MULTICLASS SENTIMENT ANALYSIS WITH RESTAURANT REVIEWS

Moontae Lee (moontae@stanford.edu)
Patrick Grafe (pgrafe@stanford.edu)

Department of Computer Science
Stanford University, June 3rd, 2010

Abstract

In the era of the web, a huge amount of information is now flowing over the network. Since the range of web content covers subjective opinion as well as objective information, it is now common for people to gather information about products and services that they want to buy. However since a considerable amount of information exists as text-fragments without having any kind of numerical scales, it is hard to classify their evaluation efficiently without reading full text. This paper focuses on extracting scored ratings from text-fragments on the web and suggests various experiments in order to improve the quality of a classifier.

1. Introduction

The goal of this project is to develop a classifier which can predict sentiment of a text-fragment along a scaled range from one to five stars. So far, most of the major research on sentiment analysis has been done to predict the polarity of text: positive or negative sentiment, but not subjective opinions along a multi-class continuum. We are able to find numerous kinds of topics and data sets on the web such as movie-ratings, book-reviews, twitter tweets, etc. Among those, we chose a data set of restaurant reviews which includes various sentiments of predefined aspects of these restaurants rated on a scale of 1 to 5. Through these restaurant reviews, we will first analyze the properties of data set and explain the basic methods we used to select the features. Moreover we will suggest two novel approaches of extracting good features and illustrate the learning result compared to previous research done on similar domains.

2. Dataset Analysis

Our data set came from a website OpenTable.com which allows users to make reservations online at restaurants around the country. In addition, the website also aggregates user opinions on various restaurants including a text-based review and star ratings for the following categories: food, ambiance, service, noise, and overall. The data set also included restaurant ID's; however, we made no use of this information. The restaurant reviews are provided exclusively by customers who have used the site to make a reservation at a particular restaurant. Thus the reviews can be relied upon as legitimate, and being voluntary the reviewers likely provide honest appraisals of the various aspects of the restaurant.

2.1 Overall Ratings

Our primary effort was in improving the prediction of the overall sentiment indicated by each review. The overall restaurant rating was on a scale of one to five with one indicating "Poor" and five indicating "Outstanding." The data set was highly biased toward five star ratings as seen in the table below.

Star-Ratings	Overall Percentage
(Poor)	1.8%
(Fair)	6.4%
(Good)	12.6%
(Very Good)	33.2%
(Outstanding)	45.9%

Table 1 : Overall Star-ranking Distribution

2.2 Aspect Ratings

One concern about the ratings of the aspects was that some reviewers may have just selected the same star rating for overall and all of the aspects, but this appears to be very uncommon in this data set and we believe the ratings are appraisals in good faith. One other factor that will limit any algorithms ability to learn the aspect ratings is that while there is a rating for all four aspects: food, ambiance, service, and noise, usually the reviews are too brief to include a mention of atmosphere or noise levels at all. Thus we do not expect any algorithms to successfully predict these particular aspect rankings without selectively discarding a majority of irrelevant reviews.

The following tables show the distribution of instances of each star rating across our data set. Our data set is heavily biased toward rankings of 4 and 5 with very few ratings of 'poor.' The Noise rating is on a scale of 1 to 3 indicating how loud the restaurant is rather than indicating the reviewer's preference.

Star-Ratings	Food	Ambiance	Service
(Poor)	1.9%	1.4%	4.2%
(Fair)	7.6%	5.4%	7.8%
(Good)	15.3%	18.8%	13.1%
(Very Good)	35.5%	39.6%	27.6%
(Outstanding)	39.7%	34.8%	47.3%

Table 2 : Aspect Star-ranking Distribution

Rating	(Quiet)	(Moderate)	(Energetic)
Noise	21.8%	48.7%	29.5%

Table 3 : Noise Ranking Distribution

2.3 Analysis

It is very difficult to handle multi-class rankings because while making a binary choice between positive or negative is straight-forward, understanding the degree of satisfaction based on the language used is highly dependent on the individual. For some, the adjective "great" would often imply an extremely strong favorable impression

and a rating of 5 out of 5 while for others this would be a weaker adjective and might correlate with a rating of 4 or even 3.

Furthermore, often the reviewer liked the food, but was disappointed with the service or vice versa. How the reviewers' opinions of these aspects of a given restaurant affect the overall score is dependent on the individuals' opinions of the relative importance of food as compared to service, ambiance, and noise. At other times, the individuals' responses can even be nonsensical as in the review below:

- I was in for a special date with my wife. While the food was good the service was a disappointment. We had to ask for everything. I had to request my water glass be filled, ask for another drink and even go looking for the server to get the bill. Not sure I would return.-

This reviewer predictably rated the service as "poor" or 1 out of 5 stars. The reviewer rated the food 4 out of 5 and the ambiance 3 out of 5. Oddly enough the reviewer, who claimed he likely would not return to the restaurant, gave the restaurant an "outstanding" overall rating of 5 out of 5 stars. Anomalies like this were common throughout the data set.

3. Feature Selection

The most important part of successful classification is selection of appropriate features. We attempted numerous different standard methods of deriving a useful set of features: removing common stop words, cleaning the review text by converting to lower case and doing spelling correction, pruning misleading reviews from our training set, and finally parsing sentences in the review to determine noun-adjective and verb-noun pairs.

3.1 Stop Words

We recognized that within unigrams, bigrams, and trigrams, many words provide no useful information for sentiment analysis. Articles such as 'a' and 'the' and pronouns such as 'he,' 'they,' and 'I' provide little or no information about sentiment. The most common trigram in our data set, "the food was" ultimately provided little information whatsoever, and the most common unigrams were also intuitively not useful. We decided to remove stop words from our unigram, bigram and trigram features in order to consider only the most relevant features in our analysis. This allowed us to have more intuitively useful features such as "food very good" and "service very good." We used the stop words from the SMART system (Salton 1971) [ftp://ftp.cs.cornell.edu/pub/smart/english.stop]. We felt this list was inappropriate as is because it contained negative adverbs such as 'not' and 'wasn't' and intensive adverbs such as 'very.' Thus, using the stop words as is would have taken a sentence, "The food wasn't very good," and would have created a bigram "food good" which is misleading and represents a loss of real information. We thus removed intensive and negative adverbs from the stop words list in our final analysis.

3.2 Pruning

One problem with the data set we are dealing with was the wide variety of sizes of reviews. Some reviews are long and explain the entire visit

to the restaurant from arriving, to ordering and eating, and paying the bill. Others are short sentences expressing opinions regarding the food or service. Finally many are not even sentences at all. One concern is that many of the reviews consisting of a short sentence or sentence fragment might not contain enough information to adequately predict its overall ranking much less the ranking for different aspects. Two such examples are below:

If Steak is what you want ….. This is the place

Ordered dine about town crab ravioli, fish and crème brulee.

These two examples provide very little information. On the other hand some reviews manage to fit a large amount of information into very few characters such as these two reviews:

Very good food, reasonable prices, excellent service.

Loved the food, service and atmosphere! We'll definitely be back.

That last review manages to indicate a strong positive sentiment for three different aspects of the restaurant as well as a strong overall sentiment all in about 65 characters. It was unclear whether many of these short sentences could be predicted effectively, and whether they adversely affected our training so we experimented with throwing out reviews of less than 75 characters. This accounted for nearly ten percent of the data set.

3.3 Unigrams, Bigrams, and Trigrams

Our first improvement was simply to add bigram and trigram features. We believed adding these features would provide crucial information that could improve both overall ratings as well as individual aspect ratings. Specifically by adding in bigrams and trigrams, we could have nouns and adjectives together in a single feature such as "good food" or "bad service," thus yielding improved accuracy. When combined with eliminating stop words, this created common trigrams such as "food very good."

4. Two Novel Approaches

Until now, our work has been developed through basic features such as unigrams, bigrams, and trigrams with some degree of manipulation. Though using up to trigrams guarantees fairly good performance as a language model, it sometimes is not enough to capture sentiments in the text. This is because people frequently express their feelings not by using a couple of words, but rather through complex sentence structures where correlated words are spread apart by more than two or three words. For example, we can easily observe some reviews like "I was very disappointed with both the service and my entrée". It is hard to capture the correlation between the verb disappointed and its target entrée because the distance between "disappointed" and "entrée" is more than two words.

For this problem, there exists conflicted research. Pang et al. (2002) discovered that surprisingly unigrams beat other features in their process. In contrast, Hang et al (2006) argued that lower order n-grams are unable to capture longer range dependencies. Pang's result is conflicted with the usual intuition that increasing n-grams would capture more subtlety in the context. Also Hang's argu-

ment has some problems because increasing n-grams greater than three may cause drastic sparsity in the actual feature counts. Thus, in this chapter, we will develop two different novel algorithms to try and improve our results while avoiding the complications of using very large n-grams.

4.1 Autonomous Spelling Corrector

Searching through large amount of restaurant reviews, we very often found misspelled words. One interesting observation is that the reason for misspelling is different in each example. Sometimes it is caused by the literal hardness of a word (e.g. restaurant vs. resturant – "au" coming from French) and other times the configuration of the keyboard is also a cause for mistakes (e.g. excellent vs. excelent – hard to type two 'l' continuously using the ring finger). Moreover similarity in pronunciation may also cause misspelled words (e.g. waiter vs. waitor – attaching "er" or "or" is often a confusing issue). Since some of these words are critical in analyzing the correct sentiments from the reviews, if Pang's argument is valid, fixing misspelled unigrams in the data set will hopefully yield meaningful improvements. We will first focus on how we implemented our autonomous spelling corrector.

4.1.1 Levenstein Edit Distance

To correct misspelled words, the primary things that we need are a dictionary of correct words and a well-defined metric to measure how far misspelled words are away from correct candidates. To measure this distance systematically, we use the most famous metric, Levenstein's distance which is the least number of edits needed to transform one word to another through the three types of possible operations: insertion, deletions, and substitution of a single character.

$$\text{Virginia} \rightarrow \text{V}\boldsymbol{e}\text{rginia} \rightarrow \text{Ver}\boldsymbol{m}\text{inia} \rightarrow$$
$$\text{Verm}\boldsymbol{o}\text{nia} \rightarrow \text{Vermon}\boldsymbol{t}\text{a} \rightarrow \text{Vermont}$$
$$\text{Distance}_{\text{EDIT}}(\text{Virginia}, \text{Vermont}) = 5$$

Since measuring the distance from a misspelled word to all legal words in the dictionary is almost impossible, we decided to generate possible candidates for a misspelled word in a range of two edit distances. We then compute the distance by using a dynamic programming algorithm. It will play a role as the fundamental metric for the following extension.

4.1.2 Keyboard Distance

One problem of Levenstein distance is that it treats every pair of words the same if their edit distance is the same. This is not appropriate in practice. For instance, the Levenstein distance of two correct words "service" and "survive" from a misspelled word "servive" are equal, but "service" is more appropriate in terms of keyboard distance. This means that changing 2nd character 'e' in "servive" to 'u' should be more expensive than changing the 6th character 'v' to 'c'. To compute the actual distance, we set coordinates of the 26 characters on the keyboard with 'q' as an origin. The coordinates of all other characters are measured as the actual distance in centimeters of that character from the origin 'q.' The exact formula that we used in our computation is composed of two cost terms: cost of changing a character 'v' to 'c' and expected cost coming from neighbor characters.

$$\text{Distance}_{\text{KEY}}(\text{servive}, \text{service}) = \text{Cost}(\text{v}, \text{c}) + \frac{1}{2}\{\text{Cost}(\text{c}, \text{i}) + \text{Cost}(\text{c}, \text{e})\}$$

$$\text{Distance}_{\text{KEY}}(\text{servive}, \text{survive}) = \text{Cost}(\text{e}, \text{u}) + \frac{1}{2}\{\text{Cost}(\text{u}, \text{s}) + \text{Cost}(\text{u}, \text{r})\}$$

Since the cost function simulates real Euclidean distance between two keys, it is clear that "service" is the more appropriate candidate for the misspelled word "servive".

4.1.3 Sound Distance

In our approach, we also utilize the difference of pronunciation as another metric to measure the cost. This is mainly because people show a tendency to spell words how they sound which means often both the misspelled word and the correct spelling will have a similar pronunciation. Thus sound can also be a valuable metric for us. For example, the misspelled word "excelent" has two candidates, "excellent" and "excrement" in a range of two edit distances. Even though Levenstein distance between "excelent" and "excellent" is slightly closer than between "excelent" and "excrement", the difference in pronunciation gave us a clear distinction to clarify that "excellent" is a far superior choice than "excrement."

To evaluate this distance efficiently, we first introduced the Soundex system which is a phonetic algorithm encoding homophones to have the same representation so that they can be matched despite minor differences in spelling. See the following example.

$$\text{getSoundex}(\text{excellent}) = \text{getSoundex}(\text{excelent}) = \text{getSoundEx}(\text{exselent}) = \text{E245}$$

$$\text{getSoundex}(\text{excrement}) = \text{E265}$$

Then the distance between two words is decided by the total number of matched characters for each digit in the value returned by the soundex system. As a result, it will become a metric varying from 0 to 4. Since the first letter of Soundex encoding is always given by the first character of target word, if their starting pronunciation is similar but the initial letter is different, we will add 0.5 to the score to make up for this.

$$\text{Distance}_{\text{SOUND}}(\text{excellent}, \text{excelent}) = 4 - \#\text{ of Matches}(\text{E245}, \text{E245}) = 4 - 4 = 0$$

$$\text{Distance}_{\text{SOUND}}(\text{excellent}, \text{excrement}) = 4 - \#\text{ of Matches}(\text{E245}, \text{E265}) = 4 - 3 = 1$$

$$\text{Distance}_{\text{SOUND}}(\text{cubic}, \text{kubik}) = 4 - (\#\text{ of Matches}(\text{C120}, \text{K120}) + 0.5) = 4 - 3.5 = 0.5$$

4.1.4 Overall Distance with Unigram

Though we prepared three different types of reasonable metrics, deciding the most probable candidate should also consider the unigram probability of each candidate in a given corpus. This is basically because frequently appearing words in a corpus are far more likely to be the intended word than some word that otherwise occurred very rarely. Furthermore, since we will formulate the unigram language model by using our training set as a corpus, this also plays an important role in finding the best candidate as a common word in our context. In our experiment, we first normalize all of our distance measures to a real value in the

range [0, 1] and combined them with a unigram probability with add-1 smoothing. The followings are sample results giving weights 0.3, 0.4, and 0.3 to Levenstein, keyboard, and sound distances respectively.

$$\begin{aligned}\text{Distance}_{\text{WEIGHTED}} &= 0.3 \times \text{Distance}_{\text{EDIT}} \\ &+ 0.4 \times \text{Distance}_{\text{KEYBOARD}} \\ &+ 0.3 \times \text{Distance}_{\text{SOUND}}\end{aligned}$$

In Appendix A of this document, we show a table of the four most commonly misspelled words in our corpus followed by words from Peter Norvig's list of words hardest to correct by any probabilistic language model. These words were correctly fixed by our algorithm using a Wordnet dictionary corpus and a unigram model from the europarl corpus (rather than using our restaurant review corpus).

4.2 Intelligent Keyword Parser

As we discussed at the introduction of the section 4, in polarity classification, Hang et al (2006) strongly argues that there is a remarkable benefit to use higher order n-grams beyond unigram, bigram and trigram. From their experiment, 6-grams model shows a significant increase of F1 scores compared to low order n-grams models. However, increasing the order of the language model is not a simple task because the more n-grams that we have, the less votes each feature would get from training samples. Since the number of total features proliferates exponentially as n increases, it causes sparsity which will eventually prevent us to grasping good features. Therefore we decided to develop a novel way of extracting features from language text. It will parse each sentence from training data and figure out what the keywords are in that sentence only using grammatical structure of the sentence. Through this parser, we can collect meaningful words distributed in a sentence and combine them as features.

4.2.1 Basic Idea

The simplest type of comments in restaurant reviews are of the form: "Good service!", "Delicious food." If these are the only types of comments that we would find in the training and test sets, we don't have to consider using more than trigrams. However people are likely to express their opinion through complex sentence structuras such as "Food and service are good!" In this case, trigrams can catch "service are good" but fail to catch "Food are good". Sometimes, it is much worse and changes the original intention of the reviewer. Think about the following example.

The atmosphere is pretty bad and food is quite good

[Unigrams]
atmosphere / pretty / bad / food / quite / good
[Bigrams]
atmosphere is / pretty bad / food is / quite good
[Trigrams]
atmosphere is pretty / bad and food / food is quite

Unigrams are confusing between bad or good because there is no context information. Bigrams cannot realize whether atmosphere is bad or good and the same can be said about food. Trigrams seem to indicate that atmosphere is pretty, but food quality is bad.

As shown in the above example, unigram and bigram are not enough to catch any meaningful sentiments and trigrams sometimes committed serious misinterpretations. Instead of just using continuous number of words, we will analyze a sentence systematically through a parser tree and dependencies map which we get from the Stanford Parser. Since there we can summarize English sentence as few numbers of basic structural types, we develop analyzers for three summarized types and extract meaningful features as a compressed form. In this case, our parser's expected output would be "atmosphere bad" and "food good".

4.2.2 Initialization

Before going to parse, we should divide each sentence into a set of clauses, the smallest structure in which we can find something meaningful.

[Algorithm]

1. Parse the whole sentence

2. Get the leaves of parse tree

3. Find leaves with label: NN(Noun) or PRP(Pronoun)

4. For each node having those labels
   - Do bottom-up search on parse tree until meeting a node with S(Clause) label.
   - Save the address of S node and stop searching

Using this algorithm, as long as there is no path from leaves with NN(Noun) or PRP(Pronoun) label to the top root of original parse tree, we don't have to parse the whole sentence. It encourages us to get correct information from simple structures and reduce the amount of time in parsing.

4.2.3 TYPE1 Feature Retrieval

The simplest structural sentence frequently shown in our data set is the [A is B] type clause. When 'A' is not a PRP, we defined this type as TYPE1 structure. Usually A and B are spread out into a few words and so it achieves basic variation form given in the following.

A(1) and…and A(n) is/are B(1) and…and B(m)
"Food and service are delicious and good"

[Algorithm]

1. Find a left child with NP (Noun Compound) label and a right child with VP (Verb Compound) label in from given subtree representing each clause.

2. Reconstruct the dependency map only on the left child and search the left keyword in left child inferring through recreated dependency map. (e.g. "Food")

3. Search conjunctively connected words to the left keyword from step 2. (e.g. "service")

4. Do the same procedures in step 2 to right VP child getting from step 1. Then it will acquire right keyword through reconstructed dependency map. (e.g. "delicious")

5. Search conjunctively connected words to the right keyword from step 4. (e.g. "good")

6. Generate the full combinations between associated nodes and right associated nodes. (e.g. "Food delicious" / "Food good" / "service delicious" / "service good")

7. If the above example contains negative adverb "are not" instead of "are", then all extraction become include negative adverb. (e.g. "Food not delicious")

You can find the TYPE1 extracted results from training data set in the Appendix B.

4.2.4 TYPE2 Retrieval

The next basic structural sentence is [A do B] type. When 'A' is a PRP, we defined this type as TYPE2 sentence. In contrast to TYPE1, we assume only 'B' can spread out into a few words in TYPE2 and can be prolonged by attaching preposition "with". Then the normal form becomes like the following

A do B(1) and…and B(m) with C(1) w/…w/ C(k)
"We enjoyed the food and the service with smooth atmosphere."

[Algorithm]

1. Find a left child with NP (Noun Compound) label and a right child with VP (Verb Compound) label in from given subtree representing each clause.

2. Reconstruct the dependency map only on the right child and search the right keyword in right child inferring through recreated dependency map. (e.g. it can be either "enjoyed" or "food" probabilistically)

3. If the right keyword from right child is not a verb
   - Reserve the current right keyword (e.g. "food")
   - Register the verb in right child as a right keyword (e.g. "enjoyed")

4. If the right keyword is not a verb
   - Search the direct object word (e.g. "food")
   - Search conjunctively connected words to the right keyword from step 2 including "with" preposition attachment (e.g. "service", "atmosphere")

5. Generate the full combinations as a form of right keyword + one of right conjunctively connected words (or right keyword + reserved word when step 3) (e.g. "enjoyed food", "enjoyed service", "enjoyed atmosphere")

6. If the above example contains negative adverb "do not" instead of "do", then all extraction become include negative adverb. (e.g. "not enjoyed food", "not enjoyed service", "not enjoyed atmosphere")

You can find the TYPE2 extracted results from training data set in the Appendix B.

4.2.5 TYPE3 Retrieval

The last basic type is similar to [A do B] TYPE2. However, we defined this form as TYPE3 only when 'A' is not a PRP. Then similar to TYPE1 structure, 'A' and 'B' can spread out into a few words and 'B' can contain the "with" preposition. Then the normal form becomes like the following.

A(1) and…and A(n) do B(1) and…and B(m) with C(1) with…with C(k)
"The restaurant became noisier and dirtier."

[Algorithm]

1. Find a left child with NP (Noun Compound) label and a right child with VP (Verb Compound) label in from given subtree representing each clause.

2. Search left keyword and conjunctively connected words by doing the same step 2, 3 in TYPE1.

3. Search right keyword and conjunctively connected words by doing the same step 2, 3, 4 in TYPE2.

4. Generate the full combinations as a form of (left keyword + left conjunctively connected words) + (right keyword + right conjunctively connected words) (e.g. "restaurant became noisier", "restaurant became dirtier")

5. If the above example contains negative adverb "do not" instead of "do", then all extraction become include negative adverb instead of verb. (e.g. "restaurant not noisier", "restaurant not dirtier")

You can find the TYPE3 extracted results from the training data set in the Appendix B.

4.2.6 Analysis

As you can see in Appendix B, this parser successfully extracted important keywords from each training sentence and reconstructed them as a feature having up to three words. By eliminating all auxiliary words, it succeeded in both catching subtle relation in distance and reducing sparsity. If we combine these features with basic unigrams, bigrams, and trigrams, then it can cover almost all important features that we can think from the text-fragments.

5. Analysis

We ran our analysis using multiple different permutations of our various improvements. As can be seen below, we successfully improved training set accuracy and MSE considerably; however, we were unable to make any significant improvement on the test set data. Particularly, the lowest MSE we achieved was done with a simple unigram language model. Most of our attempts appear to have led to over fitting to the training set. Two of our attempts tended to make very small improvements: spell-correction and eliminating stop words.

5.1 Feature Effectiveness

We were surprised to see that using bigrams and trigrams failed to improve our test set accuracy or MSE. We assumed that adding features such as “good food” and “food was good” would give us improved results across the board, but that appears not to be the case. This may be due to aforementioned weaknesses in consistency of our dataset.

5.2 MSE vs. Accuracy

One significant tradeoff when classifying reviews into multiple different classes is that between accuracy and mean squared error (MSE). The multi-class C-SVM that we used attempts to maximize accuracy, i.e. the number of correct classifications. This could be suboptimal however because it doesn’t differentiate between degrees of wrongness. For example, if the correct star rating is 5, the SVM could predict any value from 1 to 4 and it would have no effect on the accuracy. It is clear that we should prefer a 4 in this scenario. Ideally we should optimize the MSE directly; however, from our experiments it would appear that the SVM tends to guess labels close to the correct label even when wrong, and thus we settled with using the SVM.

5.3 Performance

The data set we worked with was approximately 83,000 reviews. With a data set of this size, it was reasonable to collect up to 30,000 features and use a training set of 10,000 reviews. Thus the previous table of results was collected using these settings. Spell checking did increase the training time, so we saved our spell-checked data set so as to avoid this calculation for each individual run.

Unfortunately, our parser was particularly slow and we were unable to collect any results using the aforementioned number of features and reviews. Thus in order to test our parser we used a significantly smaller data set. In the future it may be possible to decrease the time needed to parse sentences so that we can train it on much larger data sets.

5.4 Aspect Classifications

We did a limited number of tests of our algorithm on predicting the aspect ratings for food, ambiance, service, and noise. As expected, the performance in predicting these was even worse than for the overall case. For ambiance and noise our results showed slightly less than 50% accuracy on the test set albeit with MSE around 0.85 for ambiance and 0.57 for noise. Accuracy for predicting service was about 54% for the test set, but unfortunately the MSE was considerably larger than in the overall case at about 1.38. Surprisingly the worst results came in the prediction of the rating of the food. Prediction of food ratings was around 40% for the test set with an MSE of 1.5. The reasons for this have not been explored and represent an avenue of future study and improvement. Particularly we suspect if we stop training all of these independently and incorporate and agreement model like that in the Good Grief algorithm, the performance on this and other aspect ratings could improve considerably.

6. Future Improvements

There are a number of things that may be done in the future to improve classification. Below we discuss three ways we can improve our results

6.1 Subjective Analysis

One important tool in improving sentiment analysis is subjectivity analysis. Within any body of text, there are two types of statements. Objective statements describe that which can be ob-

	Training Set Accuracy	Training Set MSE	Test Set Accuracy	Test Set MSE
Unigram	84.62%	0.3398	57.36%	0.8231
Unigram with spell check	84.53%	0.3256	57.12%	0.8297
Unigram/Bigram/Trigram	95.65%	0.1058	57.18%	0.9181
Unigram/Bigram/Trigram with no stop words	95.14%	0.1199	57.26%	0.8798
Unigram/Bigram/Trigram with pruning	98.66%	0.0321	56.71%	0.9052
Unigram/Bigram/Trigram with spell checking	95.53%	0.1088	57.27%	0.8984
Unigram/Bigram/Trigram with no stop words and spell check	95.38%	0.1110	57.42%	0.8936
Unigram/Bigram/Trigram with no stop words, pruning, spell check	98.77%	0.0304	56.56%	0.8970

served by other individual. These statements consist of (presumably) factual statements about what is and what has happened. Other statements can be said to be subjective. These are statements that cannot be directly observed by others and are thus referred to as private states. These include personal opinions and feelings.Objective statements do not provide actual personal opinions and sentiments, and thus can safely be thrown out of the data set without adversely affecting the results of the analysis. In fact, throwing out objective statements can often improve sentiment analysis, because these objective statements may contain features that will affect the classification predicted by our algorithm or the classifier may otherwise attempt to classify based on a purely objective statement. For example, one review stated simply:

Ordered the prix fixe meal for my girlfriend and me

This sentence is clearly objective and provides no information about overall sentiment or individual aspects. If we performed subjectivity analysis, we could detect this as an objective sentence and safely remove it from out training set preventing it from skewing our results.

6.2 Aspect Classification

In this paper we did little to improve aspect classification explicitly. In the future it would be worthwhile to explore the dependencies between the overall rating and the individual aspect ratings, food, ambiance, noise, and service. We currently are running completely independent tests for each aspect ranking. In the future we would like to implement something akin to the Good Grief algorithm (Snyder and Barzilay, 2007) which models the level of agreement within reviews. This provides a more accurate model than our current implementation which ignores these dependencies between different aspects.

6.3 Better Learning Techniques

In this report, we performed all of our training using a Classification SVM provided in the library libSVM2.91. This C-SVM was run with cost, C=2.0 and gamma=0.03125, and attempts to maximize the accuracy of our predictions, but it doesn't necessarily minimize the mean squared error. While the C-SVM tends to decrease the MSE by chance, in the future we would like to experiment with classifiers such as Support Vector Regression (SVR), which directly minimize the MSE and thus theoretically would improve our results.

7. Conclusions

We were not totally satisfied with the results of our improvements. It seems some qualities of the dataset prevented us from achieving our desired results. First and foremost are the inconsistencies among reviewers. In typical sentiment analysis, the task is simply to identify positive or negative reviews. This can be handled rather easily through bag of words techniques with unigrams and bigrams. A preponderance of instances of negative words as compared to positive words would indicate a negative review. In our case however, we must judge how positive or how negative a review is. Unfortunately, this is a very difficult task for even humans to achieve. In our own personal experiments predicting overall rankings, we were

able to achieve around 50% accuracy, no better than the results we achieved above. The results from having an independent person attempt to guess ratings gave lower than 40% accuracy. This is likely due to the person not being knowledgeable of the bias to 4 and 5 star ratings in the data, and also due to aforementioned differences in perspective among different people. No doubt predicting ratings for individual aspects would be similarly difficult and even more so in the case of ambiance and noise level. In the future, it would be wise to implement some of the improvements that take into account correlation between ratings, but we must keep in mind the practical limitations of the data set.

References

[1] Pang, B. Lee, L, and Vaithyanathan, Thumbs up? Sentiment classification using machine learning techniques. *The Conference on Empirical Methods in Natural Language Processing, 79-86.*

[2] Hang Cui, Vibhu Mittal, Mayur Datar. Comparative Experiments on Sentiment Classification for Online Product Reviews. *American Association for Artificial Intelligence. 2006*

[3] Peter Norvig. How to Write A Spelling Corrector. *http://norvig.com/spell-correct.html*

[4] Snyder Benjamin, Barzilay, Regina. Multiple Aspect Ranking Using the Good Grief Algorithm. *ACL Anthology Network 2009*

- Note on Collaboration: We developed code side by side, with both of us contributing to every substantial part of the assignment including this report.