Predicting the Helpfulness of Amazon Reviews

Cristian Caraballo Richard Addo Hanna Kim

Abstract

With the increasing importance of online product reviews in the purchasing decisions of online shoppers, it has become important for online businesses to present helpful reviews to prospective buyers. A great majority of these businesses, if not all, use crowd-sourced "helpfulness" scores that are from binary ratings (helpful or not helpful) from review readers. Previous studies have used methods such as support vector regression (SVR) and linear regression to predict this helpfulness score with varied results. In this study, a Random Forest regression model was used to predict the "helpfulness" score from a dataset of Amazon reviews. A regularized least squares linear regression model was also implemented for baseline and comparison purposes. The Random Forest and linear regression implementation resulted in average errors of 5.22% and 5.5%, respectively, compared to 5.65% and 7.36% errors of SVR and linear regression respectively from a previous study using different features [1].

Introduction

Reviews have completely transformed the experience of online shopping. In the past, there was much risk involved for the consumer because of uncertainties in the product's quality, durability, efficiency, and etc. Now, reviews serve as a tool to minimize the distance between the product and the consumer. In theory, having a public forum on these shopping websites where previous buyers can enter in text reviews and star-based ratings should solve the problem. However, such a method does not account for the fact that undetailed textual reviews do exist and do little to inform the decision making process of a potential buyer. There have been many modifications made on shopping websites to work towards this goal of providing a potential buyer with enough information to help decide whether or not to purchase an item.

Amazon.com, for instance, poses the question, "Was this review helpful to you? Yes/No" to a potential buyer reading a review, and automatically sorts reviews based on “helpfulness”. However, not everyone reading a review responds to this question, and hence inundation of unhelpful reviews still persists.

Some work has been done to study the specific factors that affect helpfulness, while other work has focused on the prediction of helpfulness scores. It has been found that review extremity (represented by the reviewer’s star rating of the product) and review depth (including review length) tend to be taken into account by consumers when considering the helpfulness of a review [2]. In the 2006 study by Zhang and Varadarajan, SVR and linear regression were used with a feature set including lexical similarity of the review to the product specification, part-of-speech tags, and the number of subjective words.

In this present study, we propose the use of regression Random Forest using word presence, review length, review extremity, and product price to predict the helpfulness score of reviews. Random Forest was chosen because they have shown comparable accuracy to state of the art algorithms such as SVMs [3] while providing additional speed benefits.

“HELPFULNESS” DEFINITION

For the purposes of this study, “helpfulness” is defined as the following:

$$Helpfulness = \frac{\#\ of\ Helpful\ Votes}{Total\ \#\ of\ Votes}$$

DATASET

We successfully obtained links to the entire Amazon reviews dataset compiled by the Stanford Network Analysis Platform (SNAP) Group as part of the Stanford Large

Network Dataset Collection. The dataset consists of almost 35 million consumer product reviews from Amazon.com, spanning 18 years. Almost 2.5 million products are reviewed by over 6.5 million users, and over 50,000 of these users have reviewed more than 50 products.

The original dataset comprises reviews for 28 different consumer product categories. However, all training and testing so far has been performed on a subset of reviews for the "Electronics" category.

Each review in the dataset comes with the user's name and ID, the user's rating of the product, the total number of votes on the review with the corresponding number of "helpful" votes, and other information in addition to the actual review text. Additionally, the dataset provides a separate file containing descriptions of all the products. Figure 1 is a snapshot of the format of each review in the dataset [4, 5].

```
product/productId: B00006HAXW
product/title: Rock Rhythm & Doo Wop: Greatest Early Rock
product/price: unknown
review/userId: A1RSDE90N6RSZF
review/profileName: Joseph M. Kotow
review/helpfulness: 9/9
review/score: 5.0
review/time: 1042502400
review/summary: Pittsburgh - Home of the OLDIES
review/text: I have all of the doo wop DVD's and this one is as good or better than the
1st ones. Remember once these performers are gone, we'll never get to see them again.
Rhino did an excellent job and if you like or love doo wop and Rock n Roll you'll LOVE
this DVD !!
```

Figure 1: Snapshot of each review in the dataset. Most pertinent to this study are "review/helpfulness", "review/score", and "review/text".

PRE-PROCESSING & FEATURE EXTRACTION

Preliminary experimentation was done via 5-fold cross-validation to determine which representation of the review text would produce optimal results. Figure 1 and Table 1 show that the word presence representation outperformed the word frequency

representation with an average error of 5.22% compared to 5.88%. For this reason, all further experimentation were performed using the word presence representation of the review text. These preliminary results may be due to the word frequency model not generalizing as well as the word presence model, especially since the review lengths tend to be relatively short. Indeed, Zhang and Varadarajan also counted the total occurrences of words in the list, rather than individual word frequencies [1].

Review Text Representation	Average Mean Squared Error
Word frequency	0.0588
Word presence	0.0522

Table 1: Average mean squared errors for the two representations of the review text.

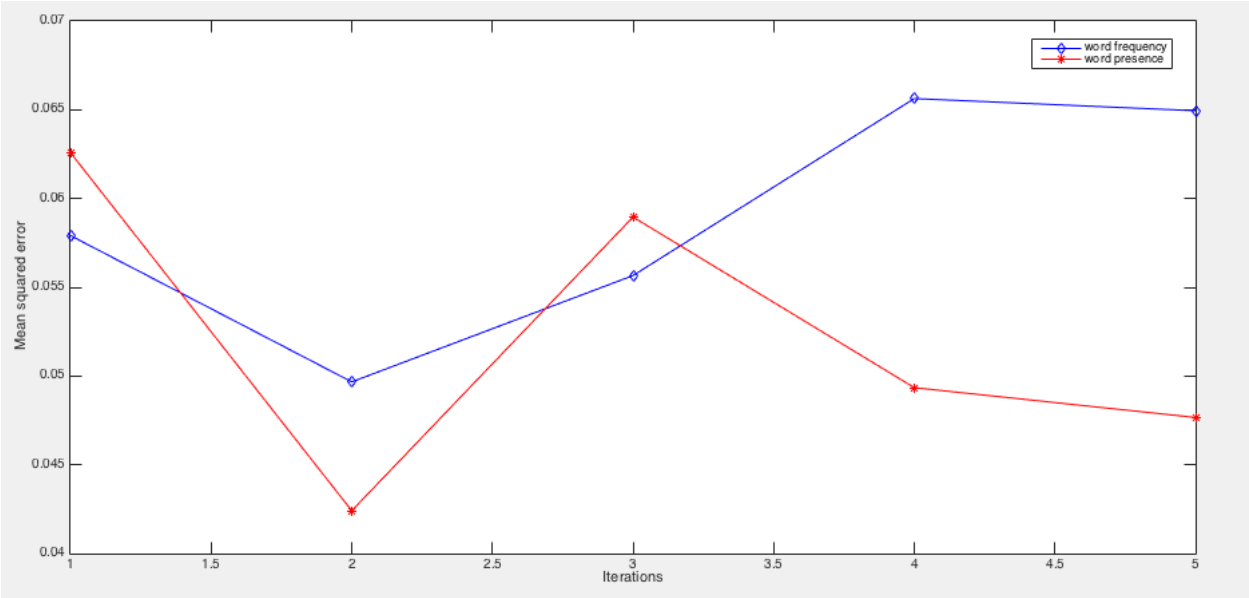

Figure 1. Comparison of review text representations (word presence vs. word frequency). These values were obtained by performing multiple runs on solely the training set. In each run, a fraction of the training set was used for testing and the remaining for training.

Previous studies have shown that the length of a review and the review writer's rating of the product correlate with the helpfulness of the review [2]. For this reason, we

included the aforementioned two attributes of a review in the features for each review. For each review, the feature vector comprises

- a word presence representation of the review text.
- the review writer's rating of the product.
- the cost of the product.
- the number of words in the review text.

Before extracting features for the reviews, a filter was applied on the dataset. Only reviews with at least 30 words in the review text and at least 10 votes on the review were included in training and test sets.

Using of the selected reviews, a vocabulary was built from the subset of the data selected as the training set. Porter stemming was applied to the review text of each of the reviews. A mapping of stemmed words to their frequencies in the training set was created and the *n* most frequent words were selected as the vocabulary.

Finally, features were extracted for each of the training and test set examples. The same Porter stemmer was applied on each word in the review. A check for the presence of the vocab words in the stemmed review words was done to obtain the word presence representation of the review text. The product rating (discrete values from 1-5), product cost, and review length were then added to the created feature vector.

RANDOM FOREST IMPLEMENTATION

The Random Forest comprises a set of regression trees. Each tree is built using CART methodology and allowed to grow to the maximum depth without any pruning. Additionally, two layers of randomness are employed in the Random Forest implementation:

- **Bagging**: A random sample of the training data is selected (with replacement) and used to build each tree in the forest.

- **Random features**: To split a node, a random sample of the features are selected and evaluated with different thresholds to identify the best feature and threshold to split on at that node.

These two layers of randomness were employed because they have been shown to produce high levels of accuracy [3].

Termination Criteria

In the current implementation, the only termination criterion is the size of the node. In the code, a variable `minSplittableNodeSize` is preset, and during the building of a tree, any node that has fewer than `minSplittableNodeSize` examples is not split any further. For testing, `minSplittableNodeSize` = *50*.

Prediction

The final predicted helpfulness score for the given review is the mean of values predicted by the regression trees in the forest.

RESULTS & DISCUSSION

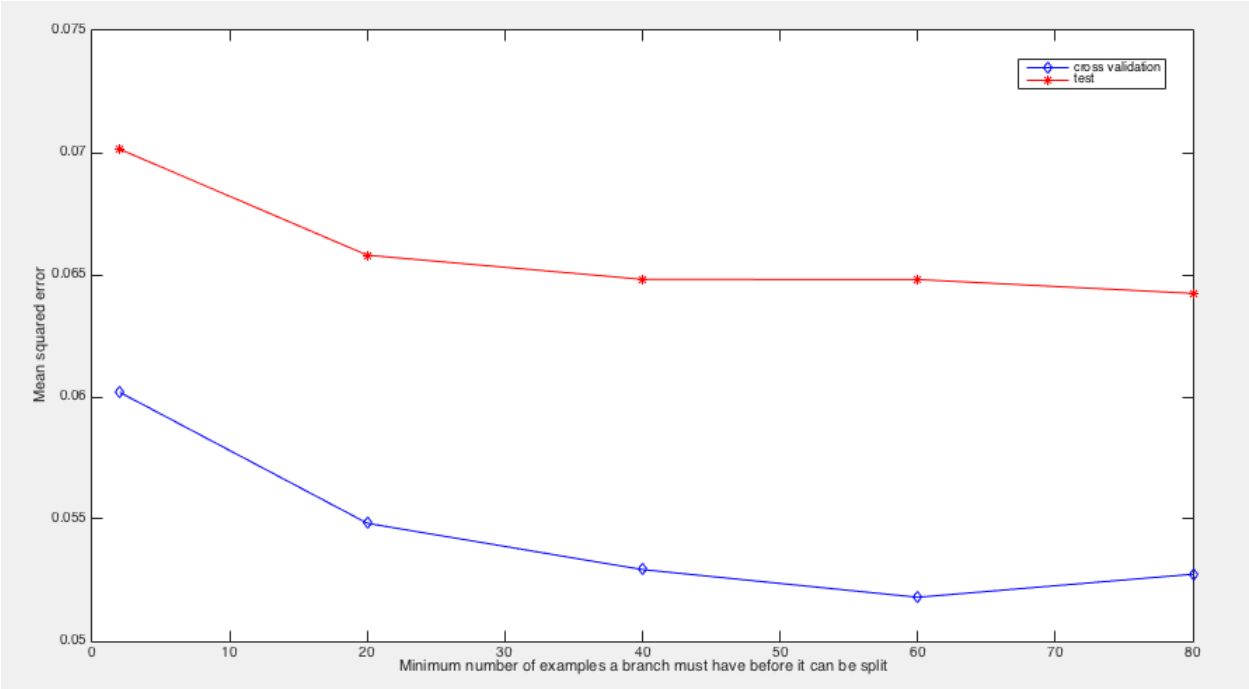

Figure 2: Cross-validation results of varying the minimum number of examples a node must have before splitting.

Figure 2 shows the effects of varying the initialization of the variable `minSplittableNodeSize` on the overall performance of the random forest. By varying the size of `minSplittableNodeSize`, what is being affected is the minimum number of examples a node must have before it can be split further. Shown on the graph above (Figure 2), as the minimum number of examples increases, the mean squared error (MSE) decreases, albeit almost negligibly, for both testing and cross-validation.

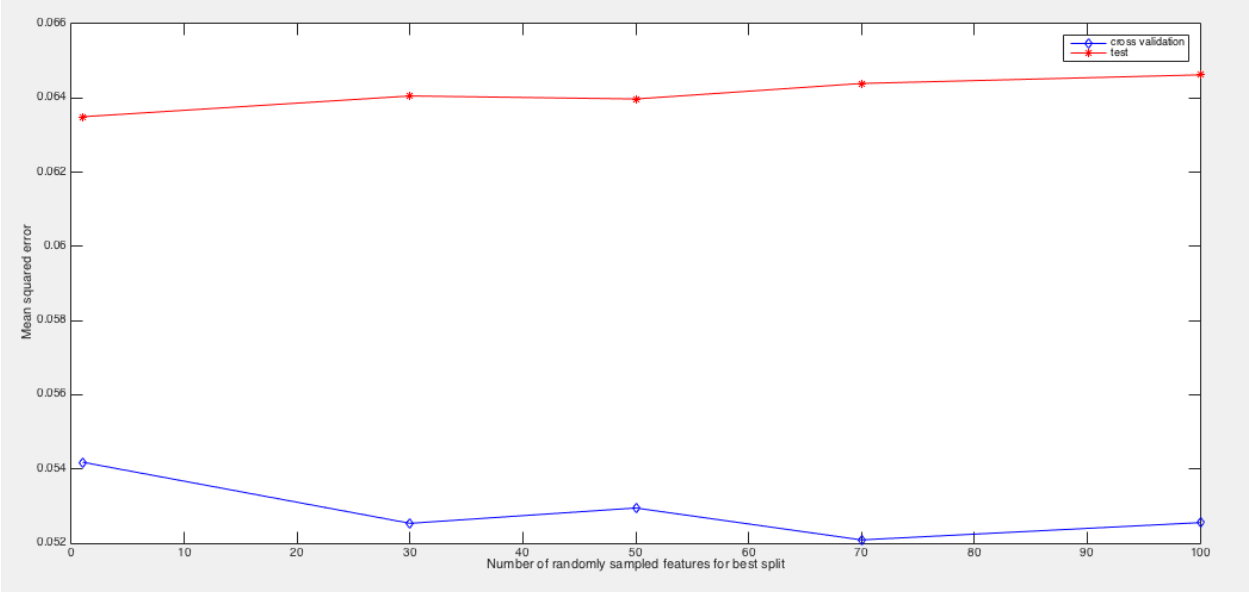

Figure 3: Cross-validation results of varying the number of candidates randomly selected and considered for the best split

Figure 3 shows how much varying the number of candidates selected and considered for best split affects the overall performance of the random forest. As shown on the graph, there is not much fluctuation in the results with randomly sampling the features for best split. The MSE of the cross validation stays between 0.052 and a 0.054, while for the test, the MSE lies between 0.063 and 0.064.

In summary, Figures 2 and 3 show that for both cross validation and testing, varying either parameter negligibly impacted performance of the random forest. However, this is expected for random forests [6].

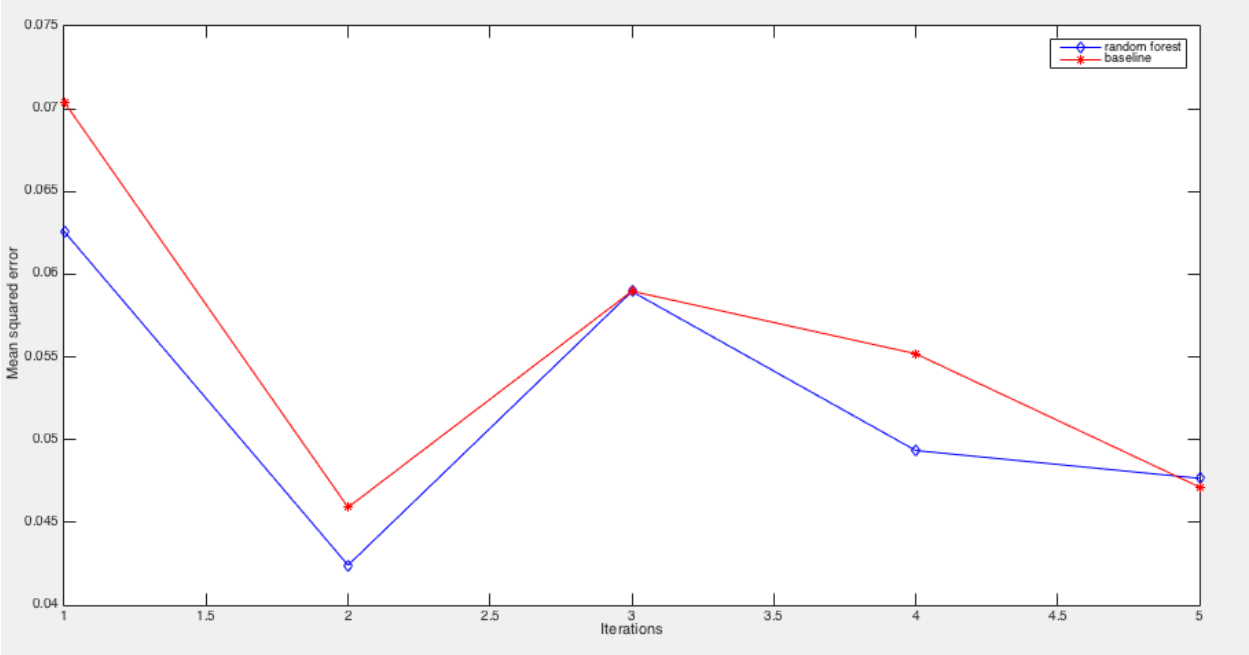

Figure 4: Random Forest performance vs. regularized least squares linear regression baseline.

Figure 4 shows the comparison between the random forest algorithm and our baseline, regularized least squares linear regression. The random forest model outperformed the baseline on an average basis with the first iteration being a difference of almost 1%. Although this difference does not seem particularly significant, it is similar to that of previous studies. Zhang and Varadarajan reported ~2% difference between their SVR and linear regression model results.

CONCLUSION

As discussed, the obtained results from a word-presence regression model of Random Forests are comparable to, and in some cases better than, previous studies using SVR and linear regression. With the added benefits of speed and ease of use, we hope for further exploration of the use of Random Forests in tackling the problem of predicting helpfulness of online reviews.

IMPLEMENTATION DETAILS

All code used in this project can be found in the submitted `AutoRateReviews` directory. This directory contains 5 main subdirectories.

- `CommonFuncsAndScripts` - This directory contains functions and scripts that are used in both learning methods that were implemented (i.e. Random Forest and Regularized Least Squares Regression)

- `Data` - This directory houses the dataset used in the project. The submitted version comes with a subset of the Electronics category reviews.

- `Baseline` - This directory contains all functions and scripts that are used in the regularized least squares baseline.

- `RandomForest` - This directory contains all major scripts and functions used in the Random Forest implementation

- `External` - This directory contains all third party source code that was used in the project.
    - *porterStemmer.m* - This implementation of the Porter stemming algorithm was used in the preprocessing step. The function was obtained from the MatlabNLP library: http://faridani.github.com/MATLABNLP
    - *reviewStopWords.stop* - This file contains the words used as stop words in the preprocessing and feature extraction stage. This file is a modified version of english.stop available at : http://faridani.github.com/MATLABNLP
    - *SanitizeComment* - This function strips a review text of any punctuations. The function was obtained from the MatlabNLP library: http://faridani.github.com/MATLABNLP

REFERENCES

[1] Z. Zhang and B. Varadarajan. Utility Scoring of Product Reviews. *Proceedings of the 15th ACM international Conference on Information and Knowledge Management (CIKM ',06)*. pp. 51-57, 2006.

[2] Susan M. Mudambi and David Schuff. What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.com. *MIS Quarterly,* Vol. 34, No. 1. (2010), pp. 185-200

[3]. Andy Liaw and Matthew Wiener. Classification and Regression by randomForest. *R News*, Vol. 2, No. 3. (2002), pp. 18-22.

[4] http://snap.stanford.edu/data/web-Amazon.html.

[5] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. ACM Recommender Systems conference (RecSys), 2013.

[6] Leo Breiman and Adele Cutler. Random Forests. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm