and so it is not surprising that it did not help for movie review data either. The accuracy using this feature was **50%**, which is no better than guessing.

Prefixes as Features

The next feature I tried was using the 3-letter prefixes of words in the reviews (i.e. word truncation to the first 3 letters of words). I had found in PA3 that using word prefixes as features improved performance of the MaxEnt classifier for biomedical terms, so it is not unreasonable to expect that using word prefixes as features might also help in classifying movie reviews.

	Accuracy
Disallowing Repeated Features:	**79.00%**
Allowing Repeated Features:	**81.75%**
Allowing Repeated Features with Counts Appended:	**74.00%**

Using prefixes, and allowing repetitions of non-unique prefixes to be added as features of reviews, the MaxEnt classifier achieved an accuracy of **81.75%**, an improvement of roughly 30% from not having any features at all.

I was curious to see what would happen if I captured the number of repetitions for a non-unique prefix and stored it as a number appended to the prefix itself as a feature. In this case, the accuracy dropped to **74%**. One reason for this drop is that capturing the exact number of repetitions is not as useful for the MaxEnt classifier because it distinguishes between the exact number of repetitions even when the numbers are very close (e.g. 1000 versus 1001).

I then tried disallowing repetitions of non-unique prefixes completely. Essentially, the classifier is collapsing all the non-unique prefixes into a single one, regardless of the exact number of repetition. For the reason above, this performed better than capturing the number of repetitions as part of the feature, achieving an accuracy of **79%**.

Suffixes as Features

I also tried suffixes, since this feature worked slightly better than prefixes in the biomedical corpora. However, for movie reviews, the converse proved to be true: suffixes used as features resulted in an accuracy of only **78.75%**, compared to **81.75%** using prefixes.

Counting all repetitions of suffixes and using just those suffixes appended with their counts as features, the accuracy dropped to **67.5%**. This drop in accuracy is worse than with prefixes.

Disallowing repetitions of non-unique suffixes completely resulted in an accuracy of **73%**. This is similar to the behavior that the accuracies followed for the prefixes feature.

	Accuracies
Disallowing Repeated Features:	**73.00%**

Allowing Repeated Features:	**78.75%**
Allowing Repeated Features with Counts Appended:	**67.50%**

In examining the Polarity data set, there are many words that end in the same 3 letters, most notably "-ing", that are equally prevalent in both positive and negative reviews. Since the last few letters of most English verbs really just indicate a grammatical role of the word being used by the author, and is not generally related to whether the review was positive or negative, this feature did not perform as well as prefixes, which tries to capture the actual meaning of an English word through a crude form of stemming.

Additionally, in both the training and test movie review data, negative reviews tended to have more of the prefix "-un" in them (e.g. "uninspired", "untalented", "uncomfortably"), which the prefix feature was able to capture somewhat, but which was completely missed by the suffix feature.

On the other hand, in the biomedical corpus from PA3, words tended to end in special suffixes such as "-ide" and "-osis" that were a signal as to what particular tag the word was more likely to have.

Letter N-grams as Features

Next, I investigated using letter N-grams as features. For each word, I pre-pended a caret (^) and appended a dollar sign ($) to capture information about the beginnings and endings of words.

I had initially excluded all words that were not long enough to yield an N-gram for the chosen N, but later decided to include them for comparison. Using letter N-grams and allowing shorter words to be used as features actually caused the accuracies to go up slightly, but the difference is so small that it could easily be attributed to noise in the test data.

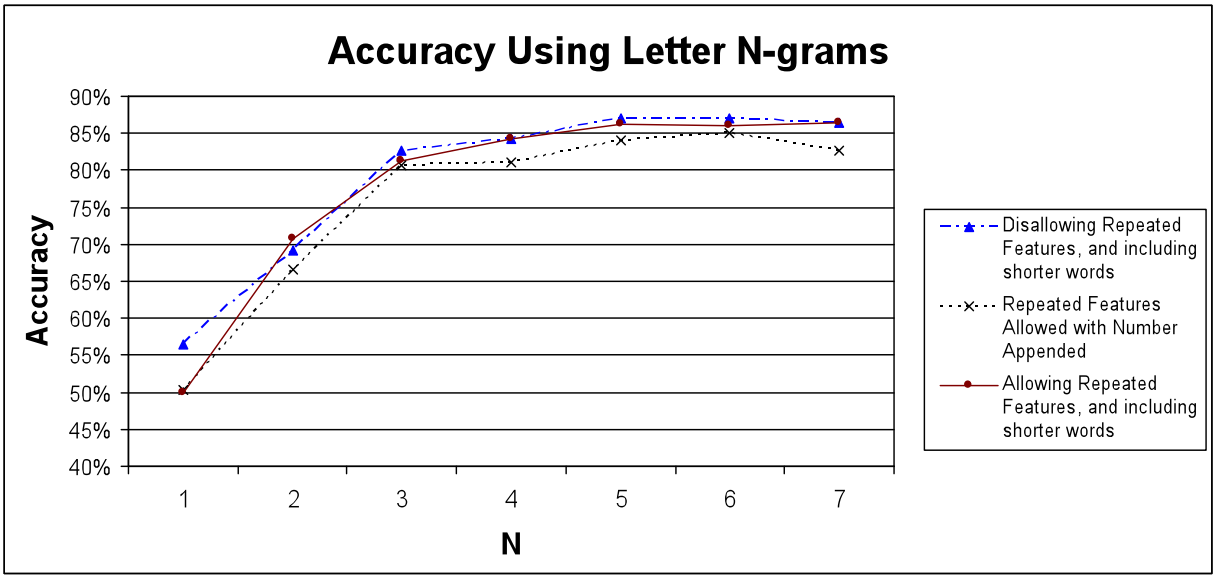

Figure 1. Accuracy of the MaxEnt Classifier using letter N-grams.

The graph shows that the MaxEnt classifier was able to classify more accurately as the value for N increased from 1 to 5. After N=5, the accuracy starts to level off, and begins to drop for features that are appended with their feature count. At this point, there are so many N-grams that each one has a more probable chance of having a different count than what appears in the test data, resulting in the inability of the MaxEnt classifier to effectively use that information for classification.

N	Disallowing Repeated Features	Disallowing Repeated Features, and including shorter words	Repeated Features Allowed with Counts Appended	Allowing Repeated Features	Allowing Repeated Features, and including shorter words
1	56.50%	56.50%	50.25%	50.00%	50.00%
2	69.25%	69.25%	66.50%	70.75%	70.75%
3	82.25%	82.75%	80.75%	80.00%	81.25%
4	83.50%	84.25%	81.00%	85.25%	84.25%
5	**86.75%**	**87.00%**	84.00%	**86.25%**	86.25%
6	84.75%	**87.00%**	**85.00%**	85.25%	86.00%
7	86.00%	86.50%	82.75%	84.50%	**86.50%**

Figure 2. Accuracies achieved by the MaxEnt classifier on Polarity Dataset using letter N-grams. The highest accuracy in each column is bolded.

In PA3, I had found that using letter N-grams worked better than just prefixes or suffixes, since the N-grams capture both as well as additional information related to each word. The results show that using letter 3-grams as features caused the classifier to perform better than using suffixes but slightly worse than just using prefixes, for the movie review data, achieving an accuracy of **80%**. Including words shorter than the N chosen, the accuracy rose slightly to **81.25%** for N = 3.

To summarize: Allowing repetition of letter N-grams as features, the best accuracy achieved was **86.25%** using letter 5-grams. Disallowing repetitions, the accuracy rose to **86.75%**. And, lastly, disallowing repetitions but allowing words shorter than N=5 to be used as features, the accuracy rose slightly to **87%** for 5-grams. In examining the movie review data, I found that some reviews, such as the one for *The 13th Warrior* in cv814_20316.txt (label = negative), contain a large number of words that are less than 5 letters long ("not", "to", "but", "it", etc.), while other reviews do not. The classifier is likely picking up on the differences between these types of movie reviews, and especially on negative words like "not", and using them to better classify the reviews, when words less than N long are allowed to be used as features.

Word N-grams as Features

After letter N-grams, I thought a reasonable next step would be to try using word N-grams as features. As in PA1, I created a START token <S> and a STOP token </S> that I pre-pended and appended to each sentence, respectively, so that I could capture information about the beginnings and endings of sentences. (For trigrams, I pre-pended two START tokens.)

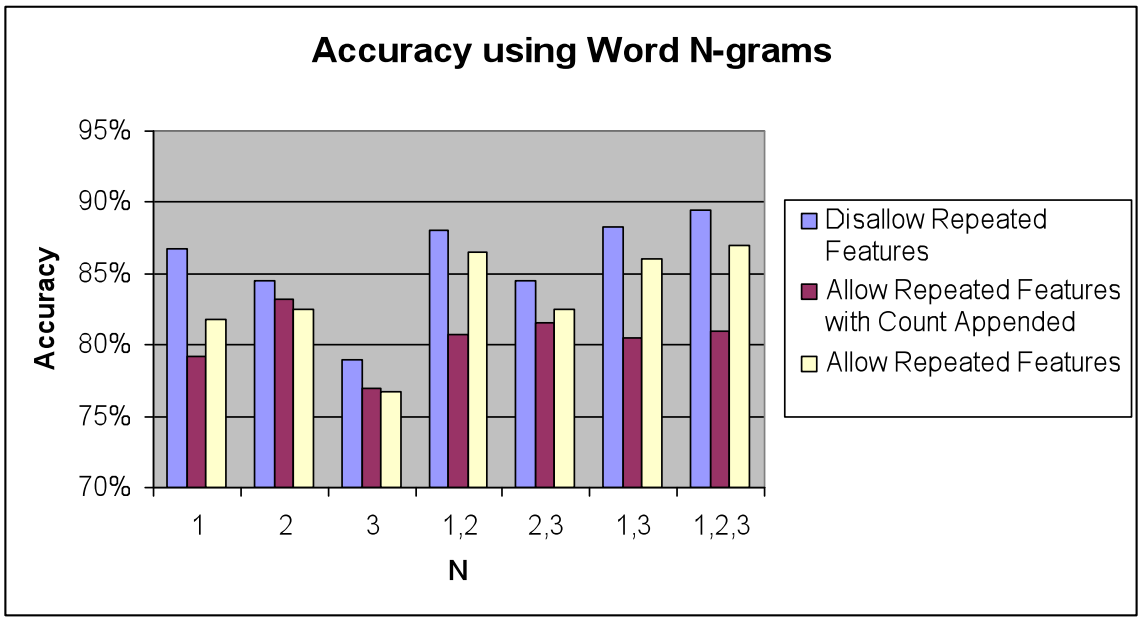

Figure 3. Performance of MaxEnt classifier on Polarity Dataset using word N-grams as features.

I tried unigrams, bigrams, trigrams, and combinations of those three as features. Of the N-grams used alone, unigrams performed the best, followed by bigrams, and finally trigrams. Combining unigrams with bigrams resulted in a slight improvement over just using unigrams. Further combining unigrams and bigrams with trigrams resulted in the best accuracy of **89.5%**. This accuracy is slightly better than the best accuracy of **87%** achieved using letter N-grams. One reason might be that word N-grams convey more information about the meanings of sentences than letter N-grams.

Final Feature Set

My attempt to find a good final features set was not as clear-cut as I had expected. When combining prefixes with the letter N-gram, word unigram, and word trigram features, the classifier achieved worse results than just using the letter N-gram feature alone. Allowing repeated prefixes, but disallowing repeated letter 5-grams, and also allowing words less than N=5 in length to be used as features, the accuracy was only **85.75%,** as opposed to **87%** using just letter 5-grams (disallowing repeats and allowing shorter words).

I thought that this might be due to the prefixes causing the classifier to overfit to the training data, since it was giving extra emphasis to letter 3-grams at the beginnings of words, which were already part of letter 5-grams, so I disallowed repeated prefixes, while keeping all other parameters for the letter 5-grams and word N-grams the same. This helped, and the accuracy went up to **87%**, which is the same as the accuracy using letter 5-grams.

Next, I decided to exclude prefixes altogether. This reduced the number of features trained on by approximately 380,000 (about a 7% reduction). Again, using letter 5-grams, word unigrams, word trigrams, and prefixes, this made the accuracy **88%**. This is about the same as word unigrams and word trigrams used together, which had achieved an accuracy of **88.25%**.

The best feature set was created using word unigrams, word bigrams, and word trigrams together, which achieved an **89.5%** accuracy. This totaled 2.8 million features from approximately 1.2 million words in the training set.

Performance on the Scaled Dataset

I made the necessary modifications to the MaxEnt classifier to use the Scaled Dataset: the exact format for this dataset is two separate files, one containing the labels and the other containing the text of the corresponding movie review on the corresponding line number.

Since the reviews are ordered based on their label in the data set, I split the reviews into a training set and test set (again, using an 80/20 split) by placing labeled reviews into 1 of 10 buckets, calculated by taking the modulus10 of the line number of the review. I combined the reviews in the first 8 buckets to create a training set, and combined those in the last 2 to create the test set. This was to ensure that the accuracies calculated for each feature remained the same through different runs. An alternative approach was to randomly select the bucket for each review, but this had the disadvantage in that the accuracies calculated varied between different runs even if the features remained the same. Because I wanted accuracies that were consistent across runs, I chose the modulus10 approach.

Features for the Scaled Dataset

I examine the same features for the Scaled Dataset as for the Polarity Dataset. The baseline accuracy with no features was **37.80%** for the dataset with 3 classes. The dataset did not have an even split of movie reviews in each class, and so this accuracy is slightly higher than what would be expected for a dataset that did have an even split.

Numbers as Features (Scaled Dataset)

Using the presence of numbers in the movie review text as a feature did not help the accuracy, as expected from its performance in both the Polarity Dataset and in PA3. The accuracy remained at **37.80%** using this feature, which is the same as using no features at all. This suggests that the movie ratings assigned by movie reviewers had nothing to do with their tendency, or lack thereof, of putting numbers in their reviews.

Prefixes & Suffixes as Features (Scaled Dataset)

Using prefixes as features, but disallowing repeated prefixes, the accuracy was **59.5%**, roughly a 20% improvement in accuracy over the baseline. Allowing repeated prefixes caused the accuracy to drop slightly to **58.4%**. Finally, counting the number of repetitions of prefixes and using just those prefixes appended with their counts as features, the accuracy was **57.6%**.

Using non-unique suffixes, the accuracy was **55.5%**, about 4% worse than non-unique prefixes. Allowing non-unique suffixes to be used as features, the accuracy also dropped, to **53.8%**. Counting the number of repetitions of suffixes and using just those suffixes appended with their counts as features, the accuracy dropped further still to **52.4%**.

We see that using prefixes as features caused the classifier to perform better than using suffixes, which is the same behavior we saw for the Polarity Dataset, but different from the behavior for the Genia corpus (which contained mostly biomedical terms).

Letter N-grams as Features (Scaled Dataset)

The results obtained are as follows:

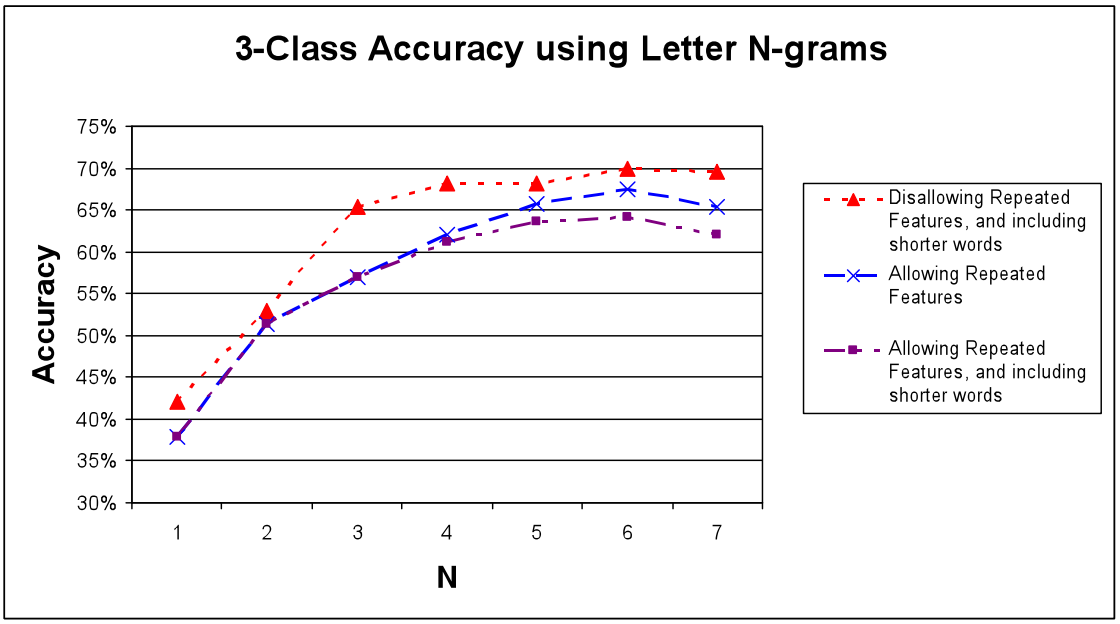

Figure 4. MaxEnt classifier accuracy on Scaled Dataset using letter N-grams as features.

N	Disallowing Repeated Features	Disallowing Repeated Features, and including shorter words	Allowing Repeated Features	Allowing Repeated Features, and including shorter words
1	42.00%	42.00%	37.80%	37.80%
2	52.90%	52.90%	51.40%	51.40%
3	65.30%	65.30%	57.00%	57.00%
4	65.20%	68.10%	62.00%	61.20%
5	68.30%	68.10%	65.80%	63.70%
6	**69.50%**	**69.90%**	**67.40%**	**64.20%**
7	66.80%	69.50%	65.40%	62.00%

Figure 5. Performance of MaxEnt classifier on Scaled Dataset using letter N-grams as features. The best accuracy in each column is bolded.

The best accuracy of **69.90%** was achieved with N = 6. This is only one off from the best N for the Polarity Dataset, which was 5. In both data sets, N = 1 achieved the worst accuracy, and the accuracy got better rapidly as N increased to 3. For both the Polarity and Scaled datasets, the improvements quickly leveled off after N = 3. This suggests that the bulk of the meanings of words are encoded in the first 3-4 letters.

Disallowing repeated features, and including shorter words, caused the accuracy to go up, which is similar to the behavior in the Polarity Dataset for the reasons stated.

Word N-grams as Features (Scaled Dataset)

START and STOP tokens were pre-pended and appended, respectively, to each sentence in a review before the word N-grams were taken. The accuracies for using word N-grams as features are summarized below.

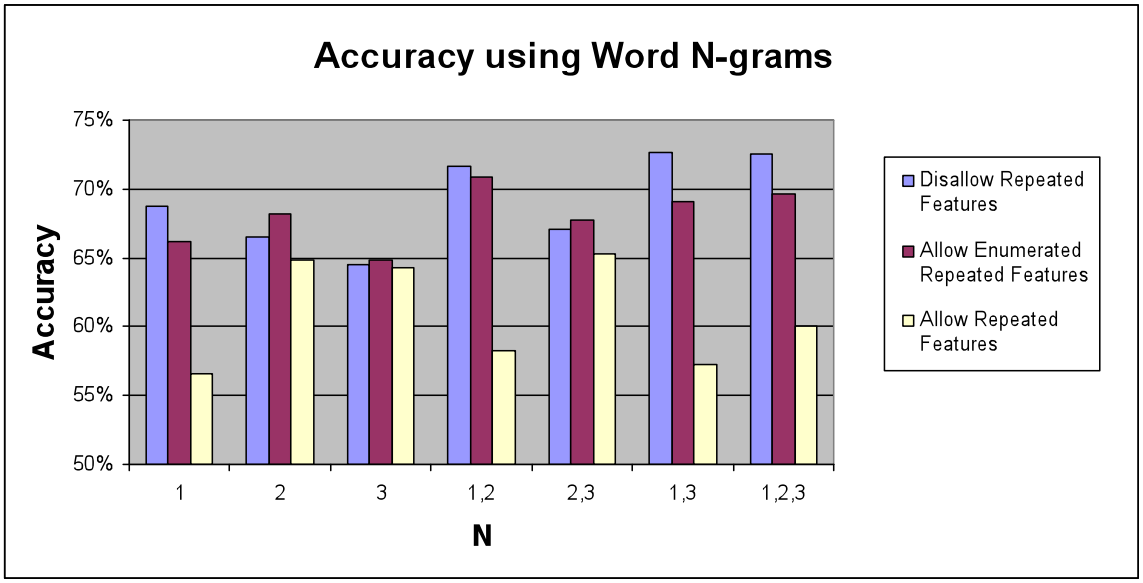

Figure 6. Performance of MaxEnt classifier on Scaled Dataset using word N-grams as features.

For both the Polarity and Scaled data sets, using word unigrams, as opposed to bigrams or trigrams alone, achieved the highest accuracy when repeated features were disallowed.

Rather than appending the counts of N-grams to their respective features like I did for the Polarity Dataset (which had resulted in poor accuracies), I decided to try enumerating the features instead. So, for example, a unigram "foo" might appear 3 times, and would be captured using the feature set "foo1", "foo2", and "foo3". The idea is that, even if the unigram does not appear the exact same number of times in a review in the test set as in a review from the training set, the classifier will still be able to use some features to help it

determine the correct label for the review. So, for instance, if "foo" appears 100 times in most of the training reviews from one class, and 101 times in a test review, the classifier would still be able to use "foo1" through "foo100" as features in the test data that have a corresponding feature in the data on which it has trained. Using this method, the classifier was consistently able to get better results than by allowing repeated features without enumeration.

Overall, using unigrams with trigrams resulted in the highest accuracy: **72.7%**. In the Polarity Data set, the highest accuracy resulted from using unigrams, bigrams, and trigrams combined. However, the difference in accuracy between using unigrams and trigrams, versus using the combined set of all three N-grams, is less than 1% in the Scaled Dataset.

Final Feature Set (Scaled Dataset)

I tried various combinations of the features mentioned above, but, similar to the Polarity Dataset, the best feature set consisted of only word N-grams. In this case, using just unigrams and trigrams resulted in the best accuracy of **72.7%**.

I was curious as to what types of errors the classifier was making in the Scaled dataset. Of the 240 reviews with 0 stars, 93 of them were mislabeled; of those 93, it turns out that only 14 of the reviews that were mislabeled had a rating that was more than 1 star away from the correct rating. Similarly, of the 306 reviews with 2 stars, 72 of them were mislabeled, but only 12 (4%) of those mislabeled reviews had a rating that was more than 1 star away from the correct rating. The majority (91.3%) of the reviews in the test set were either classified correctly or close to the correct rating, where close is defined as being at most 1 star off from the true rating.

Comparison with Related Work

Pang and Lee did similar work in their paper, Thumbs up? Sentiment classification using machine learning techniques, where they used various classifiers (MaxEnt, SVM, and Naïve Bayes) to determine the correct polarity ratings for movie reviews. They achieved a best MaxEnt classifier accuracy of **80.8%** using word unigrams combined with word bigrams, and doing 3-fold cross-validation, on a Polarity Dataset of 700 positive and 700 negative movie, available on their website under the name Polarity Dataset v1.0.

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	"	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Figure 7. Accuracies achieved by Pang & Lee on the Polarity v1.0 dataset in their paper, Thumbs up? Sentiment classification using machine learning techniques, using 3-fold cross-validation. The classifiers used are a Naïve Bayes classifier (NB), a Maximum Entropy classifier (ME), and a Support Vector Machine (SVM). As a note, Pang and Lee did not use START and STOP tokens in their word unigrams and bigrams features.

For comparison, I ran my MaxEnt classifier on the same data set, with 3-fold cross-validation, using my own features. I achieved an accuracy of **82.3%** using word unigrams combined with word bigrams, with START and STOP tokens pre-pended and appended to each sentence:

N	Accuracy
1	82.50%
2	78.57%
3	75.71%
1,2	82.29%
2,3	78.57%
1,3	82.79%
1,2,3	**83.21%**

Figure 8. MaxEnt classifier accuracy on Polarity Dataset v1.0 using Word N-grams as features with START & STOP tokens pre-pended and appended to each sentence, and 3-fold cross-validation.

Additionally, Pang and Lee achieved a best accuracy of **82.9%** using an SVM classifier with word unigrams as features. The best accuracy I achieved with my MaxEnt classifier on the dataset was **83.2%** using word unigrams combined with word bigrams and word trigrams.

Conclusion and Future Work

Trying to predict the sentiments for the 2-class, polarity movie review data set was much easier than predicting the ratings for the 3-class, scaled movie review data. The best

accuracy achieved by the MaxEnt classifier for the 3-class data set was **72.7%**, and the best accuracy for the polarity data set was **89.5%**.

Although using the MaxEnt classifier for the 3-class data set did not perform as well as for the 2-class data set when classifying the exact label of a movie review correctly, it was still able to label **92.3%** of the reviews correctly or close to the correct movie rating, where close is defined as being at most 1 star off from the true rating.

Future work could be to use a SVM classifier with different kernels to try to achieve a higher accuracy. There are also many other classifiers to experiment with, including Naïve Bayes, k-NN clustering, and decision trees, that could provide an even better accuracy when coupled with the Maximum Entropy classifier.

Additionally, in trying to use word unigrams, bigrams, and trigrams all as features at the same time, I found that my machine quickly ran out of memory and I consequently had to move my testing to the corn machines, which have much more memory. One thing to try in the future would be to build several MaxEnt classifiers, each running with just one feature set, instead of having just one classifier running with all the features, then combining their label probabilities for test set reviews via a top-level classifier (not necessarily a MaxEnt classifier itself). Such a scheme would allow several machines to work together, or one machine to build the classifiers one after the other, saving only the label probabilities, and then combining the results to produce the best label for the test data.

Appendix

Suppor Vector Machine (SVM) Formatted Data

The -outputSVM flag is available to transform the Polarity Dataset v2.0 training and test data into SVM format. You can select the fraction of the data sets to transform via the -SVM_K <integer number> flag. Since the number of SVM features per review can reach over 2 million, I highly recommend taking just a small fraction (e.g. 1/25) of the 2000 movie reviews to use for SVM classification. Otherwise, the resultant SVM-formatted data file sizes may exceed your disk space. Additionally, using extremely large data files will cause the SVM training to take a very long time.

Please consult the README file included in the java/ subdirectory for more information about how to run the MaxEnt classifier with flags.

References

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2):1-135.

Bo Pang and Lilian Lee. 2002. Thumbs up? Sentiment classification using machine learning techniques. Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm