Predicting Helpfulness Ratings of Amazon Product Reviews

Jordan Rodak
Stanford University
jrodak@stanford.edu

Minna Xiao
Stanford University
mxiao26@stanford.edu

Steven Longoria
Stanford University
sx11092@stanford.edu

Abstract

In this paper, we outline our approach to predicting the helpfulness of Amazon.com product reviews, specifically in the case of electronics products. Our developed classifier assesses a dataset of up to 5000 product reviews randomly culled from a preprocessed dataset of over 100,000 reviews to predict whether a given product review is either "helpful" or "unhelpful." Performing feature selection on anatomical, metadata and lexical features, and using Naive Bayes and SVMs with various kernels, we were able to achieve up to 76.6% accuracy on product reviews.

I. Introduction

With the growing popularity of e-commerce, online retailers such as Amazon.com and Yelp.com are increasingly relying on the ubiquity of user-supplied product reviews to crowdsource the online shopping experience, providing information to consumers and feedback to manufacturers. With the inundation of reviews and varying degree of quality of such reviews on popular online sites, we aimed to develop a machine learning approach to automatically assess and rank the helpfulness of reviews.

For the purposes of our project, we will focus on Amazon.coms product review platform. After a consumer purchases a product, Amazon prompts him or her to write a review of the product and rank it based on a star-rating scale from one to five stars. In order to differentiate reviews based on their helpfulness, Amazon has implemented an interface that allows customers to vote on whether a particular review has been helpful or unhelpful. The fraction of customers who have deemed the review helpful is displayed with the review, and Amazon uses these ratings to rank the reviews, displaying the most helpful rankings on the products front page. The drawback is that more recently written reviews are at a disadvantage since less people have voted on the helpfulness of the review. Because of this, reviews with few votes cannot be effectively ranked and will not gain visibility until they have accumulated adequate votes, which can take some time. As a result, we would like to assess the helpfulness of reviews automatically, without having to wait for users to manually vote over the course of time. If we can do this, we would be able to give users the most relevant, helpful, and up to date reviews possible, without any delay in more helpful reviews being displayed. Moreover, such an automatic classification of reviews would be able to help in rooting out poorly written reviews lacking helpful information to other consumers.

II. Dataset

To implement our project, we focused on a subset of Amazon.com product review data, which we obtained from snap.stanford.edu. In particular we are using the data set for electronic products sold through Amazon, for which there were over 1.4 million electronics reviews that span a period of 18 years up to March 2013. The raw data includes for each review the product ID, title of the product, price of the product, user ID and name of the reviewer, the fraction of users who found the review helpful, the reviewers star rating of the product (out of five), time of the review, review summary, the text of the review, and the product's description.

III. Process

A. Preprocessing

Before drawing features from the data, some preprocessing had to be done. To begin with, we iterated through the 1.4 million reviews and narrowed down our search to those reviews that had more than 10 helpfulness ratings, because the "percent helpful" attribute on our reviews would not be accurate and robust a measure for reviews with few helpfulness votes. Beyond that we also limited our search to reviews for products had at least 10 reviews, because automated helpfulness classification itself is not necessary if the given product for which the reviews are being displayed has few reviews to begin with. After narrowing down our search this way we parsed out all of the information we wanted (mentioned above). After this preprocessing we were still left with upwards of 100,000 reviews. In addition, we went through our revised dataset and calculated the average star rating for each product by aggregating across all the reviews for each product, so we could have this data present for feature extraction.

B. Feature Selection

From our raw data, for each review we obtained the text of the review, the reviewers product rating (rated out of five stars),

the reviews helpfulness rating (fraction of helpful votes out of the total number of votes), and the product description.

To derive the features for our purposes, we examined samples of Amazon product reviews and their corresponding helpfulness ratings and observed that product rating, content, and presentation of the reviews were key in assessing their helpfulness with readers. We incorporated several groups of features as our main approaches:

Anatomical features:

- Length of the review
- Sentence count
- Character count
- Number of exclamation and question marks
- Number of words in all caps

Our motivation for these structural features were to capture token-based and syntatic-based analysis of the text of the reviews. The perceived helpfulness of a review will be influenced by the review's length (for example, short reviews most likely will not contain much information about the product). Additionally, we used the exclamation marks and all-caps word counts as a rough measure of extreme emotion in reviews - reviews with high volume of exclamation sentences or all-caps tend to be overly emphatic and may prove to be not as helpful. Moreover, we purported that reviews that ask too many questions may not be too helpful to other consumers, since rather than offering information and assessment about the product they are instead asking additional questions.

Meta-data features:

- Number of stars (score rating)
- Deviation from popular opinion

Unrelated to actual text analysis, we also decided to capture the rating a reviewer gave the product at hand. In addition, we also considered a review rating's deviation from popular opinion, calculated by taking the absolute value abs(stars - avgerage stars). Our intuition was that review's which give extreme ratings that deviate considerably from the average rating will not be as helpful to a reader.

Lexical features:

- Unigrams
- Readability

We captured the unigrams features of each review text by computing the tf-idf weight for each word in a review, rather than just using the basic "bag of words" model that just takes into account word frequency. Using tf-idf ("term frequency, inverse document frequency"), we were able to scale down the weighting of stopwords such as "a," "from," "the" and other common words that appear frequently across our entire training body of reviews, and instead weigh more heavily words that appeared frequently in the single review at hand. We computed the tf-idf statistic for word t in review r using the following formulation:[1]

$$tfidf = tf * idf$$

where the tf term is calculated by the augmented term frequency (i.e. the tf factor normalized by the maximum tf of a word in the review r, and further normalized to lie between 0.5 and 1), which is normalized to prevent bias towards longer reviews:

$$tf(w,r) = 0.5 + 0.5\frac{f(w,r)}{\max\limits_{t \in r}(f(t,r))}$$

and the inverse document frequency is measured by :

$$idf(w,R) = \log\frac{N}{1 + |r \in R : w \in r|}$$

with:

- N: the total number of reviews in the training data
- r: a specified review in the training collection of reviews R
- w: the specified word in review r that we are calculating the tf-idf statistic for
- $tf(w,r) = f(w,r)$ denotes the the raw frequency of word w in review r (the number of times w occurs in r)
- $\max\limits_{t \in r}(f(t,r))$: the maximum raw frequency of any word t in the review r
- $1 + |r \in R : w \in r|$: the number of reviews in the training set R that contain the word w, adjusted by 1 to avoid any possible division-by-zero error

Additionally, we captured the readability of each review, which gauges the comprehension difficulty of each review, i.e. how easily a reader can read and process a passage. Our intuition behind using this feature was that users will generally not find reviews that are too complex or difficult to read, or at the other extreme too simple or immature diction-wise, helpful.

To assess readability of each review, we considered two such readability tests to assess review reading difficulty:[2]

- *Flesch Reading Ease Score:* computes reading ease of the material on a scale from 1 to 100, with lower numbers indicating a text that is more difficult to read.

$$206.835 - 1.015\left(\frac{totalwords}{totalsentences}\right) - 84.6\left(\frac{totalsyllables}{totalwords}\right)$$

- *Automated Readability Index:* calculates an approximate representation of the U.S. grade level needed to comprehend the text (essentially how many years of education are needed to understand the text).

$$ARI = 4.71\left(\frac{characters}{words}\right) + 0.5\left(\frac{words}{sentences}\right) - 21.43$$

We also experimented with a feature to capture the extent with which a product review contained mentions of words in the product's description, with the intuition that reviews that a higher volume of description overlap, such as mentioning certain specs of a product like "CPU" or "aperture," would contain more helpful information to the consumer reading the review. However, we observed that incorporating this feature into our feature set did not improve classification accuracy. We purport that our unigrams feature was sufficient in subsuming product description word mentions, particulary since the use of tf-idf weighting gave less weight to common words and more weight to review- and product-specific diction. As a result, we dropped this feature from our final classification evaluation.

IV. Models

In our work we modeled our problem as a binary classification problem, where we wanted to be able to take in a given review and classify it as either helpful or not helpful. To do this we took the 5000 reviews that we randomly selected from our set of 100,000 and looked at whether or not they were helpful, where we said a given review was helpful if its helpfulness rating (the percentage of people who claimed the review was helpful as opposed to not helpful) was greater than 60%. Note here that we made sure to balance these 5000 reviews, where we had an equal amount of helpful and non-helpful reviews so that our classifiers would not be biased to varied class sizes. We then took these 5000 reviews and selected a set of 1000 to test on. We left the remaining 80% as our training set, where we iterated through increasingly large set sizes from this training data and trained multiple classifiers to see how they would perform on classifying our test data. We moved our way up in set size as 50, 250, 500, 1000, 2000, 3000, 4000.

The classifiers we used were Naive Bayes, and Support Vector Machines with linear, sigmoid, radial basis function (rbf), and polynomial kernels. These classifiers are very powerful and helped model the data well. In earlier iterations of our project we attempted using linear regression, which did not fit our problem well because binary classification nature of the problem.

V. Results

Train Size	Naive Bayes	Linear	Sigmoid	RBF	Poly
50	0.541	0.561	0.503	0.585	0.587
250	0.563	0.709	0.503	0.616	0.626
500	0.575	0.530	0.497	0.655	0.658
1000	0.605	0.503	0.503	0.657	0.66
2000	0.602	0.751	0.497	0.679	0.73
3000	0.611	0.503	0.497	0.665	0.681
4000	0.614	0.766	0.497	0.682	0.685

TABLE I
TEST ACCURACY OF VARIOUS CLASSIFIERS

Figure 2 displays our results for each of the models we tested. With our most consistent models we were able to achieve

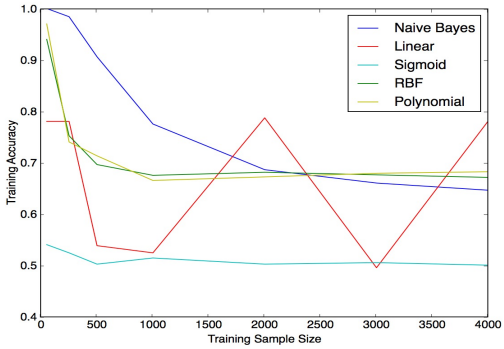

Fig. 1. Training data accuracy of models against various training set sizes.

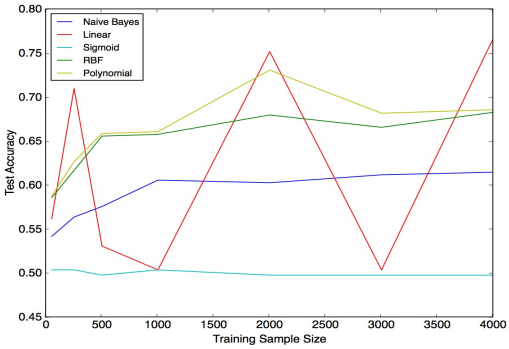

Fig. 2. Classification test accuracy of models against various training set sizes.

about 70% prediction accuracy on the helpfulness of product reviews, in particular by using rbf and polynomial kernels for SVM. Earlier in our work we had a much higher accuracy around 80%, but we then realized that it was because of a strong bias in our dataset. We had been training on a set of reviews that were 80% helpful, and so our classifiers were simply prone to predicting helpful almost all the time. Upon balancing our data our accuracy has dropped a bit more, but as we can see now there are clearly some patterns that are being detected within the data itself.

As a baseline for our discussion it helps very much to note that the sigmoid SVM, while having a bit over 50% accuracy on small sample sizes of the training set, oscillates right at the 50% mark on our test set regardless of training set size. This makes sense with respect to the fact that a sigmoid function is not a very helpful model for our type of data, and because the data is perfectly balanced we should be able to expect 50% accuracy while knowing that no patterns are being extrapolated.

Using this as a baseline we can see that our other four classifiers are doing better in most cases than 50%, which tells us that they are extrapolating some pattern(s) from the data given.

RBF and Polynomial SVMs interestingly have achieved the exact same accuracy regardless of sample size. Because RBF and Polynomial kernels are much more similar to one another than linear kernels, sigmoid kernels, or nave bayes, this makes sense to a certain extent. Another interesting pattern we see is that using a linear SVM leads to oscillations in accuracy level that are dependent upon training size, though we would expect that as our training set grows much larger the linear kernel will perform better than other models. The interesting patterns for our purposes though lie within a comparison between Nave Bayes and the RBF/Polynomial SVMs. As we can see both improve in accuracy over time, but we also notice that Nave Bayes tapers off in accuracy gain much sooner than RBF/Poly SVMs. We have a gain in accuracy of 7.3% for Nave Bayes from 50 samples to 4000 and 9.7% for the RBM/Poly SVMs. Nave Bayes has its largest accuracy gain before we hit 1000 samples though, while we see the RVM/Poly duo continuing to gain accuracy into the 2000 and 4000 sample sizes.

VI. Conclusion

To conclude we are definitely encouraged by our results. We are seeing that our classifiers are definitely extrapolating patterns from this data, and this shows promise for being able to successfully classify review helpfulness upon review publishing. An important take away though is with respect to which of these classifiers was most successful at classifying our data. Now, while linear SVM did in fact achieve the highest accuracy at certain points, it was unreliable as well, and as such we would not recommend it for the task at hand. Using the sigmoid kernel did not at all model our data well, and so we are left with Naive Bayes or rbf or a polynomial SVM. Now this comes down to the type of available data. Naive Bayes is definitely helpful for very small sample sizes, and while it does not attain the accuracy of a polynomial SVM, it gets pretty close. However, though we are able to achieve around a 70-percent accuracy, we would like to see if this statistic could be improved with hyperparameter-tuning for the rbf and polynomial kernels, and if more optimized feature selection and additional features could achieve similar improvements.

VII. Future Work

Future extensions of our work include focusing on the area of features improvement. More specifically we would like see if syntactical features such as parts-of-speech tagging, and sentiment analysis of the polarity of sentences in reviews, or capturing the subjectivity vs. objectivity levels of reviews, would improve our classification errors. And while our accuracy was promising, there is room for growth in extrapolating even deeper patterns from our data in this way.

One future goal of ours would be to incorporate pairwise ranking using the SVM linear kernel. If we could compare any two given reviews and rank one above another, this would help for specific ordering of reviews. There is some difficulty in this, as ranking reviews would require other reviews to already be present (which most of the time will be the case). Otherwise, simple binary classification as we have already accomplished can yield impressive results for times where there are no other reviews to compare to, or if we would like to provide immediate automatic feedback to a reviewer about the helpfulness of his or her review.

Additionally, we would like to generalize our classifier to any product category (i.e. Furniture or Movies), to see if our feature set could be applied succesfully to any subset of review data, not just the electronics reviews, or if developing specialized classifiers for different categories would be more optimal.

References

[1] Kim, S.M., P. Pantel, T. Chklovski, and M. Pennacchiotti, Automatically assessing review helpfulness, In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423-430, Sydney, Australia. Association for Computational Linguistics. 2006

[2] W. H. DuBay, *The principles of readability.*, Costa Mesa, Calif: Impact Information, 2004.