# Autoranking Amazon Reviews

Ivan Gozali, Ilker Karakasoglu
Stanford University
{igozali, ilker}@stanford.edu

*Abstract*—**We consider the problem of predicting real valued scores for reviews based on various categories of features of the review text, and other metadata associated with the review, with the purpose of generating a rank for a given list of reviews. For this task, we explore various machine learning models and evaluate the effectiveness of them through a well known measure for goodness of fit. We also explored regularization methods to reduce variance in the model. Random forests was the most effective regressor in the end, outperforming all the other models that we have tried.**

## I. Introduction

In the recent decade or so, online shopping has in many ways replaced the need to go to physical stores. While this is more convenient and efficient, and provides consumers with a wider range of products compared to shopping at a department store, there are drawbacks as well, one of the main ones being the inability to physically hold and assess the quality of the product before purchase.

As such, people are increasingly reliant on product reviews, as it allows them to assess the product in various aspects to see if it matches their needs. However, for popular products reaching hundreds or thousands of reviews, it is difficult for potential buyers to efficiently sort through the more helpful reviews.

Shopping websites, such as Amazon, have implemented a helpfulness vote indicator so that consumers can rate other buyers' reviews to improve the experience. However, most of the reviews that exist in Amazon have either very low or zero votes. Especially for popular products that have a high number of reviews, it's difficult for newly written reviews that are helpful to be read by people, since only the old and helpful reviews show up on the top of the review list for those products.

In our project, we pose the problem of designing a supplemental way to rank reviews by looking at their certain features, such that these reviews which could potentially be helpful can be surfaced to the top of the list. We attempt to use machine learning algorithms that are appropriate for string data and ranking. By analyzing the portion of reviews that have enough visibility and a high number of votes, we aim to build a model that could predict the helpfulness of reviews with zero or few votes.

## II. Related Work

There has been a fair amount of related work being done on Amazon product reviews. A study done by Minna et al performed very similar work in using binary classification to determine whether a review is helpful or not [1]. By extracting several features from the review text and evaluating performances of different classifiers, including Naive Bayes and support vector machines with various kernels, the authors were able to achieve an accuracy of around 70%.

Bolter used distinct classification models for each product category in Amazon, noting that different words might work better as predictors depending on the category of the product [2]. Words that are frequently used in the electronics category might not work well when it is used to classify clothing products.

Yet another study done by Kim et al performed very similar work in using supervised learning methods to generating scores to rank reviews [3]. In this study, helpfulness scores for reviews were derived from how many upvotes or downvotes a review received from human readers, which in turn were used to train the regressor. The study concluded that SVM regression showed very promising results, with rank correlations up to 0.66, while the most significant features were shown to be unigrams, product rating, and the length of the reviews.

Having taken a look at some of the studies, we note a couple of areas that we can further investigate. Most of the studies performed above attempted to fit a binary classifier to determine whether the reviews are either helpful or not. However, it is not possible to generate an ordering of reviews by helpfulness simply by assigning binary labels to reviews, because there is no quantity to compare whether some reviews are more helpful than others. Therefore, the user will still have to read through several reviews before being able to make a decision to buy the product. In this sense, a regressor would be more suited to perform the task of generating a rank.

In addition, in the study done by Minna et al, the authors did not attempt to tune the hyperparameters to optimize some of the classifiers that they were using. We note that by considering effects of regularization, we can reduce the variance of our regressors such that they may perform better.

We acknowledge that our work is very similar to that of Kim et al. Due to time constraints, we limited the scope of our project to training regressors using a subset of features used by Kim et al, and evaluating the performance by using the standard coefficient of determination metric for evaluating goodness of fit in regression problems. Our contributions are the evaluation of several other regressors than those that were considered in the aforementioned study, and the exploration of regularization methods to reduce variance and bias in the regression models.

## III. Dataset and Features

### *A. Dataset*

We utilized the set of Amazon reviews written within the date range of May 1996 - July 2014, in particular for the Toys and Games product category. These reviews were originally procured and curated for a study done by McAuley et al [4]. Each review within the set contains information about the reviewer ID, product ID, review upvotes and downvotes, the review text itself, the rating the reviewer gave the product, the title of the review, and the time the review was written, all formatted in JSON. For the implementation, we use Python utilizing the libraries of pandas and scikit-learn [5].

The original dataset contained 2.25 million reviews. In order to able to extract enough number of trustworthy features and reflect the objective of this study better, we filtered the review data with the following criteria: (1) each review needs to have more than 10 votes (thereby having enough visibility), and (2) the review should exist in popular products (with more than 15 reviews). By using these criteria, we reduced our data set from 2.25 million down to about 60,000 reviews. We then randomly selected 80% of the data as the training set, and the other 20% as the test set.

### B. Outcome Variable

Borrowing the idea from the study done by Kim et al [3], we define the following quantity to provide a measure of the sense of helpfulness a review provides given the $i$-th review:

$$y^{(i)} = \frac{\text{upvotes}^{(i)}}{\text{upvotes}^{(i)} + \text{downvotes}^{(i)}}$$

Originally, we thought of defining the outcome variable to be the difference between upvotes and downvotes of a particular review. This definition, however, doesn't differentiate the case between a review that has 105 upvotes and 100 downvotes versus a review that has 5 upvotes and 0 downvotes, although one would think that the first review could have been less helpful due to the review having downvoted by people.

We visualized the data to get a better understanding of it. The histogram of the review counts grouped by the outcome variable, as shown in Figure 1, indicates skewness in the data. Approximately 70% of the reviews have helpfulness scores between 0.8 to 1.

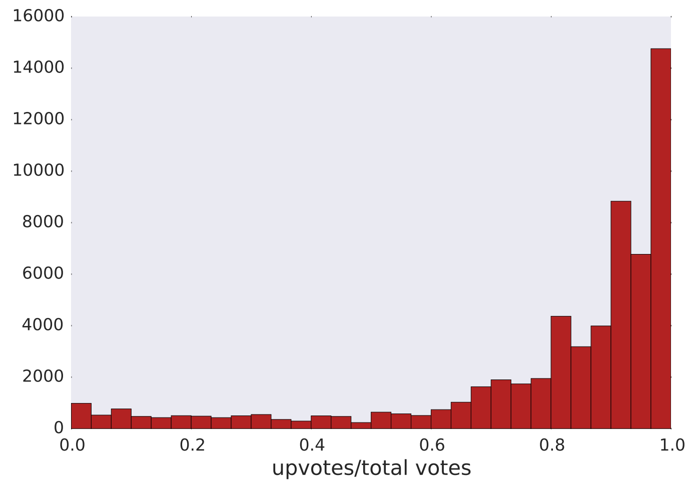

Fig. 1. Histogram of helpfulness- the outcome variable (the ratio of upvotes in total votes)

### C. Features

We used the following features extracted from each review text, classified into broad categories.

**Textual Features**

- Text Length
  The count of characters in the review text, including punctuation and spaces.
- Character Count
  The number of alphabetical characters in the review text
- Word Count
  The number of words in the review text.
- Unique Word Count
  The number of unique words in the review text.
- Sentence Count
  The number of sentences in the review text.
- Automated Readability Index [6]
  The automated readability index score, which is a measure of text readability, is computed as follows:

$$ARI = 4.71\frac{N_{character}}{N_{word}} + 0.5\frac{N_{word}}{N_{sentence}}$$

**Metadata Features**

We used the number of stars that a reviewer has given a product in his/her review as part of our feature set.

**Bag of Words**

We also used the bag of words model to generate additional features for each review. The bag of words model uses a large-dimensional, sparse vector to count word occurrences in a review text with respect to some vocabulary of words. More precisely, let $x^{(i)} \in \mathbb{R}^n$ be the sparse vector containing word occurrences in the $i$-th review, where $n$ is the size of our vocabulary. If review $i$ contains $c$ occurrences of the $j$-th word in our vocabulary, then $x_j^{(i)} = c$.

## IV. Regression Models

We modeled the problem as a regression. To evaluate the performance of our regression models, we used the coefficient of determination (denoted by $R^2$), which is a standard measure of goodness of fit for our model. Given that $x^{(i)}$ is the data point to be evaluated, $y^{(i)}$ is the actual value of the outcome variable, $\hat{y}^{(i)}$ is the predicted value of the outcome variable, and $\bar{y}$ is the mean of all values of the outcome variable in the data set, the coefficient of determination is calculated as follows:

$$R^2 = 1 - \frac{\sum_i (y^{(i)} - \hat{y}^{(i)})^2}{\sum_i (y^{(i)} - \bar{y})^2}$$

Lesser values of $R^2$ mean that the model doesn't fit the data (i.e. the model cannot explain the variance of the data). While it typically ranges from 0 to 1, for very low-performance cases $R^2$ can be negative.

Next, we explain the four models we studied: linear regression, ridge regression, support vector machines for regression and random forests.

### A. Linear Regression

Linear regression poses the problem of solving for the parameters $\theta$ that minimizes the following cost function:

$$J(\theta) = ||X\theta - y||_2^2$$

where, $X$ is the design matrix of size $m \times n$ created by having $m$ training sample as row vectors, and $y^T = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(m)} \end{bmatrix}$ is the vector containing the target variables for each sample. Also note that $||z||_2^2 = z^T z$ is the $L_2$-norm of a vector $z \in \mathbb{R}^n$.

Given $\theta$, we make a prediction given a new data point $x'$ as follows:

$$\hat{y} = \theta^T x'$$

## B. Ridge Regression

Similar to linear regression, ridge regression attempts to minimize the least squares cost function with the additional term $\alpha||\theta||_2^2$. In particular, we have:

$$J(\theta) = ||X\theta - y||_2^2 + \alpha||\theta||_2^2$$

The additional term gives preferences to a parameter vector $\theta$ with a smaller norm, which makes the regressor less susceptible to outliers in data. To understand this intuitively, note that if the parameters are unconstrained, each parameter can be arbitrarily large, and this happens when the model tries to fit outliers. By constraining the parameters, we discourage the model to try its best to accommodate outliers, hence reducing variance. The addition of this hyperparameter $\alpha$ (also called the penalty constant) allows us to control model complexity and achieve the optimal balance of bias and variance.

## C. Support Vector Machines for Regression

While typically used for classification problems, Support Vectors Machines (SVMs) can be used for regression as well. Like SVMs for classification, Support Vectors Machines for Regression (SVR) contain all the main features that characterize maximum margin algorithm.

We can see SVR's mathematical foundations and the intuition of it by illustrating on a linear model. Suppose we have a training set: $x \in \mathbb{R}^{m\times n}$ and $y \in \mathbb{R}^{m\times k}$ where $x_i'$s are the features and $y_i'$s are the observations. Then the linear regression model is given by

$$\hat{y} = f(x) = w^T x + b$$

where $w \in \mathbb{R}^{n\times k}$ is the parameter vector that we wish to estimate, and $b \in \mathbb{R}^k$ is the intercept term. To estimate $w$, we can use the "$\epsilon$-insensitive" measure, where the goal is to find a function $f(x)$ that has at most $\epsilon$ deviation from the actually obtained target $y^{(i)}$s for all the training data. Meanwhile, in order to keep the variance low, we want $f(x)$ as flat as possible. To achieve this, errors (small residuals) of size less than $\epsilon$ are ignored. We seek to minimize $\|w\|_2^2$ while using the "$\epsilon$-insensitive" measure. Formally, we can model this problem as a convex optimization problem:

$$\min \frac{1}{2}\|w\|_2^2$$

$$s.t. = \begin{cases} y^{(i)} - w^T x^{(i)} > -b \leq \epsilon \\ w^T x^{(i)} + b - y^{(i)} \leq \epsilon \end{cases}$$

This convex problem is only feasible if there actually is a function $f$ that approximates all pairs $(x_i, y_i)$ with $\epsilon$ precision. This may not be the case. To handle infeasible problems, we soften the loss function by introducing slack variables $\xi_i, \xi_i^*$. Then, the "$\epsilon$-insensitive" loss function $|\xi|_\epsilon$ is described by

$$|\xi|_\epsilon = \begin{cases} 0 & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise.} \end{cases}$$

Finally, we can write the SVR formulation as

$$\min \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

$$s.t. = \begin{cases} y^{(i)} - w^T x^{(i)} - b & \leq \epsilon + \xi_i \\ w^T x^{(i)} + b - y^{(i)} & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0. \end{cases}$$

$C > 0$ controls the tradeoff between the model complexity and the amount up to which deviations greater than $\epsilon$ are tolerated. $\epsilon$ and $C$ are two fundamental parameters which in different ways determine the model flatness, the training errors, the number of support vectors and the bias-variance tradeoff of SVR.

The use of kernels, which we shall not discuss in this paper, is a powerful method used to transform input data into a higher dimensional space, and can be used with SVR as well.

## D. Random Forests

Regression with random forests is a powerful nonparametric technique [7]. It is classified as an ensemble method since it aggregates many separately grown trees and generates a single unified model.

Regression trees are used as building blocks. A top-down, greedy approach is used to partition the feature space into a specified number of distinct non-overlapping regions. The number of partitioned regions affects the bias-variance tradeoff.

Trees typically suffer from high variance. *Bagging* is used to fight against high variance. For bagging, many deep trees are grown from bootstrapped samples. Each tree has high variance, but low bias. Averaging all these trees reduces the variance. The number of trees is not a critical parameter. Using a very large number will not cause overfitting. A sufficiently large number can be chosen after the error levels off.

Random forests are also a bagging method but with a difference: each grown tree only uses a random sample of p predictors out of all n predictors. The motivation of this is to decorrelate trees. When all predictors are used, all grown trees look like each other. Therefore, bagged trees are very correlated, which negatively affects the reduction in variance by averaging. Random forests solves this problem by considering only $(n-p)/n$ portion of predictors at each tree split. For $p = n$, random forests turn into simple bagging.

The parameters to be tuned for random forests are (1) the number of random predictors p, (2) the maximum number of leaf nodes (which determines how deep a tree is grown) and (3) the number of trees to be grown and averaged.

To estimate the test error of a random forest, cross-validation is not necessary. Out-of-bag (OOB) error is used. It can be shown that on average a tree uses two-thirds of the data. The remaining one-third is out-of-bag. For a data point, we can therefore predict the response of it using the trees in which that data point was OOB. It is a valid test error estimate since trees that were not fit using that point are used.

## V. Results and Discussion

The scores obtained by all considered methods are shown in Table I.

## A. Linear Regression

We first experimented with the simplest model: linear regression. Since linear regression has the intrinsic assumption that the considered data is linear, it did not perform well. However, due to the high number of features, the performance was still

| Regressor | $R^2_{train}$ | $R^2_{test}$ | $R^2_{CV}$ |
|---|---|---|---|
| Linear | 0.34 | 0.26 | 0.19 |
| Ridge ($\alpha = 400$) | 0.33 | 0.28 | 0.25 |
| SVR (kernel=`linear`) | 0.26 | 0.23 | 0.07 |
| SVR (kernel=`rbf`) | 0.70 | 0.21 | 0.17 |
| Random Forest | 0.65 | 0.35 | 0.34 |

Table I. Various regression models' $R^2$ scores for the training set, the test set, and cross validation(or OOB).

relatively decent. It is a very rigid method, therefore, it suffered most from high-bias.

## B. Ridge Regression

We attempted to reduce variance for our linear regression model by using ridge regression. By retraining the model for various values of the penalty constant, we determined that the optimal value for the penalty constant is $\alpha = 400$, which gave the highest score in the test set and cross validation. The learning curve for ridge regression is shown in Figure 2. As $\alpha$ increased, the penalty on the model increased and the flexibility decreased. It resulted in reduced variance. This can be seen on $R^2_{train}$ which decreased while $R^2_{test}$ and $R^2_{CV}$ increased. $\alpha$ forced insignificant features towards zero in regression. Further increasing $\alpha$ would have penalized too much and decreased $R^2_{test}$ and $R^2_{CV}$ scores.

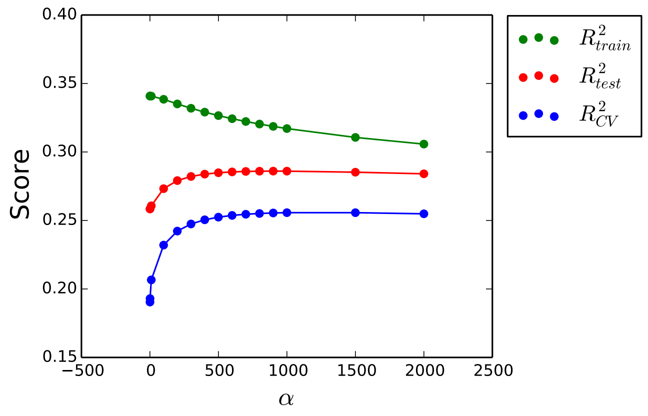

Fig. 2. Learning curve for ridge regression

Ridge regularization resulted in a relatively large improvement in the performance. At the expense of slightly higher bias, we were able to improve the $R^2_{CV}$ by 0.06 to 0.25.

## C. Support Vector Regressors

SVR has 3 main factors to tune: kernel, $\epsilon$ and $C$. Our strategy was to choose two kernels and optimize $\epsilon$ and $C$ for each one.

We first experimented with a linear kernel, and we found that SVR did not perform well. We performed an exhaustive grid search using cross validation for the hyperparameters $\epsilon$ and $C$, and the results are shown in Figure 3. Observing the results, we saw while both $\epsilon$ and $C$ affected the $R^2$ scores, $\epsilon$ was the dominant hyperparameter.

In the end, even when we chose the hyperparameters that gave us the best $R^2$ scores ($C = 163.84, \alpha = 0.06$), linear SVR still scored a relatively low $R^2$ on the training set, the test set and during cross-validation. In particular, $R^2_{CV}$ had a

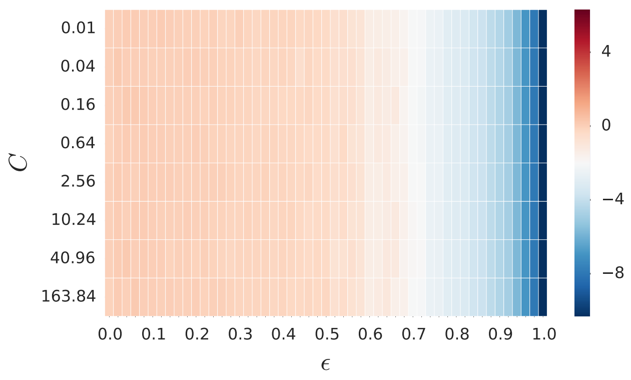

Fig. 3. Grid search cross validation for linear SVR hyperparameters. Red colors show high cross validation $R^2$ scores (up to 1), while blue colors show lower cross validation scores.

very low mean and a high standard deviation. It indicates that it suffers from both high bias and high variance.

Secondly, we considered the rbf (radial basis function) kernel. Compared to the linear kernel, it performed better on $R^2_{train}$ and $R^2_{CV}$. However, its $R^2_{test}$ was not improved. The big gap between $R^2_{train}$ and $R^2_{test}$ shows that there is a serious problem of high variance. Since rbf is nonlinear, it is more prone to overfitting. Again, cross-validation was used to pick $\epsilon$ and $C$ (see Figure 4). The pair $\epsilon = 0.1$ and $C = 10$ scored the highest.

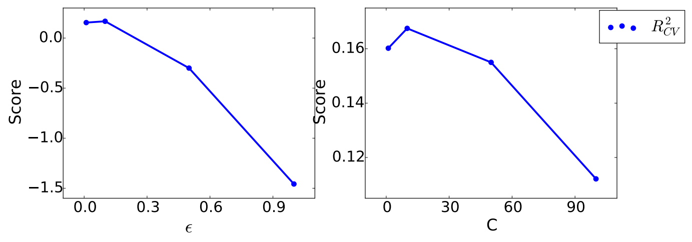

Fig. 4. Cross-validation scores of SVR with a rbf kernel. Each plot is shown for a varying hyperparameter while the other is kept constant at its optimum value.

Compared to the SVR with a linear kernel, SVR with a rbf kernel performed a lot slower. Due to its lengthy development time, we only tested for $\epsilon$ and $C$ around a limited range.

## D. Random Forests

Unlike the previous methods, random forests are nonparametric, which means that we did not need to make assumptions about the form of the data when using them.

We first chose a large enough number of trees to grow (150). How the scores settled down with the increasing number of trees can be seen in Figure 5.

We then optimized for the maximum number of leaf nodes and the number of features ($p$) considered when looking for a

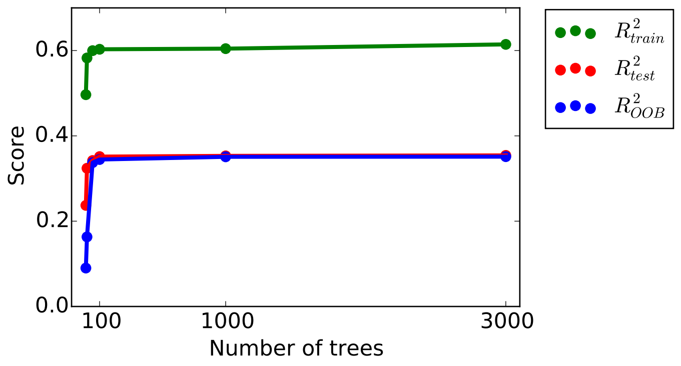

Fig. 5. The scores of random forests given for different the number of grown trees. After a sufficiently large number of many trees, scores level off ax expected.

split in a tree again using OOB scoring (see Figure 6). $R^2_{OOB}$ increased with the number of leaves but then decreased due to overfitting. Same happened for increasing $p$. As $p$ approached $n$, grown trees became more correlated with each other so there was less reduction in variance. We chose the maximum number of leaf nodes to be 800 and $p = 700$. As pointed out in the literature [8], for regression purposes $p$ is best chosen around $n/3$. Our results also confirm this.

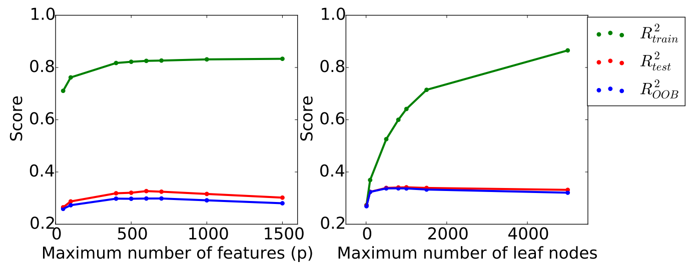

Fig. 6. Learning curves of random forests.

Random forests scored much higher than other methods. We attribute this to their nonparametric character. Data is unlikely to have an easily parameterized structure, therefore, previous parametric techniques performed worse due their wrong assumptions about the data. The big difference between $R^2_{train}$ and $R^2_{test}$ signals overfitting.

Finally, using this most successful model, we ran a significance test on the model to see the most important features on the outcome using the Gini importance criteria. As seen in Figure 7, textual features dominate the bag words features. The words relevant to the subject are at the top among bag of words. Star rating given to the product by the reviewer is the most significant feature. Investigating data, we see that most helpful reviews are the ones which rated the products highest. Here we emphasize that textual features based on text length are highly correlated with each other. Therefore, it is hard to argue which ones are more significant than others. However, their combined significance is clear.

## VI. Conclusion and Future Work

We have successfully shown that there is a possibility to automatically rank reviews. We modeled the problem as regression and extracted features that represented the data. We applied four

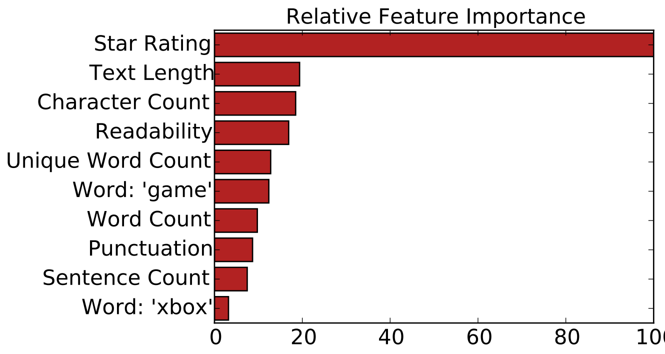

Fig. 7. Top ten most significant features according to the random forest regressor

different regression methods and compared their properties. We observed that random forests performed the best.

Due to the complexity of the problem at hand, we believe there is a lot of room for improvement in the future work by addressing three main challenges: creating a more complex feature set, reducing high dimensionality and overcoming skewness.

The textual features that we used were very basic ones that was only concerning the text length properties and readability. More advanced natural language features such as sentiment analysis can be added. Moreover, bag-of-words can be improved by using term frequency - inverse document frequency (tf-idf), or the word2vec [9] library developed by Google that implements the continuous bag-of-words or the continuous n-gram models. Another feature can be the background of a reviewer. A reviewer with previous helpful reviews may have a higher probability of making more helpful reviews in the future.

The feature space is relatively high dimensional which makes it necessary to have a large data set in order to get decent results. We are also aware that some of the features are highly inter-correlated. To address these problems, feature selection methods or principal component analysis shall be applied.

Lastly, skewness has a big negative impact on the performance of models. Especially, support vector machines are known to be susceptible to skewness of data, as studies have indicated [10]. In order to deal with this problem, simple techniques such as upsampling, downsampling, or stratified sampling of the data can be employed, while the study mentioned also provided another entire methodology to mitigate skewness.

## References

[1] J. Rodak, M. Xiao, and S. Longoria, "Predicting helpfulness ratings of amazon product reviews," http://cs229.stanford.edu/proj2014/Jordan%20Rodak,%20Minna%20Xiao,%20Steven%20Longoria,%20Predicting%20Helpfulness%20Ratings%20of%20Amazon%20Product%20Reviews.pdf, 2014, Accessed: 2015-11-08.

[2] S. Bolter, "Predicting product review helpfulness using machine learning and specialized classification models," 2013. [Online]. Available: http://scholarworks.sjsu.edu/etd_projects/348

[3] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti, "Automatically assessing review helpfulness," in *Proceedings of the 2006 Conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 423–430.

[4] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, 2015, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2783258.2783381

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel." Tech. Rep., Feb. 1975. [Online]. Available: http://www.eric.ed.gov/ERICWebPortal/detail?accno=ED108134

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A%3A1010933404324

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[9] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[10] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *In ICML 2003 Workshop on Learning from Imbalanced Data Sets*, 2003.