Amazon Reviews - Final Report

Ezra John Guia *30031697* | Ivan Suyat *30089089* | Long Ta *30069130* | Suhaib Tariq *30075751* | Garth Slaney *30061944*

I. Preamble

A. Total Contribution

- Ezra John Guia - 20%
- Garth Slaney - 20%
- Ivan Suyat - 20%
- Long Ta - 20%
- Suhaib Tariq - 20%

B. Contribution Declaration

- I, Ezra John Guia, confirm that the total contribution above is true.
- I, Garth Slaney, confirm that the total contribution above is true.
- I, Ivan Suyat, confirm that the total contribution above is true.
- I, Long Ta, confirm that the total contribution above is true.
- I, Suhaib Tariq, confirm that the total contribution above is true.

C. Link to Public Repository

GitHub Repository - Amazon-Reviews-Classifier

II. Abstract

A. Succinct description of your project and main outcomes.

This project involves the development of a classifier to analyze the sentiment of given text, particularly Amazon product reviews.

In the e-commerce industry, a customer may write a review with each purchase of a product or service. These reviews are essential for both producers and consumers. Reviews provide producers with feedback on the performance of their product or service, indicating what it did well and what they need to improve. It also impacts brand trust, building credibility and increasing purchase probability if reviews are good. For consumers, reviews provide a clearer idea of a product, thereby allowing for a better decision on purchases. Reviews allow consumers to see other customer's opinions of a product, whether it would be something they recommend or not, and what to expect from the product. Therefore, the ability to categorize whether a review is favorable or not is a valuable tool.

The development of the classifier involves the use of PySpark on the Databricks Community edition platform. The dataset used for the purpose of this project is the *Software* subset of the *Amazon Review* dataset [1], containing 12,805 reviews. This paper will cover the processes involved in developing the classifier which includes collecting, inspecting, and extracting data, as well as performing exploratory data analysis and building machine learning models to predict text sentiment.

Three machine learning algorithms were developed in this project, which are Logistic Regression, Naive Bayes, and LinearSVC. All three algorithms were viable, but the Logistic Regression algorithm had slightly better results overall. The Logistic Regression algorithm achieved the following results:

```
Area under PR: 0.86
Area under ROC: 0.77
Accuracy: 0.77
Weighted Precision: 0.78
Weighted Recall: 0.77
Weighted F Score: 0.77
```


III. Introduction

A. What is the problem you selected?

The problem selected was the 10th option from the list in Project Ideas. This was the Amazon Review dataset to develop a classifier or sentiment analyzer that can predict whether a given review is favourable or not.

B. Why is it an important problem?

Sentiment analysis is valuable information to brands since it gives details from actual users who have tested out their product. With these people telling them what they like and don't like about their product they would be able to improve on their product's services. This also includes the positives and negatives of a product. By checking the reviews rather than the rating itself, this may give more insight than meets the eye.

As mentioned the idea that someone may have given a good number of stars may miss important aspects of their written review which may possibly have negative comments which is overclouded by the high rating given. This may be the same case the other way around.

This natural language processing (NLP) project allows us to look beyond the star ratings to give more insight than what the star ratings provide to us.

C. What have others done in this space?

Sentiment analyzers are a fairly common tool that can be found online. Here is an example of an online sentiment analyzing tool : MonkeyLearn Sentiment Analyzer.

Sentiment analysis works thanks to natural language processing (NLP) and machine learning algorithms, to automatically determine the emotional tone behind online conversations.

Sentiment analysis can be achieved using different machine learning models. Logistic regression is a good model because it trains quickly even on large datasets and provides very robust results.

Other good model choices include SVMs, Random Forests, and Naive Bayes. These models can be further improved by training on not only individual tokens, but also bigrams or trigrams. This allows the classifier to pick up on negations and short phrases, which might carry sentiment information that individual tokens do not. Of course, the process of creating and training on n-grams increases the complexity of the model, so care must be taken to ensure that training time does not become prohibitive.

Most advanced sentiment models start by transforming the input text into an embedded representation. These embeddings are sometimes trained jointly with the model, but usually additional accuracy can be attained by using pre-trained embeddings such as Word2Vec, GloVe, BERT, or FastText.

D. What are some existing gaps that you seek to fill?

Millions of people order products from Amazon daily. This tool would allow buyers to get the information about the quality of the product, and it also helps these brands get valuable feedback and make their product better constantly. Just looking at the star ratings quickly can not always give all the information about the product that one may need. So having a sentiment analyzer that classifies the review will be extremely helpful to the buyers to make a decision on whether to buy the product or not.

E. What are your data analysis questions?

We had a large dataset of amazon reviews, of which we wanted to find out the following :

- Whether a review was positive or negative.
- The distribution of ratings.
- The average length of a review, this shows us how in depth and detailed a review is.
- Correlation between features.
- Most popular words used in the reviews.

F. What are you proposing and what are your main findings?

We propose 3 algorithms, then showcase results achieved with these algorithms. We used LogisticRegression, NaiveBayes and LinearSVC algorithms for our sentiment analyzer. With Logistic regression, the following results were achieved:

```
Accuracy: 0.77
Weighted Precision: 0.78
Weighted Recall 0.77
Weighted F Score 0.77
Weighted False Positive Rate 0.24
Weighted True Positive Rate 0.77
```

With NaiveBayes, the following results were achieved :

```
Accuracy: 0.76
Weighted Precision: 0.78
Weighted Recall 0.76
Weighted F Score 0.76
Weighted False Positive Rate 0.21
Weighted True Positive Rate 0.76
```

With LinearSVC, the following results were achieved :

```
Accuracy: 0.74
Weighted Precision: 0.80
Weighted Recall 0.74
Weighted F Score 0.74
Weighted False Positive Rate 0.19
Weighted True Positive Rate 0.74
```

Based on these results, it appears that all three algorithms performed relatively similarly in terms of accuracy, precision, recall, and F-score. Logistic regression and Naive Bayes both achieved an accuracy of around 77%, while LinearSVC had an accuracy of 74%.

In terms of precision, all three algorithms had similar performance, with values ranging from 78% to 80%. Precision measures the proportion of predicted positive cases that are actually positive, so a high precision score indicates that the algorithm is good at correctly identifying positive cases.

All three algorithms also had similar performance in terms of recall, with values ranging from 74% to 77%. Recall measures the proportion of actual positive cases that are correctly identified, so a high recall score indicates that the algorithm is good at finding all of the positive cases.

Finally, the F-score is the harmonic mean between precision and recall. In other words, F-score is a measure of the balance between precision and recall, and all three algorithms had similar F-scores, with values ranging from 74% to 77%.

Overall, these results suggest that all three algorithms are relatively effective at performing sentiment analysis, with Logistic regression and Naive Bayes performing slightly better than LinearSVC.

IV. Background and Related Work

A. Technical Background

This project focuses on Sentiment Analysis which has been a topic of many studies for it's informative and insightful results. Sentiment analysis is the process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral. This research has been done through large volumes of text data such as emails, social media comments and reviews. With the insights provided from the sentiment analysis, companies are able to improve the services provided.

In essence, this uses natural language processing (NLP). This means that we train computer software to understand text the ways us humans do. Typically, this would be done in 2 stages: Preprocessing and Keyword Analysis.

Preprocessing identifies key words to find the key message of the text provided. This is also broken down to 3 portions.

First tokenization, which breaks down sentences to words. Second lemmatization, which converts a word to its original root form. Finally, removing stop words, which filters out words that do not add any meaning to the sentence such as, *with*, *for*, and *of*.

Keyword Analysis is the analysis of the extracted keywords to give a score, or what would be called a *sentiment score*. It is a measurement scale indicating the emotional element in the sentiment analysis system. In our case, we had a scale of 1.0 to 5.0 with 1 being the worst and 5 being the best to accomodate for Amazon's star reviews.

These techniques will be used and described in further detail in the later sections of this report and how they were incorporated into our project.

B. Review of existing work pertinent to your project.

Sentiment analysis has been around with many contributions adding to this. In a study done at the Royal Institute of Technology [2], 60,000 reviews were analyzed from Amazon. Multinomial Naïve Bayes, Linear Support Vector Machine and Long short-term memory network algorithms were used to find the lowest error rate on the test set. Performance metrics used were the F1-score, accuracy, precision and recall in addition to a few others. While our data set strictly focuses on software products, they would use the beauty, look, electronic and home section datasets.

Another study was done at Swarthmore College [3] also using naïve bayes and decision-list classifiers for the category of books. These were then separated as positive and negative reviews through 50,000 of them. Bag-of-words and bigrams were used to compare one another. His best feature extraction were the bag-of-words and the naïve bayes performed much better than the decision list.

Finally a fascinating approach done by the National Research Council of Canada, Turney [4]. Bag-of-words (BOW) was used once more where the relationship between words was ignored. Instead, the individual words of the sentences were given values and then aggregated by some function. Other methods used in this were Sequential Minimal Optimization and Support Vector Machines.

As shown from these works, there have been many attempts to find the best ways to analyze all this data. Obviously, there isn't really a best way, however with this information, we were able to come up with a few ideas to move the project forward.

V. Methodology

A. Preprocessing

In order to improve the accuracy and reliability of the data, we have taken the 5 following steps to preprocess the data in order, which are:

- Lowering text
- Expanding contractions
- Removing stop words
- Removing special symbols
- Stemming

1) Lowering text: All used algorithms rely on word to predict sentiment of a review. However, the algorithm cannot tell the difference between the word "Happy" and "happy". As a result, we need to lower all characters in the reviews. This helps to make sure that there is no difference between capitalized and non-capitalized characters.

2) Expanding contractions: The idea is similar with lowering text. It is noticed that there is no difference between 2 words "can't" and "cannot". Therefore, expanding contractions also help algorithms to reduce ambiguity between 2 similar words. We used Python contractions library to achieve this goal.

3) Removing stop words: Stop words are words that does not contribute or hold any significant meaning in terms of sentiment in a sentence. An example for a stop word is the word "the".

4) Removing special symbols: Special symbols such as "," and ":" do not hold any significant meaning in a sentence. Thus it is removed to reduce noise in training data and number of features used.

5) Stemming: Our final step for preprocessing is called stemming. Stemming is the process of reducing words to their stem, base or root form. An example of stemming would be converting word "running" to its base form which is "run". This is done to remove ambiguity between these 2 words since they share the same meaning. We used Snowball Stemmer from nltk library to achieve this. In order to achieve better accuracy, we could have used lemmatization instead of stemming. But the tradeoff would be a significant increase in preprocessing time and training time. After testing lemmatization, we found out the trade-off was not worth it.

B. Train, Validation and Test

The dataset is split into 2 with 80/20 ratio. 20% of the dataset is used for testing. While 80% of the data set is used for training and validation. For this project, we used K Fold Cross Validation with number of folds to be 5 for training and hyperparameter tuning. The model with the best parameters will then be evaluated using the test set.

C. Experimentation factors

For our experiment, we used three different ML Algorithms: Linear Support Vector Classifier (LinearSVC) is a linear model for classification tasks. It is similar to Logistic Regression, but uses a different optimization method to find the weight coefficients that define the decision boundary between classes. LinearSVC is known for being efficient and easy to implement, and it can handle large datasets efficiently.

Naive Bayes assumes that all features are independent of one another and uses this assumption to make predictions about the class of a given data point. Naive Bayes is often used for text classification tasks and is known for being simple and fast to implement. Multinomial distribution was chosen for our Naive Bayes algorithm.

Logistic Regression is a classification algorithm that is used to predict the probability of an event occurring. It is

a linear model that uses the logistic function to transform the predicted probability into a binary outcome (0 or 1). Logistic Regression is often used in binary classification tasks, such as this sentiment analyzer. It is also known for being easy to implement and interpret, and it performs well on a wide range of classification tasks.

```
param_grid = ParamGridBuilder() \
    .addGrid(hashingTF.numFeatures, [5, 100, 1000]) \
    .addGrid(lr.threshold, [0.62, 0.65]) \
    .build()

crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=param_grid,
                          evaluator=BinaryClassificationEvaluator(),
                          numFolds=5)
```

Fig. 1. Hyperparameters and cross-validation

As for implementing the model, we used ParamGridBuilder to specify a grid of hyperparameters for each model. To choose the best tuned hyperparameters for the models, the hyperparameters for each model will be searched over a set of specified values during cross-validation, and the combination that results in the best score will be chosen for optimal performance. Cross-validation is often used in combination with hyperparameter tuning to ensure that the results are not overfitted to the training data. In the above code snippets, the CrossValidator is used to perform cross-validation, using the BinaryClassificationEvaluator as the evaluation metric and a specified number of folds. The resulting cross-validated model is then fit on the training data and evaluated on the test data using the evaluate_model function.

```
validator = cross_validator.transform(train)
validator.show()
evaluate_model(validator)
```

Fig. 2. Evaluating model

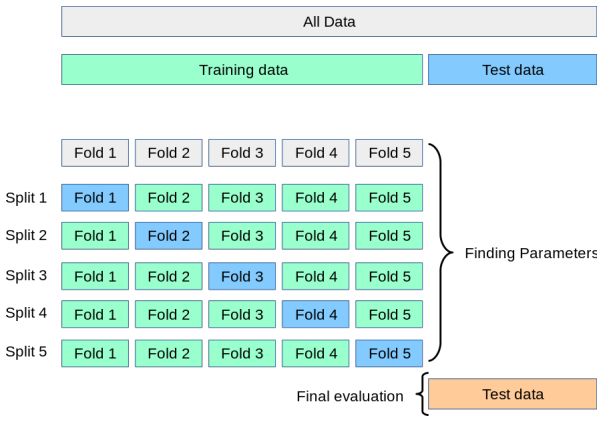

Fig. 3. Cross-validation explained

D. Experiment process

We defined a data pipeline which we used to feed our training data into our model. First, we used a Tokenizer which splits the word by spaces. We will then use TF-IDF for feature extraction. The reason is that Count Vectorizer only focuses on the frequency of the word, which results in biasing in favour of popular words. For example, the word ”the” will be counted a lot but does not hold any significant meaning. Meanwhile, using TF-IDF, we can avoid common words that do not carry any sentiment.

After, defining our pipeline we then use a ParamGridBuilder which will test different values for our parameters using cross-validation. For example, in our Logistic Regression model we have two parameters numFeatures and threshold with three values and two values correspondingly. This means we will train 3*2*5 = 30 models as we train a model for each value and each fold. We then will select the best model and use this model to evaluate using our testing data.

E. Performance metrics - accuracy, precision, recall, F-score etc.

With Logistic regression, the following results were achieved:

```
Accuracy: 0.77
Weighted Precision: 0.78
Weighted Recall 0.77
Weighted F Score 0.77
Weighted False Positive Rate 0.24
Weighted True Positive Rate 0.77
```

With NaiveBayes, the following results were achieved:

```
Accuracy: 0.76
Weighted Precision: 0.78
Weighted Recall 0.76
Weighted F Score 0.76
Weighted False Positive Rate 0.21
Weighted True Positive Rate 0.76
```

With LinearSVC, the following results were achieved:

```
Accuracy: 0.74
Weighted Precision: 0.80
Weighted Recall 0.74
Weighted F Score 0.74
Weighted False Positive Rate 0.19
Weighted True Positive Rate 0.74
```

VI. Results

A. Key findings in your exploratory data analysis and prediction. If you are trying out multiple algorithms, your results would compare them.

In the exploratory data analysis (EDA) stage, the following visualizations were produced:

- Review distribution per rating (and review count per rating)

- Review distribution per sentiment (and review count per sentiment)
- Review text length
- Most popular words in
  - Positive reviews
  - Negative reviews
  - All reviews
- Correlation between features

Figure 4 displays the percentages of review sentiments. Plotting pie chart for overall score distribution. The 'overall' column has 5 labels: 1.0, 2.0, 3.0, 4.0, 5.0 where 1.0 is very bad, 3.0 is neutral and 5.0 is very good. We can see here that the two largest categories are of rating 5 as well as 1. This helps with the clear boundary of what would be considered good or bad rather than taking a neutral stance which would not help the model.

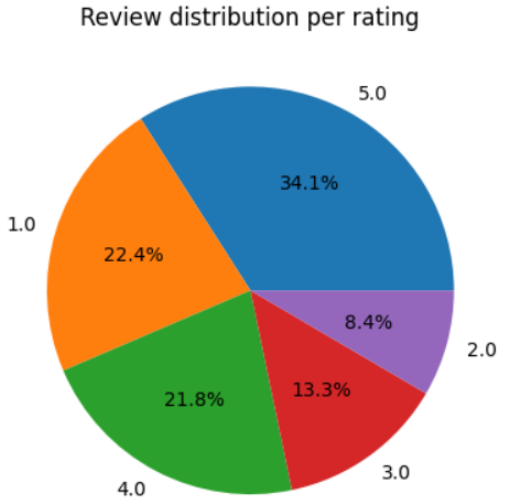

Fig. 4. Review Distribution per Rating

Figure 5 displays a similar statistic, but with sentiment. Looking at the chart, it is noticed over 55.9% of the reviews are positive, followed by 30.8% neutral reviews. Finally, negative reviews only account for 13.3%. With the imbalanced distribution of the, it is natural for our model to be bias towards positive reviews more. But through hyperparameter tuning with our models, we managed to remove the impact of the bias.

Figure 6 displays the length of each review. With a count of 3383 reviews the mean was 1110 words where the smallest review word count was 3 and the largest was 23075 words long.

Figure 7 displays the most popular words without seperating the positive and negative dataset.

With figure 8, we want to visualize the correlation between vote and rating in our dataframe. There is no correlation

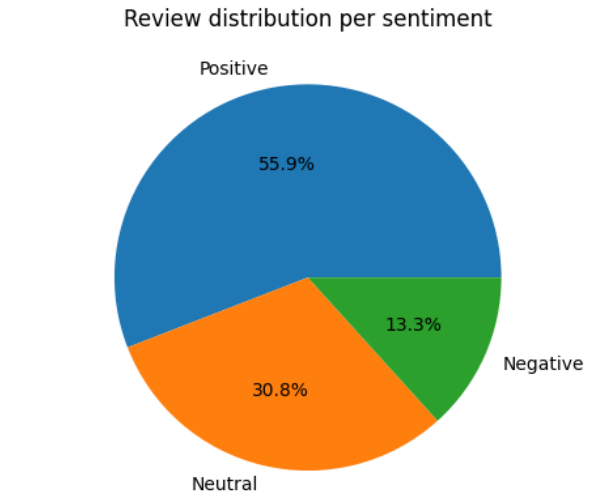

Fig. 5. Review Distribution per Sentiment

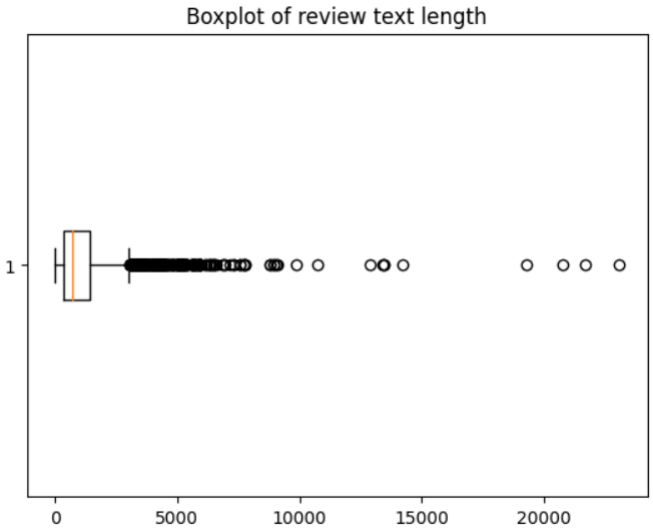

Fig. 6. Boxplot of Review Text Length

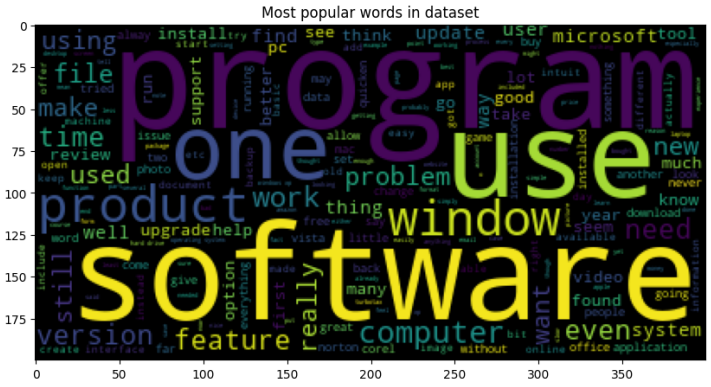

Fig. 7. Most Popular Words in Dataset

Model	Area under PR	Area under ROC	Accuracy	Weighted Precision	Weighted Recall	Weighted F Score
Logistic Regression	0.86	0.77	0.77	0.78	0.77	0.77
Naive Bayes	0.85	0.77	0.76	0.78	0.76	0.76
LinearSVC	0.86	0.77	0.74	0.80	0.74	0.74

between vote and rating. Therefore, we conclude that by removing samples that have no vote, it creates no bias for our dataset.

After EDA, three machine learning algorithms were developed. The results of each algorithm can be compared in the table above.

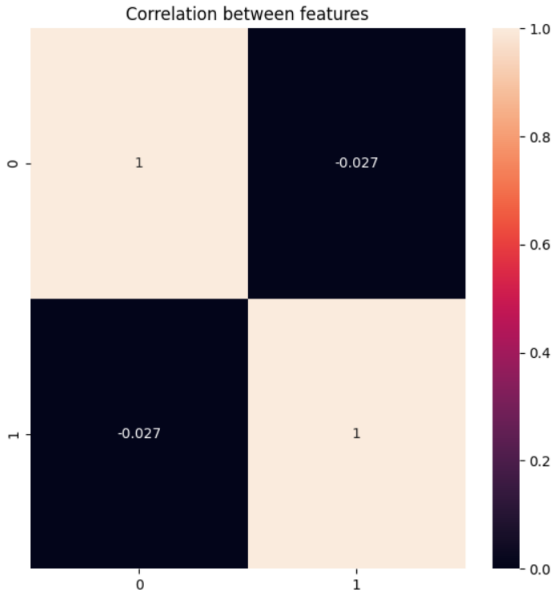

Fig. 8. Correlation between Features

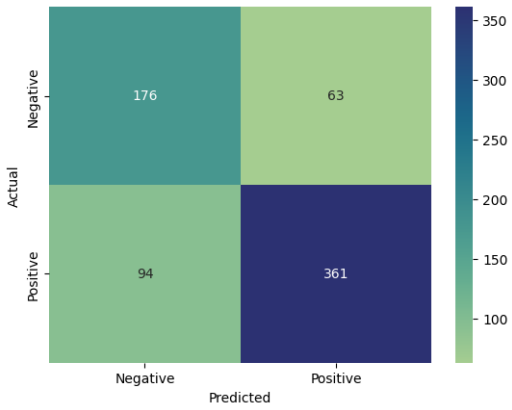

Fig. 9. The Logistic Regression model confusion matrix

B. Conclusions and future work

From our findings, we can conclude that the three produced algorithms are all viable, achieving similar results. However, they can be ranked as follows:

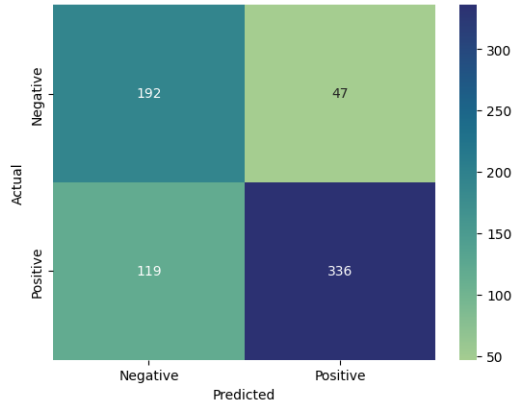

Fig. 10. The Naive Bayes model confusion matrix

Fig. 11. The Linear Support Vector Classification model confusion matrix

1) Logistic regression
2) Naive Bayes
3) LinearSVC

The linear regression algorithm slightly edges the rest in most categories. LinearSVC only has a slight advantage in weighted precision.

If this topic were to be further explored in the future, the current dataset would expand to include the entirety of the *Software* superset and include other categories. Furthermore, the viability of other algorithms and the effects of other n-gram sizes may be investigated. Optimizations would also be made to increase the performance and resolve potential bugs.

Finally, in this problem, we only considered unigram word in out dataset. To achieve a better classification between positive and negative review, we need to use both unigram and bigram. Using bigram adds more context of sentence to the model, allowing to correctly determine sentiment of a review.

REFERENCES

[1] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.

[2] Levent Guner, Emilie Coyne and Jim Smit, "Sentiment Analysis of Amazon.com Reviews", March, 2019, Available : https://www.researchgate.net/publication/ 332622380 _Sentiment_analysis_for_Amazoncom_reviews.

[3] Callen Rain, "Analysis in Amazon Reviews Using Probabilistic Machine Learning", 2012. Available: https://www.semanticscholar.org/paper /Analysis-in-Amazon-Reviews-Using-Probabilistic-Rain/ f0afe9ea9d286248336ee9dc4e954aecde3475bb

[4] Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, USA, 417–424. https://doi.org/10.3115/1073083.1073153