# Amazon Product Reviews Sentimental Analysis

## Dataset

### Exploration of the dataset:

The amazon review dataset after extraction contains 25 folders, one for each domain such as 'Books', 'automotive', and 'video', etc. Each of these folders have unprocessed reviews along with their ratings and other details stored in 3 main files - "all.review", "positive.review", and "negative.review". Out of these, I consider the consolidated all.review file for this project.

### Loading the dataset:

The python XML parsing libraries cannot be used right out of the box on such files due to the structure of these files as well as due to use of special symbols in some of the review texts. One main reason I found was that none of these XML files have a root node which causes problems when parsing.

So, I wrote a custom function load_data(). The input to the function is folder path containing the 25 folders (one for each domain) which hold the review files.

The function works as follows:

1. Using the folder path provided, navigate to the domain directory. There, the function reads the "all.review" file line by line and using tags such as '<review_text>…</review_text>, stores the review texts, its ratings, and product type, etc.
2. If the rating is >3, it assigns the value 1 (positive sentiment). If the rating is <3, it assigns the value 0 (negative sentiment). The reviews with ratings = 3 are filtered out.

Repeat the above steps for all domains.

The entire dataset once read stores 1,422,530 reviews and its details. This is a large file and to save computation time, I store the loaded data into a "reviews.pickle" for future use if needed.

The extracted dataset is stored in Pandas data frame as shown in Figure 1.

| | review_text | sentiment |
|---|---|---|
| 0 | I want to start by saying Fred Flare- shipped ... | 0 |
| 1 | I have to say that I was disappointed when I o... | 0 |
| 2 | I am sorry but I did not like it nor will I we... | 0 |
| 3 | A red star!?!? I bet this won't sell well in ... | 0 |
| 4 | Perhaps it is my own fault for not reading mor... | 0 |
| 5 | THe pants that I ordered for my size were very... | 0 |
| 6 | I ordered the black one. According to the phot... | 0 |
| 7 | Suit was too small, not enough information ava... | 0 |
| 8 | the swim suit was so unsatisfactory looking th... | 0 |
| 9 | IT was advertised and NOT IN STOCK so I was un... | 0 |

**Figure 1. Structure of the loaded data**

```
'I want to start by saying Fred Flare- shipped this product very fast!! And the transaction itself was very smooth. I
do however, have extreme problems with the product itself. The product is not leather, its nylon, and it sort of look
s cheap? The inside material is sued, but that\'s only the lining for the base of the wallet. Also, The wallet part i
s very hard to use. You cant really put too much in the wallet- The credit card slots are a little too snug, and ther
e is no place for my I.D. The wallet included a small "note book" but it also doesn\'t fit in the wallet? I was very
excited about this product, but now I feel duped. The pictures made the wallet seem like it was of higher quality, an
d that it was user friendly, but it\'s not. I do not recommend this product'
```

**Figure 2. A sample unprocessed review**

The dataset that is loaded contains reviews that are unprocessed. As seen in figure 2, it contains a lot of features that are not useful for classification such slash, extra spaces, and punctuation, etc. That means, it has to be ‘cleaned’. For this, I removed the punctuations, irrelevant features such as slash and extra spaces as well as transforming uppercase words to lower case.

Next, the stop words such as ‘a’, ‘across’, ‘am’ and so on are removed. Note: For stop words, Only those words given in the ‘stopwords’ file provided along with the dataset is used.

The cleaned dataset is lemmatized. The processed review text from figure 2 is shown in figure 3.

```
'i want start saying fred flare shipped product very fast transaction itself wa very smooth i however extreme problem
product itself product not leather nylon sort look cheap inside material sued only lining base wallet also wallet par
t very hard use you cant really put too much wallet credit card slot little too snug no place my id wallet included s
mall note book also doesnt fit wallet i wa very excited about product now i feel duped picture made wallet seem like
wa higher quality wa user friendly not i not recommend product'
```

**Figure 3. Clean and Lemmatized review**

I tried stemming the texts instead of lemmatization but I found out that I got better results with lemmatization than with stemming.

Note: The functions performed for processing the data such as removing punctuations, lemmatization is in data_util.py file residing in the code folder.

Once the data is processed, I store the processed data as pickle file so computation time in future is less.

## Modeling

### Preparing the data for Modelling

The processed data that is obtained above is still not ready to be interpreted by classifier models. Concretely, the raw text has to be in a numeric representation. To do this, I vectorized the raw text.

For this assignment, I explored two techniques. First, I vectorized the raw text using 'one hot encoding technique' where I created a matrix where the columns are made up of the unique words present in the corpus. So, if a particular word is present in a review, then the cell value in the column that corresponds to that word would be 1 for that review.

Second, I vectorized the raw text using TF-IDF or term frequency – inverse document frequency. I used unigrams and bigrams when considering the word sequences in the raw text.

### Model Selection

The vectorized train dataset is split into 75% train and 25% validation set. I classified the reviews with three models. 1. Logistic Regression 2. Gradient Boosting 3. Decision Trees

The TF-IDF vectorized data was used for Logistic Regression and Gradient Boosting classifier and 'one hot encoding' vectorized data was used for the Decision Tree classifier.

### Training

1. Logistic Regression – For hyperparameter optimization, I tried C values = [0.01, 0.05, 0.25, 0.5, 1] and evaluated the accuracy on the validation set. I found out that the accuracy generally increased with C and the best model was with C = 1. The entire training process took about 25-30 minutes.

2. Gradient Boosting – I used the default parameters for the gradient boosting classifier. The entire training process took around 20 minutes.

3. Decision Tree – For Decision Tree, I first sampled around 30,000 reviews from the training dataset and tuned the values of max_depth, criterion and min_samples_split

using Grid Search and 5-fold cross validation on the train set that was generated after splitting the entire dataset into train and validation sets during model selection phase.

I found that the best model is one with criterion = 'entropy' and default values of max_depth and min_samples_split. The entire training process took about 2 to 2 ½ hours.

**Evaluation**

To choose a good evaluation metric. I looked at the distribution of the classes. There are 1,238,299 positive sentiment reviews and 184,231 negative sentiment reviews. This makes the dataset heavily skewed towards the positive class. A good metric for such datasets is F1-score which computes the precision and recall for both the classes.

The evaluation of the trained classifiers on the held-out data is shown in Table 1. Logistic Regression is able to achieve the highest F1-score for both the classes and has the highest accuracy amongst the three classifiers.

As a result of the skewness, it has trouble with the negative sentiment reviews but its still able to produce a decent result when compared to the Gradient Boosting and Decision Tree classifiers that have F1 score of just 0.21 for negative sentiment reviews.

| **Classifier** | **Negative Sentiment** | | | **Positive Sentiment** | | | **Accuracy** |
|---|---|---|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-Score** | **Precision** | **Recall** | **F1-Score** | |
| Logistic Regression | 0.95 | 0.67 | 0.79 | 0.95 | 0.99 | 0.97 | 0.9526 |
| Gradient Boosting | 0.91 | 0.12 | 0.21 | 0.88 | 1.00 | 0.94 | 0.8841 |
| Decision Tree | 0.92 | 0.12 | 0.21 | 0.88 | 1.00 | 0.94 | 0.8841 |

**Table 1. Model Evaluation**

Since Logistic Regression was the best classifier. I used the trained logistic regression classifier to determine which words had the most influence for classifying a text as positive sentiment as well as which words had the most influence for classifying a text as negative sentiment. The five most influential words from both classes are shown in Table 2.

| **Positive** | **Negative** |
|---|---|
| excellent | worst |
| great | disappointing |
| not disappointed | disappointment |
| best | not recommend |
| wonderful | boring |

**Table 2. Model Inference**

Note:

Since the training the classifiers takes so much time. The vectorized data and the trained classifier are stored in pickle files for faster computation in future.

The code for Dataset and Modeling is written in the SentAnalysis.py file present in the code directory

## Cross Domain Sentiment Classification

For cross domain sentiment classification, I used the reviews from "video" domain as my training set and reviews from "kitchen and housewares" domain for testing. There are 36,180 reviews in the train set and 19,856 reviews in the test set. As in the previous sections, the reviews are cleaned, lemmatized and the stop-words are removed.

I used the TF-IDF technique considering unigram and bigrams for vectorization and Decision Tree for classification. The result of a model trained on "video" domain reviews when tested on "kitchen and housewares" domain can be seen in table 3. Evidently, the performance isn't as good as achieved above. The negative sentiment reviews in particular are being classified really bad with an F1-score of 0.02. The model, however, does a decent job of classifying the positive sentiment reviews with a F1-score of 0.88. The final accuracy on the test I achieved was 0.7840.

| **Negative Sentiment** | | | **Positive Sentiment** | | | **Accuracy** |
|---|---|---|---|---|---|---|
| **Precision** | **Recall** | **F1-Score** | **Precision** | **Recall** | **F1-Score** | |
| 0.19 | 0.01 | 0.02 | 0.79 | 0.99 | 0.88 | 0.7840 |

**Table 3. Model Evaluation on the test set**

One reason for this reduction can be inferred when a closer look is taken on the review texts of the two sets. As seen in Figure 4, the review text for the video domain contain strong negative words such as hate whereas the review text from kitchen and housewares use a lot of words / word sequences that signify negative personal experiences such as "refused" that are negative given the context and domain.

"Not only did this product have no effect on the manic barking of our own dogs, but the company also refused to provide a refund/replacement after the product arrived with a broken chunk out of the cheap plastic casing. After sitting out in a rainstorm, the Super Bark Stop shorted out and is now totally worthless. Find another product - You'll thank me later"

'I have rented this video for the sake of a trip down Memory Lane. Of course, fourteen years produce a great shift in gullibility. I hated this video for its teaching of American ideology in perhaps the least open manner imaginable. For example, in'

**Figure 4. Comparison between excerpts of negative sentiment review texts. Top one is from Kitchen and Housewares domain and bottom one is from video domain**

One of the ways we can try improve the accuracy on cross domain sentimental analysis is by only considering majority of the features that are common between the two domains. This means extracting domain invariant features that can classify a text across domains.

One such word for negative sentiments could be ‘disappointing’ and for positive sentiments - ‘great’. By having a good balance between the domain specific features and domain invariant features, we can effectively utilize the training domain data to build a good model as well as make sure that its able to generalize relatively well.

Note: The code for cross-domain sentimental analysis is written in the CrossDomain.py file in the code directory.