Predicting product ratings using reviews

Information Retrieval

Xiaoying Chen (s2714140), Lennart Faber(s2500523)

11-01-2017

Abstract

In this report, we will discuss an experiment done on Amazon product reviews. The goal will be to predict the score that a user will give to a product using the review the customer has written.

1 Introduction

Online shopping has become more popular than ever and the growth doesnt seem to stop very soon. There is nothing one cannot buy on the internet. This huge market brings us interesting challenges. The incredible amounts of data need to be interpreted to better understand the needs of customers and allow further expansion of online retail.

A key feature of online shopping is the possibility for customers to give reviews on products they purchased. These reviews can then be used by other possible customers to decide which product to buy and which to avoid. Retailers on the other hand can use the reviews to monitor the quality of their products and adjust their stock to the needs of the customers.

For huge websites like Amazon.com, the amount of reviews that are given is far too big to be processed by humans. Using basic rating-systems on scales of for example 1 to 5 makes it easier to find out what buyers think of their product. By looking at the reviews, we could try to predict what rating the writer gave to the product he or she wrote about. This is exactly what we are going to try in this research. The success of an algorithm can be graded by looking at the average accuracy of the system when guessing user ratings.

The database we are going to use comes from Amazon. This huge website provides us with enough data to gain realistic results. Using all data (142.8 million reviews) however would result in extremely long processing times, so we will use a subgroup of this data. We are going to look at the reviews and ratings given in the category Musical Instruments. This category contains 10,261 reviews, which we should be able to process in an acceptable timespan.

We are going to look at different classifier techniques to find the best method for predicting scores. Then we will look at the best method and determine whether this system performed good enough. Our research question will be: Will one of the classifier techniques discussed in the lectures of Information Retrieval be able to predict ratings of Amazon products using their reviews with an accuracy of at least 0.6?

Our hypotheses is that we will be able to reach an accuracy score of at least 0.6. After cleaning the reviews of information we do not need and finding words that are typical for expressing positive or negative thoughts, predicting scores should not be a real problem.

2 Literature review

Classifying reviews has been done before. Turney (2002) developed an algorithm that classified reviews in the categories positive or negative. He managed to achieve an average accuracy of 74% when categorizing 410 reviews done on Epinions. The classification was done by looking at the semantic orientation of the phrases in the reviews. The adjectives and adverbs were extracted and averaged and then used to decide. An interesting fact is that there was a difference in success rate between different kinds of reviews. Classifying movie reviews resulted in an accuracy of 66%, while the accuracy for automobiles and banks was about 80% to 84%.

Dave et al. (2003) found multiple problems while classifying reviews from all over the web. The first problem they encountered was the inconsistency of ratings. While users might give similar qualitative descriptions on a product, the quantitative score they give can be quite different. Some users do not understand the rating system, giving a 1 instead of a 5, which makes the data noisier.

Another problem they found when dealing with customer reviews is the fact that people use comparison and write ambivalent reviews. Users might write negatively about a comparable product to express their positive thoughts about the actual product they are reviewing. Also, people could sum up the negative aspects of a product but conclude with the statement that in general they are satisfied. In these situations, it is hard to separate out the core assessment of the review and therefore harder to classify it.

Dave et al. (2003) also concluded that reviews that are very short should be filtered out to improve classification performance. Reviews on for example Amazon are easier to classify because they contain more words on average.

The last problem they found was the fact that the reviews are often not evenly distributed. They found that positive reviews were predominant and some types of products were reviewed more often than others. Negative reviews often contained more words than positive ones, but because they contained a bigger variation of words it was difficult to achieve good recall on that set.

3 Methodology

The experiment can be divided into two parts: feature selection and classification. First, we read the dataset and select features to make classification more efficient by removing unimportant words. The approaches we used for feature selection were tokenization and part-of-speech tagging. The resulting data was splitted into two sets, one for training and one for testing. To classify the data, we used two classifiers that were discussed in the lectures of Information Retrieval: Naïve Bayes and the decision tree classifier.

3.1 Feature selection

3.1.1 Tokenization

By tokenizing the raw data, we were able to remove punctuation and other characters and extract the important words from sentences. We used Regex tokenizer to split the sentence into words. As the name suggests, Regex tokenizer uses regular expressions to achieve this. For this research, we only kept sequences of alphabetic characters (words) from the sentences for further processing.

3.1.2 Part of speech tagging

Not every word in a review is useful for classification. In order to decide whether a review is mainly positive or negative, the adjectives and adverbs were most important. When words like "nice", "good", "satisfied", and "perfect" appear in a review, we could consider it as a positive review. For this experiment, we used the NLTK POS-tagger to label the lexical categories. We extracted the adjectives (e.g. "good", "great"), adverbs (e.g. "nicely", "smoothly"), and verbs in past tense (e.g."satisfied", "disappointed"). Some words were not correctly tagged by the POS-tagger: words like "perfect", "nice", and "love" were sometimes categorized as nouns. This is why we decided to also keep words with the tag "NN". After POS-tagging we were left with the actual words we were going to use for classification.

3.2 Classification

The distribution of the ratings that were given by the reviewers was imbalanced. In the dataset of the category "Musical Instruments", there were only 217 reviews with the rating 1. 250 reviewers gave rating 2 to the products, and for rating 3, 4, and 5 there were 772, 2084, and 6938 reviews respectively. There were a lot more positive than negative reviews. As said before, the data was splitted into train set and a test set. For this research, we used 90% of the features for the train set, and the remaining 10% for testing. When we randomized and split the data, there were a lot less reviews with a rating of 1, 2 or 3 in the testset than reviews with ratings of 4 and 5. The total amount of words that the classifier collected for the training and test sets in the categories with ratings of 1, 2, and 3 was relatively small. After doing some tests, we found that categorizing reviews in five groups wasnt very effective. Because of this, we decided to classify the reviews in the categories ”positive” and ”negative” instead of a category for each score. We considered 4 and 5 as positive, and the rest negative.

3.2.1 Naïve Bayes classifier

The principle of Naïve Bayes is to calculate the product of the probabilities of each word that occurs in the category by examining the frequency of each word in the training set (Bird et al., 2009). It is a simple and popular classifier, but sometimes less accurate. When a word only appears in the test set but not in the training set, the probability of that word will be 0. Since our collection of negative words was relatively small, this could have affected the performance of the classification system. We used the Nltk Naïve Bayes classifier to classify the training features. After the training of the dataset, we calculated the accuracy of the classification using the Python module ”nltk.classify.accuracy”.

3.2.2 Decision tree classifier

The decision tree model classifies the input in a tree form flowchart with a root node, decision nodes and leaf nodes. For each feature, it starts at the root node, loops through the branches of decision nodes until it gets to the leaf node which represents the right label of the input features (Bird et al., 2009). This is also a simple approach of classification. But since the classifier has to get from the top of the tree to the bottom for each word, this approach is very time consuming when dealing with larger datasets. We trained the training set with NLTK decision classifier, and then calculated the accuracy in the same way as we did before while classifying with Naïve Bayes.

3.3 Implementation

For this research, we performed different experiments in order to find the most effective method to predict the ratings on the Amazon product reviews.

Experiment 1: We read the Amazon review dataset, tokenized each review into tokens with Regex Tokenizer, and stored the rating alongside every review when creating the token lists. Then we extracted the features and split the remaining words into a train set with 90% of the words and a test set with the remaining words. After that we classified the reviews into five groups, one for each of the possible scores (1 to 5), with the Naïve Bayes as well as the Decision tree classifier. The results of the classification were evaluated by looking at the average accuracy scores.

Experiment 2: We tokenized the data, used POS-tagging to extract the important words, splitted the dataset, and classified it the same way as in experiment 1, but in this case we classified the reviews into the categories positive (ratings 4 and 5) and negative (rating 1 to 3).

Experiment 3: In this experiment we will do the same as in experiment 1, but with one added step in which we preprocess the data. After tokenizing we used the NLTK POS-tagger to tag the words, and extract only informative features. Finally, we split the resulting features in a train and test set as done before.

Experiment 4: This experiment is the same experiment as experiment 2, but this time we used the POS-tagged features for classification.

4 Results and Discussion

4.1 Results

Accuracy	All words	Extracted features
Naive Bayes	0,50	0,28
Decision tree	0,69	0,69

Table 1: Results experiment 1 and 3 (classified reviews in 5 groups)

Accuracy	All words	Extracted features
Naive Bayes	0,34	0,55
Decision tree	0,88	0,88

Table 2: Results experiment 2 and 4 (classified reviews in 2 groups)

4.2 Discussion

The results showed that whether or not we used part of speech tagging did not have impact when we classified the reviews with decision tree model, while it did affect the accuracy of classification with Naïve Bayes. When we categorized the reviews into the ratings 1 to 5, the accuracy of training with POS-tagged features was lower than training with all words of the reviews. When the reviews were divided in only two categories, training with extracted features resulted in higher accuracy than training with all words. When we removed unimportant words from the reviews, the word variety in categories with a small amount of reviews became even smaller. Words which do not appear in the training set cannot be classified, so the performance became worse. When we replaced the categories for rating 1 to 3 with just a negative category, the collection of words was bigger than when we separated the reviews with rating 1 to 3 into separate categories. Because the problem of words not being present in the training data now occurred less often, a greater accuracy could be achieved. This could be the reason of the unstable accuracy with Naïve Bayes classifier.

The performance of Decision tree classifier was better than the performance of the Naïve Bayes classifier. The accuracy was lower when categorizing reviews in five groups than when using just two groups. Because many reviews of rating 4 and 5 contained the same words like ”good”, ”nice”, and ”great”, classification was harder. When we concatenated these two categories into one, the classification became easier and accuracy increased.

5 Conclusion

We can conclude that our hypotheses was correct. We were indeed able to create a system that can predict rating scores on Amazon reviews using the text of a review, with an average accuracy of at least 0.6. During our research we found that predicting ratings can however be quite hard, because of uneven distribution of words amongst the categories. This is what Dave et al. (2003)also found in their research. Concatenating the five categories in just two groups resulted in even better results. When we look at the goal of automatic analysis of product reviews, this might not even be a big problem. Customers often just want to know if they should buy a product or not and classifying reviews in just two categories but with higher accuracy might be more valuable to them than a less reliable categorization in five groups.

References

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python.* ” O’Reilly Media, Inc.”, 2009.

Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.