Ratings Prediction Research Report

Pol Monroig Company

September 2020

1 Data Analysis

As for the data analysis, we must explore two different paths, the labels, and the reviews that represent the features of the dataset.

1.1 Label Exploration

The dataset was a table of 10000 rows, each row was labeled with a discrete value between 1 and 5 that represents the rating or quality of a given review, where lower means a bad review a higher means a good review. This prediction problem is a NLP classification problem, more specifically, a sentiment analysis problem. The following table represents the number of samples per label, we can see that the dataset is perfectly balanced, that is great for training since the model won't be biased towards a specific class.

Label	1	2	3	4	5
Count	2000	2000	2000	2000	2000

The last thing to note about the dataset is that each entry is independent, anonymous and they are not associated between them with a specific user ID. If this was possible we could get a better predictor by taking into account the person that wrote it and creating a profile per user. The username could also be helpful to create a personalized product recommendation using some model-based or memory-based collaborative filtering.

1.2 Reviews Exploration

Each review is written in English, and it includes alphanumerical characters as well as special characters such as curly brackets, quotes, and parentheses. Reviews are often easy to analyze but some of them contain contradicting sentences that confuse the model later on. Another thing that I noticed while testing the model was that using exclamation marks led to greatly positive ratings, which could mean that most reviews with exclamation marks are positive. It is important to highlight that the following exploration is done using the training data, in order to remove any personal bias, thus, a different train/eval split might lead to a different data exploratory analysis.

One of the most important things to consider when working with natural language processing problems is how to approach the tokenization of the sentences. Different tokenizations lead to very different datasets, as a convention I used the *NLTK library* tokenizer because it is very standardized and is often used in machine learning contests. Other text preprocessing methods such as case folding and removing stop words were used to help the model generalize and reduce the dimensions of the input, thus avoiding "the curse of dimensionality". The stop words are also downloaded from the NLTK library. Techniques such as lemmatization and stemming were not chosen as they required a more extensive exploration of each specific word in the dataset, these techniques were not necessary and would have damaged the generalization of the model since we already had a relatively low vocabulary size. Some tests show that removing stop words do not increase a lot the vocabulary and do not affect the performance of the model, although more experimentation is needed to confirm this hypothesis.

After all the previous text preprocessing techniques had been applied we got the following lengths (here we use length as the number of tokens in each review). Some reviews can be classified as outliers where their tokenization length is of size 1. Although for the purpose of this project they were left untouched. After tokenization, the vocabulary size was 7964.

Reviews length			
Mean	STD	Min	Max
26.28	19.62	1	219

A very important analysis to understand the importance of tokens in the dataset is to make **TF** (term frequency) analysis. This metric is also extremely dependent on the tokenization technique. After calculating the TF we could also calculate the probabilities of words. The following plot shows the distribution of the most common words.

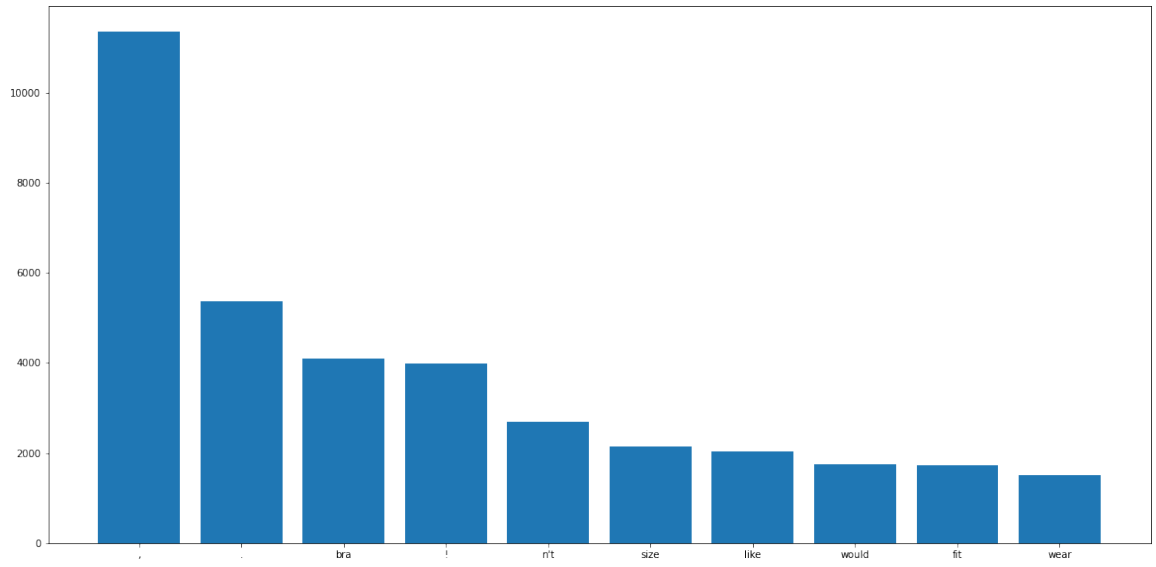

Figure 1: Most common tokens

Unfortunately, the most common one is the **,** symbol with a very high frequency of 11355, bigger than the number of instances in the entire dataset. That means that the comma does not provide any useful information at all, even if we suppose that is a useful token grammatically speaking, it is extremely repeated to the point that it has lost its purpose. And we have not taken into account commas that are a postfix of a word. And as we explained earlier, the exclamation mark is biased towards positive predictions. All this points to one thing, we must improve the tokenizer.

2 Model Review

For the training, a Naive Bayes model was selected because it is a very simple and powerful model. Its training and prediction speed is incredibly fast and uses low memory and storage space. A final reason for considering this model before others is that it is often used in email providers as a spam filter, so it might work similarly but with 5 classes instead of 2.

2.1 Preprocessing

For the data preprocessing, the dataset was first split into a train and test samples. The eval split size was of 20%. After that, tokenization was done for each sentence to spot words of interest. Case folding allowed better generalization by removing identical words. Finally, stop words from the NLTK library where removed. After applying this process to both, the training set and the test set, a lot of similar words were found although the test set can approximately 300 more tokens. This 300 tokens where removed, this might be one of the reasons why the model failed to generalize its knowledge.

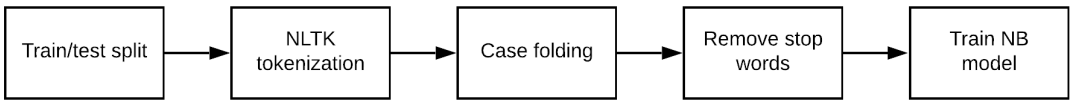

Figure 2: Train pipeline

A problem that was found with this tokenizer was that many reviews had useless tokens like at the end of the following text.

> *The shirt was more of a smock. I expected a soft little cotton t shirt, but I suppose itś my fault for not reading the description carefully. Also, the white shirt was completely see through. I returned mine."}},{r:{id:46830320,si:1,pi:10860142,mu:"{f7dedd4c-02d2-49ba-9553-4f9be0bdb9e5*

2.2 Performance

To measure the performance of the model simple classification metrics were used, although the mean absolute error was really helpful too. The performance of the model on the test set was worse than on the train set as it usually happens with machine learning models.

Metric/Dataset	**Train**	**Eval**
Accuracy	0.86175	0.7645
Precision	0.86175	0.7645
Recall	0.86175	0.7645
MAE	0.2175	0.3675

One of the downsides of the model was not in the model itself but the tokenization, since a lot of the most used tokens were not representative of any specific class. Another issue was that some phrases like *not good* or *not bad* where not being taken into account because there is no temporal relationship in the bag of words that was used for training, neither a bigram was used fix it. As explained earlier the exclamation mark was also an issue because it was biased towards positive ratings. Finally, the most notorious issue with the model was that is was naive (meaning that it couldn't find relationships between variables) and that it couldn't find a temporal relationship between words; this problem happens often in sentences where there is a contradiction in the review.

This contradiction might arise when a user describes the product positively but it also describes some unrelated thing negatively. These negative keywords were the main issue for the accuracy loss. The following sentence is an example of this problem, the user loves the product but hates some other unrelated one.

Great item to have, instead of buying those expensive, unflattering maternity jeans. Comes in different sizes as your belly grows! Must have!

Metrics show that recall and precision have the same value as accuracy, if we had used lemmatization or stemming precision would probably be lower. A thing to take into account is that the mean absolute error is very small, this metric as well as some manual testing can tell us that even though the model does not calculate the exact rating always, it is generally off by 1 star (rating measure) With the confusion matrix, it is easy to reassure the previous statement, where labels are often off by one star. We can see that because most predictions are correct and the incorrect ones are adjacent to the predicted ones. All this is caused by the naive problem of the model.

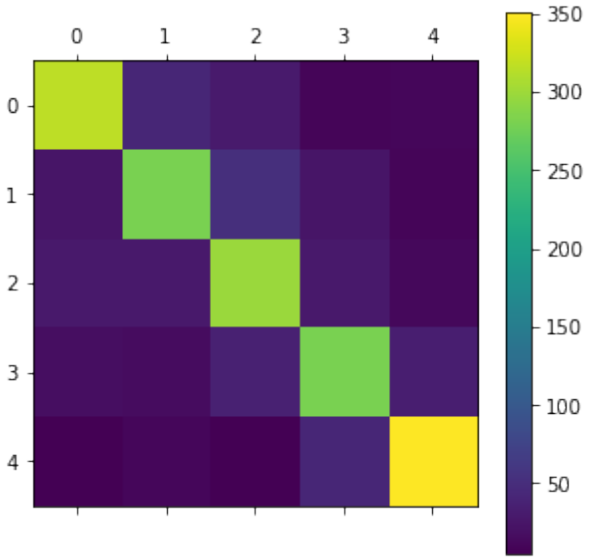

Figure 3: Confusion matrix

2.3 Improvements

- Change the tokenization is a must-do since it is one of the things that affect the most.
- Adding n-grams or at least bigrams can help with spatial word representation.
- Using a vectorized data representation with probabilities smoothing instead of a BOW might help to generalize.
- Removing outliers could help to remove some bias.
- Removing biased terms such as the exclamation mark and useless characters.
- Try more complex models such as Decision trees and SVM. RNNs could help in identifying temporal relationships in sentences.

2.4 Viability

In the end, how viable is this model in a real-world application? Well, it depends on the application of course, but if we improve the tokenization and apply some of the improvements mentioned previously we could get a model with an extremely great performance and generalization. If we leave the model as it is we get a great performance too, although you need to consider that the prediction will probably be off by one star in some reviews (which is a great prediction because it is sometimes difficult for a user to decide a rating for their own review).