Predicting Helpfulness of Online Reviews

Ignacio Thayer
thayer@cs.stanford.edu

December 11, 2009

Abstract

On the problem of predicting the helpfulness of online product reviews, I was able to reproduce some of the results from previous work, and made an attempt to use Latent Dirichlet Allocation on unlabeled (or weakly labeled) data to improve performance. The end result was numerically positive, but was likely not statistically significant, because it was well within the variance of cross-validation measure. I used SVM regression (SVM_{light})[2] with a radial-basis-function kernel, and evaluated the proposed rankings against held out data with the Spearman rank correlation coefficient.

1 Introduction

My project involves trying to predict utility of online product reviews. I have 48933 reviews from Amazon.com, provided by professor Chris Potts, which look like:

```
<helpful>30 of 64 people found the following review helpful</helpful>
<rating>5.0 out of 5 stars</rating>
<summary>Liberals would rather keep their fingers in their ears.</summary>
<date>October 17, 2006</date>
<author>John Steinbeck "John Steinbeck"</author>
<location>USA</location>
<review>Read the book. What is the worst threat .... [rest cut]</review>
```

The objective is to predict whether or not a given review is "helpful" or not, based on the other features of the review (eg: the text). The motivation for this is to improve user experience on commercial webpages and encourage reviewers to write more helpful reviews.

1.1 Terminology

A "reviewer" is someone who writes a review (shown above in the "author" field of the data). In the example above the "reviewer" is someone who identified himself as "John Steinbeck". A "star rating" is defined as a subjective score that a "reviewer" assigns to a product (in this case a book). A "reader" is someone who reads a review, and casts a "vote" indicating whether a review is helpful or not. The term "helpfulness" indicates the percentage of readers that found a review to be helpful.

2 Related Work

After beginning the project, I found previous work in the form of a paper published in EMNLP (Empirical Methods in Natural Language Processing) in 2006 [3]. They formulated the problem as a ranking

problem, trying to rank reviews for a given product by helpfulness. Originally I had cast this as a binary classification problem, but I changed to this formulation to compare against their work. They evaluated several features using SVM regression with a radial-basis-function kernel, and compared rankings using Spearman's ρ (accounting for ties). They determined that length, star rating and unigrams were most helpful in determining ranking. They used a different dataset than was used here.

3 Data

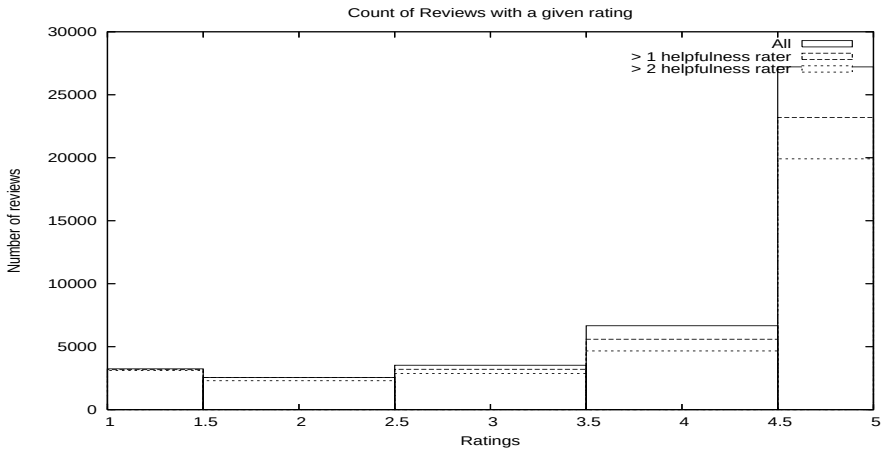

Figure 1: Distribution of star rating by number of helpfulness raters

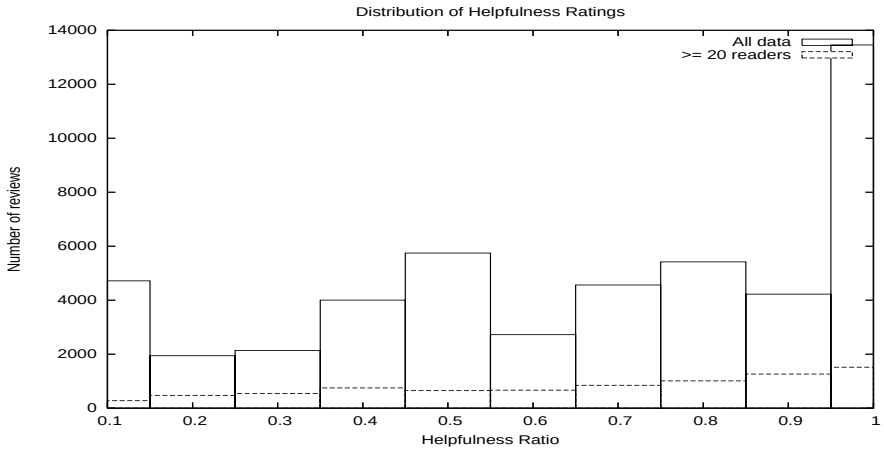

Figure 2: Distribution of helpfulness

There are clear biases present in the data. About 75% of the reviews have a star rating of 5 (Fig. 1), and most of the reviews are deemed to be helpful by users (Fig. 2).

To filter out reviews which have only been rated by few people, and experiment more quickly, I removed all reviews with < 30 readers in further experiments, leaving 4869 reviews. Where noted, I thresholded the reviews at < 20 readers, leaving 8001 reviews.

4 Goal

A normal procedure when faced with data that has low counts is to throw away data that has count below some threshold because it's unreliable. In this problem, that means throwing away reviews with few review votes. For example, if only a single person labeled a review as unhelpful, this is a weaker indication than if 100 people have labeled a review as unhelpful. In the previous work mentioned above, they filtered reviews with fewer than 5 readers (they started with much less data). One of my goals in this project was to find a way to effectively use this low-count data.

5 Experiments and results

For the following experiments I used SVM$_{light}$ in the regression mode, with the reported numbers averaged over 5 folds. The folds are split at product boundaries, so there is no product overlap in training and test.

For each fold, I do 5-fold cross-validation to determine the optimal setting of the SVM parameters C and γ over a grid of 18 possible values (this grid should be more refined). The target value of the regression for each review is the number of helpful votes divided by the total number of votes. For ranking, the SVM is run on an example, and the reviews within a completely unseen product are ranked by the output value of the SVM. I normalized the data according to the guidelines in [1].

5.1 Weighting the data

I first tried weighting the data by again adding binary-valued training instances, and including one example for each reader. In other words, if 18 out of 20 readers had indicated that a review was helpful, I would include 18 training examples of class "1", and 2 of class "0". This was unsuccessful, however - with the data prepared in this way, SVM$_{light}$ would not converge, even with a single feature. Although I removed null features, prepared the data according using the tips in [1] and the SVM$_{light}$ FAQ, I did not see convergence.

5.2 LDA-based Approaches

Another approach to use the low-count data that I tried was to learn a topic model over it that could be used to generate features for the high-count training and test data. The hope was that LDA would "discover" terms that were indicative of helpful reviews.

I tried two approaches here: the first was "seeding" initial term counts of a two-topic latent dirichlet allocation model with reviews that had most readers indicate that they were either positive or negative. For simplicity, these documents were completely distinct from the documents used for training and test. The topic model was learned over all of the low-count documents, but the "seeding" was done only with those that were predominantly labeled either helpful or not. "Predominantly" here means more than 10 readers, and more than 80% of the readers agreeing on the helpfulness (either helpful or not) of the review. When keeping documents with more than 30 votes for training/test, 4271 reviews were used for seeding, while the topic model was trained on 44,044 reviews in total. When keeping documents with more than 20 votes for training/test, 2960 reviews were used for seeding, and the topic model was trained on 40,932 reviews in total.

After the LDA model was learned, I would generate two features for each review in the training/test set by computing the expected term counts of that review in each of the two topics (helpful/not helpful). This is the "LDA" experiment included in the table below.

In addition to seeding the term vectors with particular documents, I also tried constraining the labels of the documents during estimation using the "Labeled LDA" technique (labeled as "LDA-constrained" below) [4]. These were hard constraints that were applied to low-count reviews that were predominantly

```
      Training and test performance (Spearman rho) on 5-fold CV (>= 30 readers, 4869 reviews)

                                      Training          Test
Length                                0.4681(0.0158)    0.4718(0.0548)
Length + Stars                        0.6266(0.0117)    0.6354(0.0462)
Length + Stars + TFIDF               *0.6519(0.0185)    0.6013(0.0580)
Length + Stars + Author               0.6403(0.0113)    0.6347(0.0520)
Length + Stars + Author + TFIDF       0.6432(0.0215)    0.5987(0.0609)
Length + Stars + Author + Unigrams    0.6296(0.0137)    0.6004(0.0517)
Length + Stars + Author + Product Info 0.6084(0.0089)   0.6144(0.0429)
Length + Stars + LDA                  0.6247(0.0128)   *0.6364(0.0378)
Length + Stars + LDA-constrained      0.6127(0.0139)    0.6176(0.0419)


      Training and test performance (Spearman rho) on 5-fold CV (>= 20 readers, 8001 reviews)
Length                                0.4315(0.0157)    0.4354(0.0580)
Length + Stars                        0.5730(0.0089)    0.5777(0.0526)
Length + Stars + Author              *0.6120(0.0063)   *0.5968(0.0546)
Length + Stars + LDA                  0.5691(0.0115)    0.5733(0.0512)
Length + Stars + LDA-constrained      0.5596(0.0158)    0.5665(0.0666)
```

labeled as helpful or not. This meant that throughout training of the model, only a single topic could generate words in those constrained documents, but was unconstrained on the other documents.

A third approach which I attempted, but could not get successfully working in time, was to treat the reader helpfulness ratings as draws from a binomial random variable (where the outcome is either helpful or not helpful), that would be parameterized from a true helpfulness. The observed helpful/not helpful counts would be parameters of a Beta (Dirichlet) *per-document* prior for the topic distribution of the document.

5.3 Conclusion

The only novel result I was able to show was that including author information is sometimes useful for ranking online reviews. Other experiments were not definitively conclusive. The performance obtained is in the same range as previous work, and baseline features perform as expected in relation to each other, so this project has served useful in establishing a baseline for continuing research.

5.4 Acknowledgements

Thanks to professor Chris Potts for the Amazon data and discussion, and Chris Manning for suggestions.

References

[1] Department Of Computer, Chih wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. Technical report, 2003.

[2] T. Joachims. SVM light, http://svmlight.joachims.org, 2002.

[3] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423–430, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[4] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore, August 2009. Association for Computational Linguistics.