Star Rating Prediction of Amazon Movie Review

Yuyan Li
yuyanli@bu.edu

1 Preliminary Analysis

There are total 1397533 data in the training set and each of them have 9 properties: Id, ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text.

I plot a histogram to show the score distribution for the training set. It shows that the number of score 5 is much larger than the others.

I group the data in training set by their product id and found out that some products have a constant score(mean of their score is stable).

constant rating	number of product
5.0	2200
4.0	2689
3.0	904
2.0	249
1.0	23
sum	6065

Compared to the total data set, these data with such a property is just a small portion of the data.

2 Feature Extraction

In the data features, HelprulnessNumerator, HelpfulnessDenominaotr, Summary, Text contains most of the information.

1. Helpfulness

   In the given *feature_extraction* file, we calculate the **Helpfulness** feature by dividing HelpfulnessNumeratror by HelpfulnessDenominator. In the observation, there are some data have more Numerator than Denominator. That must be an outlier, so we need to remove them before data cleaning.

2. WordCount

   In the **Text** feature, customers describe their feeling of the product. We can find out how many words are there in the review by separating the string using space.

3. Word_clean

   We can clean the review text by removing words we don't need. In the clean function, we filter out special characters in the review, And then in the clean_word function, we remove stopwords in the *nltk* package.

4. word_stem

   I also found out that some words are in different tense. So I tried to apply stem function for the text message in the data set. Words with same stem are tend to express the same emotion, so I tend to group them together.

5. CountVector

   I applied the CountVector function to both the text and summary feature in the data set. It is a simple way to both tokenize the text feature into numerical features. It transform the data based on the frequency of each word in the whole training set. In my assumption, the review that have same score have a same preference of using the same word in their review.

6. TF-IDF

   TF-IDF is also a word vectorizor. Different from CountVector, it not only focuses on the frequency of words in the feature, but also provides the importance of the words. In some way I hope this can help distinguish the preference of viewers.

3 Models

After processing the data, I split the training set into 2 part. Three forth of them are used for **training set** and the rest are saved for **test set** in model evaluation.

1. KNN-Word Count

   In the *wordcount* file, I attempts to model the relationship between number of words in the review and their score by fitting K-neighbour classifier into the KNN model. The guessing is that higher score may have more word counts in their review. In the assumption, the value is decided by the average of the values of k nearest neighbours.

2. Naive Bayes - Text & Summary

   In the *Naive Bayes*file, I attempts to model the relationship between the word use in review and the score. I applied Naive Bayes model to the matrix of both text and summary after CountVector and Tfidf. I also tried the matrix of each after transforming the word into their stem form.

Naive Bayes is easy and fast to predict the class of the test data set. It is suitable for solving multi-class prediction problems like this one.

3. SVM

   In the *SVM* file, I assume that SVM is a good model to predict the text. Since the prediction based on summary is not very well, I append the summary to the text to make them as one feature.

   SVM is good for classification questions. They are very effictive in high dimensional spaces since we have so many information after we convert the text into numerical informations.

4. Logistic Regression

   In the *Logistic Regression* file, I tried to apply logistic regression model to the combination of **Text** and **Summary** after CountVector and Tf-idf.

   Logistic regression can help with predict the likelihood of the rating of review being made based on the information in their text.

4 Work Flow

I mainly used the RMSE function to evaluate the performance of models. The lower the RMSE is, the better the model.

1. KNN-Word Count

   I tired different number of neighbours for KNN. And here's their performance:

N	5	12	15	25	30	1000
MSE	1.98	1.86	1.85	1.87	1.88	2.21

   In conclusion, the MSE value decreased and then increased in the range of 12 15. It indicates that the best N value is in that range. However, it also shows that the best RMSE of wordcount and KNN pair is around 1.8, which is relatively high in our test. So in the end, I dicided not to generate the submission set for this model and switch to another model.

   I also tried to plot the confusion matrix of the test set. However it's hard to adjust model based on the matrix.

2. Naive Bayes - Text Summary

   Since we don't know which pair of text component and vectorizor is better for the model, I tried them all in the *Naive Bayes* file. Here's their result:

Feature	CV	Tfidf	stem-CV	stem-Tfidf
Text	1.29	2.20	1.32	2.21
Summary	1.38	1.45	1.40	1.63

   In Naive Bayes model, we find out that it performs much better than the first method. After comparison, CountVector vectorizor performs best in this model, and since Text usually contains more information, the performance of text feature after ConterVector works better for prediciton. As a result, I tried to output a submission file for this model.

3. SVM

   Since Stem function takes too long to perform and do not perform better than the original feature, I decided not to use this function in the following models. Similar to the Naive Bayes model, I applied all pairs into SVM model to see their performance:

Feature	CV	Tfidf
Text	1.12	0.98
Summary	1.40	1.36

   In comparison, Tf-idf works better for SVM model. And since **Text** contains way more information than **Sumaary**, it have more data to compute. As a result, it goes better than models on Summary.

   Therefore, I create a new file calles *SVM-combine*. In this file, I append **summary** after **Text**, and use Tfidf as vector. After testing, the model get a RMSE of 0.87, which is a great improvement.

   Other than that, we observed before that some product have constant review rating. So I extract the **ProductId** of these data in the training set, and set that if we met any one in these special product, we change the score in our prediction into its constant rating. However, after submitting the revised prediction file, the prediction score goes worse.

4. Logistic Regression

   In the *Logistic Regression* file, I tried to apply logistic regression model to the **combination** of Text and Summary after CountVector and Tf-idf. And here's their performance:

Feature	CV	Tfidf
LG	0.89	0.86

   After that, I adjust the cost based on the output:

C	0	20	40	60	80	100
MSE	0.865	0.862	0.857	0.851	0.843	0.912

   Since cost = 80 works best in our local test, I generate a submission file based on this model. After that I tried cost = 90. Even though cost = 90 has a lower MSE on the local test, the test on Kaggle seems worse. It might because the 90 cost model is overfitting our local set.

   Then based on this cost, I increase the max iteration parameter to make the model better.

5 Creativity,Challenges, Effort

1. I tried to find out the products that have constant rating and test my assumption on one of my prediction set.

Even though it didn’t work out, it was funny to test it out.

2. I train most of the model based on **Text** and **Summary** because I think they are the most valuable variable in the whole data set.
3. I applied different ways to clean the data and vectorize the data. And I found out some times the word frequency of data works better, sometimes it doesn’t work so well.
4. I also applied differernt models to see their differences. And I also adjust the parameters in the model to get a better result.
5. The data set is huge. Each iteration will cost a lot of time.