Amazon Rating Prediction with Review Text

Ming-Chieh Lin
A53262891
milin@ucsd.edu

Jie Gu
A53269144
jgu@ucsd.edu

Shuo Wang
A53272439
s8wang@ucsd.edu

ABSTRACT

This research reveals how sentiment analysis on review text could predict user rating on a purchased product using bag-of-words technique to process text data into unigrams and bigrams and using their TF-IDF values and word counts of the most popular 1000 words as features. Comparing the prediction results of different models with respective MSE values and a baseline model using global average as a constant predictor, the result shows that ridge regression on unigram TF-IDF features has the best prediction results. It achieves 0.8380 MSE on the test set, which is 28.95% lower than the global average rating of the training set baseline.

1. DATA SET DESCRIPTION

This dataset contains Amazon's Digital Music category product reviews, including 64,706 reviews spanning May 1996 - July 2014 from website 'jmcauley.ucsd.edu/data/amazon'. Specifically, this dataset is a 5-core dense version, meaning that each of the users and items have 5 reviews each.

It contains the following fields:

1. ReviewerID - ID of the reviewer
2. asin - ID of the product
3. reviewerName - name of the reviewer
4. helpful - helpfulness rating of the review
5. reviewText - text of the review
6. overall - rating of the product
7. summary - summary of the review
8. unixReviewTime - time of the review (unix time)
9. reviewTime - time of the review (raw)

1.1 BASIC STATISTICS

- Review Count: 64,706
- Reviewer ID Count: 5,541
- Product Count: 3,568
- Average Rating: 4.22

1.2 EXPLORATORY ANALYSIS

1.2.1 Rating

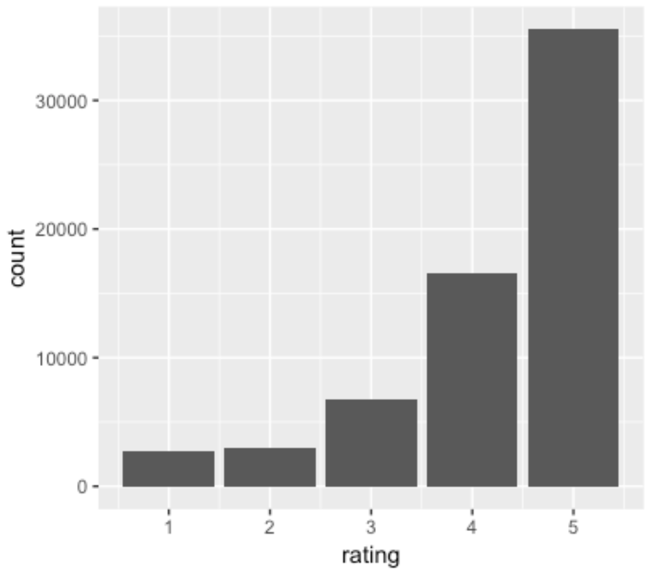

Figure 1.1 Rating Distribution

This category has over half (35,424) of the ratings are 5-star and far less (2,689) 1-star ratings. It is far more positive than negative and is a reasonable distribution on platforms like Amazon because people tend to review only when they really like or hate it, but the dissatisfaction could be sufficiently resolved by a well-designed return or refund policy.

1.2.2 Review Texts

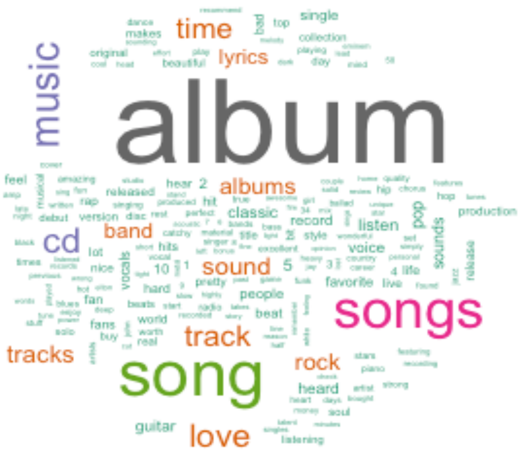

Figure 1.2 Top 25 Unigrams in Word Cloud

Most keywords are category-specific such as albums, songs, and music.

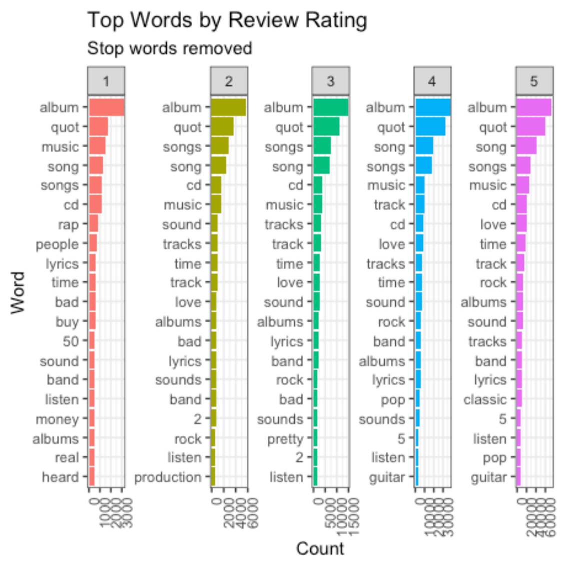

Figure 1.3 Top Unigram by Rating

Word 'pop' shows on 4-star and 5-star rating lists. 'Classic' shows on 5-star rating list only. 'Rap' shows on 1-star ratings list only. This graph gives us a general idea that types of music may be of influence on user rating.

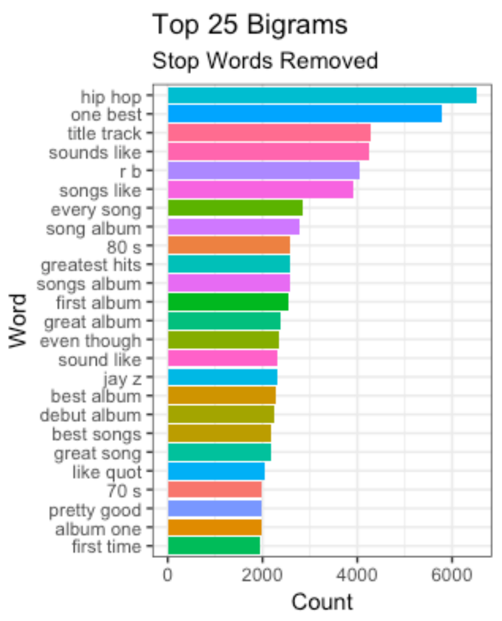

Figure 1.4 Top 25 Bigrams in Histogram

This category has 7,057,472 bigrams without stop-words and upper- or lower-case differences. Music types such as 'Hip-Hop' or 'R & B', ages such as '80s' or '70s', and artists such as 'Jay Z' shows up.

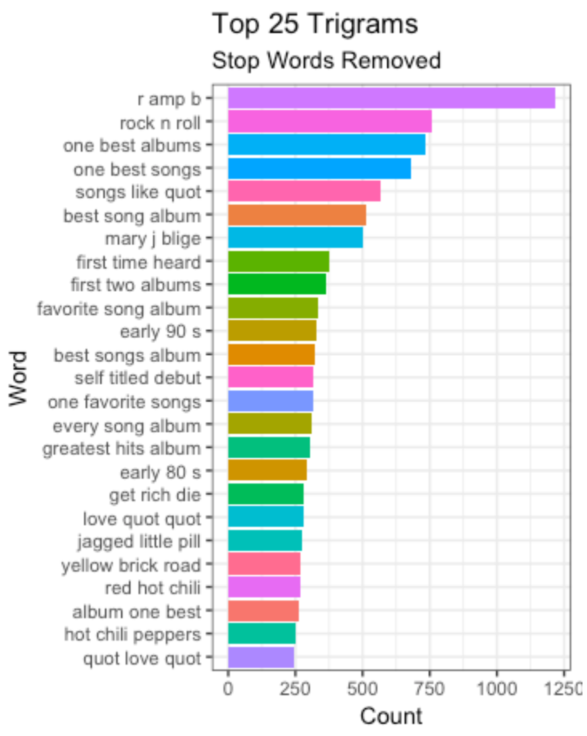

Figure 1.5 Top 25 Trigrams in Histogram

This category has 6,992,985 trigrams without stop-words and upper- or lower-case differences. We can tell from the list that most of the trigrams are not of too much use for text analysis except some words like ‘R & B’, ‘Rock & Roll’, or ‘Mary J Blige’. Therefore, we will not proceed with trigrams in further predictive tasks.

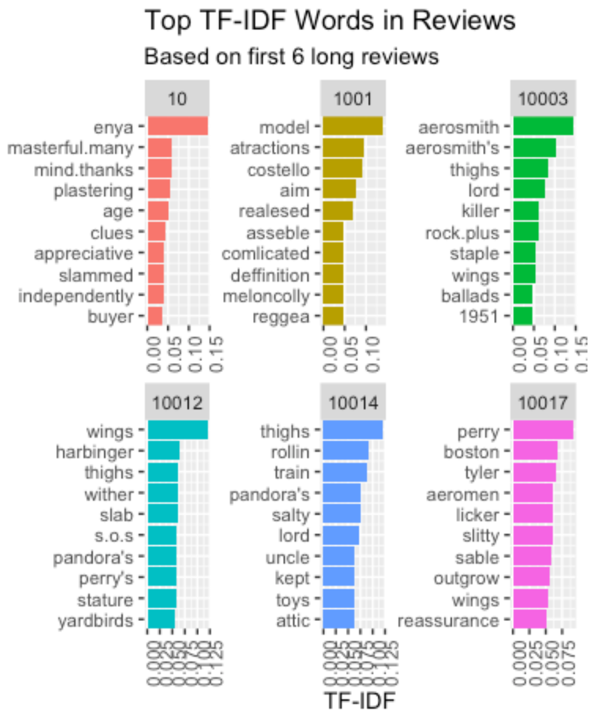

Figure 1.6 Top 10 TF-IDF unigrams from 6 reviews.

Artist- and song-specific words such as ‘Enya’ or ‘Aerosmith’ pop up in the lists.

2. PREDICTIVE TASK

2.1 Predictive Task and Feature Extraction

Our model is designed to predict user rating based on what review he or she wrote on the product. This analysis could understand the review sentiment based on word usage and its associated ratings. The features are the review texts users wrote, removing punctuations, non-case-sensitive, and keeping only the most common 1000 words. We further used and compared the performances of the following feature processing methods:

I. Bag of Words: Unigram vs. Bigram

Compared to the more straightforward concept of unigram, bigrams are supposed to preserve more meaning of words because they preserve the sequence of words assuming sequence plays an important role in text sentiment. Furthermore, when discussing a topic like music, a lot of proper nouns show up because the artists and artworks are the center of discussion and usually have names composed of two or more words. Therefore, we are interested in comparing the results of unigrams and bigrams.

II. Calculate word counts vs. calculate TF-ID

We chose only the top 1000 used words for further analysis because it would reasonably get rid of the long-tail words in reviews. TF-IDF represents how much weight on a specific topic a document discusses compared to other documents across the whole corpus. TF-IDF draws a meaningful comparison to word counts because it potentially presents the latent topics of documents better than simply count the frequency of word usage.

2.2 Model Evaluation and Validity

We used MSE (Mean Squared Error) to evaluate model performances on testing sets because we assume the error terms should follow a Gaussian distribution, and practically speaking, far-off rating predictions would have more negative impact on user experiences and therefore should be punished more.

2.3 Relevant baselines

Our model would be compared with a constant predictor using global average, which is the best constant predicting value with least MSE compared to other constants.

- Global average of rating: 4.2225
- MSE of the constant predictor: 1.17955

3. LITERATURE

The dataset we used is described in section 1, which we found at 'jmcauley.ucsd.edu/data/amazon'.[1]
That article used this data to create image-based recommendations on styles and substitutes.
A lot of article has been found to study topics related to rating prediction with text mining.
Basically, they focused using:

i) binary classification:
Pang, Lee and Vaithyanathan (2002)[2] approach this predictive task as an opinion mining problem enabling to automatically distinguish between positive and negative reviews. (Reviews are considered as positive when they have at least 3 stars out of 5, while reviews are said to be negative when they received less than 3 stars.) Among others, they compare the performance of Naïve Bayes and SVM with linguistic features such as Part-of-speech.
ii) multi-class classification:
'Yelp Dataset Challenge[3]: Review Rating Prediction', the authors used classification models like logistic regression, Naïve Bayes classification and perceptrons, with features of unigram, bigram and latent semantic indexing. Also, Fan and Khademi (2014)[4] experiment a combination of four different machine learning algorithms with three feature generations methods using a corpus from the Yelp dataset.
iii) Creative methods:
Li, Liu, Jin, Zhao, Yang and Zhu (2011)[5] bring a new perspective to the review rating prediction task. They incorporate the reviewer and product information in order to predict review ratings more accurately. This is achieved with a three-dimension tensor medialization where each dimension pertains to the reviewer, product or text feature.

Similarly, McAuley and Leskovec (2013)[6] intend to understand hidden dimensions of customer's opinions to better predict the numerical ratings. They also find out that the user information is a rich source of information to take into account when predicting ratings with review text.

4. MODEL

We used three classifiers from scikit-learn to complete the prediction task. They are ridge regression, logistic regression, and support vector classification with a linear kernel. For each classifier, we used unigrams or bigrams in combination with their word count value or TF-IDF value to construct feature vectors, and then we trained the model and tested the performance respectively. Therefore, there are twelve different models in total.

4.1 Data Preprocessing

The raw data includes 64,706 reviews. We first shuffled them and then divided the whole data set into three parts: 20,000 for training, 20,000 for validation, and 24,706 for test. We removed punctuation marks (that are in Python's string.punctuation collection) and stop words (that are in nltk's corpus.stopwords collection) in all reviews. For each model, a word ID dictionary (using unigrams or bigrams) is built and feature vectors are extracted by the dictionary.

4.1.1 N-grams
In this case, an n-gram represents a sequence containing n consecutive words. We only used unigrams and bigrams in our models, because apparently trigrams or n-grams with larger n are not likely to give better performance.

Unigram list can be constructed simply by Python built-in list data structure. Bigram list can be constructed by firstly building unigrams list and then calling nltk's bigrams functions.

With a unigram/bigram list, a dictionary can be built with feature values corresponding to unigram/bigram keys. We used two kinds of feature values: word count and TF-IDF. For word count method, we calculated the frequency of each unique unigram/bigram in all review texts in the training set, and then we built the dictionary based on 1,000 unique unigrams/bigrams with 1,000 highest frequencies. For TF-IDF method, we initially built the IDF directory and selected top 1,000 unique unigrams/bigrams with 1,000 highest IDF values, but the MSE turned out to be high. We think the disadvantage of this building method is that it only keeps the rarest unigrams/bigrams and sometimes they are trivial to the predicting ability of the model. Therefore, we changed our method to firstly building the word count dictionary and then calculating the IDF value of each unigram/bigram in the dictionary and built a new IDF dictionary. Final TF-IDF values would be generated when constructing feature vectors.

4.1.2 Feature Vector

Each feature vector has a length of 1,001 (1,000 most common unigram/bigram, and an offset feature). We assigned each unigram/bigram an ID to represent its position at the feature vector.

4.1.2.1 Word Count

For each review text, the feature vector is constructed using the appearance time of a unigram/bigram (that must be in the dictionary).

4.1.2.2 TF-IDF

For each review text, we firstly computed the TF value of each unigram/bigram that is in the dictionary. With the TF values and IDF values (already stored in the dictionary), we generated the feature vector by multiplying them.

4.2 Model Selection

We used three kinds of classifiers from scikit-learn and compared their performances. The first one is ridge regression, which is a linear regression with penalty on the estimates. The second one is logistic regression, in which ratings are treated as different classes (0.0, 0.5, …, 5.0). The third one is support vector classification with a linear kernel, in which ratings are also treated as different classes.

4.3 Training, Validation, and Test

For the validation process, we used 20,000 reviews to train the model (by scikit-learn built-in methods) with different regularization strength, and then we tested the model's performance on the validation set. We found that too large lambda or too smaller lambda would not bring significant change to MSE, and MSEs with lambda values in a short interval were similar. Therefore, we only used lambda values from {0.005, 0.01, 0.1, 1.0, 10, 100, 500} and recorded the validation MSEs respectively.

After we found the optimal regularization strength, we set the "alpha" (or "C") parameter of the classifier correspondingly and trained the model on the combination of training set and validation set (40,000 reviews in total). For all three kinds of classifiers, we set the "fit_intercept" parameter to False. All other parameters were set to default initially.

However, after we trained several models and tested their performances, we found that it was more difficult for logistic regression and support vector machine to converge completely. Besides, support vector machine's MSE were extremely high comparing to the other two classifiers. To fix the convergence problem, we set the "max_iter" parameter of logistic regression and support vector machine to 500 to obtain complete convergence. We also figured out that support vector machine's default "rbf" kernel was not suitable for our task, and it led to overfitting due to higher complexity of features. Therefore, we changed the kernel parameter to "linear" to eliminate the overfitting problem.

After training models on the 40,000 reviews data set, their performances were tested on the test set (containing 24,706 reviews).

We also considered other models like Naïve Bayes classifier and lasso regression. The features in the vector were not independent from each other (e.g., "good" is very likely to appear if there is a "nice" in some text), so we gave up the Naïve Bayes model. We also tested the performance of lasso regression, but its performance was worse than ridge's, and its advantages could not be applied to our feature vectors (we do not plan to zero out some coefficients). Therefore, we only used ridge regression as a representative of linear regression. Besides, the result will show that ridge regression generally works better in our task, which indicates that it is more natural to use linear regression to handle rating data. The number of classes (11) did affect the predicting ability of classifiers like logistic regression and support vector machine.

5. RESULTS

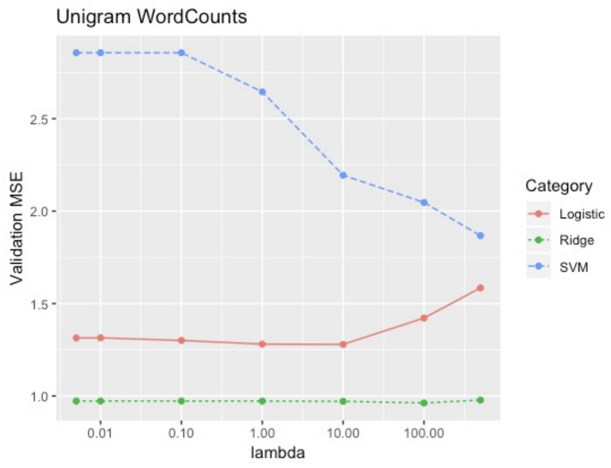

Figure 5.1

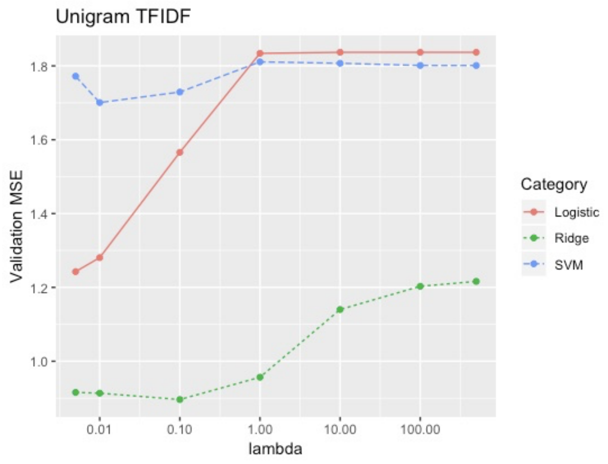

Figure 5.2

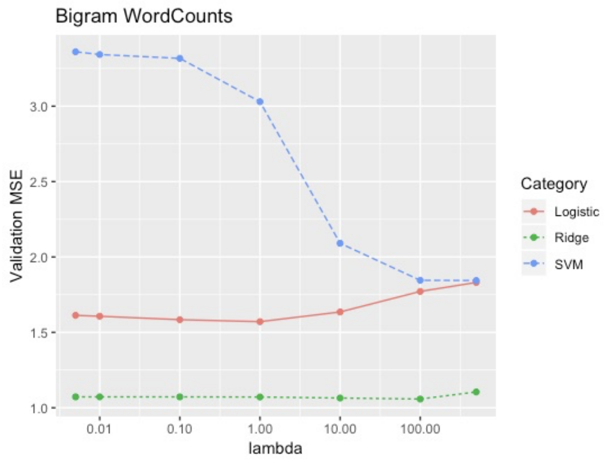

Figure 5.3

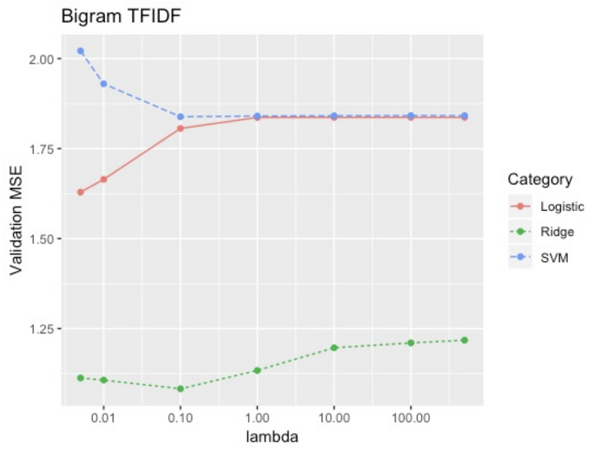

Figure 5.4

Figure 5.1 and Figure 5.3 show how validation MSE changes as lambda (regularization strength) changes if we use word count values. For both unigrams and bigrams, the MSE of ridge regression barely changes and keeps the lowest among all three classifiers. The MSEs of logistic regression and support vector machine has evident variance. Logistic regression prefers smaller lambda values while support vector machine prefers larger ones.

Figure 5.2 and Figure 5.4 show how validation MSE changes as lambda changes if we use TF-IDF values. It turns out that small lambda values work fine for all three kinds of classifiers .The reason behind this might be that the values of TF-IDF are already very small comparing to values of word count (i.e., the variance of TF-IDF data is numerically smaller), so the importance of penalty terms is weakened for TF-IDF models.

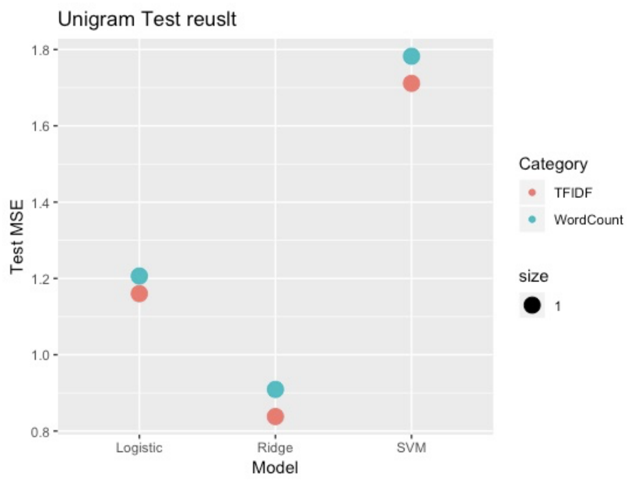

Figure 5.5

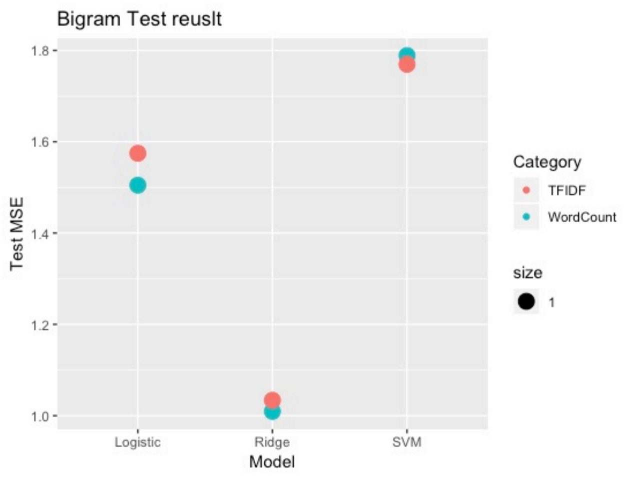

Figure 5.6

Figure 5.5 shows that TF-IDF works better for unigram models no matter we use which kind of classifiers. Figure 5.6 shows that word count works better than TF-IDF for bigram models using logistic regression or ridge regression, and the performances are similar using support vector classification.

For both unigram and bigram, we found that ridge regression worked best among all three classifiers. In our task, logistic regression and support vector classification need to distinguish 11 classes, which reduces their accuracies. For this multi-class problem, we tried both one-vs-rest method and multinomial method, but neither of them worked very well.

We also found that unigram models worked better than bigram models. The result is quite reasonable if we look at the bigrams statistic in Figure 1.4. For some top common bigrams, such as “greatest hits” or “great album”, there is actually no need to use two words to analyze the sentiment because “greatest” or “great” could already represent the features. The redundancy in bigrams can affect the performance of our models.

From the result, we conclude that ridge regression with unigrams and TF-IDF is the best of all twelve models for our task. It achieves 0.8380 MSE on the test set, which is 28.95% lower than the baseline (global average rating of the training set).

For our best model, we also analyzed words with 20 highest and 20 lowest coefficients:

Top 20 positive words (higher coefficient)

(8.953773125508844, 'decided')
(7.887107336402777, 'range')
(7.653750607619889, 'talent')
(6.607062995092129, 'taste')
(6.599718541502019, '45')
(6.57546923644013, 'peace')
(6.57309815071944, 'inspired')
(6.485733652438709, 'dont')
(6.387091925221731, 'la')
(6.233712145214604, 'dre')
(6.207245595282777, 'beauty')
(6.090263045588114, 'tori')
(5.983020978009756, 'contemporary')
(5.783594083295869, 'roll')
(5.780064675601929, 'quite')
(5.726373670026841, 'beautifully')
(5.4709887691245385, 'rain')
(5.396976529430144, 'upon')
(5.2455206043533265, 'isnt')
(5.237427764650378, 'lady')

Top 20 negative words (lower coefficient)

(-24.852795624512787, 'word')
(-19.218138221264173, 'kind')
(-14.096410526509526, 'late')
(-13.983636141005395, 'rappers')
(-13.448657880781361, 'check')

```
(-12.9402053987302, 'buying')
(-12.07786147236827, 'rest')
(-11.526707178955364, 'ability')
(-10.85206385433479, '10')
(-9.698530136313108, 'lack')
(-9.477742921726657, 'pac')
(-8.82191174156563, 'instead')
(-8.692601991290498, 'riffs')
(-8.62439489750667, 'cry')
(-8.433029525183267, 'ways')
(-7.284161390435394, 'popular')
(-7.282261387400745, 'minutes')
(-7.191492922146162, 'car')
(-6.8658016994650355, 'artists')
(-6.604918922134236, 'biggest')
```

We expected to find most positive words and most negative words. However, the result turns out to be interesting. It is reasonable for some words to stand out, such as “talent” or “inspired” indicating positive feeling, and “lack” indicating negative feeling, while many words’ appearance does not make sense, e.g., “45” or “taste” in positive words and “word” or “rest” in negative words all seem neutral. Besides, another interesting finding is that we can see people’s general attitude towards some artists or genres. The ratings tend to be high when they are relevant to “contemporary” or “roll” (supposed to represent rock n roll), and Dr. Dre has a good reputation in ratings. Words like “rappers” exert negative influence on ratings perhaps because rap’s listeners are more fastidious.

6. Reference

[1] McAuley, J. (2018). Amazon product data. Retrieved from http://jmcauley.ucsd.edu/data/amazon/

[2] Pang, B., Lee, L. & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of EMNLP, 271-278. doi:10.3115/1118693.1118704

[3] Nabiha Asghar. Yelp Dataset Challenge: Review Rating Prediction 2016

[4] Fan, M. & Khademi, M. (2014). Predicting a Business’ Star in Yelp from Its Reviews’ Text Alone. ArXiv e-prints. doi: 1401.0864

[5] Fangtao Li, Nathan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, Xiaoyan Zhu. (2011). Incorporating reviewer and product information for review rating prediction. Artificial Intelligence – Volume Three. Pages 1820-1825

[6] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In RecSys, 2013.