# Sentiment Analysis on Amazon Reviews Using Spelling Correction and Ensemble Learning

**Natural Language Processing**

**Ali Khurram**

National University of Computer and Emerging Sciences, Islamabad, Pakistan

i170069@nu.edu.pk

## Abstract

A number of projects for sentiment analysis have been developed and made available in the Natural Language Processing community. It has proven to be quite a valuable technique of recent times, including reviews on shopping platforms. This study performs a large-scale, data driven empirical analysis of product reviews on shopping websites' data. An analysis on the reviews is done by determining the sentiment of the review, whether it is positive or negative. The reviews are spell-checked and corrected, while keeping the context of the sentence in perspective, to dismiss human errors. Multiple machine learning algorithms are applied using ensemble learning like Logistic Regression, Support Vector Machines, Decision Trees and ensemble methods that include Random Forest, Bagging and Voting.

## 1 Introduction

Sentiment analysis is a classification process in which machine learning techniques are applied on text-driven datasets in order to analyze its sentiment, e.g. a message being positive or negative about a certain topic. We want to investigate if these sentiment analysis techniques are also feasible for application on product reviews on Amazon.com.

Online shopping has been growing for 20 years and many e-commerce websites such as Amazon, have been created to meet the increasing demand. Consequently, a specific product can be bought on several websites and the prices may vary. As customers usually want the best quality for the lowest price but can't directly check it, reviews from other customers seem to be the most reliable way to decide whether to buy the product or not. Therefore, sentiment analysis has proven essential to understand a product's popularity among the buyers all over the world.

Businesses, for example, are always interested in public or consumer thoughts and feelings about their products and services. Before using a service or purchasing a product, potential customers want to know what other people think about it. Last but not least, researchers use this data to conduct in-depth analyses of market patterns and consumer sentiment, which could lead to improved stock market forecasting. The goal of this research is to identify customers' positive and negative reviews of various products and to develop a learning model to divide vast amounts of data.

## 2 Research and Dataset

After exploring a number of datasets for sentiment analysis, Amazon reviews for sentiment analysis was selected. The dataset used in this project comes from Amazon Reviews for Sentiment Analysis[1]. This dataset consists of a few million labelled Amazon customer reviews (input text) and star ratings (output labels) for learning how to train fastText for sentiment analysis. The training dataset contains 3.6 million reviews and the testing dataset contains 400,000. A total size of 2.5GB, approximately 2.1GB text file for training and 400MB text file for testing.

Since the dataset is quite extensively large, to improve performance and speed up the process, a part of the dataset was used, while sacrificing some accuracy, but nothing major. However, before splitting the dataset, it was shuffled so that the whole dataset was still being used. After shuffling the dataset, it was split – ten thousand reviews for training and twenty-five hundred reviews for testing were used.

1 https://www.kaggle.com/bittlingmayer/amazonreviews

The training dataset contains a few spelling errors. It was spell checked and corrected, while keeping the context in perspective.

## 3 Project Specification

The idea for the project was conceived in early April of 2021. Sentiment Analysis seemed like a good area to apply some key Natural Language Processing techniques that we learned throughout our course. Including Sentiment Analysis, spelling correction was also applied, while keeping the context of the sentence in perspective. The concepts of ensemble learning were also applied during the implementation phase.

After exploring a number of datasets for sentiment analysis, Amazon reviews for sentiment analysis was selected. Following the analysis of the dataset, the project's objectives were established. The aim was to achieve an F1-score of above 80 percent. The score achieved, is stated later in the paper.

Google Colaboratory was used for creating the python notebook and utilizing the free access to their powerful GPUs. Although the idea for the project was confirmed in April, the execution in code form started in late May.

## 4 Problem Analysis

A few problems were faced during the implementation phase of this project. Firstly, the dataset was too big to start with. It needed to be split up in such a way that the whole dataset was still being used. Although the accuracy that would have been achieved using the entire dataset would be marginally higher, it had to be compromised in favor of performance and efficiency.

Another challenge faced during the implementation phase was of spelling correction. Some spelling correction techniques were tested, but adding the feature of keeping the context of the sentence in perspective, while using those techniques, was a challenge.

Testing out different machine learning models and selecting which one maximizes the overall accuracy, when used in combination with other models as well (since ensemble learning was applied), was an added concern.

## 5 Solution Design

### 5.1 Overcoming the Problems

The previously described issues were effectively resolved. To overcome the problem of the enormous dataset, it was decided that it should be shuffled and only a portion of it should be used, ensuring that the complete dataset was still utilized.

A number of spelling checking and correcting tools/libraries were explored like TextBlob, Pyspellchecker and autocorrect, until finally the most appropriate spelling correction tool was discovered – JamSpell[2].

JamSpell is a spell-checking library that is sufficiently accurate while keeping the context of the sentence in perspective. It has an impressive processing speed at nearly five thousand words per second. It's written in C++ so it requires a swig binding when applied in python.

Here is an example of how JamSpell functions, particularly in the case of same misspelt word used in different sentences with different context.

```
jsp.FixFragment("Everyone just wants pece in their life")
```

```
'Everyone just wants peace in their life'
```

Figure 1: JamSpell Example 1

```
jsp.FixFragment("I ate a pece of chicken yesterday")
```

```
'I ate a piece of chicken yesterday'
```

Figure 2: JamSpell Example 2

This example is to demonstrate the ability of JamSpell. Notice the similar word "pece" in both the examples above. It is purposefully misspelt to showcase how it takes context into persepective to make spelling corrections.

In Figure 1, it corrects the word "pece" to "peace", as it clearly should. In Figure 2, it corrects it to "piece". This shows that it does keep the context of the sentence in view as it corrected the same word differently according to the sentence it was being used in.

A number of supervised machine learning algorithms were tested – like Decision Trees, Logistic Regression, Support Vector Machines as well as ensemble learning methods like Random Forest, Bagging, Boosting and Voting Classifiers. The ones providing the maximum results were then finally selected.

2 https://github.com/bakwc/JamSpell

## 5.2 Solution Implementation

The first step of the implementation stage was to read the dataset and pre-process it. Since the Amazon Reviews dataset was compressed in bz2 format, the bz2 library was required for reading. The labels and the text were split and stored in separate lists. This process took a lot of time as the dataset was very large.

After reading the dataset and storing it into lists accordingly, the text and labels of training and testing lists were shuffled using the shuffle library from sklearn.utils.

The training and testing data were then split – ten thousand reviews for training and twenty-five hundred for testing.

The next step was to set up the spelling correction method. Spelling correction was applied to every piece of text in the dataset.

After spelling correction was applied to the dataset, it was time to clean it. Any word within quotation marks, anything that was not a letter and stopwords, except for the word "not" (as it is essential for sentiment analysis), were removed. The text was converted into lowercase, which meant that our data was now cleaned.

The training dataset was them vectorized using TfidfVectorizer, and was split for validation, to test the best algorithms. The machine learning models used for validation Random Forest, Bagging Classifier with base classifier as Logistic Regression, Support Vector Machines and Decision Trees. All of these algorithms were utilized through the Voting Classifier by ensemble learning. The results obtained from validation are as follows:

```
1.0
Accuracy: 0.8628
f1 Score: 0.8627962900116819
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1199
           1       0.90      0.82      0.86      1301

    accuracy                           0.86      2500
   macro avg       0.86      0.86      0.86      2500
weighted avg       0.87      0.86      0.86      2500
```

Figure 3: Validation Set Results

The F1-score achieved through the validation set was 86.279%. Note that these results change every time a new session is started as the dataset is shuffled differently each time – so, they vary by approximately 0-2% each time. The lowest seen was ~85% and the highest was ~88%.

The same models using the same techniques were then used on the testing set. The results obtained through that are as follows:

```
0.9998
Accuracy: 0.8632
f1 Score: 0.8628404444547115
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1281
           1       0.88      0.83      0.86      1219

    accuracy                           0.86      2500
   macro avg       0.86      0.86      0.86      2500
weighted avg       0.86      0.86      0.86      2500
```

Figure 4: Testing Set Results

The F1-score achieved through the testing set was quite similar to the validation set.

To further visualize these results, a confusion matrix was constructed, which shows the picture in much more detail about how things went on.

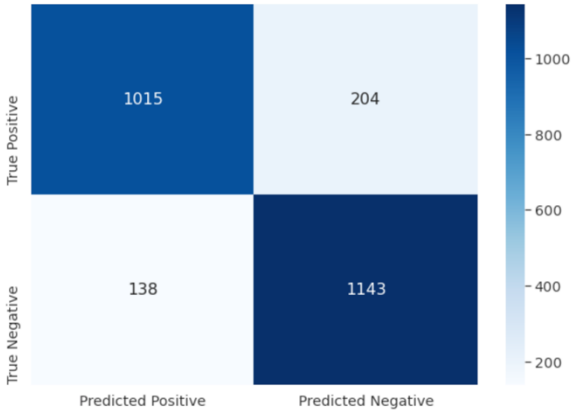


Figure 5: Confusion Matrix – Testing Set

To explain this confusion matrix, out of the twenty-five hundred reviews, the total number of true positive reviews were 1,219 (1,015 + 204), out of which the results obtained through the algorithms used predicted 1,015 of them correctly (positive) and got 204 incorrect (negative).

Similarly, the total number of true negative reviews were 1,281 (138 + 1,143), out of which 1,143 were predicted correctly (negative) while only 138 were predicted incorrectly (positive).

So, it can be seen in much more detail that the results obtained through all the techniques used were quite good.

# 6 Evaluation

The original plan proposed in April was strictly followed and we did not deviate from it. A lot of effort went into achieving this as it was not an easy

task, since we were not very familiar with all of the techniques used in this project. However, we see it as a plus as we learned a lot through this project, especially spelling correction and ensemble learning and how they work.

The total time it took for this entire project to run on Google Colaboratory was approximately thirty minutes. The training part takes a lot of time since we are using ensemble learning techniques, so multiple machine learning algorithms are working together to deliver results.

Time was somewhat of a limitation to this project. As training the models took a lot of time, we were unable to tune them to improve our results further. Otherwise, we definitely would have.

The good part was that we achieved everything that we had originally proposed back in April. We believe what we implemented is quite useful in the practical world as well.

We took a lot of help through the sci-kit learn's official documentation website - (scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation, 2021).

## 7 Conclusion

To summarize, everything that we had initially proposed was achieved successfully. Our dataset was first shuffled (to balance the dataset), then shortened (as it was too large). We applied spelling corrections to our data and cleaned it. The algorithms used were in ensemble through Voting Classifier using Random Forest, Bagging with Logistic Regression, Support Vector Machines and Decision Trees. These algorithms when used together, provided the highest accuracy.

## References


Mayer, B., 2021. Amazon Reviews for Sentiment Analysis. [online] Kaggle.com. Available at: <https://www.kaggle.com/bittlingmayer/amazonreviews>

Scikit-learn.org. 2021. scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/> [Accessed 10 June 2021].

Scikit-learn.org. 2021. 1.11. Ensemble methods — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/ensemble.html> [Accessed 10 June 2021].

GitHub. 2021. bakwc/JamSpell. [online] Available at: <https://github.com/bakwc/JamSpell> [Accessed 10 June 2021].

Acl-org.github.io. 2021. Paper formatting guidelines - ACLPUB. [online] Available at: <https://acl-org.github.io/ACLPUB/formatting.html> [Accessed 10 June 2021].

Brownlee, J., 2021. *How to Develop Voting Ensembles With Python*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/voting-ensembles-with-python/> [Accessed 10 June 2021].

Python, S., Barnes, M., Narasimhan, R., uppal, s. and Sahrawat, R., 2021. *Spell Checker for Python*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/13928155/spell-checker-for-python> [Accessed 10 June 2021].

Scikit-learn.org. 2021. *sklearn.ensemble.BaggingClassifier — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html> [Accessed 10 June 2021].

Scikit-learn.org. 2021. 1.4. Support Vector Machines — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/svm.html> [Accessed 10 June 2021].

Scikit-learn.org. 2021. *sklearn.linear_model.LogisticRegression — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html> [Accessed 10 June 2021].

Scikit-learn.org. 2021. *sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> [Accessed 10 June 2021].

Scikit-learn.org. 2021. 1.10. Decision Trees — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/tree.html> [Accessed 10 June 2021].

Scikit-learn.org. 2021. sklearn.metrics.confusion_matrix — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html> [Accessed 10 June 2021].

Semicolon, T., 2021. *Scikit Learn Ensemble Learning, Bootstrap Aggregating (Bagging) and Boosting*. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=X3Wbfb4M33w&t=276s> [Accessed 10 June 2021].