Mining Users' Opinions from Amazon Reviews

by Xinyu Zhang

Contents

Goals and desired outcomes

The goal of this project is to mine opinions of users towards health care products from their review text. To do this, I collected Amazon review dataset and product metadata. Specifically, I aim to answer the question:

When reviewers like a certain product or dislikes a certain product, what do they like/dislike about?

Data Collection

Data is collected from website http://jmcauley.ucsd.edu/data/amazon/. It stores a collection of Amazon reviews and product metadata from various departments of Amazon. The data I use are reviews and metadata for products from the department of "Health & Personal Care". Both reviews and metadata are stored in json files. I downloaded them from online and save the files in my local computer.

Methods and Results

To answer the two questions, I propose to do analysis that follows these four steps:

- Data cleaning and integration
- Exploratory data analysis
- Sentiment analysis to identify the positive opinions and negative opinions
- Extracting the relevant product features or perspectives mentioned in these opinions

In the following sections I will present how each step is done.

I. Data Cleaning and Integration

Product metadata. As described earlier, both the review data and product metadata are stored in json format. So first I read in the json data using Jupytor notebook. Let's first take a look at the first five rows of metadata:

The meta dataset provides information of each product in the Health care and personal care department on Amazon. Each is for a specific product and each column provides one aspect of the products:

- 'asin' is the product unique ID.
- "description" gives a brief description of the product: its function, ingredients etc.
- 'related' lists what other products that users who bought this product also bought.
- 'salesRank' gives the rank of the sale amount of product.
- "categories" shows all categories that a product can belong to.
- 'price' and 'brand', are price and brands of the products.

There are in total 263032 products in the department of Health & Personal Care. Although in the same department, these products can have vastly different functions. They can be either music CDs, Vitamins or clothes. If we put all products together for sentiment analysis, the model may work poorly due to such diverse group of items, since some words may only describe some of these products but never used for some others. For example, "rythms" may be used for CDs but may never for clothes or medicines. Therefore, the first thing I did is to group these products into subcategories, so that product with the same subcategory has similar functions.

The current category column stores all possible categories that a product can be assigned to. For example, a product titled 'Zaner-Bloser Classic Handwriting Pen Single Unit (9228)' is assigned to four categories: 'Health & Personal Care', 'Stationery & Party Supplies', 'Stationery', 'Fine Writing Instruments'. However, there are two problems associated with such categorization system. Firstly, each product has a unique combination of multiple categories, so that there are in total many combinations (over 1800). Secondly, some of categories to define a product are informative while some others are not. Using this product as an example again, 'Fine Writing

Instruments' or 'Stationery & Party Supplies' can be more informative than ‘Stationary’. In addition, since all products belong to the 'Health & Personal Care' department, this category isn’t informative either. Therefore, for every product I picked the most informative and commonly used category. This ensures that products under the updated category share similar functions and the total number of categories is also considerably reduced.

After the products categorization system is updated, the number of categories is reduced from 1805 to 41. Following is a figure showing the amount of products under each category after the categorization system is updated:

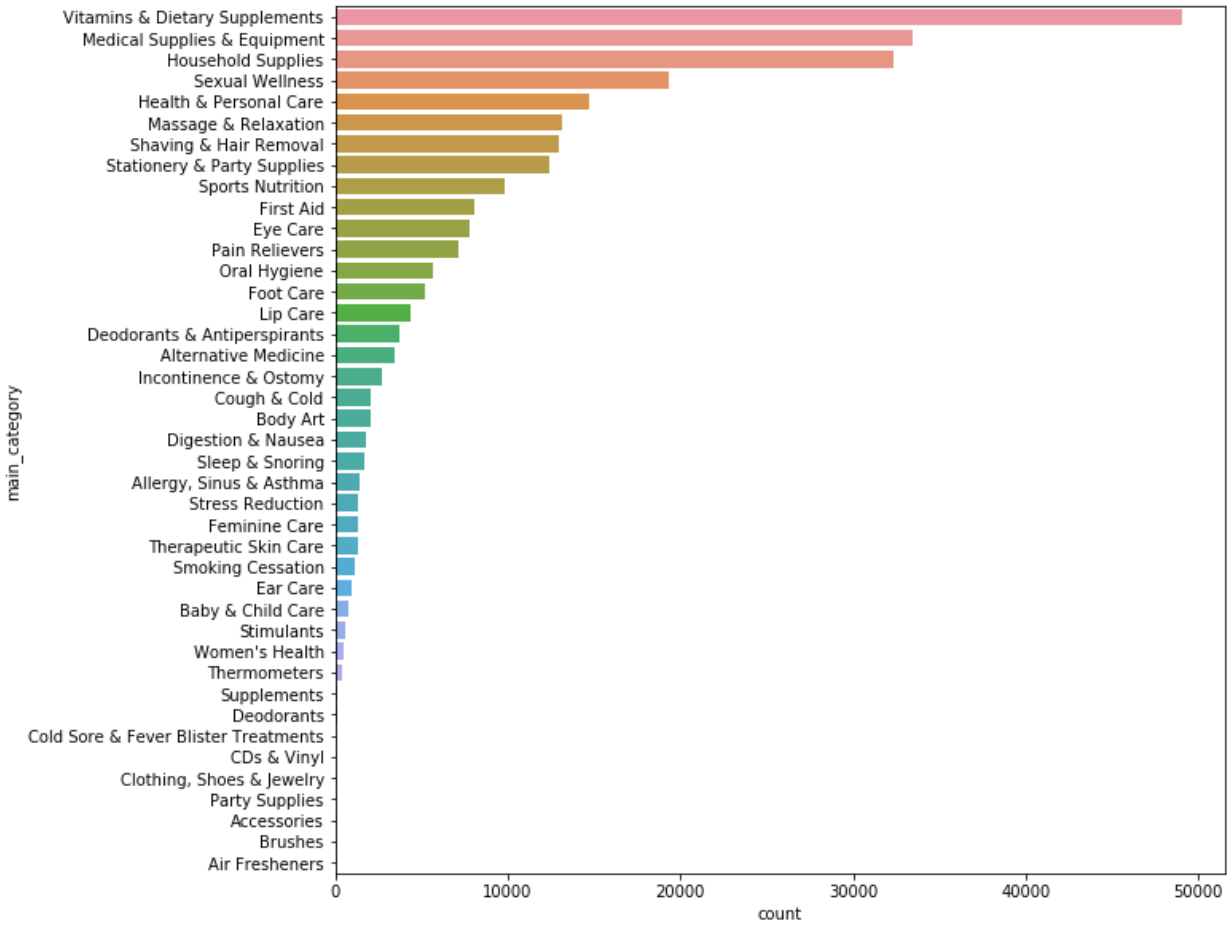


Figure 1. Amount of products in department of Health & Personal Care under each category after the categorization system is updated

It shows that the biggest category is 'Vitamines and Dietary Supplements', while the smallest is 'Air Fresheners', the number of which is too small that it is hardly seen on the figure. In this exercise I will first focus on one category: "Sleep & Snoring". I choose it because there is intermediate number of products in this category, which makes it possible that the analysis used for this category would fit the larger categories and smaller ones.

Review data. The review dataset keeps information of the reviewers and the review text they wrote for a specific product. Most columns are information relevant to the review:

- reviewerID: the reviewers' ID
- reviewerName: reviewers' user name
- helpful: with the first element as the number of people who find the review helpful and the second as the number of people who find it not helpful
- reviewText: the review text
- overall: the rating from 1 to 5
- summary: a sentence to summarize the review
- unixReviewTime: time the review was uploaded in unix time stamp
- reviewTime: time the review was uploaded in regularly format
- asin is the product ID that can link the reivew data to the product metadata.

Column 'asin' links the review data to the product metadata. I merge the review data with metadata on this column and select the subset of data where the products are under "Sleep & Snoring" category.

II. Exploratory Data Analysis

Purpose of the exploratory data analysis (EDA) is to detect insights for later analysis, especially the sentiment analysis. There are three columns of the integrated dataset that can be valuable for sentiment analysis: "overall", "reviewText" and "summary".

"Overall" is the rating, which is an integer that a reviewer scored the product, 1 to be lowest and 5 as the highest. Since I cannot label each sentence manually, the overall rating will be a source of label for the sentiment analysis. Let's take a look at how overall rating distributes:

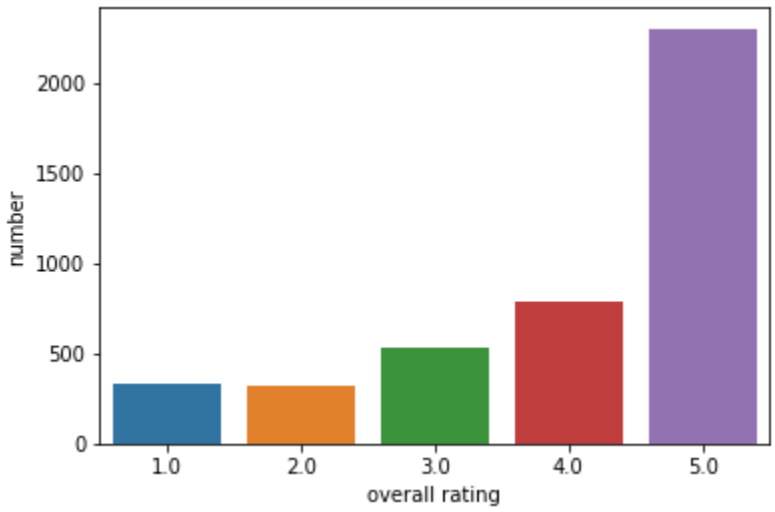


Figure 2. Number of reviews with each rating

Above figure shows that the number of reviews increases as rating increases, where majority of reviews have a rating of 5. This suggests users who have positive experience may more tend to write a review.

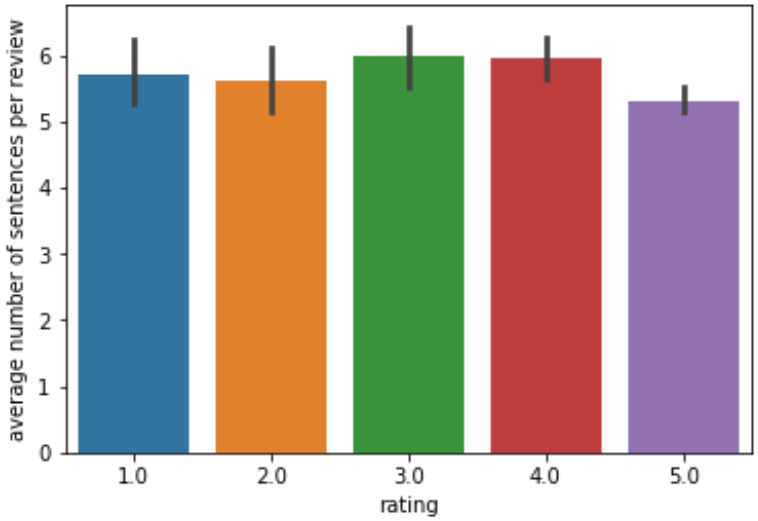
Column "reviewText" is the review text that a reviewer wrote about a product, would be the major column where users' opinions are mining from. Let's take a look at an example:

> *"Ok... so I got this because a friend recommended melatonin to help me sleep at night. The first night I tried it I got a little relaxed, fuzzy sleeping feeling for maybe 5 minutes (I was already a little tired) and then BAM I felt hyper. Every other time I tried it since then? Nada. I might as well be drinking water.As for taste... I love me some straight up spirits, so the alcoholy part didn't bother me. It reminded me a bit of NyQuil meets a bit of, I don't know.. some sort of flavored vodka. I'm certainly drank worse things."*

This reviewer rated the product 1 star, which is the lowest one. The review text gives details of the user's experience, negative and positive and something in between. For example, "The first night I tried it I got a little relaxed" apparently is spoken in a relatively positive tone. "I love me some straight up spirits, so the alcoholy part didn't bother me." is in between positive and negative. The taste does not bother the user although the user did not love it. "then BAM I felt hyper" is a negative experience. It is therefore inappropriate to see the whole review text as being negative, although the overall rating is 1.

This example (together with many other reviews) suggest that it is inappropriate to treat the whole review text as either entirely positive opinions or negative opinions only based on their rating. Instead, each review text needs to be broken into so that opinions are mined from each sentence.

Therefore, I further tokenize each review text and look at the number of sentences (potential opinions) for each rating:



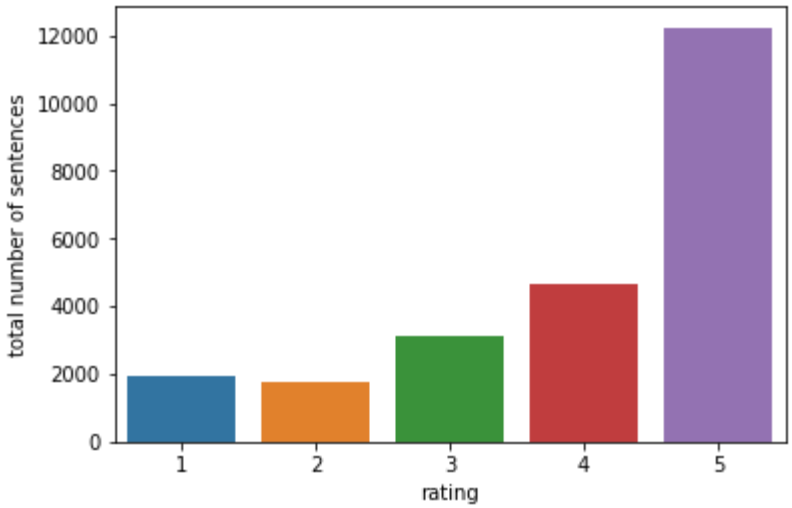


Figure 3. Upper: number of sentences per review for each rating; Lower: total number of sentences in reviews with each rating

Figure 3 shows that the length of review text, in terms of number of sentences is quite similar among different ratings. However, the total number of sentences increases as rating increases, largely drive by the number of reviews (Figure 2).

The last column that is useful for sentiment analysis is the summary column, which is a short summary sentence for the review text. Using the example above to see that how a user wrote a review paragraph and summarized it with a sentence:

> *review text: "Ok... so I got this because a friend recommended melatonin to help me sleep at night. The first night I tried it I got a little relaxed, fuzzy sleeping feeling for maybe 5 minutes (I was already a little tired) and then BAM I felt hyper. Every other time I tried it since then? Nada. I might as well be drinking water.As for taste... I love me some straight up spirits, so the alcoholy part didn't bother me. It reminded me a bit of NyQuil meets a bit of, I don't know..some sort of flavored vodka. I'm certainly drank worse things."*
>
> *Summary sentence: Can't decide if I should dump the rest or just keep using it.*

The summary sentence more straightforwardly reflects the user's opinion (mainly negative) than the review text. Therefore, they will be another valuable sets of data for sentiment analysis.

III. Sentiment Analysis

Data Collection. As discussed earlier, long review text are mixtures of users' opinions, thoughts, past experience etc. So it is inappropriate to treat the whole text as purely positive or

negative only based on the rating. In contrast, shorter reviews (no more than two sentences) most concentrate on users' opinions. In addition, summary sentences are also clear opinions. These two types of data can be appropriately labeled (by the rating). Therefore, I collect these two types of text data for training, validation and testing. I label those with rating 4 or 5 as positive and those with rating 1 or 2 as negative. These text data will be used to train, validate and test a classifier. The review text with more than 2 sentences, all review text with rating at 3 and all summary sentences with rating at 3 will remain unlabeled. For these unlabeled data, each sentences will be fit to the trained classifier and the probability of that sentence being positive or negative opinions will be predicted. The whole process can be presented by the following decision flowchart of collecting data to train, validate and test a classifier:

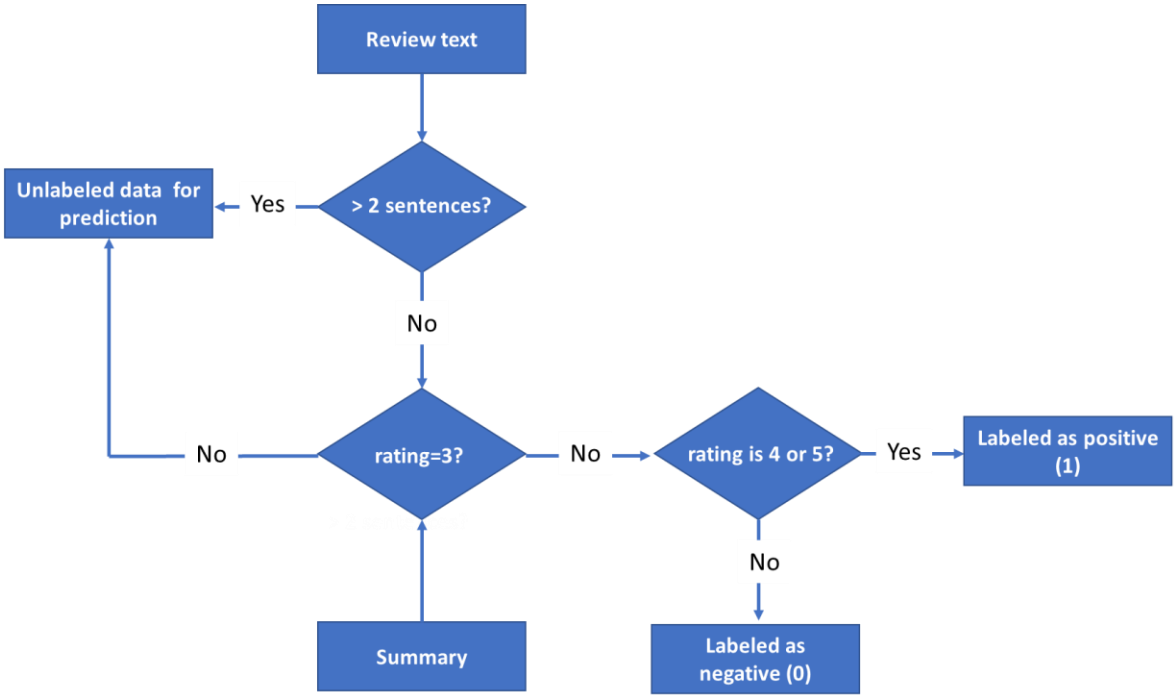


Figure 4. Decision flowchart of collecting review text and summary as labeled data for training, validation and testing of sentiment analysis or as unlabeled data for sentiment prediction.

Data Preprocessing. Before feeding the data into a classification model, the raw text data needs to be preprocessed and ready to be transformed into features, such as words and n-grams. The preprocessing follows five steps:

1. All letters changed to lower cases;
2. Separating "not" as a single word. For example: don't -> do not, won't -> will not.
3. Spelling correction. For example, amazzzzing -> amazing.
4. Removing punctuations
5. Lemmatization

For example, a sentence is originally like,

"It is amazzzzing! I can't believe that I fell asleep within mins!"

It would be preprocessed into:

"amazing not believe fell asleep within min"

Word 'not' is singled out because it is very important to distinguish the positive opinions and the negative opinions. All review text and summary collected for training-validation-testing will be preprocessed in this way.

Test-training split. As indicated earlier, I collected short reviews and summary sentences as the training-validation-testing data. However, the classes are unbalanced: there are 3755 positive opinions but only 777 negative opinions. This is consistent with what is observed for the whole datasets (Figure 2, 3). Therefore, I choose not to oversample the negative opinions or under-sample the positive ones. Rather, I did stratified sampling and do the training-testing split (70%/30%) separately for positive opinions and negative opinions.

Tf-idf model. Words and n-grams (2, 3,4,5 grams) are extracted as features from the text data. I use TfidfVectorizer from sklearn to calculate the Tfidf matrix of the words and n-grams. Tfidf is the frequency of a word/ngram in a particular sentence adjusted by its frequency in all sentences. For example, if a word appears once in a sentence and in total five times from all sentences, its tfidf value for that sentence is 1/5=0.2. By contrast, if another word appears once in this sentence but in total 100 times, its tfidf value would be only 1/100=0.1. It reflects the probability of a word is associated with a sentence's topic instead of just randomly showing up in that sentence.

Each row of the TfIdf matrix corresponds to a sample sentence, while each column corresponds to a feature, which can be a single word or an n-gram. A value in ith row and jth column is the tfidf value of jth feature in ith sample.

Feature Selection. Not every word or term is valuable for the sentiment analysis. Only those that more likely to occur within one class than the other would help with the classification. I use chi square feature selection to select these useful features. It calculates the deviation from the condition that the occurrence of a feature is independent from the occurrence of a class. The higher the deviation is, the less likely a feature occurs independently from a class or more likely that they are associated with each other. For each class, the chi square feature selection technique calculates the extent to which a feature is associated with a class (degree of deviation), ranks such association from high to low and picks the top n features.

Pipeline of Tuning classifiers. I fit four classification models to the preprocessed text data and pick the one with the best performance. These models are Multinomial Naïve Bayes model, Random Forest, Support Vector Machine (SVM) and Gradient Boosting Model.

When fitting each of the model to the data, I construct a pipeline with stages:

Tfidf → Chi square feature selection → 5-fold cross validation

Table 1 lists the hyperparameters to tune at each stage of pipeline.

Table 1. Stages of pipeline, corresponding tuned hyperparameters and tuning range

Stages of pipeline	Hyperparameters to tune	Tuning range
Tfidf	n grams	1-3: single word, bigrams, and trigrams 1-4: single word to four grams 1-5: single word to five grams
Feature selection	Number of features to keep	100, 300, all
Classification (5-fold cross validation)		
Naïve Bayes	Smooth factor (alpha)	0.01, 0.1, 1, 10
Random Forest	Number of random trees used to aggregate the predictions	
	Maximum number of features used to grow a tree	20, log2(N), (N)^0.5 N= total number of features
	Maximum number of splits in a tree	8, 10, 40
SVM	Penalty coefficient of the error term	0.01, 0.1, 1, 10
	Kernel coefficient (gamma)	0.01, 0.1, 1, 10
	Kernel	'rbf', 'poly'
Gradient Boosting	Number of trees to assemble	50, 100, 500
	Max number of features used to grow a tree	20, log2(N), (N)^0.5 N= total number of features
	Minimum number of leaves on a tree	3, 10, 20
	Minimum number of split	3, 5, 10

Evaluating the classifier. The metrics to be evaluated are F1 score, precision, recall and accuracy. There metrics are calculated when the classifier is fit to the training data and testing data. The best classifier is one with high F1 score, precision, recall and accuracy and small gap between its performance on training data and on testing data, or in other words, least overfitting.

Below figure shows that performance of the four classifiers on training data and testing data when n grams range is set at 1-3, 1-4 and 1-5.

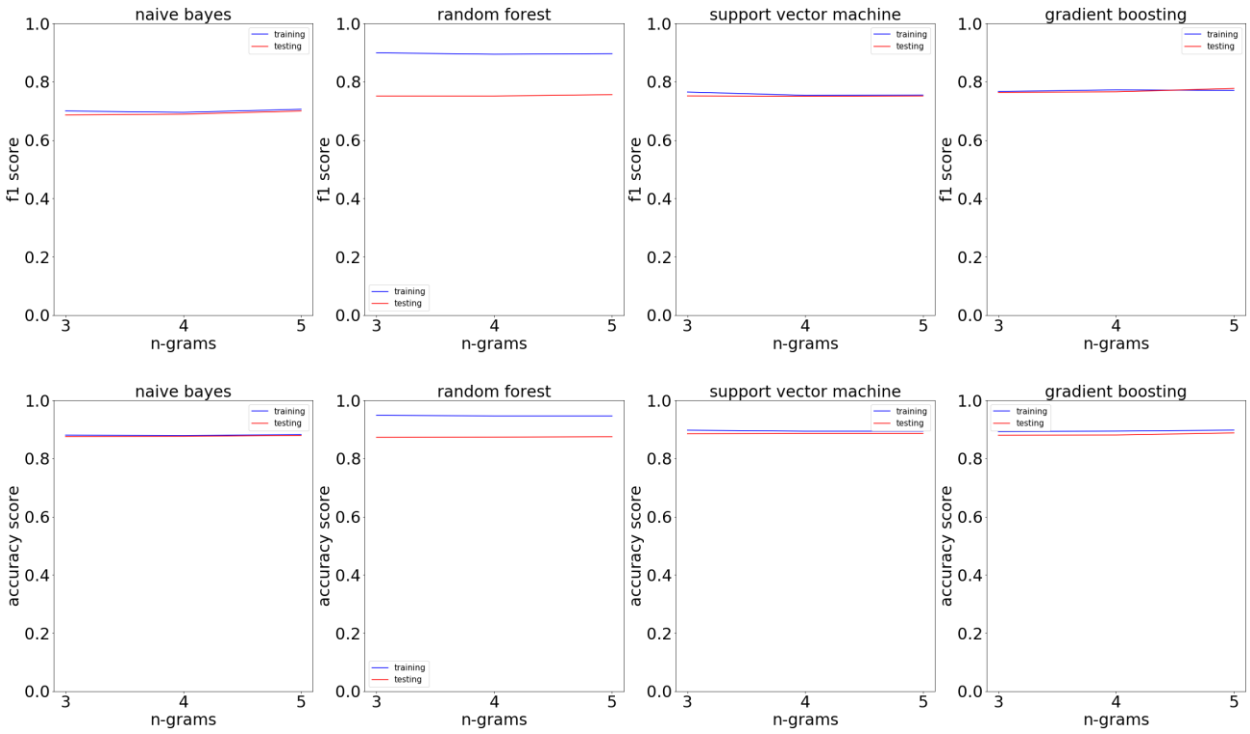


Figure 5. F1 score (upper) and accuracy score (lower) when fitting tuned Naïve Bayes, random forest, support vector machine or gradient boosting to training data (blue) or testing data (red) as n-gram range is set at 1-3, 1-4 or 1-5.

The above figure shows that overall SVM and Gradient Boosting perform better than Random Forest or Naive Bayes (where random forest has the poorest performance).

Compared to the other two classifiers, SVM and gradient boosting show more similar performance on testing score (red) and training score (blue), indicating less overfitting. Given the small size of training-validation data (about 17%*0.7=12% of the whole data), it is important that the model does not overfit.

Overall the score does not considerably change as ngram range increases. In order to avoid underfitting, I choose 5 grams. The difference between SVM and Gradient Boosting is small. Given that Gradient boosting classifier gives a higher recall (not shown here), it is preferred than SVM.

Figures below show the ROC curve of the Gradient Boosting classifier. The area under ROC curve is 0.82, considerably beyond the performance of a random classifier, 0.5.

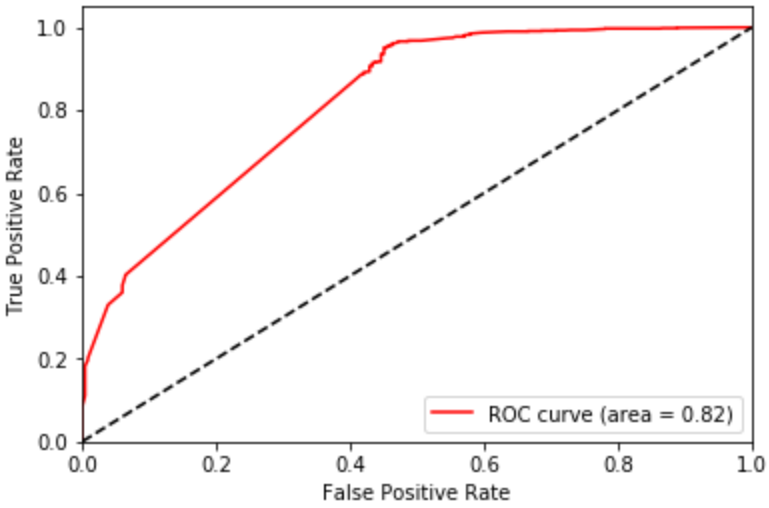


Figure 6. ROC curve of selected gradient boosting classifier on training data.

The average precision-recall curve also shows a good balance between these two metrics: precision only slightly decreases as recall increases, indicating it is quite robust to change in threshold.

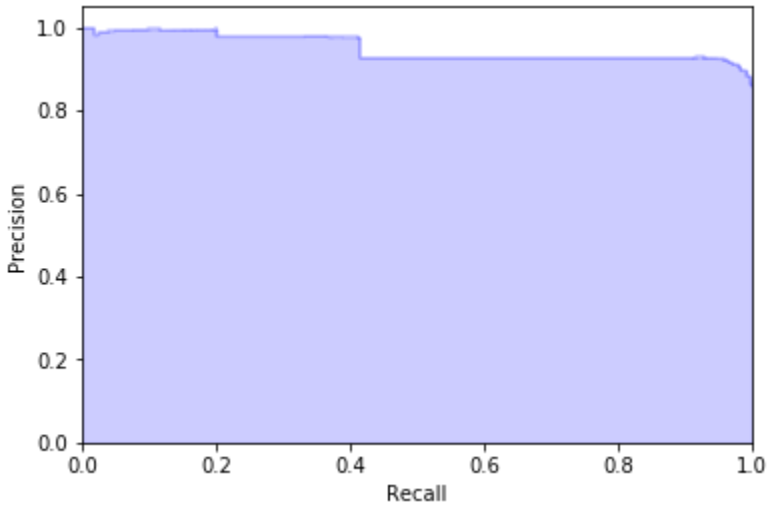


Figure 7. Precision-recall curve of selected gradient boosting classifier on training data.

How about these metrics for each class? This can be seen in the figure below.

```
the classification on training data are

confusion matrix:

[[ 242  302]
 [  22 2606]]
accuracy=0.8978562421185372

cohen kappa=0.5483991169481277

             precision    recall  f1-score   support

          0       0.92      0.44      0.60       544
          1       0.90      0.99      0.94      2628

avg / total       0.90      0.90      0.88      3172


the classification on test data are

confusion matrix:

[[ 122  111]
 [  39 1088]]
accuracy=0.8897058823529411

cohen kappa=0.5573051165980201

             precision    recall  f1-score   support

          0       0.76      0.52      0.62       233
          1       0.91      0.97      0.94      1127

avg / total       0.88      0.89      0.88      1360
```

Figure 8. Performance of Gradient Boosting Classifier on training data (upper) and on testing data (lower)

Firstly, recall is only high for class 1, which is the positive opinions but low for class 0, the negative opinions. This is true for both training data and testing data. It means that the classifier is better at detecting positive opinions than negative opinions (positive recall>negative recall). It also means that the classifier tends to misclassify a higher fraction of negative opinions than it does on positive opinions.

Secondly, precision of class 1/positive opinions is similarly high for training data and testing data. However, although precision of class 0 is only high for training data but considerably decreases when model fits to the testing data.

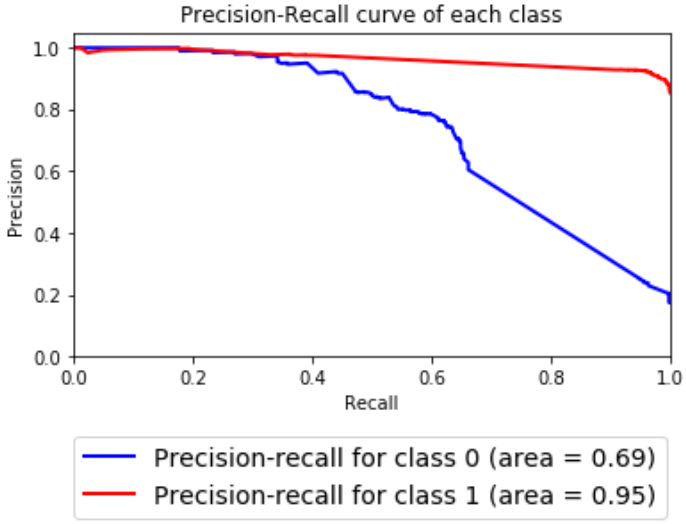


Figure 9. Performance of Gradient Boosting Classifier on positive opinions (red) of training data and on negative opinions of training data.

Above figure shows the precision-recall curve for each class when fitting the classifier to the training data. The red one is for positive opinions and blue is for negative opinions. It clearly shows that the classifier performs better on positive opinions than on negative opinions.

All suggest that the classifier performs better on the sampled positive opinions than on the sampled negative opinions. This is because of the unbalanced class: the positive opinions are 5 times of the negative opinions.

Custom Cutoff to Optimize Precision. The above figures show that this default cutoff probability, 0.5, does not work well to detect the negative opinions. Considering my aim is to identify the strong positive opinions and strong negative opinions, I want more custom cutoff values below which I am confident that the majority are negative opinions and above which I am also confident that the majority are positive ones. This means that precision is more important than recall. Therefore, the cutoff value is one at which the precision is relatively high for both classes.

However, there are two things to note before choosing the cutoffs. Firstly, it cannot be a single threshold. This is because that moving a single threshold always enhance the precision of one class with the cost of precision of other classes. Secondly, the tradeoff between precision and recall are quite different for the two classes (shown by Fig.9).

I choose the cutoff values by separately optimizing the F_β score for each class. Its formulation is,

$$F_\beta = (1+\beta)\frac{precision \cdot recall}{\beta \cdot precision + recall}$$

Parameter β is the ratio of the weight of recall relative to that of precision and can range from 0 to infinity. When $\beta = 1,\ F_\beta = F_1$. When choosing the cutoff values for positive opinions, I set this β at 0.1, which suggests that the importance of recall is 10% of that of precision. Then I find the threshold probability that optimizes F_β for positive opinions. When choosing cutoff values for negative opinions, I set weight of recall/weight of prediction ratio at 0.3, and find the threshold probability that optimizes F_β for negative opinions. Recall is given a higher weight for negative opinions because the classifier overall tends to misclassify more negative opinions than it does for positive opinions.

Finally, the cutoff probability of positive opinions is 0.91, which means that all sentences predicted score above 0.91 will be classified as being positive. The cutoff probability of negative opinions is 0.42, meaning that all sentences with predicted score lower than 0.42 will be classified as negative opinions. Sentences with predicted scores between 0.42 and 0.91 are those with weaker polarity or fewer key words/features. Classification on sentences in this range has a greater uncertainty than classification on sentences with more extreme predicted scores. I temporarily call them neutral opinions, which are not of interest in this project. However, this does not mean they are truly neutral, but means that this group of comments contain the majority of the neutral comments that I want to exclude from the analysis.

IV. Extracting Phrases and Representative Sentences

Classifying Unlabeled Data. The unlabeled data (as shown on the left side of the decision flowchart) are tokenized into sentences and fit to the tuned classifier. The sentences are classified as negative, neutral or positive based on rules described in last paragraph. Below figure shows the distribution of each type of opinions by rating in the unlabeled data.

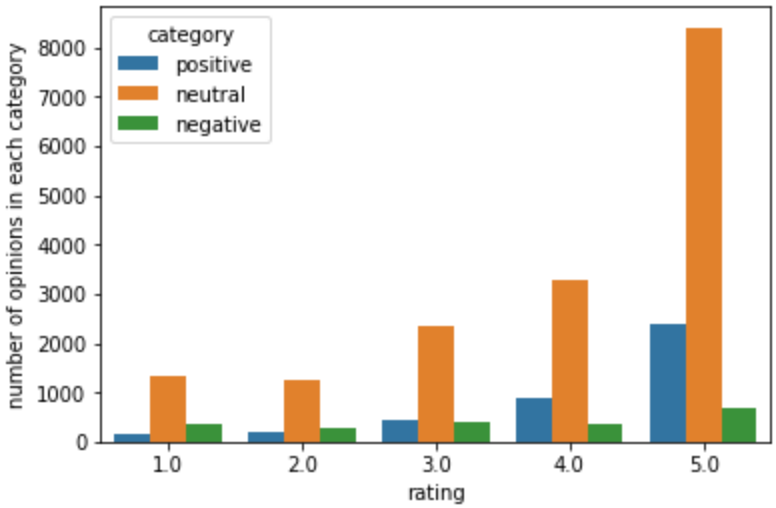


Figure 10. Distribution of predicted negative, neutral or positive opinions by rating from the unlabeled data.

It shows that for each rating the majority of sentences are classified as “neutral”. This is reasonable since most sentences would have intermediate polarity while a few others would belong to either of the two extremes. Below figure shows how the strong negative opinions and strong positive opinions distribute among ratings after I remove all neutral opinions:

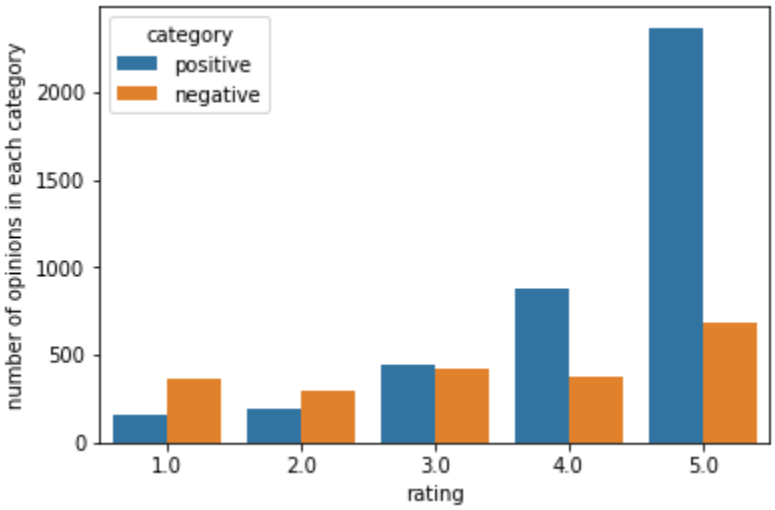


Figure 11. Distribution of predicted negative and positive opinions by rating from the unlabeled data.

The result looks reasonable: review text with higher rating have fewer negative comments (orange) and more positive comments (blue).

Extracting Meaningful Phrases. The goal of this project is to extract the features of the products that the users like or dislike from the negative and positive opinions. These features can be expressed in single words or phrases. Here I focus on phrases since compared to single words they contain clearer and richer information.

I focus on two words phrases, or bi-grams, such as “good sleep” or “bad taste”. A bigram is identified as a phrase for a class of opinions if it meets three criteria:

i. Two words appear in a bigram not by chance

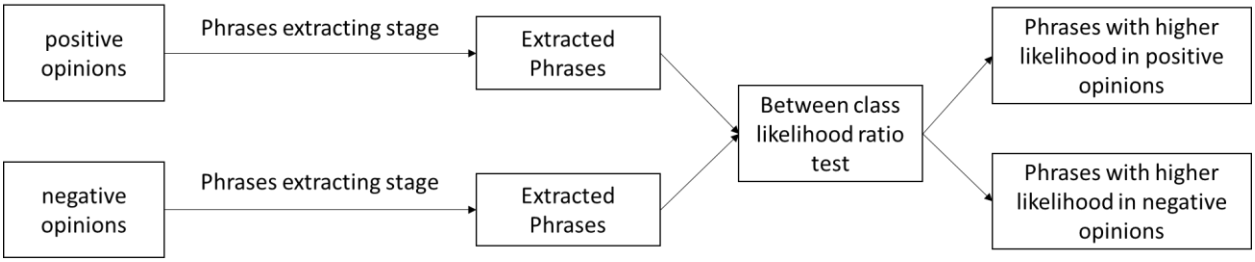
ii. The bigram preferentially occurs in one class of opinions than the other

iii. The bigram follows certain structures. For example:

adjective + noun, e.g. good sleep;
verb+noun, e.g. like taste;
verb+adverb, e.g. sleep well;
adverb+verb, quickly worked;
noun+noun, sleep time.

The diagram of how phrases are extracted from the raw sentences are shown below,

Overall work flow to extract phrases for two classes of opinions



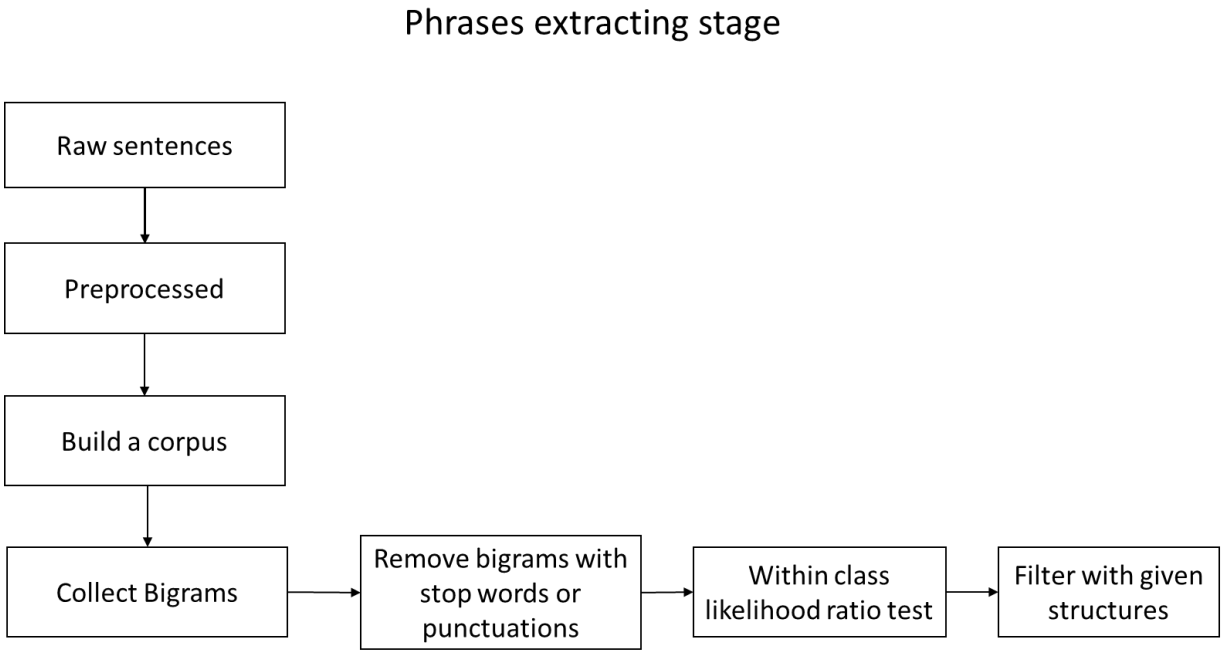


Figure 12. Diagram of workflow that extract meaningful phrases from raw sentences that are predicted or labeled as positive or negative opinions.

I start from the raw sentences which are either labeled or predicted as positive or negative opinions. All sentences classified as positive opinions are preprocessed in a similar way as that for sentiment analysis. However, punctuation and stop words are not removed. It is because I want to collect the true bigrams, where the two words do occur side by side in the sentence, rather than appearing together after the punctuations and stop words are removed.

From the preprocessed sentences, I collect all bigrams, which are all pairs of two consecutive words (including punctuations). Among these bigrams I select those that are most likely meaningful phrases.

I made such selection in three steps. Firstly, I remove all bigrams where there is at least one punctuation or one stop word. For example, bigrams such as ‘to sleep’ or ‘, and’ will be removed from the list of candidates. Secondly, a within class likelihood ratio test calculates the ratio of the probability that two words in the bigram are associated with each other over that when the two words occur together by chance. Thirdly, all bigrams are further filtered by certain structures that opinions are usually written in, such as verb+noun, adverb+verb, adjective+noun+noun etc. Bigrams with high likelihood ratio and fit the given structures or part of the given structures are selected as the candidates of meaningful phrases.

The phrase candidates for positive opinions and negative opinions are further compared with each other. Bigrams that preferentially occur in one class are saved as candidates for that class and removed for the other.

Here are the top 30 phrases (with highest within class likelihood ratio) identified for positive opinions:

Bigrams	Likelihood ratio
sleep mask	inf
fall asleep	3.246981e+215
doe not	7.644850e+160
dream water	1.111200e+159
work well	4.158263e+150
great product	1.323021e+150
work great	2.313019e+133
good night	1.596020e+128
ear plug	5.910845e+121
jet lag	1.146892e+87
doxylamine succinate	1.157488e+69
year old	2.807142e+63
side effect	1.883415e+62
mg sleep	3.214381e+61
next day	4.774278e+60
go sleep	8.225597e+58
time release	5.060111e+58
also sleep	3.014924e+58
eye sleep	2.554948e+58
much better	2.036825e+58

Here are the top 30 phrases (with highest within class likelihood ratio) identified for negative opinions:

Bigrams	Likelihood ratio
not work	inf
not help	4.594080e+70

Bigrams	Likelihood ratio
anything not	1.769477e+67
not not	3.918912e+65
side effects	2.840317e+56
taste not	4.025552e+55
night not	5.350249e+54
find not	1.617184e+54
strap not	1.054834e+53
think not	3.958870e+52
not like	1.248547e+52
thing not	9.385651e+49
not know	5.681983e+48
product not	5.695425e+47
well not	2.455697e+47
sleep aid	2.166788e+45
not sure	1.087599e+41
not notice	1.420797e+37
not feel	1.164237e+33
trying not	1.318783e+30
not use	7.833462e+22
simply not	1.032731e+22
not stay	1.453739e+21
not recommend	7.177878e+20

Finding the Representative Sentences. The representative sentences are ones that contain the meaningful phrases and act as summaries of previous users' experience. Therefore, they should be concise and readable. Therefore, I will extract the original raw sentences that contain the bigrams as listed above as the representative sentence. To do this, I first find the preprocessed sentences that contain any of the bigrams in the list and find the corresponding original raw sentence.

Taking two products as examples,

First example: Product "Biotab Nutraceuticals Alteril Sleep Aid with L-Tryptophan, Tablets 30 ea"

Positive opinions:

"Works Well for Our 8 Year Old"

Negative opinions:

"I've taken double and triple the recommended dosage and I don't feel a thing, not even a little sleepy. Don't waste your money on this one. "

Second example: "Dream Zone- Earth Therapeutics Sleep Mask, 1ct"

Positive comments:

"My son doesn't wake up at 5:30 a.m. anymore! Yea!... "
"works well... "
"When I take two tablets, they seem to work well. I'm able to get to sleep within 1/2 hour, and stay asleep for at least 4; many times, longer.... "

Negative opinions:

"Strap not comfortable..."
"This product in no way made me sleepy, not to say that it won't work for you.... "
"I hate to say this product did not work for me...."
"I'm not sure if it did or did not; I used for about 2-3 months then went back off.... "
"That's okay on an rare basis, but would not work nightly."