Prediction of Star Rating of Appliances Based on Text Reviews from Amazon

Ruojia Tao - A15609292 rut013@ucsd.edu

Liuyang Zheng - A15409562 lizheng@ucsd.edu

1. Dataset and Exploratory analysis

For this assignment, we are interested in predicting the rating of appliances in Amazon by text. We are going to use the complete Appliances.json(602,777) review data from the pre-category dataset of Amazon that was released in 2018 and the dataset was obtained from the given course website. The dataset originally has 602,777 reviews, but considered the runtime of the whole experiment, we took 602453 reviews that have `reviewText`in it and randomly shuffled the whole dataset and took the first 100,000 reviews for the purpose of this assignment. And we also took the first 80,000 reviews for training and the last 20,000 reviews for testing. Each review in the JSON format is presented below.


```
{'overall': 5.0,
 'verified': False,
 'reviewTime': '05 9, 2014',
 'reviewerID': 'A13IW3A6W43U0G',
 'asin': '1118461304',
 'style': {'Format:': ' Hardcover'},
 'reviewerName': 'Mmf',
 'reviewText': "I so appreciated the insights and perspective this book.  Allen Gregerman organIzes and presents ideas beautifull
y - reminding us that seeing things in new ways is so very valuable.  So much about life encourages us to seek the known and comf
ortable and yet so infrequently is that where growth and progress is realized.\n\nThe writing style is wonderful. I've given copi
es of the book to my HR friends - and it has been very well received\n\nImagine if more organizations embraced this thinking!",
 'summary': 'The necessity of this book!',
 'unixReviewTime': 1399593600}
```


We might notice that one of the most indicative features of each review is the `reviewText`, which is still in the plain text format. And we are going to apply some text mining techniques and to extract the values out of the 'reviewText' in order to predict the `overall` ratings. And we might also find that the `overall` rating for each review is an ordinal integer ranging from 1 to 5 that shows how each users feel about that specific appliance product, with

that being said, the whole experiment could be casted as a classification problem later when we build our classifier to predict the rating.

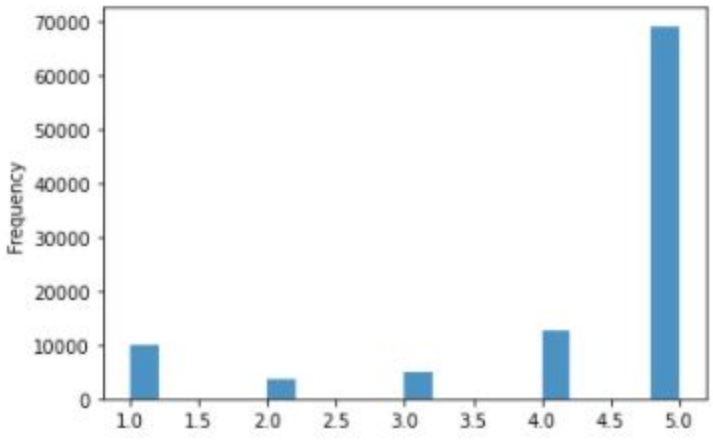
We decide to focus on the text data to predict our star rating, we need to see the variables from the dataset to decide which variable is efficient for our model. Here is the explanation of each variable:

- `reviewerID` - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- `asin` - ID of the product, e.g. 0000013714
- `reviewerName` - name of the reviewer
- `vote` - helpful votes of the review
- `style` - a disctionary of the product metadata, e.g., "Format" is "Hardcover"
- `reviewText` - text of the review
- `overall` - rating of the product
- `summary` - summary of the review
- `unixReviewTime` - time of the review (unix time)
- `reviewTime` - time of the review (raw)
- `image` - images that users post after they have received the product

- The reviewer ID is used as an identifier for who leaves the review, and can be used as the factor to compare the similarity between the different product reviews from the same user, but since we mainly focus on the text, we will not use it in this experiment.
- Asin ID is used as an identifier for what product is this review about, and can be used as the factor to compare the similarity between different reviewers from the same product, but since we mainly focus on the text, we will not use it in this experiment.
- Reviewername is the literature name of the reviewer and has similar usage as reviewer ID, but to protect reviewers' privacy and take some ethical consideration. We will not use it.
- Vote is the count as consumers think this review is pretty helpful, so they give the credit to the comments. Since we mainly focus on the text only, we will not use it in the model.
- Style is the comment format of the comment which is not related to the text information, so we will not use it in the model

- **reviewText** is the text body of the comment. It is our main source of information and we will apply some text mining techniques on it to use it to predict the User's ratings
- **Overall** is the start rating of the product, which is the information we want to predict.
- **Summary** is the summary of reviews, which may contain some useful text information, so we will include in the model.
- unixReviewTime and reviewTime are both time elements of the reviews made by users, but it is not relevant to our text data, so we will not use it in the model.
- Image is the picture left by the users. Since it is not relevant to text data we will not use it.

An interesting finding for the dataset is that this dataset is imbalanced. The distribution of the `overall` rating shows the majority of the customers give 5 stars for the product and then 4 stars and 1 star are also relatively common. However, the plot also shows that people would tend to give 2 stars and 3 stars much less often than the other three ratings, and the whole disparity of rating distribution is the issue that we need to solve when we build our classifiers.



Besides, we also notice that the `summary` feature in each review also gives us some indicative information about the user’s attitude towards the product. And some users would even

directly put their ratings for the product in this field, so we can also incorporate this feature in our model and see how it can affect our predictions.

2. Predictive Task

We are going to predict the `overall` star rating for each review based on mainly the `reviewText` we have. Note that we are also going to incorporate `summary` features in the model through sentiment analysis. Since the dataset given is pretty clean, we do not need a lot of preprocessing. We take only features that are relative to the text and ratings. One of the most trickiest problems associated with this task, just as described above, is the imbalanced ratings we have. Thus, we need to develop some meaningful evaluation metrics in order to keep the model from giving us a trivial answer. In this case, we have almost 70% of the ratings being the 5-stars, and if the model keeps classifying the rating as 5-stars, it should also achieve very good accuracy but this is not what we want here. However, **overall accuracy**(fraction of correct predictions) is still a very important evaluation metric for classification problems, so we still need to keep this metric here. Since we know that solely achieving high accuracy might make our classifier become a trivial one, we need to have evaluation metrics that could detect the problem. Thus, we also have **BER** and **False Positive Count** as our metrics and let the positive be the prediction being 5-stars and negative being every rating other than 5-stars to catch the case when our model is always predicting 5-stars. Besides, we also want to see other than the 5-stars, how well our model performs on the other four ratings. So we also need to have an evaluation metric of **Accuracy performed only on the other four ratings**. Last but not least, we would also give the most common metric for ordinal classification: **Mean Absolute Error (MAE)**, which is the average absolute deviation of the predicted class from the true class (measuring the distance as the number of categories of the ordinal scale), and its value range from zero to Q-1 where Q is the number of different ordinal data in our dataset.

In order to compare our models' performances, we will have two baseline models. The first one is to predict 5-stars all the time, which should give us a relatively high overall accuracy and false positive count should be very high and accuracy over the other four ratings should be 0. This baseline model is intended to test whether our model is trivial and we need to beat this baseline model's performance in order to prove that we have a meaningful model. In the second baseline model, we will build Bag-of-Words feature vectors by counting the 1,000 most common words across all reviews(removed all punctuation and capitalization), and perform multiclass classification using Logistic Regression without tuning the imbalanced dataset. In this way, we will have a relatively simple model to compare with and we can see how well the bag-of-words model is predicting.

We will access the validity of our model's predictions by first comparing it to our baseline model 1 to see if it is a valid model and then we will compare it with our baseline model 2 to see if it's improving by using the evaluation metrics we have above.

Since we are dealing with the text, TF-IDF is a technique that can't be overlooked and it should be appropriate for a classification task like this. It allows us to know which word is unique to that specific document across all corpus and it also can be very useful when combining it with n-grams to study the sentiment of the text, which could help to predict the ratings from customers. We can first compute TF by counting how many times 1000 most common n-gram appears in the document, and loop through the whole dataset to get the IDF for that n-gram and multiply these two components to get the TF-IDF for each word. Thus, We would get feature vectors of TF-IDF scores for the unigram model and fit the vector and ratings into Logistic Regression to predict the rating from the testing set. We would also try to include direct sentimental scores obtained from nltk library for each `reviewText` and `summary` in another model because the `summary` feature in the review seems to be more sentimental oriented, so combining them with the Bag-of-Words model also seems to be appropriate for this task.

3. Models

We have three models in total for this assignment, and all of them are related to text features in the review.

For the first model, we took the baseline model 2 to a further step in which we balanced the dataset when we fit the Logistic Regression with the `class_weight` set to “balanced”. Besides, we also feel like by removing the english stop words and applying stemming to each token, it might improve the overall prediction power for the ratings because for most of the time stop words do not offer much helpful analysis and it doesn’t really have a emotion associated with it, and stemming could remove many tokens that have the same meaning while standing for different tenses or plural. And after we built our Bag-of-Words model using the specification described above, we tried to optimize the model by using different dictionary sizes, and changing regularization constant C. But when we compared it with the other two models and two baseline models, we found that even though it outperformed two baseline models in terms of the accuracy performed on the other four ratings, false positive counts and BER, it couldn’t beat the other two models. It should be a rather unsuccessful attempt along the process of building the model. And we noticed that the possible weaknesses of this model is that sentiment analysis is pretty sensitive to stop words removal and the removal might cause some important information to be missing in this case.

The second model we have, just as mentioned in part 2, is to combine the TF-IDF and 1000 most common unigram. Just like we have stated above, TF-IDF is a pretty strong sentiment analysis tool, and it’s also very simple to encode different n-grams to the model. And we can also simply compute the similarity between two reviewText based on the TF-IDF which could also be useful when predicting the rating of similar documents. However, we need to know that TF-IDF is still derived from the Bag-of-Words model and it doesn’t have the information for the position of the words in text, which could be its weakness in this case. We

optimized this model by trying different n-grams and regularization constant C. The performance of this model didn't let us down either, it not only outperformed the two baseline models, it also beat the first model we have by increasing the overall accuracy and decreasing the MAE.

The third model we have is by far the most predictive one we have in terms of the evaluation metrics. In this model, we used the basic Bag-of-Words model and besides, we also added two sentimental scores that were obtained from nltk library for each `reviewText` and `summary` at the end of the feature vectors. This model is also very straightforward and simple to build but it also has some weaknesses. First of all, the Bag-of-Words model doesn't care about the position of each token in the text, so even texts of different meaning or different sentiment could have the same vectorized representations. Besides, it doesn't account for the relationship between words and assuming that each word is independent from each other and just counting the occurrences of each word in the text. However, the strength that made this model outperform the others was the fact that we took the `summary` feature into account. We optimized this model by adding several other features of the text(e.g. Length) to the feature vector and tuning regularization constant C. And the performance also proved that the `summary` feature contains even more precise sentimental elements that made this model even more predictive.

4. Literature

Since this dataset is provided by the professor, we have research provided by the professor which uses this data set . In the article, "Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects", the authors tried to justify the recommendations to match users' interests by generating reviews. Since a lot of machine learning processes are mainly black box, it is hard to explain how the recommended system works (Ni). The black box makes the process not visible to the people, so the researchers had to find some other way to prove the effectiveness of the prediction. According to this reason, the

researchers need a bridge to connect the result of prediction and consumers' idea of the prediction. The researchers choose to create the user preference and user writing styles to create the comments and the tip generator as the way of representation of reason why recommend consumers this product. By applying this approach, researchers can reveal the interpretation of the black box. If the generated reviews and tips somehow match the true reviews provided by the consumer, that can prove the prediction model provides recommendations that are in the correct direction for the consumer. They use Reference-based Seq2Seq Model and Aspect Conditional Masked Language Model as the way to generate the comments. Also, they only take the justification of good reviews, without bad reviews. The result of the two models beat the baseline in different ways. The research we did is pretty different from this research but we both use the text data and implement some text mining techniques on the experiments. From this research, we noticed that they only use the good reviews. We can somehow understand that using the text in the good reviews is pretty effective in this data set, but we are not sure what happened about the bad reviews. The dataset we get from it is mainly about ratings of five stars, so we need to be careful of the imbalanced dataset.

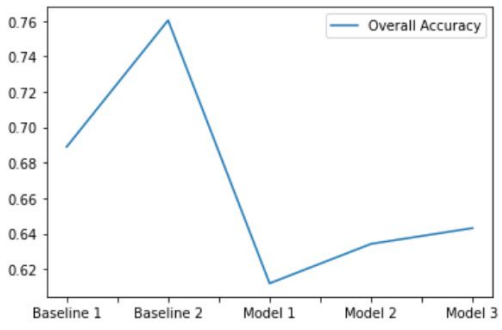
We also found research relative to the prediction of star rating from the review text. In the article, "Prediction of Star Ratings from Online Reviews", the researchers use the dataset from Yelp reviews and make a prediction of star ratings of the restaurant by the reviews left by the customers. They also did some text cleaning process: they remove the punctuations, stop words from the text data (C. S. C. Reddy). The punctuations may influence the analysis of the text data, because adding a punctuation after a word without space will create a new token for the text data. However, it is not very different from the original words. Also, stop words are not very effective in measuring the meaning included in the text data, but since they are necessary in the natural language the stop words always appear in the text data. They also tokenized the rest of text data and applied the stemming on the text data (C. S. C. Reddy). So split all the words and put them as a token, also if they have similar etyma, they will be counted as the

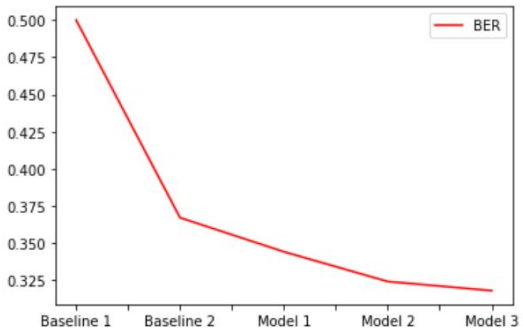
same token. I think the state of art this research have is the they have use several advance classifiers to train the model: s Multinomial Naïve Bayes, Bigram Multinomial Naïve Bayes, Trigram Multinomial Naïve Bayes, Bigram-Trigram Multinomial Naïve Bayes, Random Forest (C. S. C. Reddy). They use accuracy and precisions as elements to measure the performance of their model, and one of the models from this research reaches more than 0.69 accuracy, which is the model used the random forest classifier.

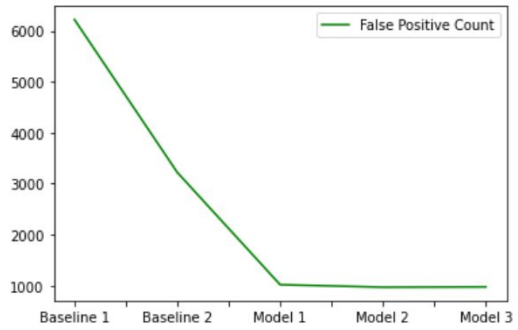
From this research, we noticed that tokenizing the words can be a good choice of text mining and also some of the text cleaning is needed, such as stemming, cleaning the stop words and punctuation. Our model 1 is pretty similar to this research, but our result is not as good as the report.

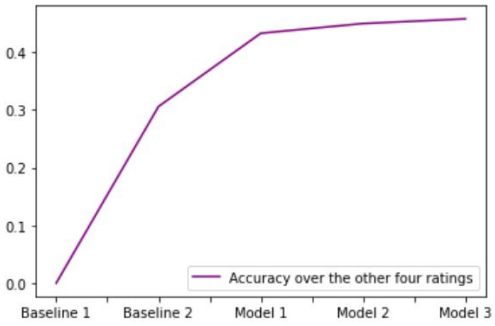
5. Results and Conclusions

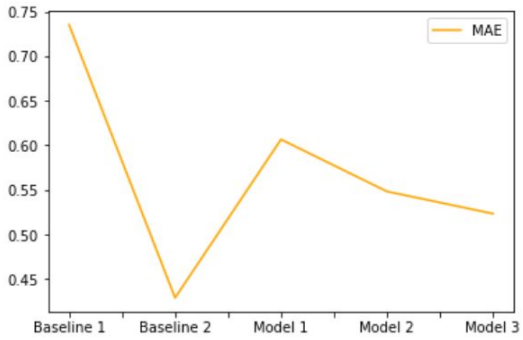
	Baseline 1	Baseline 2	Model 1	Model 2	Model 3
Overall Accuracy	0.68895	0.76035	0.612000	0.634250	0.643150
BER	0.50000	0.36709	0.344165	0.324141	0.318030
False Positive Count	6221.00000	3222.00000	1019.000000	970.000000	976.000000
Accuracy over the other four ratings	0.00000	0.30606	0.433049	0.449928	0.457965
MAE	0.73495	0.42885	0.606500	0.548150	0.523250











Just like what we got from the model part, the most indicative model we have so far is model 3 in terms of the fact that almost all of the evaluation metrics outperformed the other two models and baseline models. This result should be very significant because people would generally think TF-IDF would have a better performance compared to the Bag-of-Words model for its cheaper and efficient regularization power, but in my experiment, it was indicating the opposite. And we believe that the reason is partially due to the extremely imbalanced dataset we got, which has a 70% of chance that rating goes to 5. And maybe the following two sentimental scores that were added at the end of the Bag-of-Words feature vectors were really determining the prediction power here. And the regularization constant C we used for the Logistic Regression in model 3 is 1.0 to keep the model from largely overfitting or underfitting the data. We can see that the two baseline models have very high overall accuracy because they are being trivial and they always predict rating 5 which results in a very high false positive Count and high BER value. And we solve this problem later when we build our own models at

the cost of losing some overall accuracy but significantly improve on the other metrics. Thus, the results of this assignment also informed us the fact that most of the real dataset are either imperfect or unbalanced, people need to take the special property of their dataset into account before developing their predictive tasks and models.

6. References

[1] Ni, Jianmo, et al. “Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects.”

[2] C. S. C. Reddy, K. U. Kumar, J. D. Keshav, B. R. Prasad and S. Agarwal, "Prediction of star ratings from online reviews," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 1857-1861, doi: 10.1109/TENCON.2017.8228161.