Monitoring Customer-Generated Product Image

Team roaster:

Xun Zha

Ao Pan

Henan Wang

Peiwei Li

Yanran Liu

Submitted To:

Prof. Kai Zhao

Nov. 2017

Monitoring Customer-Generated Product Image

Xun Zha, Ao Pan, Henan Wang, Peiwei Li, and Yanran Liu

Introduction

Customer reviews are becoming increasingly available in the E-commerce website like Amazon.com for a wide range of products and services (Mudambi & Schuff, 2010). As the number of product reviews grows rapidly, a popular product can obtain thousands of reviews in a short period. It is difficult for customers to read all of these reviews comprehensively before make purchases. Also, the huge amount of reviews makes the product tracking more difficult for manufacturers (Hu & Liu, 2004). Text mining tools and algorithms can help uncover customer attitudes and potential products issues and highlights from massive customer reviews for a particular product (Jack & Tsai, 2015).

Now, more and more data are stored digitally. People could publish their content and materials by better and advanced tools. We are witnessing more and more text data has been collected and published in various channels, such as e-books, blogs. So-called big data era not only enable people to collect more and more data through different forms but also provide a set of tools to analyze. In order to enhance the product features according to user-generated feedbacks, i.e., customer review, we apply text mining techniques to identify highlights and issues of each product. Understanding the overall positive and negative perceptions of a product enable manufacturers to be more responsiveness to the customers' feedbacks, to identify issues of products, and resolve issues uncovered (Jack & Tsai, 2015). In fact, after receiving all the reviews data from amazon's products, we could figure out these products' highlights and issues much easier through text mining techniques.

In this study, we present a framework for using text mining to gather customer feedback. The review text gives insight as to what contributes to product's overall rating, it is critical to understanding the top topics that customers are concerned for each product (Hu & Liu, 2004). In this study, we built topic models based on Amazon customer reviews to identify potential topics in each product. A topic model is a kind of a probabilistic generative model that has been used widely in the field of computer science with a specific focus on text mining and information retrieval in recent years. Since this model was first proposed, it has received a lot of attention and gained widespread interest among researchers in many research fields.

We tried to use LDA as our method to identify the potential topics embedded in the customer reviews of each product. There is a growing number of probabilistic models that are based on LDA via combination with particular tasks (Wikipedia). Nonetheless, most topic models have initially been introduced in the text analysis community for unsupervised topic discovery in a corpus of documents. In LDA, each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. LDA assumes that documents cover only a small set of topics and that topics use only a small set of words frequently. In such assumptions, LDA provides better disambiguation of words and a more precise assignment of documents to topics (Wikipedia).

Is There Any Related Work?

Social Media

Several topic modeling techniques have been proposed in the social media marketing industry. Most of these models are based on the Latent Dirichlet Allocation. More and more people are using social media such as Twitter and Facebook. So, large amount of data is available on these platforms presents new opportunities for mining information about the real world. Besides, it can be also used in several ways to gather information about the world, such as trending topics, breaking news and popular events. Since a majority of people are using smart phones to access these platforms, many of data available on those is geotagged. This information can be useful to determine various location-specific aspects around the world.

Potential Contributions

The use of text analytics in real-world e-commerce platform is growing very fast as the unstructured data boomed. The use of text analytics will likely continue to grow exponentially in the future. People are listening what we say, so our time is not wasted go reviews the product so we could get better product or service.

Our project introduces a method to use text mining to help customers efficiently understand comments and feedbacks listed on the e-commerce platform. People who use our model can quickly extract the crucial information that they are interested in from the products comments. This mythology meets current customers' need by make customers efficiently know what the comments of the potentially purchased products are saying about.

Also, our model can help customers figure out which product is the best seller, which product contains the most words of reviews and which product has the best price. Our model can be used to compare a variety of products by their features, score of comments and price. Our model can be also used by the product manager, who can make improvement on the low scored products before a new product launch to the market.

This project also introduces an automatic method to reflect which context in which this review was written. The identification of such information is not an easy task. By make further improvement on our model, every customer can have his/her own thin client text analytics application to analyze the huge volumes of product reviews, knows the pros and cons.

Project Design

This project uses a data set of reviews on Amazon to answer our research question. This section elaborates our research work and consists of three parts. First, we describe our review data over product IDs and over ratings. Second, we explain, step by step, how we used topic modeling techniques to mine highlights and issues of selected products. Third, we interpret the results of our topic modeling.

Data Description

In our data set, there are 2038 reviews for 88 products with product IDs from 1 to 88. Table 1 shows the numbers and percentages of reviews over ratings. Furthermore, by examining the review distribution over ratings for each product, we found that five-star reviews dominate other reviews for each of the products (for details, see review-count results of grouping reviews by "ProductID" and "ReviewRating").

Table 1. A Summary of Review Distribution over Ratings

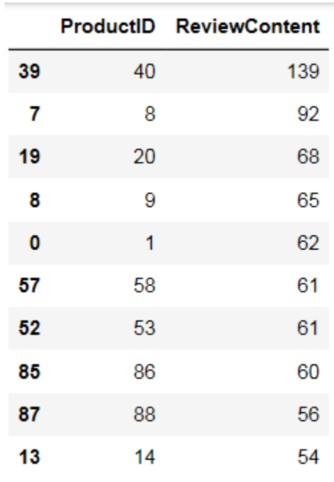
Review Rating	Number of Reviews	Percentage
1.00	113	5.54%
2.00	56	2.75%
3.00	106	5.20%
4.00	239	11.73%
5.00	1524	74.78%
Total	2038	100.00%

Topic Modeling for Highlights and Issues of Products

Data extraction

Since the reviews over ratings are very unevenly distributed, we included only a subset of data into our topic modeling in order to make sure that we have considerable data for both positive and negative aspects (i.e., highlights and issues) of products. Specifically, we ranked all the products by their numbers of reviews and included only review data for top ten products into our analysis. Those ten products have relatively considerable number of reviews in both high-level stars and low-level stars. Table 2 shows a list of the top ten products and their corresponding numbers of reviews. The largest number of reviews is 139 for Product 40. The smallest number of reviews is 54 for Product 13. The mean number of reviews for all selected products is 71.8.

Table 2. Review Counts for Products



	ProductID	ReviewContent
39	40	139
7	8	92
19	20	68
8	9	65
0	1	62
57	58	61
52	53	61
85	86	60
87	88	56
13	14	54

Although we selected products with top-ten largest number of reviews, the reviews of those selected products are still apparently skewed to five-star, as shown in Figure 1. Figure 2 further, uses a pie chart to illustrate the coverage of each rating for Product 40, which has the largest number of reviews. Five-star reviews for this product cover 80.6 percent of its reviews in total.

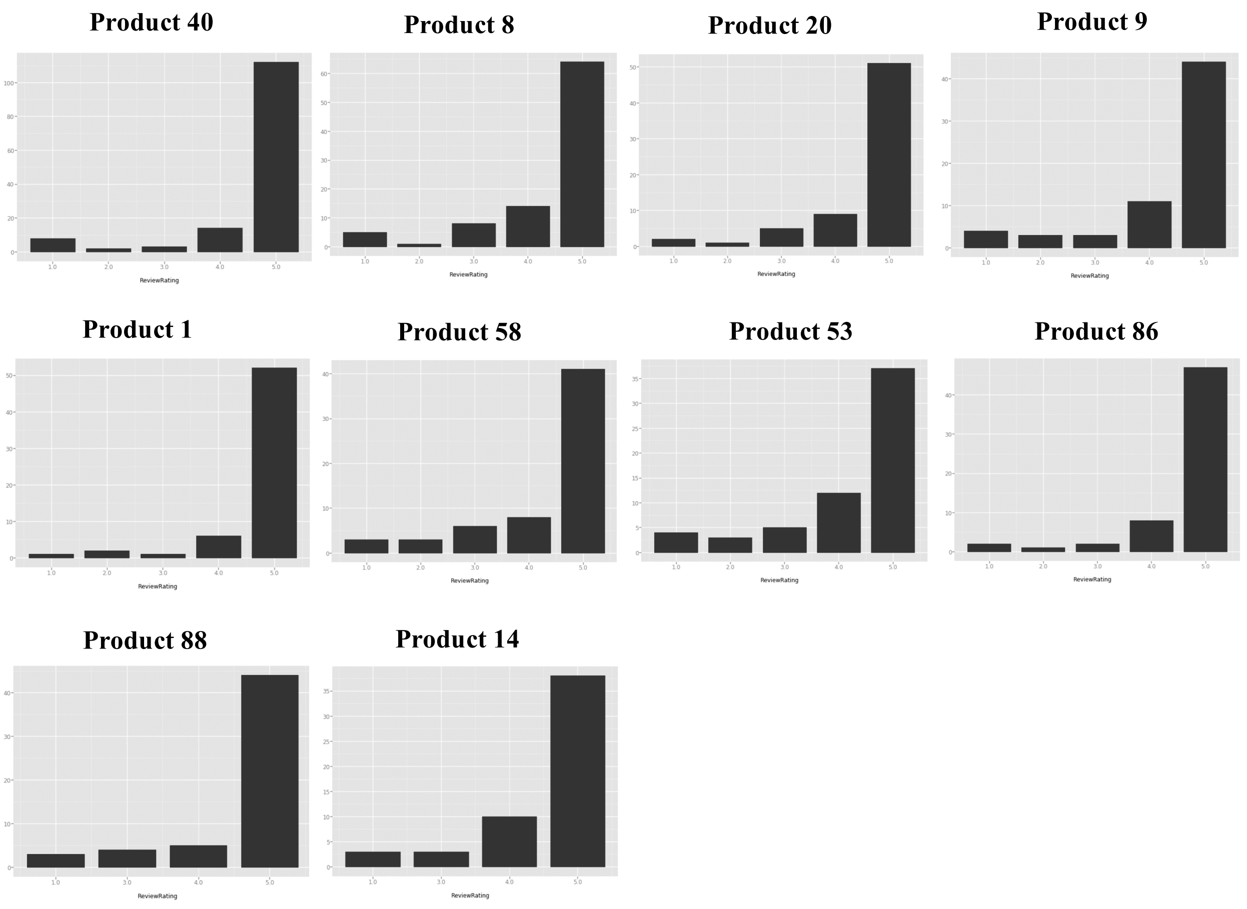


Figure 1. Number of Reviews over Ratings of Selected Products

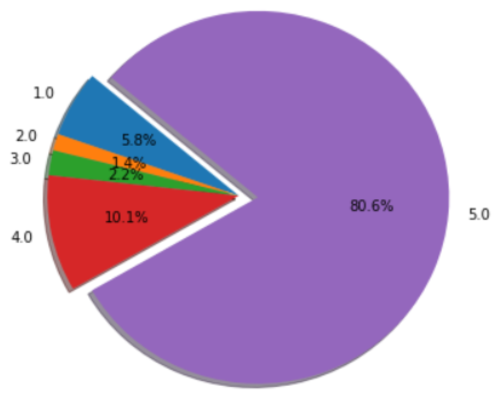


Figure 2. Percentage of Reviews for Each Rating of Product 40

Figure 3. A Presentation of Selected Products

Therefore, in order to get enough data for analyzing issues, we aggregated one-star and two-star reviews for mining user-generated issues. The aggregation is reasonable because 2-star reviews are quite negative and reviewers talk extensively about issues of a product. However, we used only five-star reviews for mining user-generated highlights. The main reason is that five-star reviews talk unequivocally about highlights and the number of five-star reviews for each product is already large enough to do analysis. We do not need to include 4-star reviews, which occasionally talk about issues.

Construct the good and bad text data. We called the review data for mining highlights of products "good text data" and the review data for mining issues of products "bad text data". The good text data are five-star reviews, and the aggregated five-star review text for each selected product is regarded as one document. The bad text data are one-star and two-star reviews, and the aggregated one-star as well as two-star review text for each selected product is regarded as one document. Thus, for either good or bad text data, we have ten documents.

Data cleaning

We cleaned data in five steps. First, we made all the words lowercase and stripped away the punctuations. Second, we kept only the adjective, noun, adverb, interjection, and verb words because those words are much more meaningful for describing highlights and issues of products than others, such as pronouns. Third, we tokenized and lemmatized the texts. We did not stem because lemmatizing could keep the integrity of words and make words easy to be understood, smoothing our interpretation of topic-modeling results.

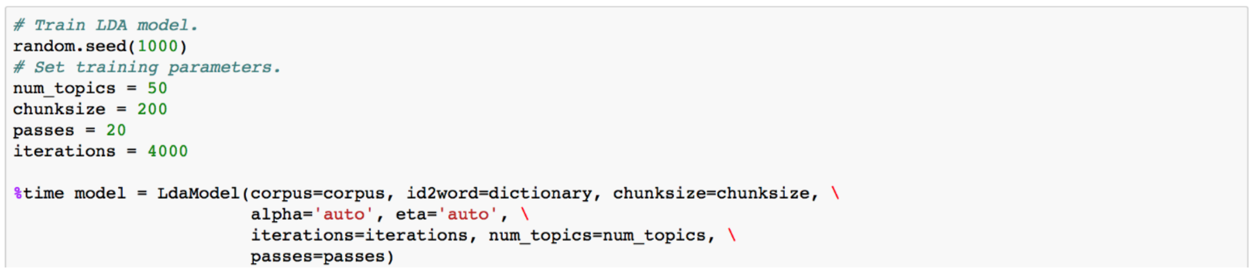
Fourth, we computed bigrams in the texts. Bigrams are sets of two adjacent words. We can get phrases like "machine_learning" (in which a space is replaced with underscores) in our output by using bigrams. Without bigrams, we would only get "machine" and "learning", which were much more ambiguous and meaningless than the phrase that combined the two words together. Note that in our algorithm, we found the phrases in the original data and then, replaced them with bigrams, so that we could keep, for instance, both the separated "machine" as well as "learning" and the phrase "machine learning" (replaced as "machine_learning" after bigram computing). Fifth, we removed the stop words.

Model training

After we cleaned the data, we are ready to train the LDA model. First, how many topics we need? In case that we may have some similar products in our corpus, we decided to use 50 topics here because we wanted to have a more concise topic for each product that we could interpret, and because that turned out to give me reasonably good results.

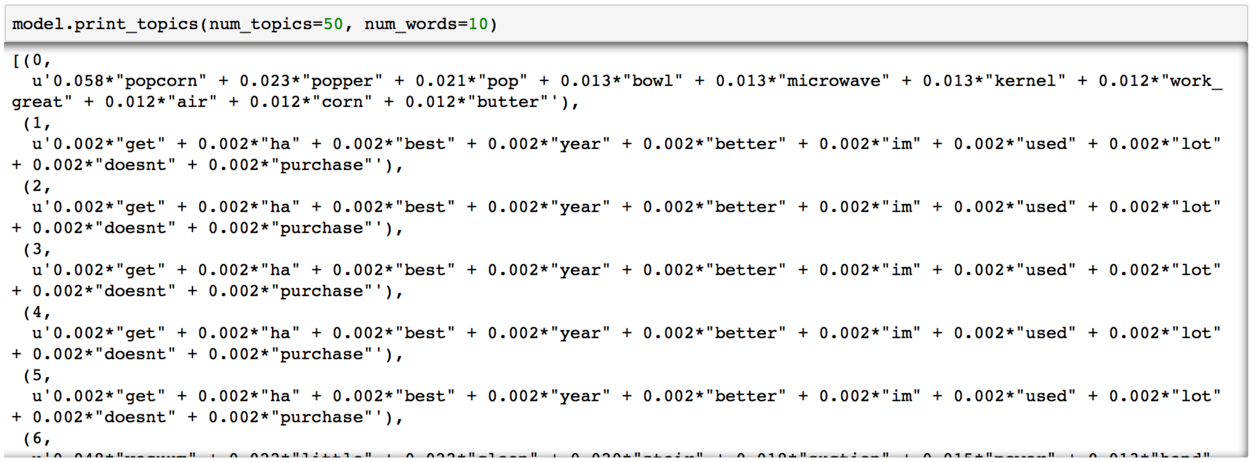
The “chunksize” controls how many documents are processed at a time in the training algorithm. We have set “chunksieze” = 200, which is more than the number of documents, so I process all the data in one go. We used the 20, 4000 for the “passs” and “iterations” parameters. As we know, It is important to set the number of "passes" and "iterations" high enough.

After training the model for the first time and analyzing our results, we found that there are a lot of words which are meaningless for each product.



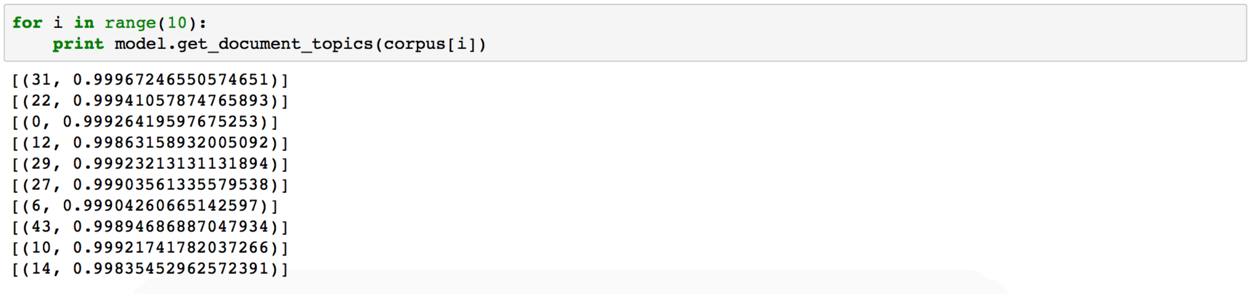
```
# Train LDA model.
random.seed(1000)
# Set training parameters.
num_topics = 50
chunksize = 200
passes = 20
iterations = 4000

%time model = LdaModel(corpus=corpus, id2word=dictionary, chunksize=chunksize, \
                       alpha='auto', eta='auto', \
                       iterations=iterations, num_topics=num_topics, \
                       passes=passes)
```



```
model.print_topics(num_topics=50, num_words=10)
```

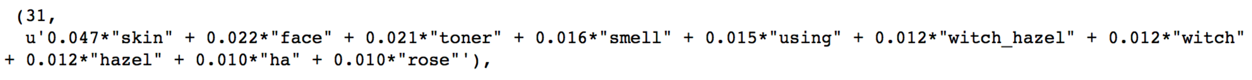
```
[(0,
  u'0.058*"popcorn" + 0.023*"popper" + 0.021*"pop" + 0.013*"bowl" + 0.013*"microwave" + 0.013*"kernel" + 0.012*"work_
great" + 0.012*"air" + 0.012*"corn" + 0.012*"butter"'),
 (1,
  u'0.002*"get" + 0.002*"ha" + 0.002*"best" + 0.002*"year" + 0.002*"better" + 0.002*"im" + 0.002*"used" + 0.002*"lot"
+ 0.002*"doesnt" + 0.002*"purchase"'),
 (2,
  u'0.002*"get" + 0.002*"ha" + 0.002*"best" + 0.002*"year" + 0.002*"better" + 0.002*"im" + 0.002*"used" + 0.002*"lot"
+ 0.002*"doesnt" + 0.002*"purchase"'),
 (3,
  u'0.002*"get" + 0.002*"ha" + 0.002*"best" + 0.002*"year" + 0.002*"better" + 0.002*"im" + 0.002*"used" + 0.002*"lot"
+ 0.002*"doesnt" + 0.002*"purchase"'),
 (4,
  u'0.002*"get" + 0.002*"ha" + 0.002*"best" + 0.002*"year" + 0.002*"better" + 0.002*"im" + 0.002*"used" + 0.002*"lot"
+ 0.002*"doesnt" + 0.002*"purchase"'),
 (5,
  u'0.002*"get" + 0.002*"ha" + 0.002*"best" + 0.002*"year" + 0.002*"better" + 0.002*"im" + 0.002*"used" + 0.002*"lot"
+ 0.002*"doesnt" + 0.002*"purchase"'),
 (6,
  u'0.048*"vacuum" + 0.022*"little" + 0.022*"clean" + 0.020*"stair" + 0.018*"suction" + 0.015*"power" + 0.012*"hand"
```



```
for i in range(10):
    print model.get_document_topics(corpus[i])
```

```
[(31, 0.99967246550574651)]
[(22, 0.99941057874765893)]
[(0, 0.99926419597675253)]
[(12, 0.99863158932005092)]
[(29, 0.99923213131131894)]
[(27, 0.99903561335579538)]
[(6, 0.99904260665142597)]
[(43, 0.99894686887047934)]
[(10, 0.99921741782037266)]
[(14, 0.99835452962572391)]
```

For example, for topics 31 which highly corresponded with product 40, we found some words, such as “ha,” “using,” are not related to the product very well.

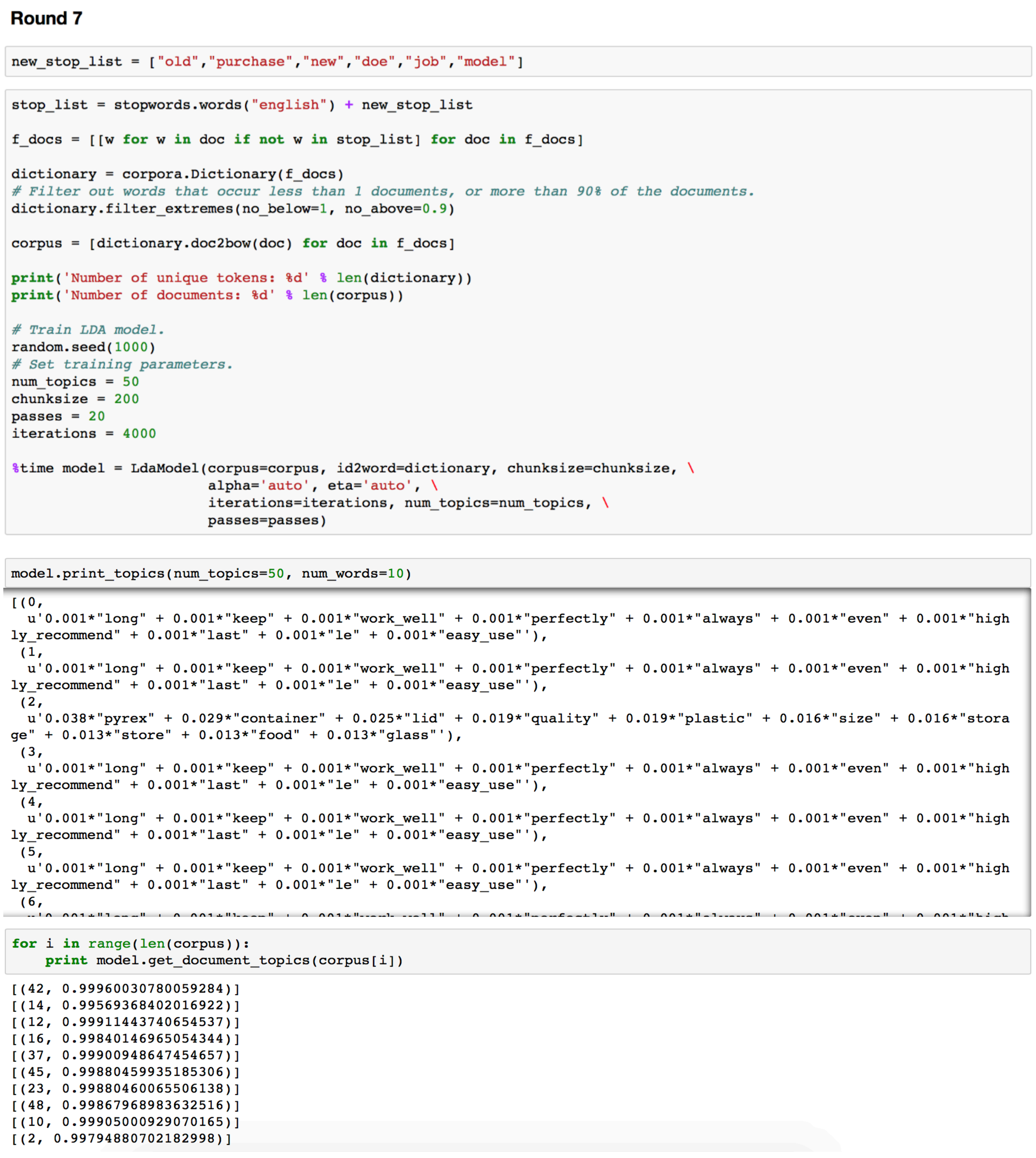


```
 (31,
  u'0.047*"skin" + 0.022*"face" + 0.021*"toner" + 0.016*"smell" + 0.015*"using" + 0.012*"witch_hazel" + 0.012*"witch"
+ 0.012*"hazel" + 0.010*"ha" + 0.010*"rose"'),
```

Therefore, we decided to add these meaningless words to the stop words list, then repeat the process of model training to improve our results.

Model improvement

As mentioned in previous part, we repeatedly trained our model to improve the accuracy by appending the meaningless word to the stop word list. For the better results, we entirely retrained our model 7 times.



Round 7

```
new_stop_list = ["old","purchase","new","doe","job","model"]
```

```
stop_list = stopwords.words("english") + new_stop_list

f_docs = [[w for w in doc if not w in stop_list] for doc in f_docs]

dictionary = corpora.Dictionary(f_docs)
# Filter out words that occur less than 1 documents, or more than 90% of the documents.
dictionary.filter_extremes(no_below=1, no_above=0.9)

corpus = [dictionary.doc2bow(doc) for doc in f_docs]

print('Number of unique tokens: %d' % len(dictionary))
print('Number of documents: %d' % len(corpus))

# Train LDA model.
random.seed(1000)
# Set training parameters.
num_topics = 50
chunksize = 200
passes = 20
iterations = 4000

%time model = LdaModel(corpus=corpus, id2word=dictionary, chunksize=chunksize, \
                       alpha='auto', eta='auto', \
                       iterations=iterations, num_topics=num_topics, \
                       passes=passes)
```

```
model.print_topics(num_topics=50, num_words=10)
```

```
[(0,
  u'0.001*"long" + 0.001*"keep" + 0.001*"work_well" + 0.001*"perfectly" + 0.001*"always" + 0.001*"even" + 0.001*"high
ly_recommend" + 0.001*"last" + 0.001*"le" + 0.001*"easy_use"'),
 (1,
  u'0.001*"long" + 0.001*"keep" + 0.001*"work_well" + 0.001*"perfectly" + 0.001*"always" + 0.001*"even" + 0.001*"high
ly_recommend" + 0.001*"last" + 0.001*"le" + 0.001*"easy_use"'),
 (2,
  u'0.038*"pyrex" + 0.029*"container" + 0.025*"lid" + 0.019*"quality" + 0.019*"plastic" + 0.016*"size" + 0.016*"stora
ge" + 0.013*"store" + 0.013*"food" + 0.013*"glass"'),
 (3,
  u'0.001*"long" + 0.001*"keep" + 0.001*"work_well" + 0.001*"perfectly" + 0.001*"always" + 0.001*"even" + 0.001*"high
ly_recommend" + 0.001*"last" + 0.001*"le" + 0.001*"easy_use"'),
 (4,
  u'0.001*"long" + 0.001*"keep" + 0.001*"work_well" + 0.001*"perfectly" + 0.001*"always" + 0.001*"even" + 0.001*"high
ly_recommend" + 0.001*"last" + 0.001*"le" + 0.001*"easy_use"'),
 (5,
  u'0.001*"long" + 0.001*"keep" + 0.001*"work_well" + 0.001*"perfectly" + 0.001*"always" + 0.001*"even" + 0.001*"high
ly_recommend" + 0.001*"last" + 0.001*"le" + 0.001*"easy_use"'),
 (6,
```

```
for i in range(len(corpus)):
    print model.get_document_topics(corpus[i])
```

```
[(42, 0.99960030780059284)]
[(14, 0.99569368402016922)]
[(12, 0.99911443740654537)]
[(16, 0.99840146965054344)]
[(37, 0.99900948647454657)]
[(45, 0.99880459935185306)]
[(23, 0.99880460065506138)]
[(48, 0.99867968983632516)]
[(10, 0.99905000929070165)]
[(2, 0.99794880702182998)]
```

```
 (42,
  u'0.056*"skin" + 0.026*"face" + 0.025*"toner" + 0.019*"smell" + 0.014*"witch_hazel" + 0.012*"rose" + 0.011*"bottle"
 + 0.009*"dry" + 0.009*"gentle" + 0.009*"acne"'),
```

As the results shown above, in the 7th training, the topic for product 40 has been much more decent. However, all these processes are just for the good reviews. For next step, we have to redo the whole process for the bad reviews to find the potential issues for each product.

Redo the above process for bad reviews

Bad

```
bad_text
```

...

```
cleaned_text = clean_text(bad_text)
```

```
tagged_text = tag_text(cleaned_text)
```

```
docs = token_lemma(tagged_text)
```

```
bi_docs = add_bigram(docs)
```

```
stop_list = stopwords.words("english")
```

```
f_docs = [[w for w in doc if not w in stop_list] for doc in bi_docs]
```

Round 3

```
new_stop_list = ["ever","come","much","lot","grinding","grinder","bought","go"]
stop_list = stopwords.words("english") + new_stop_list
f_docs = [[w for w in doc if not w in stop_list] for doc in f_docs]
dictionary = corpora.Dictionary(f_docs)
# Filter out words that occur less than 1 documents, or more than 90% of the documents.
dictionary.filter_extremes(no_below=1, no_above=0.8)

corpus = [dictionary.doc2bow(doc) for doc in f_docs]

print('Number of unique tokens: %d' % len(dictionary))
print('Number of documents: %d' % len(corpus))

# Train LDA model.
random.seed(1000)
# Set training parameters.
num_topics = 50
chunksize = 200
passes = 20
iterations = 4000

%time model = LdaModel(corpus=corpus, id2word=dictionary, chunksize=chunksize, \
                       alpha='auto', eta='auto', \
                       iterations=iterations, num_topics=num_topics, \
                       passes=passes)
model.print_topics(num_topics=50, num_words=10)
```

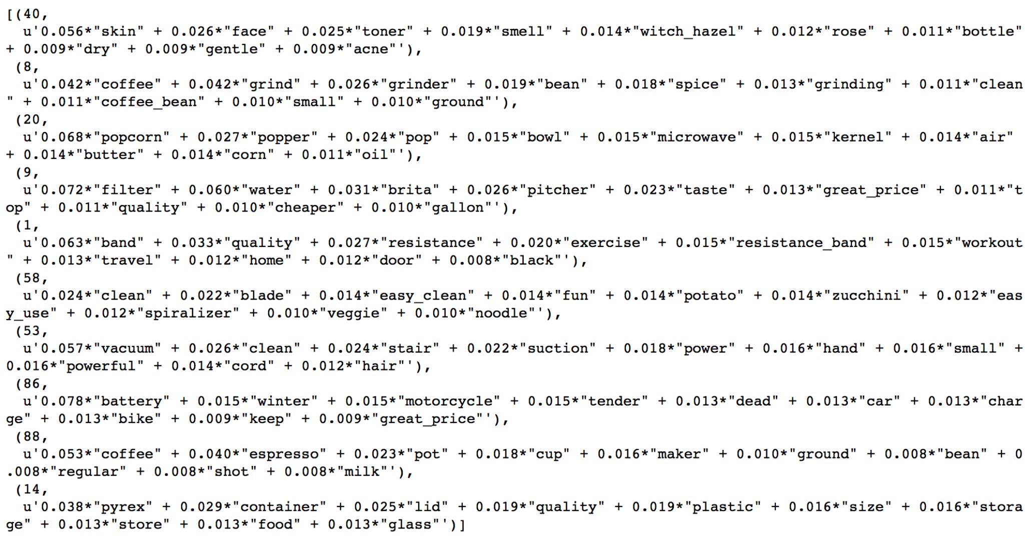
```
Number of unique tokens: 728
Number of documents: 10
CPU times: user 2.03 s, sys: 14.8 ms, total: 2.04 s
Wall time: 2.05 s
```

```
[(0,
  u'0.073*"popcorn" + 0.025*"quality" + 0.025*"smell" + 0.025*"disappointed" + 0.025*"weird" + 0.025*"worst" + 0.025*
"bag" + 0.025*"dint" + 0.025*"corn" + 0.025*"threw"'),
 (1,
  u'0.003*"year" + 0.003*"time" + 0.003*"something" + 0.002*"dont" + 0.002*"buying" + 0.002*"look" + 0.002*"old" + 0.
002*"worked" + 0.002*"thought" + 0.002*"still"'),
 (2,
  u'0.003*"year" + 0.003*"time" + 0.003*"something" + 0.002*"buying" + 0.002*"dont" + 0.002*"look" + 0.002*"worked" +
0.002*"thought" + 0.002*"old" + 0.002*"still"'),
 (3,
  u'0.003*"year" + 0.003*"time" + 0.003*"something" + 0.002*"buying" + 0.002*"dont" + 0.002*"look" + 0.002*"worked" +
0.002*"thought" + 0.002*"old" + 0.002*"still"'),
 (4,
  u'0.003*"year" + 0.003*"time" + 0.003*"something" + 0.002*"buying" + 0.002*"dont" + 0.002*"look" + 0.002*"worked" +
0.002*"thought" + 0.002*"old" + 0.002*"still"'),
```

After three rounds training, we found the outcomes are not very well. We believe the primary reason for that is the sample size is so small for bad reviews. Besides, we noticed people tend to just add a negation word before a positive word which is also a big challenge for us when we were trying to find the issues of a product. We will talk about our results in next part comprehensively.

Results Interpretation

Final results for good reviews(Highlight):



```
[(40,
  u'0.056*"skin" + 0.026*"face" + 0.025*"toner" + 0.019*"smell" + 0.014*"witch_hazel" + 0.012*"rose" + 0.011*"bottle"
+ 0.009*"dry" + 0.009*"gentle" + 0.009*"acne"'),
 (8,
  u'0.042*"coffee" + 0.042*"grind" + 0.026*"grinder" + 0.019*"bean" + 0.018*"spice" + 0.013*"grinding" + 0.011*"clean
" + 0.011*"coffee_bean" + 0.010*"small" + 0.010*"ground"'),
 (20,
  u'0.068*"popcorn" + 0.027*"popper" + 0.024*"pop" + 0.015*"bowl" + 0.015*"microwave" + 0.015*"kernel" + 0.014*"air"
+ 0.014*"butter" + 0.014*"corn" + 0.011*"oil"'),
 (9,
  u'0.072*"filter" + 0.060*"water" + 0.031*"brita" + 0.026*"pitcher" + 0.023*"taste" + 0.013*"great_price" + 0.011*"t
op" + 0.011*"quality" + 0.010*"cheaper" + 0.010*"gallon"'),
 (1,
  u'0.063*"band" + 0.033*"quality" + 0.027*"resistance" + 0.020*"exercise" + 0.015*"resistance_band" + 0.015*"workout
" + 0.013*"travel" + 0.012*"home" + 0.012*"door" + 0.008*"black"'),
 (58,
  u'0.024*"clean" + 0.022*"blade" + 0.014*"easy_clean" + 0.014*"fun" + 0.014*"potato" + 0.014*"zucchini" + 0.012*"eas
y_use" + 0.012*"spiralizer" + 0.010*"veggie" + 0.010*"noodle"'),
 (53,
  u'0.057*"vacuum" + 0.026*"clean" + 0.024*"stair" + 0.022*"suction" + 0.018*"power" + 0.016*"hand" + 0.016*"small" +
0.016*"powerful" + 0.014*"cord" + 0.012*"hair"'),
 (86,
  u'0.078*"battery" + 0.015*"winter" + 0.015*"motorcycle" + 0.015*"tender" + 0.013*"dead" + 0.013*"car" + 0.013*"char
ge" + 0.013*"bike" + 0.009*"keep" + 0.009*"great_price"'),
 (88,
  u'0.053*"coffee" + 0.040*"espresso" + 0.023*"pot" + 0.018*"cup" + 0.016*"maker" + 0.010*"ground" + 0.008*"bean" + 0
.008*"regular" + 0.008*"shot" + 0.008*"milk"'),
 (14,
  u'0.038*"pyrex" + 0.029*"container" + 0.025*"lid" + 0.019*"quality" + 0.019*"plastic" + 0.016*"size" + 0.016*"stora
ge" + 0.013*"store" + 0.013*"food" + 0.013*"glass"')]
```

This is the good (5 stars) outcome from our topic modeling.

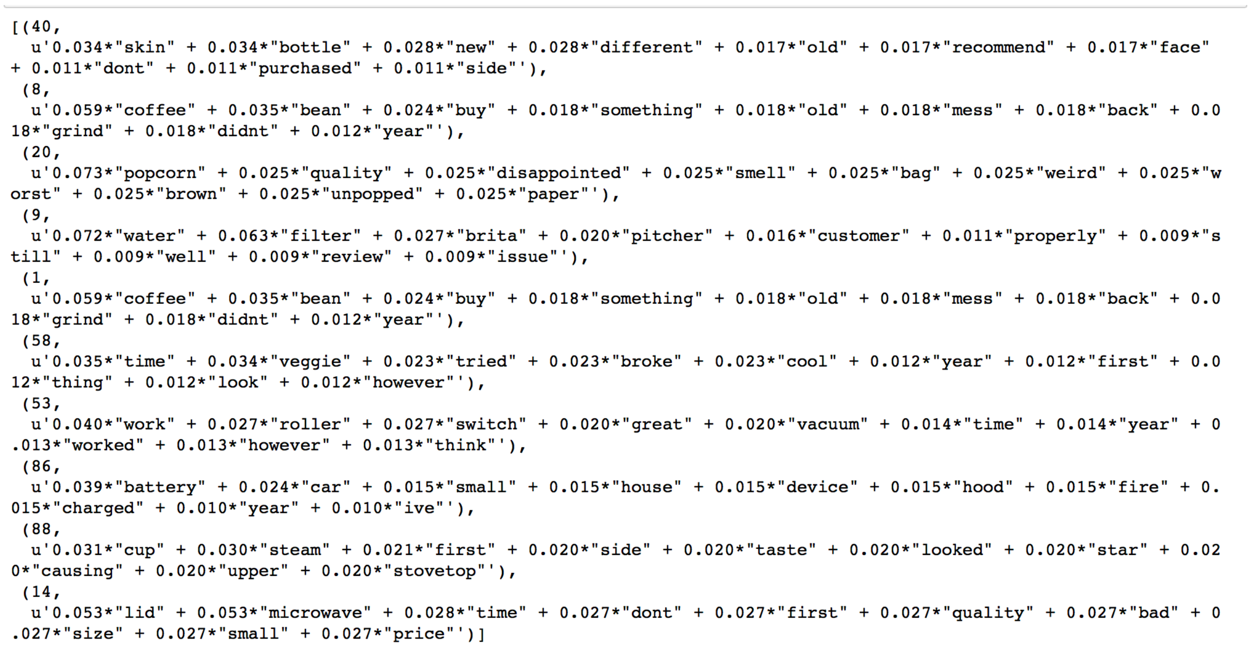
Product ID	Product Name	Pros
40	Toner	Smell good. Witch hazel. Good for Acne. Good for skin
8	Grinder	Easy Clean. Small.
20	Popcorn Popper	Can put in butter and oil, Better than microwave
9	Brita Pitcher	High quality, Cheap, Tasty water
1	Resistance Band	High quality, Bring to travel, Good for exercise
58	Spiralizer	Easy Clean, Can make for potato and zucchini noodle
53	Vacuum	Easy Clean, Powerful
86	Battery	Use for winter. Can charge for Car, Motorcycle

88	Coffee Peculator	Can make espresso
14	Pyrex Container	High quality, Glass

We grouped 1 and 2-stars reviews as our data for bad results, and only 5-star review as our data for our good result interpretation. The reason we gave up 3 and 4-stars is that most of people in this range would leave very neutral statements. For example, this is not a very great product, but the quality worth the price.

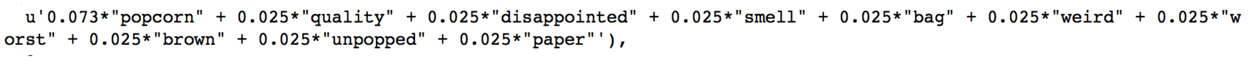
Through our analysis, we could figure out that the Product 40 is Toner. It smells really good and can resolve the acne skin problems. The other good example is Vacuum. We could see the that product is easy clean, very handy and powerful. Besides, the Product 14 is Pyrex Container. It has the glass body materials with a very good quality.

Final results for bad reviews(Issues):



```
[(40,
  u'0.034*"skin" + 0.034*"bottle" + 0.028*"new" + 0.028*"different" + 0.017*"old" + 0.017*"recommend" + 0.017*"face"
 + 0.011*"dont" + 0.011*"purchased" + 0.011*"side"'),
 (8,
  u'0.059*"coffee" + 0.035*"bean" + 0.024*"buy" + 0.018*"something" + 0.018*"old" + 0.018*"mess" + 0.018*"back" + 0.0
18*"grind" + 0.018*"didnt" + 0.012*"year"'),
 (20,
  u'0.073*"popcorn" + 0.025*"quality" + 0.025*"disappointed" + 0.025*"smell" + 0.025*"bag" + 0.025*"weird" + 0.025*"w
orst" + 0.025*"brown" + 0.025*"unpopped" + 0.025*"paper"'),
 (9,
  u'0.072*"water" + 0.063*"filter" + 0.027*"brita" + 0.020*"pitcher" + 0.016*"customer" + 0.011*"properly" + 0.009*"s
till" + 0.009*"well" + 0.009*"review" + 0.009*"issue"'),
 (1,
  u'0.059*"coffee" + 0.035*"bean" + 0.024*"buy" + 0.018*"something" + 0.018*"old" + 0.018*"mess" + 0.018*"back" + 0.0
18*"grind" + 0.018*"didnt" + 0.012*"year"'),
 (58,
  u'0.035*"time" + 0.034*"veggie" + 0.023*"tried" + 0.023*"broke" + 0.023*"cool" + 0.012*"year" + 0.012*"first" + 0.0
12*"thing" + 0.012*"look" + 0.012*"however"'),
 (53,
  u'0.040*"work" + 0.027*"roller" + 0.027*"switch" + 0.020*"great" + 0.020*"vacuum" + 0.014*"time" + 0.014*"year" + 0
.013*"worked" + 0.013*"however" + 0.013*"think"'),
 (86,
  u'0.039*"battery" + 0.024*"car" + 0.015*"small" + 0.015*"house" + 0.015*"device" + 0.015*"hood" + 0.015*"fire" + 0.
015*"charged" + 0.010*"year" + 0.010*"ive"'),
 (88,
  u'0.031*"cup" + 0.030*"steam" + 0.021*"first" + 0.020*"side" + 0.020*"taste" + 0.020*"looked" + 0.020*"star" + 0.02
0*"causing" + 0.020*"upper" + 0.020*"stovetop"'),
 (14,
  u'0.053*"lid" + 0.053*"microwave" + 0.028*"time" + 0.027*"dont" + 0.027*"first" + 0.027*"quality" + 0.027*"bad" + 0
.027*"size" + 0.027*"small" + 0.027*"price"')]
```

The best result for issues is the outcome for Product 20.



```
  u'0.073*"popcorn" + 0.025*"quality" + 0.025*"disappointed" + 0.025*"smell" + 0.025*"bag" + 0.025*"weird" + 0.025*"w
orst" + 0.025*"brown" + 0.025*"unpopped" + 0.025*"paper"'),
```

The bad reviews result example is Product 20. We could see this Popper machine has some unpopped popcorn. And some of their customers were also disappointed with the quality. Since we have 500 more good reviews and only about 50 bad reviews. The sample size of bad reviews is not quite enough to have a good analysis. In fact, in a lot of products' reviews, some bad meaning words did not show as the most frequent word.

Therefore, based on our results for highlight and issues, we have to improve our model or choose another approach for issues finding in order to achieve a better result.