Contents

[Introduction 3](#_Toc109571398)

[Run instructions 3](#_Toc109571399)

[Goal 4](#_Toc109571400)

[Dataset 4](#_Toc109571401)

[Process 4](#_Toc109571402)

[Preprocessing 5](#_Toc109571403)

[Classifiers 8](#_Toc109571404)

[Multinomial NB 8](#_Toc109571405)

[LSTM 10](#_Toc109571406)

[Results 11](#_Toc109571407)

[BERT 12](#_Toc109571408)

[Results 12](#_Toc109571409)

[Review Findings 14](#_Toc109571410)

[Future Work/Applications 14](#_Toc109571411)

[Conclusions 14](#_Toc109571412)

# Introduction

This project is part of the evaluation for the ITC 6010A1 Natural Language Processing course. In the project the theoretical knowledge and practical steps gained throughout the class were applied regarding NLP topics around sentiment analysis.

## Run instructions

Download and extract the file included in the email. Download the dataset from Kaggle (URL HERE) and place the test.csv and train.csv inside the ‘data’ folder. First run the ‘amazon\_sentiment\_analysis.py’ file found inside the project folder. A new file will be created inside the ‘data’ folder called ‘text\_title.csv’, this includes the preprocessed dataset that will be used for the rest of the project. Next run the ‘multinomial\_NB\_approach.py’ that will produce the baseline results for the rest of the classifiers. Finally run the rest of the classifiers and Neural Networks to produce the results. Since in some cases the results may take up to multiple hours to be produced, all confusion matrixes and scores are already provided in the ‘Results.xlsx’ file.

Expected runtime for each method:

* Multinomial NB: 5 Minutes
* LSTM: 30 minutes
* BERT: 17 Hours per epoch

Required downloads:

* nltk.download('stopwords')
* nltk.download('wordnet')
* nltk.download('omw-1.4')

# Goal

The scope of the project is to create a sentiment analysis classification using different NLP methodologies, in order to find the polarity of Amazon product reviews. The classifiers are trained to find if an Amazon product review is favorable or not and can be used to promote certain favorable reviews in the product page to increase product sales. In other words, the classifiers are used to flag a product review as favorable, to appear more times as a suggested review when prospective customers are viewing the product page on Amazon’s website. The project achieves this using three different classifiers, specifically Multinomial NB, Long Short-Term Memory (LSTM), and BERT.

# Dataset

The dataset used in the project is called ‘Amazon Reviews’ and contains 34,686,770 Amazon reviews from 6,643,669 users (about twice the population of Oklahoma) on 2,441,053 products. The data was collected over 18 years, up until March 2013. The dataset contains three columns: Polarity, Title, and Text.

Polarity takes two values, 1 for Negative reviews and 2 for Positive. The dataset has already been preprocessed and the Rating has been reduced to two values instead of the usual 5-star rating. Reviews for 1-2 stars have been grouped under ‘negative’ reviews, while reviews with 4-5 stars have been grouped under ‘positive’ reviews. Reviews with a 3-star rating have been left out of the dataset.

Title contains the Title of the review where applicable

Text column contains the main body of the review

# Process

The two datasets are loaded in memory and merged together. A subset is used as the working dataset which will then be split into the actual training and test datasets used for the classifiers. The working dataset is preprocessed to clean bad quality data, merge the two text columns, remove foreign languages, and split all reviews to n-grams. This process produces a new file which will be used as input for the classifiers. The contents of the file are the merged text with all the preprocessing. Each classifier is stored in a separate file which produces a confusion matrix and f1 score as an output. For the LSTM and BERT classifiers, further preprocessing and initialization takes place to prepare the dataset and the neural networks.

# Preprocessing

The dataset contains two classes, class 1 with review scores of 1 and 2 as negative and class 2 with review scores of 4 and 5 as positive (neutral comments of score 3 will not be included to increase the data quality). Each class has 1,800,000 training and 200,000 testing samples which can be found in files train.csv and test.csv as comma-separated values. As previously mentioned, the dataset consists of 3 columns corresponding to polarity (1 for negative and 2 for positive), title and text. The review title and text are escaped using double quotes, and any internal double quote is escaped by 2 double quotes. New lines are escaped by a backslash followed with an "n" character, that is "\n".

Since the original dataset is very large, a smaller sample is used, specifically, a random subset of 50000 samples from train\_df and 10000 samples from test\_df. In addition, both training and test dataset are balanced, meaning that they both contain the same number of representatives from both classes. Moreover, the title and text features are merged in a single text column and all NA/NaN values of the title column are replaced with blanks.

Graphical user interface, text

Description automatically generated

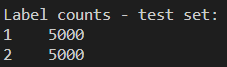


Figure Label Counts for each Dataset

The reviews contained in the training and test sets might be in many different languages. This means that the same text preprocessing techniques cannot be applied on all of the samples. The langid library is utilized to check the language of each review in the dataset. The results are that non-English reviews compose a small minority of the dataset, so a decision is made to drop the rows and not include them in the final dataset.

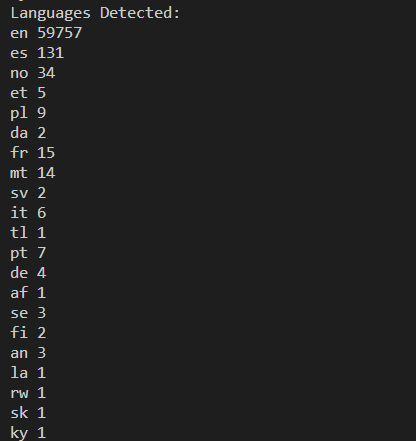


Figure Language Frequency for all Reviews in the Corpus

After that the entire corpus is tokenized for each one of the sample-reviews in the dataset. The tokens are single words (1-grams) and sequences of contiguous words of length 2 and 3 (2-grams and 3-grams respectively). By using the CountVectorizer() method to achieve that, all text is also converted to lowercase. By reviewing the results, the most frequent tokens in the dataset are 1-grams. However, they do not seem to be particularly helpful to determine whether a review is positive or negative. The same thing applies with 2-grams in the case of n=2,3.

To improve the results, it was decided to remove the stopwords from the reviews. Stopwords are words that may appear frequently in the reviews, but they do not indicate anything useful for the task of sentiment analysis, and so they can be removed to facilitate the whole procedure. Such words in this case might be: “my”, “you”, “in”, “a”, etc. So, a set of stopwords was defined which was obtained from the ‘nltk’ library, and then those words were removed from the dataset. Afterwards, the dataset was tokenized again.

After the second tokenization the most frequent tokens in the dataset are 1-grams. However, after the stopwords were removed, the most frequent tokens now are more helpful for the task of sentiment analysis (there are more adjectives and verbs that describe feelings).

The results are even better in the case of n=2,3, where it is observed that the majority of the most frequent 2-grams in the dataset are indeed very indicative of the sentiment of the review.

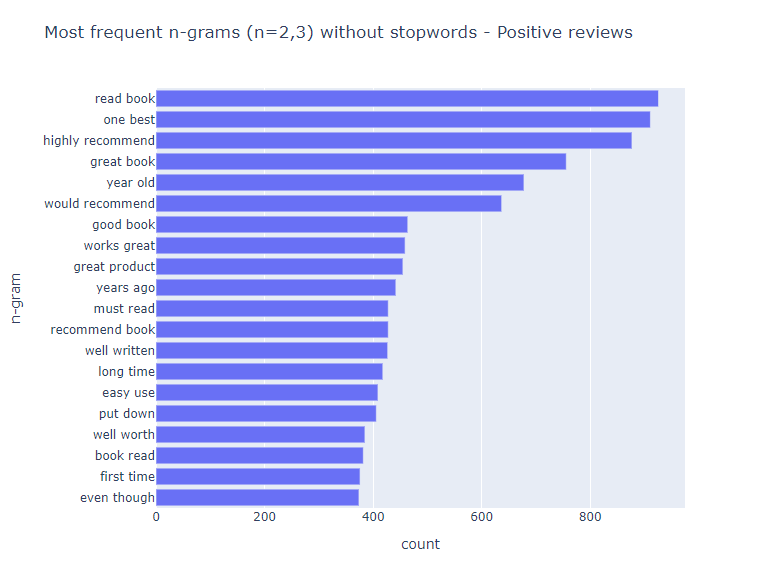


Figure Most Frequent 2-grams without stopwords - Positive Reviews

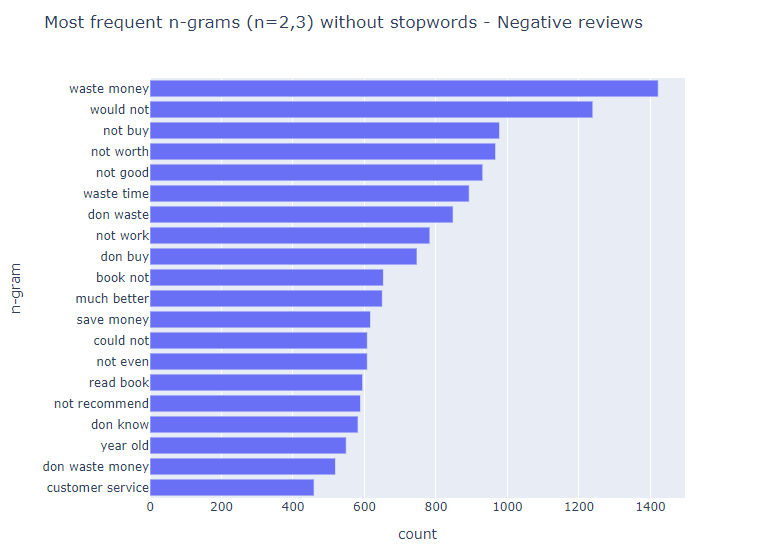


Figure Most Frequent 2-grams Without Stopwords - Negative Reviews

Even though the most frequent tokens appearing, especially in the case of n=2,3, are in their vast majority very indicative of the review's sentiment, it can still be observed that there are still some tokens that appear frequently in both positive and negative reviews, and thus they are not helping perform the sentiment analysis. This is most common in the case of n=1,2,3. A decision was made to include them in order to increase the data quality of the results of the classifiers (e.g. LSTM).

It is important to underline that after the preprocessing is finished, a csv file is created called text\_title.csv, which will be used later as input for the classifiers.

# Classifiers

## Multinomial NB

We start with the Multinomial Naive Bayes algorithm which will be our baseline in order to compare with LSTM and BERT classifiers which are more advanced. The expected results will not be as accurate as the other two but will serve as a comparison. Furthermore it is the easiest to implement since you only have to calculate the probability, it is highly scalable and can easily handle large datasets which means that will be our fastest classifier. Count vectorizer converts characters into n grams by grams and tri grams and then it counts the sum of all words and the frequency of each one in the text. This way we find the most frequent used words grouped by 1 for unigrams, 2 bigrams and 3 for trigrams. Moreover it calls the stopwords to eliminate them as well. . Multinomial Naive Bayes scans the whole dataset to find the frequencies. Then the model finds if a review is close to 1 then it belongs in the ‘1’ class and if it is close to 2 then it belongs to the class ‘2’ and counts all the possibilities which are in this case 50% (either 1 or 2). Afterwards it categorizes them as bad if the result is 2 or good if the result is 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Model: CountVectorizer + MultinomialNB w/o stopwords | | | |
| Number of features: 3187038 | |  |  |
| **Accuracy: 0.8767** |  |  |  |
|  | | | |
|  | recall | f1-score | support |
| class 1 | 0.91 | 0.88 | 4783 |
| class 2 | 0.84 | 0.87 | 4974 |
|  | | | |
| accuracy |  | 0.88 | 9757 |
| macro avg | 0.88 | 0.88 | 9757 |
| weighted avg | 0.88 | 0.88 | 9757 |

Chart, treemap chart

Description automatically generated

A scenario which used TfidfVectorizer was also examined which produced equivalent results.

|  |  |  |  |
| --- | --- | --- | --- |
| Model: TfidfVectorizer + MultinomialNB w/o stopwords | | | |
| Number of features: 3187038 | |  |  |
| **Accuracy: 0.8756** |  |  |  |
|  | | | |
|  | recall | f1-score | support |
| class 1 | 0.92 | 0.88 | 4783 |
| class 2 | 0.84 | 0.87 | 4974 |
|  | | | |
| accuracy |  | 0.88 | 9757 |
| macro avg | 0.88 | 0.88 | 9757 |
| weighted avg | 0.88 | 0.88 | 9757 |

Chart, treemap chart

Description automatically generated

## LSTM

The LSTM Neural Network requires further preprocessing and parameterization during the setup steps.

The stop words are imported from the preprocessing part of the code, and a tokenizer and lemmatizer are initialized. All the stop words are removed from the text and then the lemmatizer is applied to generate the lemma for each word. The labels are also updated so instead of 1 and 2, 0 represents the bad reviews and 1 represents the positive reviews. Train and test sets are created.

Some parameterization takes place in order to make the LSTM run more efficiently. For example, the top 3000 words in terms of overall frequency are used as the vocabulary for the LSTM, while the maximum length of a review is set to 200 tokens/words. Everything over 200 tokens is removed to speed up the neural network. For words that fall out of the vocabulary, an OOV token is used to remove them. Both train and test datasets are then converted to padded sequences.

The Neural Network has 4 layers, the first one creates the embeddings for the tokenized sentences, the second uses a bidirectional LSTM, meaning that for each embedding, the LSTM considers both the previous and subsequent tokens. Next a dense layer with 24 nodes exists, and finally another dense layer provides the output. The output for each review is a number between 0 and 1, which corresponds to the polarity of the review. If the number tends to 1 then the review is positive, else if the number tends to 0 then the review is negative.

The model is set up to produce the accuracy score of the LSTM, which is the baseline metric used by all methodologies. The LSTM runs for 15 epochs and then produces the results.

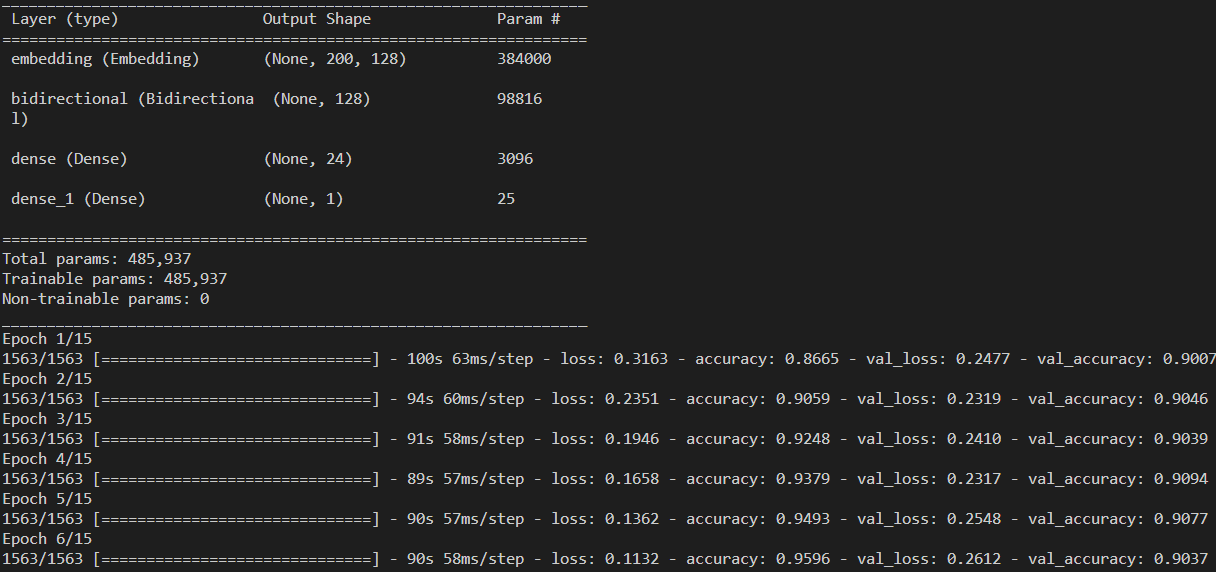
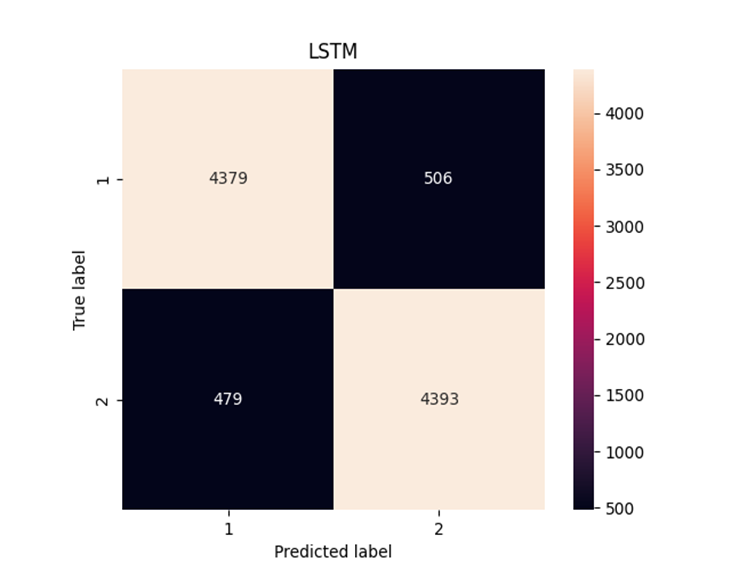


Figure Layer Configuration and Execution of the LSTM NN

### Results

Overall, the LSTM produces much better and highly accurate results than the two previous more traditional methods with an accuracy score of 90.7%. Both classes have similar, high true positive rates, meaning that the LSTM can accurately predict both positive and negative reviews based on the tokenized text.

|  |  |  |  |
| --- | --- | --- | --- |
| Model: tokenize + lemmatization + LSTM | | | |
| **Accuracy: 0.9069** |  |  |  |
|  |  |  |  |
|  | recall | f1-score | support |
| class 1 | 0.90 | 0.91 | 4885 |
| class 2 | 0.91 | 0.91 | 4872 |
|  |  |  |  |
| accuracy |  | 0.91 | 9757 |
| macro avg | 0.91 | 0.91 | 9757 |
| weighted avg | 0.91 | 0.91 | 9757 |



## BERT

The final methodology used, is by applying a pretrained model and fine tuning it to produce the wanted results. BERT stands for Bidirectional Encoder Representations from Transformers, which uses a pretrained representation from unlabeled text by considering both left and right context, thus bidirectional. This pretrained model can be further modified to produce very accurate results for a wide range of NLP tasks.

The pretrained tokenizer model used is the ‘bert-base-uncased' tokenizer, which is the most commonly used tokenizer for BERT. In this model only 10000 datapoints are taken, to increase both the speed of the model, but also because BERT requires a smaller amount of train data to properly function. The labels are updated here too to represent with 0 the negative and with 1 the positive reviews, and the train and test datasets are split and tokenized using BERT.

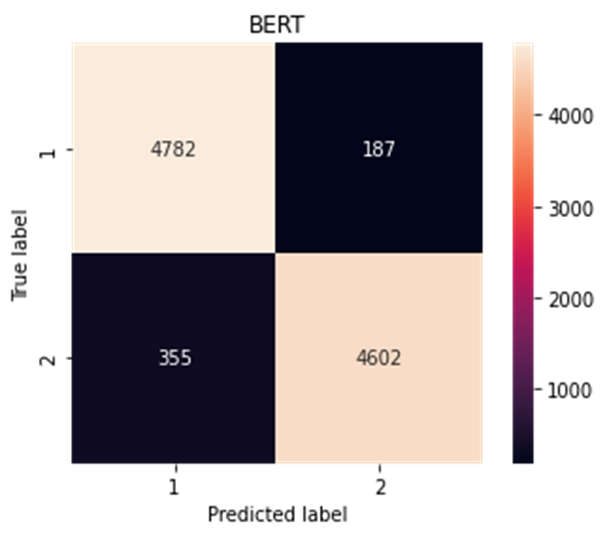
To save time only two epochs are created, meaning less passthroughs of the entire text, but much faster runtime considering the depth and resource intensity of BERT.

Even with these optimizations the program took an extremely long time to run and produce results, which is logical. Instead, the file was uploaded to GoogleColab, to use external processing power and produce the results shown below.

### Results

The BERT method produced the most accurate results out of all the other methods shown in the project. With an accuracy of 94.5% using bert-base-uncased pretrained model the program managed to produce 3.9% more accurate results than the LSTM method. Based on the confusion matrix, a small number of false positives and negatives can be seen, meaning that BERT can understand the sentiment behind the review and flag it correctly as a positive or negative review.

|  |  |  |  |
| --- | --- | --- | --- |
| Model: BERT tokenizer + BERT(bert-base-uncased) | | | |
| **Accuracy: 0.9454** |  |  |  |
|  |  |  |  |
|  | recall | f1-score | support |
| class 0 | 0.96 | 0.95 | 4969 |
| class 1 | 0.93 | 0.94 | 4957 |
|  |  |  |  |
| accuracy |  | 0.95 | 9926 |
| macro avg | 0.95 | 0.95 | 9926 |
| weighted avg | 0.95 | 0.95 | 9926 |



# Review Findings

Overall, we can say that the test was successful as the scores of the classifiers are rather good. If we compare our classifiers, we will notice that BERT has the best score, but it requires multiple processing power to operate, in contrast to the other classifiers which have scores over 85% and require dramatically less time to operate.

# Future Work/Applications

By proving the application of NLP methodologies to find the polarity of free text in the case of Amazon product reviews, there is a wide range of potential applications. As explained in this report these methodologies can be used to classify a product review as favorable or not, and by further refining the process, to decide if the review is helpful to other customers and highlight it on the product’s page. A slight variation of the work shown here is to further analyze the unfavorable reviews, to gain insight into the specific issues the previous buyers had with the product, to help with the product development and enhancement cycle.

In terms of the actual code written in the project, an effort could be made to increase the preprocessing steps to have a cleaner working dataset, for example remove the URLs and hashtags, clean up product names with extra characters, and in some cases remove emojis and other images/attached files. The runtime of each classification method is very hard to change, and in cases of executing the code in a home computer the runtime is very lengthy. This can be slightly improved by implementing a multiprocessing approach, but the recommendation still is to run the resource intensive parts of the code through GoogleColab. Finally, more classifiers could be used to have a larger variety of results such as Support Vector Machines (SVM), but due to hardware and time limitations, they were not examined in this project.

# Conclusions

In conclusion, this project has shown that by using different NLP models we can accurately predict the sentiment of a text. This can be used to find the polarity of a product review in a large online store like Amazon, and specific reviews can be identified as positive or negative. This can be further applied to achieve the original goal of the project and display the most useful reviews to prospective customers. In a more practical view, through our project we create a basis to be able to match the appropriate review to the appropriate user.