Classification of Amazon Reviews

Mayank Agarwal, Maneesh Bhand

December 11, 2009

1 Introduction

Product reviews are a valuable source of information when it comes to making online purchases. However, some reviews are more helpful than others. The goal of this project is to predict the usefulness of a product review. Our corpus is a set of online reviews taken from amazon.com (http://www.amazon.com). Given the title of the review, the author's name and location, the author's rating of the product on a scale of 1 - 5, and the text of the review, we want to predict the percentage of users who find the review helpful.

2 Problem Statement

This problem is a supervised learning classification problem. We are given an input vector $x \in \mathbb{R}^n$ which can consist of various things, such as the summary text, review text, etc, and our output vector is the percentage of people who find the review helpful, $y \in [0,1]$. Thus, the goal is to learn the mapping $h : \mathbb{R}^n \to [0,1]$

There are two ways of approaching this problem: one way is to treat it as a classification problem by discretizing the percentage we wish to predict into bins. This allows us to use classification algorithms like Naive Bayes and SVM. The other strategy is to treat the percentage as a continuous value and use regression algorithms to predict its value. We explored both strategies and implemented various algorithms in order to find a good model for the data set.

3 Methods and Results

3.1 Multinomial model

The first model we implemented was the multinomial event model, as described in the lecture notes [3]. If we divide the interval $[0,1]$ in K bins, then the r^{th} interval corresponds to

$$\mathcal{I}_r = \left\{ y \mid \frac{r-1}{K} \leq y \leq \frac{r}{K} \right\}$$

The parameters of our model are $\phi_r(y) = \mathbf{prob}(y \in \mathcal{I}_r)$ and $\phi_{i|y \in \mathcal{I}_r} = \mathbf{prob}(x_j = i | y \in \mathcal{I}_r)$. The goal is to choose these parameters so as to maximize the log-likelihood estimates of the data.

We applied Laplace smoothing to the parameters so that the final expression is

$$\phi_r(y) = \frac{\sum_{i=1}^m \mathbf{1}(y^{(i)} \in I_r)}{m} \tag{1}$$

$$\phi_{k|y\in\mathcal{I}_r} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} \mathbf{1}\{x_j^{(i)} = k \wedge y^{(i)} \in \mathcal{I}_r\} + \alpha}{\sum_{i=1}^{m} \mathbf{1}\{y^{(i)} \in \mathcal{I}_r\} n_i + \alpha|V|} \tag{2}$$

Here α is the Laplace coefficient which we choose as 1 for our first run, $|V|$ denotes the length of our dictionary which was constructed by accruding all the distinct words in the training reviews, m denotes the total number of reviews, and n_i denotes the length of the text of the review. We divided the corpus into roughly 70% training data and 30% test data. We removed all the reviews where the number of people who voted whether they found the review helpful was less than 10. The results are tabulated below:

Number of Bins	Train Error %	Test Error %
2	16.55	28.89
3	28.125	48.76
4	31.00	57.83

Next, we removed all the low frequency words and extreme high frequency words such as 'of' and 'the', since we believed these words to be "content-free" and not indicative of the quality of the review. Originally, we had 90,000 words in our dictionary, which was truncated to 15,000. Unfortunately, however, as the results below depict, the error actually worsened.

Number of Bins	Train Error %	Test Error %
2	26.55	33.98
3	39.73	52.51
4	44.76	63.09

3.2 Stemming and Removal of Stop Words

We realized that removing high frequency words is not necessarily the right thing to do because it removes words such as 'good' and 'worst' which are very relevant to our classification. So we got a list of stop words from the course taught by Prof. Chris Manning and removed only those words. This list of stop words had around 550 words and consisted of content free words such as 'and', 'of' etc. Next, we applied a standard stemming algorithm, due to Porter[1], to our dataset. The length of the dictionary after the above two steps reduced to about 66000 words. We achieved significant improvement in the results. The table below summarizes the result achieved:

Number of Bins	Train Error %	Test Error %
2	19.92	21.89
3	31.30	38.18
4	38.43	52.44

3.3 Changing the smoothing parameter

We tried changing the smoothing parameter in the laplace smoothing equation (2) above. However, it did not make a noticable difference to the output error. By decreasing the Laplace coefficient, the training error increased while the testing error decreased.

3.4 Bigrams and Trigrams

Next, we included all the bigrams and trigrams of the words in our dataset to our model. The size of our dictionary increased to about *4 million* words. We then applied all the previous methods on this extended model and Multinomial Naive Bayes stood out. The results below indicate that the train error was close to 0% which means that the model was able to learn the training set with remarkable accuracy. The impact on testing error was significant but not as dramatic as the training error.

Number of Bins	Train Error %	Test Error %
2	0.42	20.34
3	1.22	36.90
4	2.24	48.83

The very low training error indicates that some overfitting may have occurred. A sample of n-grams the model found most indicative of a helpful or unhelpful review is below. Looking at the n-grams which were most indicative of helpfulness, some phrases, like ”arrived in” and ”shipment”, make sense as useful features (as some customers complain about shipping problems they experienced instead of reviewing the product). Other phrases, like ”man of babylon” and ”benedict xvi”, seem to be specific to certain products or reviews and may not generalize well. Nevertheless, perhaps because the corpus is so large, all these review-specific features, when aggregated, produce a model that does generalize to other, similar review data.

Indicative of Helpfulness	Indicative of Unhelpfulness
benedict xvi	man of babylon
peter schiff	money of this
liesels	her kids
a lifestyle	condition
hard things	with oprah
days of	zombies
perfume	ready the
rate of	in great condition

3.5 Looking at Other Features

There is other information contained in the reviews besides the words in the text; the length of the review and usage of capitalization/punctuation can also be used as features in predicting the usefulness of a review. Longer reviews might be more likely to be helpful, for example, while reviews written entirely in upper case might not be as helpful.

We used a set of features that included the word count, average word length, count of words beginning with a capital letter, count of words entirely in uppercase, and counts of various punctuation marks.

In order to use these features in the context of a multivariate Bernoulli model, the feature vectors, which contained continuous values, were discretized by assignment into bins. Then, each feature in the vector was treated as a “word” in the review, and the modeled probability of a review was adjusted accordingly. The results are below:

Number of Bins	Train Error %	Test Error %
2	16.51	35.90
3	18.84	37.18
4	22.79	56.99

Although some of the features selected, such as review length, did provide information on the distributions of reviews, the performance of the classifier did not improve when given these extra features; training error decreased, while testing error increased, indicating that the inclusion of extra features led to overfitting.

One possible explanation for the lack of improvement is the discretization procedure; if values are assigned to a small number of bins, a lot of detail is lost, whereas if values are assigned to a large number of bins, the bins are sparsely populated and not representative of the true distribution.

Another explanation is that simply treating the features as additional words is not an effective way of integrating them into a Naive Bayes model. A different approach to combining feature probabilities with word probabilities is needed.

3.6 Logistic Regression

In order to explore predicting the percentage as a continuous value, we implemented a logistic regression model based on just the non-word features. The predicted values were placed into bins and evaluated, so as to make the results comparable to those of the discrete classifiers:

Number of Bins	Train Error %	Test Error %
2	34.69	45.35
3	48.90	56.78
4	59.70	66.27

From the results, it is clear that the non-word features are not nearly as effective as n-grams of the review text in predicting usefulness. This is a somewhat surprising result, as one might expect features such as capitalization and punctuation to impact the legibility of a review and, consequently, its perceived usefulness.

3.7 Locally-weighted Linear Regression

We also attempted locally-weighted linear regression [3]. However, in a large corpus such as this one, the non-parametric nature of the algorithm makes computation prohibitively expensive. Therefore, we used k-means clustering on the training data in order to produce a smaller, representative set of examples, and then performed locally-weighted linear regression on these clusters. In order to obtain a reasonable measure of the distance between two feature vectors, we mapped each feature to a normal distribution and measured the distance between feature z-values.

However, even at low bandwidths, predicted values almost always fell into the 60-70% range, which is close to the average helpfulness of a review. This suggests that the non-word features alone were not sufficient for describing the helpfulness of a review, or that the clustering process removed too much information from the training data.

3.8 Other methods

We tried various other classification methods on our dataset as well. We used the WEKA software package to try SVM, Max-Ent, and the K-Star clustering algorithm. We also implemented the

Complementary Naive Bayes model, as described in [2]. However, none of these classifiers produced significant improvements in the classifier accuracy. The multinomial Naive Bayes model, using n-grams of the review text, remains, by far, the most simple and accurate model for our problem.

4 Conclusions

Various algorithms as described above were tried on the dataset. It is surprising that the multinomial model performed the best among the lot of models that we tried. The performance of the Naive Bayes model improved by applying Porter's stemming algorithm, removing stop words, and including bigrams and trigrams of the review text. We looked at some non-word features, and found that, while a few features, such as review length, did correlate with review quality, most non-word features were not effective in predicting review usefulness. Another observation is that the classifier is relatively good at identifying the very helpful and very unhelpful reviews; the confusion matrix indicates that the majority of the error is in the middle - with the classifier sometimes mistakenly predicting a neighboring bin.

The problem of predicting review usefulness is somewhat different from more standard text classification problems like spam filtering. Because the helpfulness of a review is a very subjective quantity, conventional models such as the ones described in this paper were unable to capture all of the variance in the data. For example, in the case of two bins, it is very difficult to get the classifier accuracy more than 80%,without overfitting the model.

Overall, the model is reasonably able to predict the usefulness of a review. This work is useful because it can be directly applied in a variety of ways; for example, online markets like www.amazon.com can use it to automatically serve better reviews to its customers, or to flag potentially unhelpful reviews for moderation.

5 Acknowledgements

This project is the idea of Prof. Chris Potts; we are grateful to him for providing us with the corpus and some code to process it.

6 References

[1] C. J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library

[2] Tackling the Poor Assumptions of Naive Bayes Text Classifiers, Jason Rennie et al, Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003

[3] Andrew Ng. CS 229 Lecture Notes: Supervised Learning, Discriminative Algorithms