# Predicting the Helpfulness of Customer Reviews

Alex Hankin, Charlie Perkins, Sam Schick

**Abstract**

Online shopping is becoming the standard way to buy a product. As this shift occurs, it becomes more difficult to examine a product personally, and we rely more on reviews submitted by other customers. However, these reviews can be very mixed in quality. Amazon crowdsources the determination of which reviews are helpful by allowing customers to mark which reviews they found most useful, but makes no attempt to determine whether a newly submitted review is of reasonable quality. We present a system that uses the random forest algorithm to examine the text of a review and predict how helpful it will be, based on data collected from Amazon's crowdsourced data.

## 1 Introduction

Many online reviews will never be marked helpful. However, when first published these reviews are put on equal footing with reviews that could nominally be very well written. A system that could sort these by how helpful they are likely to be found could improve customer outcomes by making it easier for them to find a wide variety of helpful reviews. For a prototype system, we gathered data from Amazon reviews on popular books, which have many reviews and also tend to have a somewhat standard format. We implement a random forest classifier that examines normalized word frequencies and attempts to predict whether the book will be in the top 90% in terms of helpfulness. In order to create a stronger training set, rather than randomly selected reviews we select the 50 most helpful reviews and the 50 least helpful reviews from each book we examined, and examined a set of books.

## 2 Data Sources

## 3 Text Processing

## 4 Feature Extraction and Regression

We used the bag of words methodology to extract features. We represented each document as a vector, where each dimension was a word that occurred in the document and had value equal to the number of times the word occurred in that document, normalized with TFIDF (Term Frequency Inverse Document Frequency). This gave us a matrix where each row was a word and each column was a review.

We then used the random forest algorithm as implemented by SKLearn to create a classifier. Random forest generates decision trees based off of a random subset of features (in this case, words). Each decision tree produces a regression, and these predictions are combined to reduce variance in the output. each review was considered "bad" if it was in the bottom 10% in terms of helpfulness for the book it had come from, and "good" otherwise. The threshold was arbitrary, but this allowed us to improve the quality of the model by reducing it to a classification problem. This remains useful as a potential tool, as comments marked by the automated filter to be unhelpful could be listed last, improving user experience.

## 5 Performance Evaluation

We used stratified K fold cross validation to examine the quality of the model to avoid bias from the training set, separating the data out into K subsets each of which had approximately a 50-50 split of good and bad reviews. We then built K versions of the model. Each model was built on K-1 of the folds as a training set and the remaining fold as the tuning set. We then generated ROC curves and measured mean squared error for each fold.

## 6 Analysis and Conclusions