Product Review Rater for Video Games

Fangrui Guo, Ruobin Hu, Bowei Sun

December 2019

1. Project background

For internet companies who try to sell products online, the consumer review is a huge decision maker for potential buyers. Over the past two decades, electronic commerce has grown into a thriving industry. Under the shift towards the online marketplace, customers can’t physically assess items in consideration and therefore, they must rely on other resources to help inform their purchase decisions. For instance, product review is one of the most important resource that will directly impact the sales of online products. Both Internet retailer and consumers would like to effectively identify the reviews that provide the most insight. However, given the large amount and variation in the quality of product review, it is difficult to identify useful feedback.

In order to determine the quality of a product review, online retailer Amazon.com gives consumers the option to vote on the helpfulness of a review. And based on the helpfulness rating of product reviews, Amazon ranks and prioritizes the presentation of the most helpful reviews.

Although this system may work well for popular products, a large amount of Amazon products reviews is left with limited or no helpfulness feedback. In addition, when a particularly insightful review is shared, it takes time for people to read and up-vote the review. It’s also time consuming and frustrating for consumers to found out insightful information from a large amount of superfluous comments. Without an automated method for assessing the quality of reviews, useful information would remain hidden in the millions of Amazon product listings.

The goal of the project is to train a model to tell the helpfulness of any given product review. With the Amazon dataset of video game reviews, we want to train a model to predict how helpful a review is, thus makes it possible to improve the customer's experience by sorting the reviews according to the helpfulness.

This project can be found on GitHub at GitHub Repository, and the dataset for reviews can be found at Amazon Review Data (2018). Currently the project focus on the category of video games because nowadays video games are mainly sold online, and therefore the weight of other consumers' reviews is considerably greater than other products. However, in this dataset, any kind of categories should work with the program since they are of the same format.

2. Data Cleaning

The dataset is an updated version of Amazon review dataset released in 2014. The dataset includes reviews with several properties such as rating, text, helpfulness votes, etc. There are more than 233 million reviews in the datasets including 29 categories, but we only care about the video games category in this project for simplicity. We first requested the data from the link and got the data in .json format. Each line in the file corresponds to one single review data, and it looks like:

```
{
 "reviewerID": "A2SUAM1J3GNN3B",
 "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "vote": 5,
  "style": {
    "Format:": "Hardcover"
  }
  "reviewText": "I bought this for my husband who plays the piano.  He is having a wonderful time
playing these old hymns.  The music is at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

Since our goal is to predict the helpfulness vote from the reviewText, we only need the properties ‘vote’ and ‘reviewText’. And we just ignored all other features from the datasets. One would find that not all data would contain both properties, so we also abandon the data that missed either properties. Then we polished the data by doing operation as described in the following text:

For the ‘reviewText’ property, we simply tokenized the string using ‘nltk.word_tokenize()’, and it would provided us with an output as list of strings

As for the ‘vote’ property, the value can span from 0 to ~500, the vote distribution is shown as Figure A.

We can find most of the data is confined in the range (0,100). So we tried to build our own parameter ‘vote level’ based on this distribution with an additional function called ‘helpfulness()’. So, the ‘vote level’ has 5 value ranging from 1 to 5 (as string):

Level 1 means the ‘vote number’ is among the first 20% of the distribution (vote number from 0-2),

Level 2 means 20%-40% (vote number from 3-4),

Level 3 means 40%-60% (vote number from 5-6), Level 4 means 60%-80% (vote number from 7-10)

Level 5 means last 20% (vote number >10)

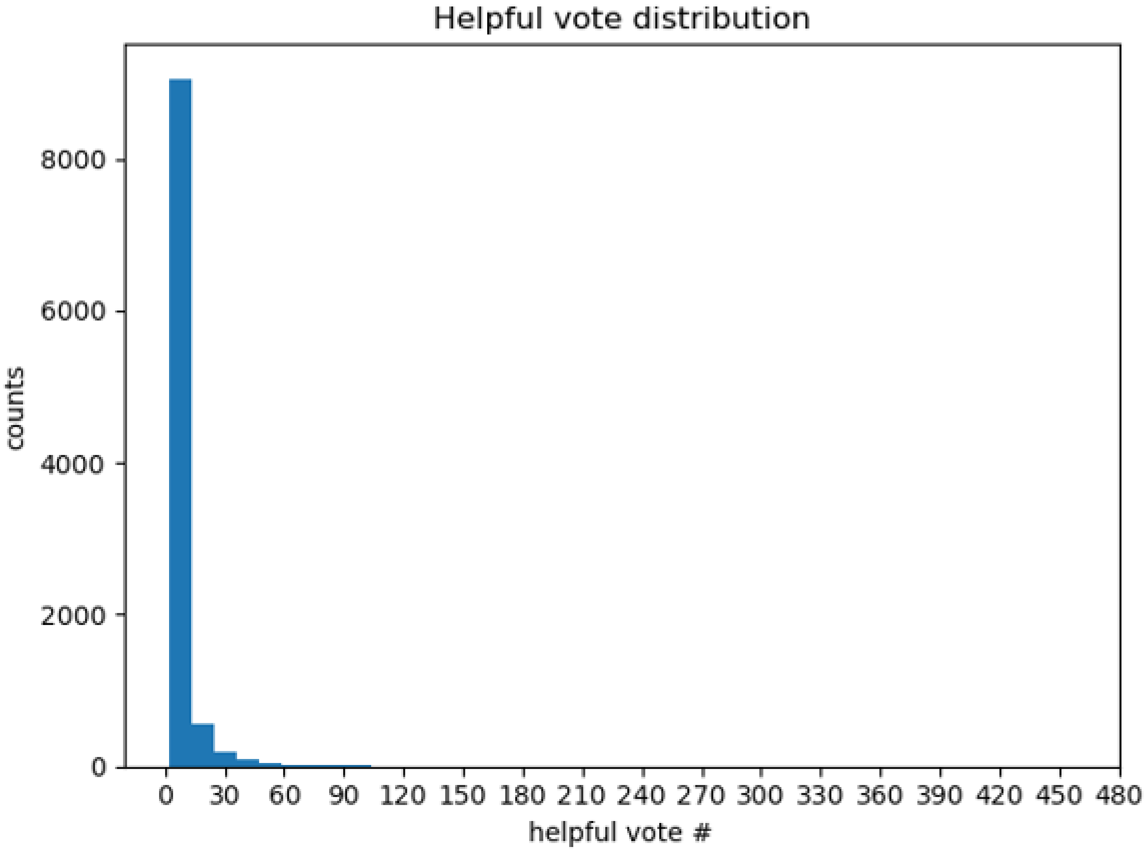


Fig A. Helpful vote distribution.

The final data generated is like, for example:

```
(['Whoa', '!', 'This', 'game', 'is', 'awesome', 'a', 'lot', 'of', '90s', 'dreaded', 'pop', 'culture', 'managed', 'to', 'crawl',
'its', 'way', 'into', 'this', 'game', 'but', 'that', "'s", 'a', 'good', 'thing', 'in', 'a', 'cheesy', 'way', '.', 'Anyways', 'this',
'game', 'is', 'full', 'of', 'entertaining', 'levels', ',', 'enemies', ',', 'fun', 'gameplay', ',', 'and', 'lots', 'of', 'awesome',
'funky', 'music', '!', 'This', 'game', 'is', 'much', 'different', 'from', 'the', 'first', 'and', 'for', 'those', 'who', 'could', "n't",
'understand', 'the', 'first', 'one', 'might', 'like', 'this', 'one', 'better', '.', 'The', 'gameplay', 'is', 'a', 'little', 'bit', 'more',
'straightforward', 'this', 'time', 'around', '.', 'This', 'time', 'the', 'game', 'takes', 'place', 'on', 'their', 'home', 'planet',
'of', 'funkatron', 'instead', 'of', 'Earth', ',', 'and', 'guess', 'who', 'invaded', 'your', 'turf', ',', 'Earthlings', '!', 'So', 'you',
'got', 'fend', 'off', 'the', 'huge', 'Earthling', 'invasion', 'with', 'exploding', 'jars', ',', 'Super', 'Jars', ',', 'Funk', 'Vacs',
'(', 'those', 'are', 'handy', 'sometimes', '!', ')', 'and', 'other', 'funky', 'alien', 'powers', '.', 'Also', 'like', 'in', 'the',
'original', 'Toejam', 'and', 'Earl', 'there', 'is', 'an', 'element', 'of', 'search', 'and', 'collect', '.', 'A', 'little', 'side',
'mission', 'you', 'can', 'collect', 'some', 'of', 'the', 'creature', 'called', 'the', 'funkapatmus', '(', 'or', 'Lamont', 'is', 'his',
'nickname', ')', 'and', 'if', 'you', 'can', 'get', 'him', 'to', 'come', 'out', 'at', 'the', 'end', 'of', 'the', 'game', 'its', 'an',
'awesome', 'ending', '!', 'Overall', 'really', 'good', 'game', ',', 'and', 'even', 'though', 'its', 'different', 'from', 'the',
'1st', 'one', 'in', 'terms', 'of', 'gameplay', 'its', 'very', 'similar', 'as', 'it', 'is', 'full', 'of', 'cool', 'characters', ',', 'power',
'ups', ',', 'good', 'levels', ',', 'graphics', ',', 'and', 'replayability', '.'], '3')
```

3. Model training

The Helpful rank prediction Model is trained based on three different predict model: the Bayesian models, the tree classifier and the cross-validation. And the text corpus is dealt with three distinctive ways: the raw text, the content text, and the negation text. The content text is based on the raw text but both punctuations and non-content words are excluded. The negation text is based on the raw text but every word with negation suffix contributes a 'not' to the token counts. So there are 9 different helpful rank prediction models in total.

In order to figure out which model is best for this situation without waste too much time and space, the 9 models are first trained by a limited data set. A random data set with 1000 different reviews is fed to the training. 800 of them are randomly split into the training set while 200 of them are randomly split into the text set. The 9 helpful rank models are trained based on the 800 reviews and the accuracy score is detected by the 200 reviews. Besides the accuracy score, a close score which counts the percentage of the prediction that with error different less than one to the ground truth also be recorded (Figure B, C.)

Then the 9 models are also tested by a new fresh data set of 1000 randomly picked up reviews. A new accuracy score and close score is reported. (Figure D, E.)

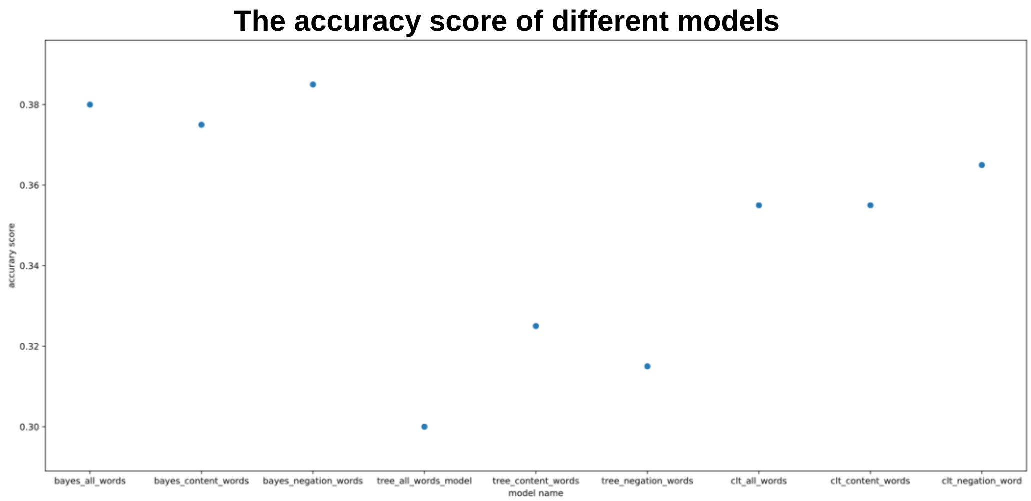


Fig B. The accuracy score of 9 models. Among these models, Bayes models show the best accuracy scores.

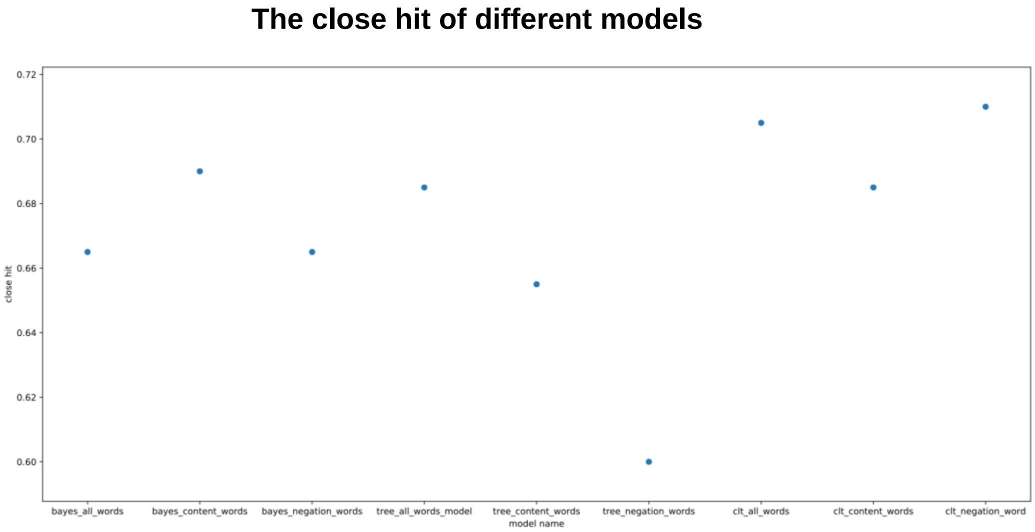


Fig C. The close score of 9 models. Among these models, Bayes-content words model, tree-all words model, and cross-validation all words as well as negation words show good scores.

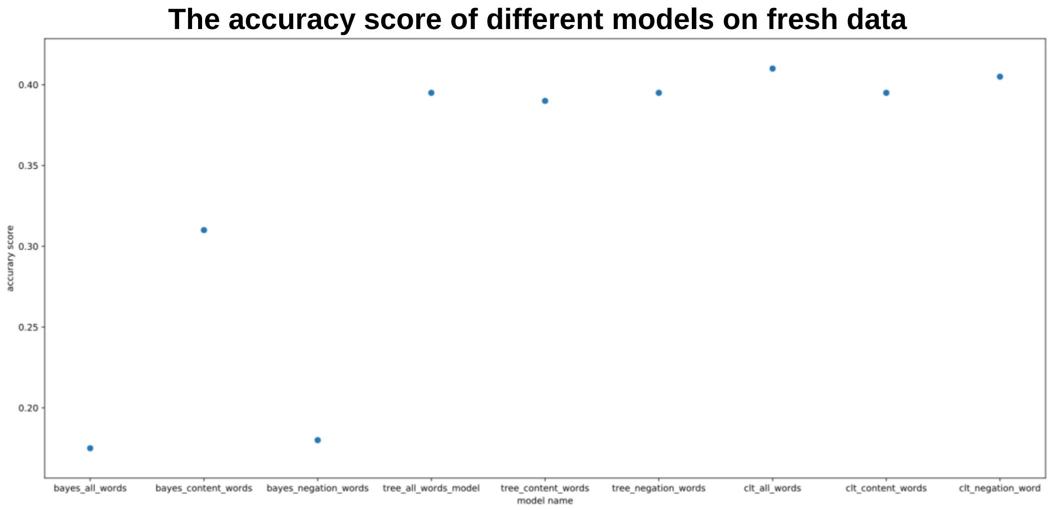

The accuracy score of different models on fresh data

Fig D. The accuracy scores on fresh data. The Bayes models show unreliable accuracy. On the contrary, tree models and cross-validation models show good scores on the prediction.

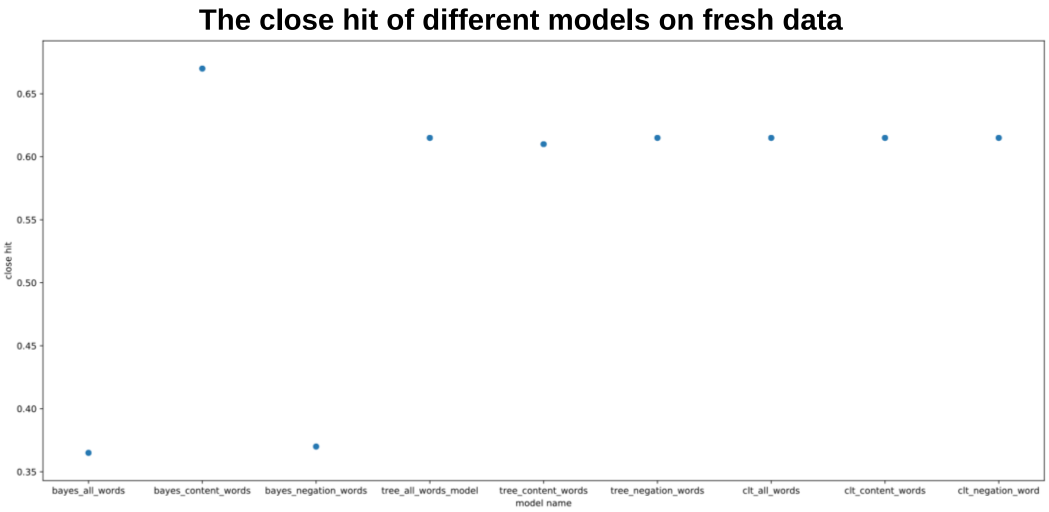

The close hit of different models on fresh data

Fig E. The close hit scores on fresh data. The Bayes models show unreliable accuracy except for the content words one. On the contrary, tree models and cross-validation models show good scores on the prediction.

Since we have 5 ranks of different helpful rank in total. The probability of having a correct random guess is 0.2 and the probability of having a random guess with an error less than 1 is 0.6. Both tree classifiers models and cross-validation models show reliable scores on the accurate predict which is much better than the random guess. However, the close hit guess both makes no sense compared with randomly guess.

Besides the accuracy score, the time consumption and space consumption of constructing the models are also taking into consideration. (Table A.) The tree_all_words model is dominated in the accuracy score on the prediction of fresh reviews, the time consumption as well as the space consumption. So that this model is selected as the prototype of the final model. The final model is fed with 3000 reviews and it comes out with an accuracy score of 0.395 as well as a close hit of 0.645.

Model name	**accuracy score on same set**	**close hit on same set**	**accuracy score on fresh set**	**close hit on fresh set**	**Time consume**	**space consume**
bayes_all_words	0.38	0.665	0.175	0.365	about 40s	18.9MB
bayes_content_words	0.375	0.69	0.31	0.67	about 40s	18.9MB
bayes_negation_words	0.385	0.665	0.18	0.37	about 40s	18.9MB
tree_all_words	0.3	0.685	0.395	0.615	about 40s	53KB
tree_content_words	0.325	0.655	0.39	0.61	about 40s	53KB
tree_negation_words	0.315	0.6	0.395	0.615	about 40s	53KB
cross_validation_all_words	0.355	0.705	0.41	0.615	about 5 min	1.45GB
cross_validation_content_words	0.355	0.685	0.395	0.615	about 5 min	1.45GB
cross_validation_negation_words	0.365	0.71	0.405	0.615	about 5 min	1.45GB

Table A. The summarize of the score, close hit, time consumption and space consumption of different models.

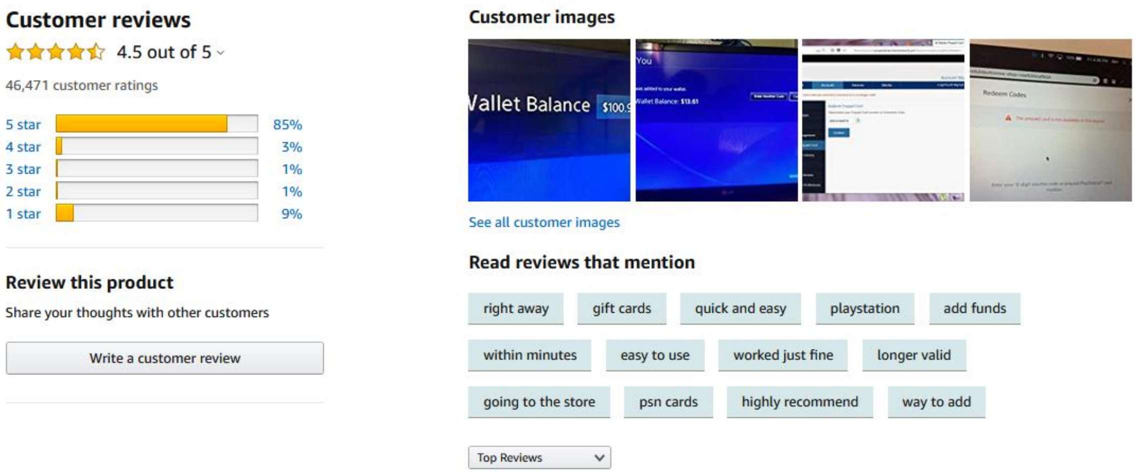
4. Conclusion and future steps

This project is currently able to provide a model to predict the helpfulness of the video games review. A five level rating system is used to separate different reviews, and as the result shows, the most accurate model in the same set is not below 0.3, and the most accurate model in the close set is not below 0.6, which is twice as better as blind guess, but clearly less accurate than the model trained when only two level system is considered (as assignment 5 shows, which can be as accurate as 0.8). With the expanding of the number of levels, models with single strategy becomes less suitable, a more complex strategy is required to ensure the accuracy of the review rater.

However, there is a more brute way to achieve this goal, which is counting the number of words and sentence, then considering the readability and relativity of the review to the product category. These macro-level statics are not used in this project because the readability and relativity are not easy to judge, which may consider duplicated trash talk of high helpfulness, leading to loss of accuracy.

As for the dataset, although it is about video games, the model training process and code can be used to deal with other categories in the dataset. However, for different product, the best model may not be the same, which means the tree_all_words model that performs best in video games reviews is not necessarily the best model for other product reviews. It is an interesting topic to figure out which model performs better for certain category of product reviews, which may be our future works.

Besides, the helpfulness may differ to different customers, and some key words may have opposite influence on different customer groups. One solution to this situation is to extract key words from the review, and display them at the beginning of all the recommended reviews (as Fig F shows). It is a much more complicate process since the preference of the current customer will be taken into consideration, the five level of rating is far from enough in such a project, and the required information is not available to regular users, such project is only possible inside the online shopping company.



Customer reviews

4.5 out of 5

46,471 customer ratings

5 star	85%
4 star	3%
3 star	1%
2 star	1%
1 star	9%

Review this product

Share your thoughts with other customers

Customer images

Read reviews that mention

Fig F. Example of key words extraction in review (PlayStation Store Gift Card Review in Amazon)