CSE 255 Assignment 1: Helpfulness in Amazon Reviews

Kristján Jónsson
University of California, San Diego
9500 Gilman Dr
La Jolla, CA 92093 USA
kjonsson@eng.ucsd.edu

Devin Platt
University of California, San Diego
9500 Gilman Dr
La Jolla, CA 92093 USA
dwplatt@eng.ucsd.edu

ABSTRACT
In this paper we consider models for predicting the helpfulness rating of Amazon book reviews. We examine features such as the review's star rating, the length of the review text, the readability of the review text, and the amount of comparisons made in the review. We compare Support Vector Machine and Random Forests models both for regression and classification.

Keywords: Amazon, reviews, helpfulness.

INTRODUCTION

Multiple websites such as Amazon and eBay rely on user reviews to provide the consumer with an objective review of a product. Amazon allows its users to vote on the helpfulness of a review. The ratio of positive votes reflects the review quality. Unfortunately, new products and low traffic products don't have enough helpfulness votes to accurately assess the review quality. For these scenarios its important for these sites to be able to automatically assess the review quality in order to display useful reviews to its customers and thus improve user experience.

EXPLORATORY ANALYSIS

We chose to explore the Amazon reviews data set[1]. The reviews range in date from June 1995 to March 2013. The format of the data is

- product/productId
- product/title
- product/price
- review/userId
- review/profileName
- review/helpfulness
- review/score
- review/time
- review/summary
- review/text

[1]http://snap.stanford.edu/data/web-Amazon.html. See [3].

We focused our work specifically on Amazon book reviews, which totaled in 12,886,488 reviews.

We started by limiting our data set to reviews with at least 5 helpfulness ratings. Reviews with few ratings don't have enough granularity to reflect the true helpfulness of the review. On the other hand we don't want to lose too much data. We found 5 to be a reasonable cutoff. After filtering we were left with about 4.2 million reviews.

In similar fashion to [1], we define our estimate of review helpfulness as the helpfulness ratio:

$$\text{HR} = \frac{\text{number of positive helpfulness ratings}}{\text{number of helpfulness ratings}}$$

Out of our filtered set of reviews the average helpfulness ratio was 70% with a median of 80%. Most ratings seemed to be positive which suggest that people are more likely to rate if they found a review helpful. This could also be the result of a positive feedback loop since Amazon is more likely to present the most helpful reviews to its users.

User Helpfulness Over Time

We were curious to see how review helpfulness of individual users changed over time. Do users become more adept at writing helpful reviews? In order answer this we tried looking at users who had written at least 10 book reviews. Unfortunately many of the reviews were duplicates so we had a hard time of finding a pool of users who actually had written more than 10 reviews. We tried removing duplicates based timestamp, userId and productId but only to realize that most duplicates had differing productIds. Removal would have required matching of review text were and since the filesize was large and reviews sorted by productId we abandoned this quest.

The search for features

We started by looking for correlations between various features of the reviews and helpfulness. For a review to contain useful information it would likely need to be longer than some threshold. We searched for a lower threshold under which reviews were unlikely to be helpful. From figure 2 it seemed such a threshold existed around 100 characters. Likewise we looked at very long reviews and saw that reviews of length greater than 1000

seemed more likely to be helpful (Figure 3). At any rate review length seemed like a promising feature to include.

We hypothesized that reviews with extreme ratings, like 1 star or 5 star, would be less helpful than more balanced and objective reviews. Looking at figure 1 however we found similar helpfulness distribution among all 5 different star ratings suggesting there was little direct correlation between. However, Korfiatis et al.[2] hypothesize that star rating becomes meaningful in conjunction with review length and that long reviews with extreme ratings are qualitatively different than short reviews with extreme ratings. A review with both extreme rating and a long text could suggest an explanation of a good/bad experience while a short review with extreme rating could indicate emotional rambling.

A helpful review must provide information to the users and so we expect that a review needs to be accessible to be helpful. We looked for features that measure the readability of text and found numerous measures in the literature. Korfiatis et al.[2] looked at four different readability scores in the context of predicting review helpfulness. We tried using the Gunning-Fog Index, the Smog index and the Automated Readability Index (ARI). They were all highly correlated so we chose ARI because it was the most computationally efficient one. The Automated Readability Index indicates the educational grade level required for understanding text. It ranges from 1 - 12 where grade 1 is the most readable and grade 12 the least. It is defined by [2]

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

Figure 4 shows a plot of helpfulness against readability. We can see that there is some correlation between the variables.

Another feature that we explored was how comparative a review is. A review that makes comparisons could indicate an objective comparison that provides useful information to the reader. This feature was suggested in the future work of [1] and as far as we know has not been tried before. Ideally such a feature should count the number of comparisons, but also normalize by text length in order to minimize correlation with length. We define the Comparative Index (CI) as

$$CI = \frac{\text{comparative words}}{\text{words}}.$$

To discern whether a word was comparative we used a part of speech (POS) tagger. We counted words tagged as comparative adjectives (JJR) or comparative adverbs (RBR) such as “bigger” or “better” respectively. Figure 5 shows CI against HR. Although unclear there appears to be a potential relationship between the variables.

Observations of Specific Samples

We might expect that reviews with lot’s of comparisons are more helpful. Some of the randomness in Figure

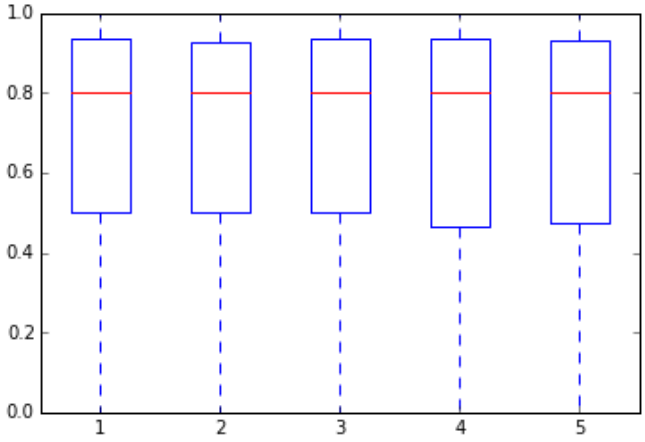

Figure 1: Review star rating vs. Helpfulness Ratio.

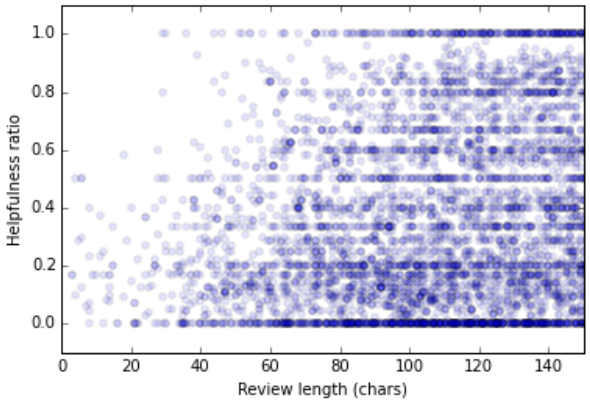

Figure 2: Review length (in characters) vs. Helpfulness Ratio (to just 200 characters)

5 might seem to go against this point, but by inspecting the review text we see that the very highest indices mainly happen for very short reviews. For example, the sample with the highest Comparative Index is

“Good seller, quick delivery, could have put a better description about book on Web”

So although the graph appears noisy, some information may be encoded when review text length is taken into account. We also can see the influence of sentence length on readability as captured by the ARI with the "most readable" and "least readable" samples:

“This book was fascinating. I could not put it down. I hope that there will be a sequel.”

“What can I say about this book that hasn’t already been said? It’s an invaluable resource for both published and non-published writers alike. The only complaint I have is that I would have liked to have seen a CN designation for "Creative Non-Fiction."”

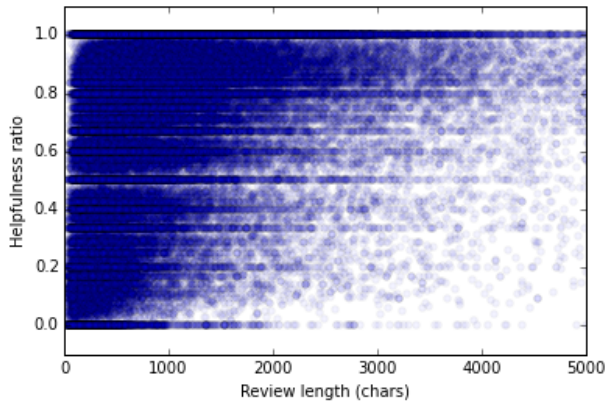

Figure 3: Review length (in characters) vs. Helpfulness Ratio

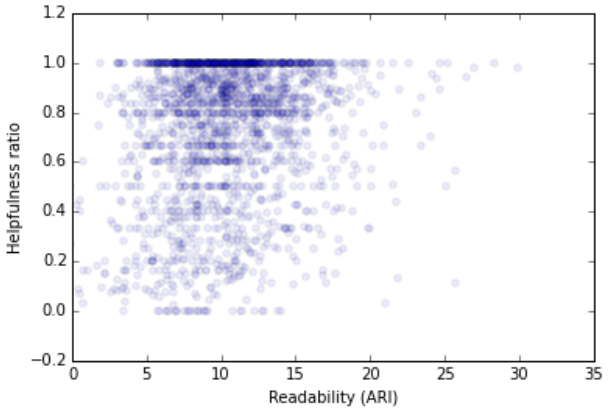

Figure 4: ARI vs. Helpfulness Ratio

PREDICTIVE TASK

We considered two predictive tasks. Predicting the helpfulness ratio and classifying a review as helpful or not. The former is a regression task while the latter a classification task. For the regressive task we decided to use mean squared error (MSE) for evaluation and both accuracy and F1 score for the classification task. For baseline comparisons we use a naive regressor that always predicts the mean of the training set or the mode in the case of classification.

LITERATURE

Data Set

Our data set came from McAuley and Leskovec [3]. This data set was used for rating prediction, product recommendation, and genre discovery. The paper does indeed have a brief section on review usefulness as well. That section discusses analysis of review text based on expected language. As far as we know though, this particular data set has not yet been used to assess helpfulness or review quality at length.

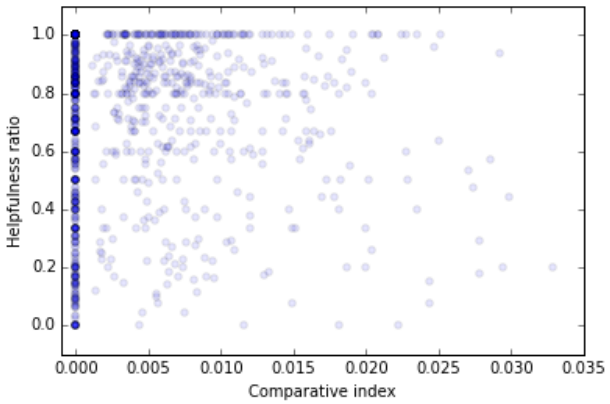

Figure 5: Comparative Index vs. Helpfulness Ratio

Previous work with helpfulness

Kim, Pantel et al. [1] studied models of predicting helpfulness ratings in Amazon reviews. Using MP3 player and digital camera reviews for their data set, and an SVM regression as their model, they found that "the most useful features include the length of the review, its unigrams, and its product rating."

They also suggested "the use of comparatives (e.g., more and better than)" as a feature for potential future work. We incorporate the use of comparatives as part of our feature set.

Korfiatis et al.[2] studied models of predicting helpfulness ratings in Amazon reviews as well, but using readability features of the review text and a Random Forests model. They found that readability was correlated with review helpfulness and that readability even had "a greater effect on the helpfulness ratio of a review than its length". For this reason readability is included as part of our feature set.

Interestingly, the Kim paper found that review score worked well as a feature for their SVR classifier. This conflicts with our exploratory analysis that indicated that there was little correlation between star rating and helpfulness score. This might be due to differences between categories, or to interplay between features (ie. maybe rating becomes relevant when taking other features into account).

Other work has been done with regards to reviewer characteristics and subjectivity of the review text [4], but we did not investigate such features because extracting such features would have required additional data sets.

RELEVANT FEATURES

Kim et al. found that review rating, review length, and review text were the most useful features for predicting helpfulness. In our exploratory analysis we saw a correlation between review length and helpfulness (figures 2 and 3). We also saw that reviews had a lower length threshold for being informative. We didn't find any di-

rect relationship between star rating and helpfulness, but hypothesized that rating might be relevant in conjuncture with review length. We also explored the use of features related to the accessibility of the text such as the readability (ARI) and came up with the Comparativeness Index which measure how comparative a review is.

For length we used 4 features. We used the text length in characters which we normalized by dividing it by 1000 and three binary variable indicating whether a review was short, medium or long. We hoped that the binary variables would capture the thresholds we saw in the exploratory analysis (Figures 2, 3).

For the star rating we used vectorized binary features. We could have used a single numeric feature but we wanted a more expressive representation.

The following is a list of our features:

- float: Normalized review length (in characters)
- binary: Short review? (< 100 characters)
- binary: Medium review? (100 to 1000 characters)
- binary: Long review? (> 1000 characters)
- Binary 5-tuple: Star rating
- float: CI (Comparativeness)
- float: ARI (Readability)

For the regression model our labels were the helpfulness ratio (HR), a floating point number between 0 and 1. For the classification task we labeled a review as helpful if the HR was larger than 0.8, otherwise as unhelpful.

We also considered using a bag-of-words feature representation, but we couldn't beat our baseline models with it and it slowed training and processing considerably.

Preprocessing

The text features required a fair amount of pre-processing, as discussed in the exploration section. Calculating the comparative index required tokenization and tagging of the review text. The authors tried various taggers before settling on a fast enough implementation. Taggers found in the standard NLTK were orders of magnitude too slow. We used an implementation of a combined Brill, regular expression, affix, unigram, bigram, trigram tagger to process the data efficiently.[2]

Readability required calculating the number of words and sentences in the review text. We note that in our experience calculating the ARI is actually substantially faster than calculating other indices such as the SMOG, which requires counting syllables (or the Gunning-Fog index, for which SMOG is intended as a quicker alternative).

MODEL

We looked at two models, SVM and Random Forest, for both regression and classification tasks.

[2]See http://streamhacker.com/2010/04/12/pos-tag-nltk-brill-classifier/

SVM Regression

An linear SVM regressor minimizes [6]

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\chi_i + \chi_i^*),$$

subject to

$$\begin{aligned} y_i - < w, x_i > -b \leq \epsilon + \chi_i \\ < w, x_i > +b - y_i \leq \epsilon + \chi_i^*. \\ \chi^i, \chi_i^* \geq 0 \end{aligned}$$

where C is a penalty parameter, ϵ the insensitive tube parameter. Similarly to a SVM classifier the Kernel trick can be applied to increase the model expressiveness. Kim, Pantel et al. [1] predicted helpfulness rating with a SVM regressor using a radial basis function (rbf) kernel. Following that we experimented with both a linear kernel and rbfs. Like them we had the best result with radial basis functions. An SVM regressor with an rbf kernel has three hyperparameters; C (the penalty parameter), γ (the kernel width parameter) and ϵ the insensitive tube parameter.

We did a grid search over these parameters trying out around 50 different possible combinations geometrically spaced. We quickly ran into scaling issues with this. The running time seemed to increase quadratically with the number of samples. To deal with this we decreased our training set to 30K samples for the parameter tuning and used 3-fold cross-validation on it. After we found the best parameters we fitted the model on a set of 100K samples. Interestingly our best parameters; $C = 1$, $\gamma = 0.1$, $epsilon = 0.3$ were very close to scikit-learns default values and performed only a little bit better suggesting that scikit-learn has good default values.

Our training error and cross-validation error tracked each other very closely which gave us confidence that our model wasn't overfitting the data. This wasn't surprising since our feature dimension was low. Since we seemed to have some room for increasing the model complexity we tried increasing the degree of our kernel from 3 to 4 but that gave us worse validation error.

SVM Classification

As the average helpfulness ratio was 80 percent, classification required care with unbalanced classes. We experimented with various forms of discretization of the helpfulness ratio: different splits for a binary classifier and also a ternary classifier split along the 35 and 65 percent lines for "unhelpful", "neutral", and "helpful". All of these variants suffered from over guessing the mode ("helpful"), even when attempting to correct for class imbalance by adjusting weights inversely proportional to the class frequencies.

In the end we decided to discretize the helpfulness ratio into two classes, "unhelpful" and "helpful", split along the median ratio of 80 percent helpfulness. This balanced the classes, and also seems reasonable since in

practice we would only want to label truly helpful reviews as helpful. Helpfulness would likely be used for the ordering of reviews presented in a user interface.

Tuning of parameters worked similarly to that with the SVM regression; a grid search yielded values close to the defaults in the scikit-learn implementation. Scalability was also an issue as it was with the SVR.

Random Forests

After trying out SVMs for both regression and classification we decided to test a different model, Random Forests. These models were used successfully for predicting helpfulness and sales of Amazon reviews and products in [4] (where they were shown to outperform the more commonly used SVMs). The Random Forests model constructs multiple Decision Trees and by sampling the training set and features differently among the various constructed trees the variance and risk of overfitting is drastically reduced [7]. They can rate feature importance and are resistant to redundant features [7]. Another great advantage of Random Forests is that they lend themselves well to parallelization and run blazing fast on a small feature space. It had no trouble training the model on 900K samples. Random Forests can be used both for regression or classification by outputting either the mean or the mode, respectively, of the decision trees outputs. We tested Random Forest on both the regression and classification tasks and got better results. We used a hold out set of 100K training example to come up with a good number of decision trees for the forest. As we increased the number of trees in the forest the mse got better but training and predicting with the model got slower. For this reason we stopped at 150 trees in the forest for the regressor and 100 trees for the classifier. The biggest disadvantage is that our final forest of 150 trees took about 6GB of space when saved to the HD compared to the SVR, which took less than 10MB.

For the classification task we used a gini index criteria to determine splits.

Unsuccessful attempts

We tried to use bag-of-words (unigrams) features in our model as suggested in [1]. Bag-of-words model have been successfully used in sentiment analysis for predicting whether a review is positive or negative [5]. The task of discerning helpfulness is different but we expected that bag-of-words features could be indicative of review quality since some words might be frequently associated with review quality. We experimented with using different vocabulary sizes, with and without term frequency (tf) scaling, stemming and smoothing. We then trained a SVM regressor on the features. This model performed very poorly and we could not get it to beat our base benchmark MSE, which is the MSE gotten by guessing the average every time. This also increased the dimensionality of our features by a lot and slowed down processing and training considerably. For this reason we ended up abandoning the bag-of-words in search of simpler features.

Results and Conclusions

Results

Table 1 shows the performance result on both the regression task and the classification task. We held out a set of 100K samples during both training and validation and tested our final models on it. In the case of the SVM classifier we only used 20K samples from the test set because it ran so slow. Both Random forests and SVM regression got a higher mse on the on the test set than the validation set indicating that there was mild overfitting. Both performed better than the baseline classifiers (guessing the mode or mean) but Random Forests did significantly better with 75% accuracy, a .74 F1 score, and 0.48 MSE.

Table 1: Comparison of Models

	Classifier		Regression
	Accuracy	F1 Score	MSE
SVM	0.577	0.678	0.0583
Random Forest	**0.754**	**0.737**	**0.0477**
Baseline	0.470	—	0.0847

Important Features

Using the random forest regressor we could estimate which features were important and how important each feature was.

Table 2: Comparison of Feature Importance

Feature	Importance
Normalized review text length	0.41
Readability (ARI)	0.21
1 star rating	0.15
Comparative Index	0.12
2 or 3 star rating	0.06, 0.04
4 or 5 star rating	0.004, 0.009
Discrete review text lengths	< 0.001

Table 2 show the features in order of importance. review text length, readability, comparativeness, and star-rating. We expected the length and readability to be important features like we had seen in the literature. We were quite pleased to see how well the comparativeness worked because it had only been mentioned in the Kim et al. paper as a possible feature.

The prominence of the star rating importance demonstrates that given other features, star-rating becomes relevant. It appears that 5 features is unnecessary to encode most information; if we know that a review is not 1,2, or 3 stars, then it must be 4 or 5 stars What's interesting, is that a one-star review is so important. In fact, it's value is 2 to 3 times the value of a two star review. Thus, a simple binary feature (good review or bad review) probably isn't enough to encode all relevant information for helpfulness.

Future improvements

To improve our result we could have cleaned our data better before training. The data we used was quite messy and contained a lot of duplicate reviews. We

removed duplicates for the case when the product ids matched but later realized that in most cases they didn't. These duplicates might have introduced a bias in the classifier and furthermore might have caused our test set to be contaminated with already seen samples. This could mean our test accuracy is too high.

With cleaner data we also could have looked at the temporal aspects of user helpfulness and added temporal features to our model.

We also were unable to replicate any good results with a bag-of-words representation like in the Kim paper. This may be because of our product category (there may be a larger vocabulary for books in general vs. mp3 players), but without attempting a model on both categories we cannot reach any conclusions to that point.

One area with room for improvement could be our Comparative Index. It lends too much weight to very short reviews and a slightly more complicated calculation might be able to attenuate this problem. Perhaps normalizing with the log(words) instead of just the word count would have improved our score.

The prominence of the low-star rating features remains unexplained. Are these one-star ratings helpful, or not helpful? Long or short? Readable, or use comparatives? It would be interesting to investigate in exactly which situations the one-star rating becomes important.

Conclusion

In this paper we considered models for predicting the helpfulness rating of Amazon book reviews. We examined features such as the review's star rating, the length of the review text, the readability of the review text, and the amount of comparisons made in the review. We confirmed the results of previous work that review length, readability, and rating are relevant features for predicting helpfulness, and we found that comparisons also work well as a feature.

We also compared Support Vector Machine and Random Forests models both for regression and classification, and reproduced the findings in [4] that Random Forests perform more accurately with much quicker training.

REFERENCES

1. Kim, S. M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006, July). Automatically assessing review helpfulness. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (pp. 423-430). Association for Computational Linguistics.

2. Korfiatis, N., García-Bariocanal, E., & Sánchez-Alonso, S. (2012). Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. Electronic Commerce Research and Applications, 11(3), 205-217.

3. McAuley, J., & Leskovec, J. (2013, October). Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on Recommender systems (pp. 165-172). ACM.

4. Ghose, A., & Ipeirotis, P. G. (2011). Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. Knowledge and Data Engineering, IEEE Transactions on, 23(10), 1498-1512.

5. Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10 (pp. 79-86). Association for Computational Linguistics.

6. Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and computing, 14(3), 199-222.

7. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.