Analyzing Food Reviews and Giving Product Recommendations

David Randall
University of Utah
Salt Lake City, Utah, USA
u1164746@utah.edu

Bashar Al-Habash
University of Utah
Salt Lake City, Utah, USA
u1067631@utah.edu

1 ABSTRACT

For our project, we analyzed how to predict attributes of Amazon food reviews [1] and how to recommend products from those reviews. We used document embedding and clustering to show how embedding review text as vectors using BERT can be useful for predicting users' ratings of products. Furthermore, we logistic regression and bag-of-words to predict the sentiment of users' reviews. We also used various techniques like SVD to give product recommendations to users.

2 PROBLEM AND MOTIVATION

Originally, we were aimed at finding what attributes make up Amazon food reviews. However, we quickly learned that this problem was extremely broad and hard to completely "solve". Therefore, we narrowed our problem down to two areas: how to predict attributes of a review from other attributes of that review (e.g. predict the sentiment of a user's review) and how to give product recommendations. Our motivation for solving these problems are that they can be useful for predicting missing data (e.g. predicting a product's rating from the reviews of the product), understanding current data, and helping users find more of what they want.

3 DATA-SET

We used the Kaggle Amazon Fine Foods Review data-set [1]. This data-set includes around 500,000 reviews of fine foods taken from Amazon, going back as far as October 2012. The data-set consists of ratings, reviews, and other info about food products from Amazon users.

4 DATA MINING TECHNIQUES APPLIED

4.1 Document Embedding

Since a lot of the information in our data-set is contained in the product reviews, we needed to find a way to extract information from those reviews. One of the ways that we chose to analyze the text of the product reviews was to encode the reviews into numerical data. There are many techniques of doing this such as bag-of-words, k-grams, etc. However, most of these methods only store the words of the reviews themselves and do not encode the context of the words. Since the context and meaning of words in the reviews is important for our analysis, we used document embedding to encode our reviews and keep the context/meaning behind each review.

4.1.1 Google's "Bidirectional Encoder Representations from Transformers" (BERT). The document embedding process that we chose to use is Google's "Bidirectional Encoder Representations from Transformers", more commonly known as BERT. BERT is a relatively new (introduced by Google in 2018), state-of-the-art method of embedding sentences and documents which provides more context and meaning than other embedding methods like GloVe or word2vec. BERT utilizes multiple layers of transformers to encode sentences/documents into vectors (see figure 1). Transformers are relatively new natural language processing (NLP) models that allow sentences to be analyzed both left-to-right and right-to-left, hence "bidirectional" in BERT's name [4]. These transformers work by taking in specially structured text data and outputting the encoding of that data (see figure 2).

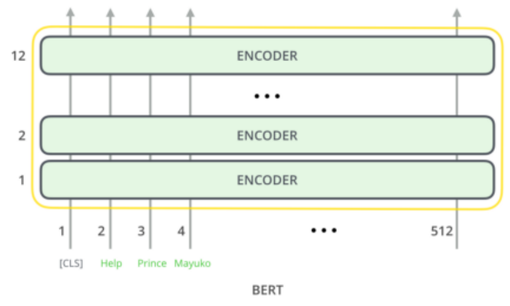


Figure 1: Overview of BERT's Base Model [5]

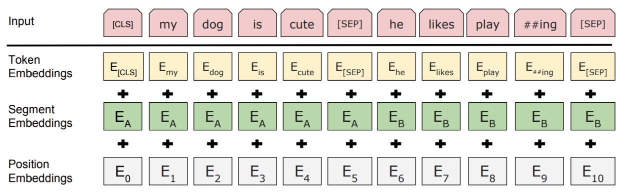

Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2: One Transformer Layer of BERT's Base Model [2]

Since BERT has seen such good results in NLP, we decided that it would be the best option for us to use for our review document embedding. We used Huggingface's pretrained BERT model which uses a modified version of BERT's base model to generate sentence/document embeddings [7].

4.1.2 Data Preprocessing. Before we could encode our reviews using BERT, we needed to preprocess them. We standardized the reviews by converting each word in them to lowercase, removing any special characters, and removing any stop words (stop words are words like "a" and "the" that do not provide meaning to the document embedding). For example, the review:

"Very good for digestive system. Easy to eat with cold milk. Very flaky. Has a nice taste and texture. Sometimes hard to find in stores and the prices are high. You cannot beat the price if you buy from amazon."

after preprocessing becomes

"good digestive system easy eat cold milk flaky nice taste texture sometimes hard find stores prices high cannot beat price buy amazon"

Doing this preprocessing gives us more meaningful document embeddings when using BERT than if we did not process the data.

4.2 Clustering

After getting the document embeddings of our reviews, one of the most natural ways of analyzing the data was looking for clusters. We ran k-means clustering and hierarchical clustering on a subset of our data-set to see if we could find any meaningful trends. To visualize the clusters, we ran PCA on our review embeddings to get the best three-dimensional representations of them. As seen in figure 3, the k-means clusters with k=5 did not produce very meaningful clusters. However, as seen in figure 4, hierarchical clustering with five clusters gives meaningful results. Compared to figure 5, which shows the data grouped by ratings (1-5, given by the user), the hierarchical clustering gives a similar result. This shows that our document embedding can be used to roughly predict the ratings that a user gave to the product they reviewed.

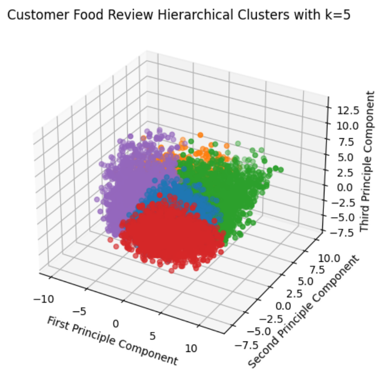


Figure 3: K-Means Clustering of Product Reviews

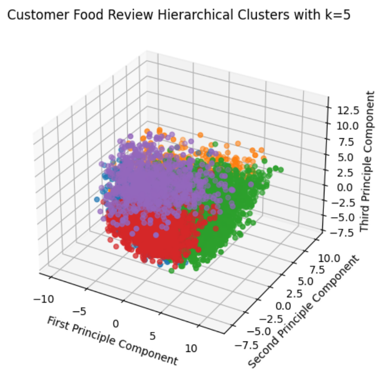


Figure 4: Hierarchical Clustering of Product Reviews

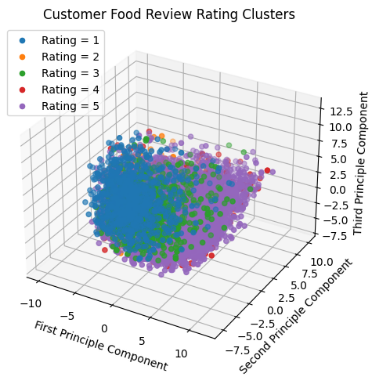


Figure 5: Product Reviews Clustered by Product Ratings

4.3 Misra-Gries

We decided to use the Misra-Gries algorithm to get a better overview of the data-set that we worked on. In particular, we wanted to look at the distribution of the scores for all the reviews which would help us get a general idea for the next two sections, Sentiment Analysis and Recommendations. We decided to use the Misra-Gries algorithm to carry out this analysis due to the large size of our data-set. In other analysis we have done, we had to take a subset of the whole data-set, which does not capture the whole picture, and since the Misra-Gries algorithm is one of the few data analysis methods that is fast enough to use the whole data-set, we incorporated it.

4.3.1 Data Preprocessing. To get a stream of the scores for the reviews to the Misra-Gries algorithm, we first dropped all the other columns in our data-set, except for the score. From there, we created a list of the scores stream and ran the algorithm on it.

4.3.2 Results. After running the Misra-Gries algorithm, we were able to see that we had a total of 568,454 reviews, which was the length of the stream. The score per review is on a scale of 1 to 5, with 5 being the best review score, this gave us five different labels: 1, 2, 3, 4, 5. In the Misra-Gries algorithm, we set $k = 6$ to account for all the 5 labels and used that to initialize the *counter* array and L. The final counter for the stream on each label for the 568,454 reviews can be seen in Table 1 and in Figure 6

Label	1	2	3	4	5
Counts	52,268	29,769	42,640	80,655	363,122

Table 1: Misra-Gries Counts and Labels

4.4 Sentiment Analysis

In this section we wanted to use our data-set to carry out sentiment analysis on review statements. We wanted to leverage logistic regression to be able to classify a review statement as either a positive review or a negative review.

4.4.1 Data Preprocessing. To carry out the analysis we only needed the score and text of the reviews, and thus we created a subset of our data-set only containing those. Moreover, the subset only

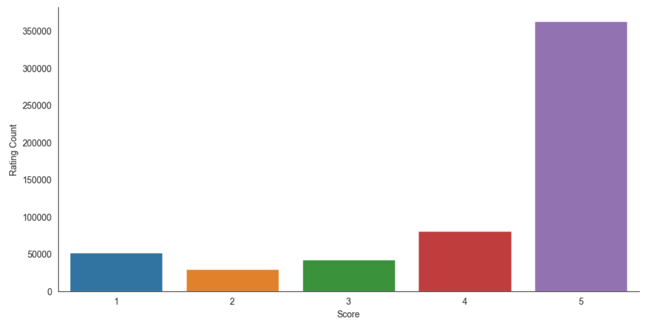
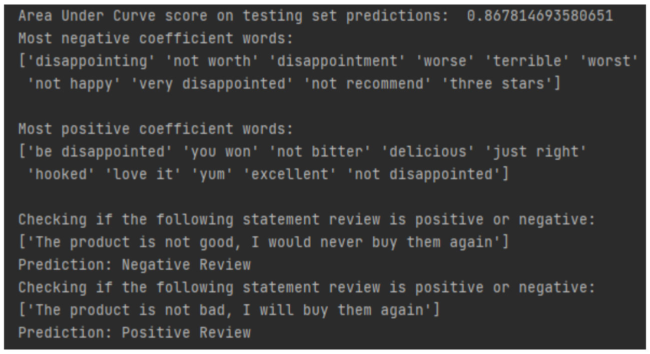


Figure 6: Misra-Gries Results

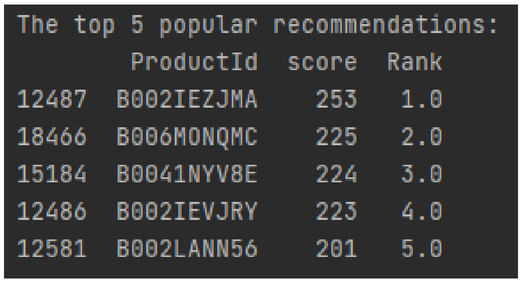


```
Area Under Curve score on testing set predictions:  0.867814693580651
Most negative coefficient words:
['disappointing' 'not worth' 'disappointment' 'worse' 'terrible' 'worst'
 'not happy' 'very disappointed' 'not recommend' 'three stars']

Most positive coefficient words:
['be disappointed' 'you won' 'not bitter' 'delicious' 'just right'
 'hooked' 'love it' 'yum' 'excellent' 'not disappointed']

Checking if the following statement review is positive or negative:
['The product is not good, I would never buy them again']
Prediction: Negative Review
Checking if the following statement review is positive or negative:
['The product is not bad, I will buy them again']
Prediction: Positive Review
```

Figure 7: Sentiment Analysis Result

used 100,000 random samples from our full data-set and classified positive reviews as those with a score 4 or 5 and negative reviews as those with a score of 1 or 2. Reviews with scores of 3 are considered neutral and were removed prior to sampling.

4.4.2 Results. From there, we split our data-set randomly 70:30 to create training and testing subsets. We added a “positivity" value to each review that shows a positive review (score 4, 5) as 1 and a negative review (score 1, 2) as 0. We then used bag-of-words and n-grams on the text of the reviews to create a matrix of token counts. Finally, to create our classifications we used logistic regression on the training set matrix where we have a minimum frequency of 5 with 1-grams and 2-grams. After training our model using logistic regression, we used our testing set to find the accuracy of our model, and the result for the area under the receiver operating characteristic (ROC) curve for our model was 0.8678 = 86.78%. We also show the words in our matrix with the highest coefficients for positive reviews and the words with the highest coefficients for negative reviews in figure 7. Finally, we created a function that uses our model and a testing review statement to classify it as either a positive or negative statement. An example of a positive and negative statement being classified correctly can also be seen in figure 7.



```
The top 5 popular recommendations:
          ProductId  score  Rank
12487  B002IEZJMA    253   1.0
18466  B006MONQMC    225   2.0
15184  B0041NYV8E    224   3.0
12486  B002IEVJRY    223   4.0
12581  B002LANN56    201   5.0
```

Figure 8: Popular Recommendation Result

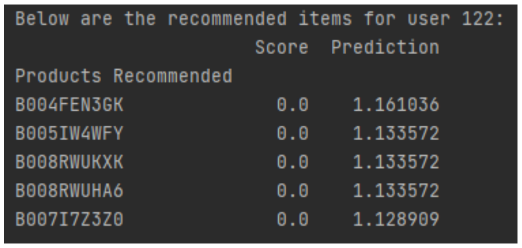
4.5 Recommendation

With the help of our analysis and data mining techniques, we wanted to use our data-set to give users recommendations for products using the reviews that we have. Here we have three different approaches to provide the user with some popular recommendations and user-specific recommendations.

4.5.1 Data Preprocessing. To implement this we first had to do some preprocessing on our data due to its size, this would give us a reasonable run-time for this application and would also give us higher accuracy in our results. In order to accomplish this, we took a subset of the data-set which only included reviews that were made by users who had 20 reviews or more. Doing so resulted in a denser model and allowed more accurate reviews to be taken into consideration. The resultant subset contained 68,015 reviews from 1,891 unique users on 19,488 unique products. We also ran Misra-Gries on the scores in this “experienced" user subset and the results showed a decrease in the ratio of the 5 score reviews from before. The results are (score 5: 38,908 reviews, 4: 14,076 reviews, 3: 7,224 reviews, 2: 3,778 reviews, 1: 4,029 reviews), as we can see the ratio of 5 score reviews to any other score reviews is a lot less significant than what we saw in Section 4.3, which implies a higher review accuracy.

4.5.2 Popular Recommendations. To get the popular recommendations for users, where the recommended products are the same for all users and only depend on the reviews found in our data-set. We did the following: first we got the number of reviews for each unique product and then added up the scores of the reviews to get the product's recommendation score. From here we sorted all the products using their recommendation scores and displayed the top 5 products to the users. The results for this can be seen in figure 8.

4.5.3 User-Specific Recommendations. To get user-specific recommendations, we first split the subset of the data-set randomly 70:30 into training and testing data-sets. Here the training data-set had 47,610 reviews and the testing data-set had 20,405 reviews. From here, we created a table for each user and all of the available unique products. Then we were able to use singular value decomposition to create a diagonal array with a user-specific “prediction rating" on all the available products that they have not reviewed. We then sorted all products for each user using the user's “prediction rating" to get the products they are likely to rate the highest, based on their previous reviews. Finally we returned the user-specific



```
Below are the recommended items for user 122:
                              Score   Prediction
Products Recommended
B004FEN3GK                      0.0     1.161036
B005IW4WFY                      0.0     1.133572
B008RWUKXK                      0.0     1.133572
B008RWUHA6                      0.0     1.133572
B007I7Z3Z0                      0.0     1.128909
```

Figure 9: User-Specific Recommendation Result

recommended products. The results of this method when run on the user with user ID (122) can be seen in figure 9.

4.5.4 Document Embedding Recommendations. Another way that we could recommend products to a specific user is by looking at products that have similar review embeddings to the product the user reviewed. There are many ways of computing the similarity between vectors, however, we decided to use cosine similarity for our purposes since it compares the vectors' orientations rather than their magnitudes (e.g. we do not have to normalize each document embedding vector). Furthermore, cosine similarity is frequently used for looking at document similarity [3].

We recommend products to a user using this method by looking at the document embedding of the user's review on a product. Then, we compute the cosine similarities of the document embedding of the user's review with the reviews of other users. Finally, we take the reviews with the top similarities and recommend the products that correspond with those reviews.

For example, if we want to recommend the top three similar products to the user who gave the review on "Arrowhead Mills Organic Cereal, Oat Bran Flakes" from section 4.1.2:

"Very good for digestive system. Easy to eat with cold milk..."

we get recommendations for "Stash Tea Orange Spice Black Tea", "Snyder's of Hanover Mini Pretzels", and "RiceSelect Couscous Variety Pack". This shows that our recommendation system recognizes that the user originally liked a product that was starchy and/or could be labeled as "healthy" and "natural" and recommended similar products (see summary in figure 10).

5 RESULTS AND CONCLUSION

We were able to use the data mining techniques that we applied to solve the problems that we aimed to solve: predicting attributes of user reviews and recommending products to users. Clustering showed that users' ratings of products could be predicted directly from the document embeddings of their reviews. Misra-Gries showed the distribution of the users' ratings of products, giving us more information to predict users' ratings. Furthermore, sentiment analysis allowed us to understand the overall feeling of users' reviews, which can also be used to better predict their ratings.

Using SVD and vector similarity allowed us to give product recommendations to users and proved to give good results in many cases (e.g. the example in figure 10).

In conclusion, working with large data-sets can be difficult but using data mining techniques can help us extract practical and

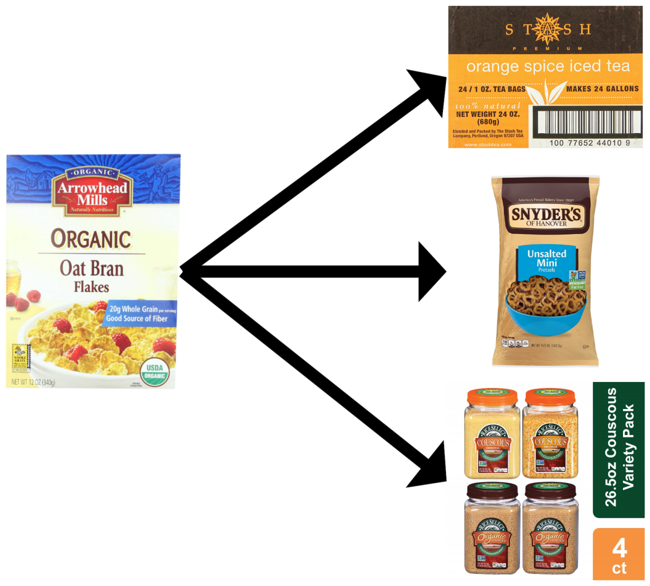


Figure 10: Example User Recommendation Using Review Document Embeddings

useful information efficiently. Moreover, from these results we can learn that data mining techniques are very powerful at giving us insights into our data-set and allow us to use those results for practical purposes.

6 GITHUB REPO

For the code used to generate our results, feel free to see our GitHub Repo [6].

REFERENCES

[1] 2017. *Amazon Fine Food Reviews.* Retrieved February 10, 2020 from https://www.kaggle.com/snap/amazon-fine-food-reviews

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[3] Jiawei Han, Micheline Kamber, and Jian Pei. 2012. 2 - Getting to Know Your Data. In *Data Mining (Third Edition)*, Jiawei Han, Micheline Kamber, and Jian Pei (Eds.). Morgan Kaufmann, Boston, 39–82. https://doi.org/10.1016/B978-0-12-381479-1.00002-2

[4] Ben Lutkevich. 2020. *BERT language model.* Retrieved May 1, 2020 from https://searchenterpriseai.techtarget.com/definition/BERT-language-model

[5] Andreas Pogiatzis. 2019. *NLP: Contextualized word embeddings from BERT.* Retrieved May 1, 2020 from https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b

[6] David Randall and Bashar Al-Habash. 2020. *GitHub Repository.* https://github.com/nightdr/cs5140FinalProject

[7] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. http://arxiv.org/abs/1908.10084