National and Kapodistrian University of Athens

Department of Informatics and Telecommunications

Computational Linguistics - Assignment report
Summer semester 2023-2024

Student 1	Triantafyllou Thanasis
	7115132300010
Student 2	Tsiompikas Dimitris
	7115112300036

Introduction

For this assignment, we have performed a series of experiments on a dataset that contains 19000 electronic product reviews from Amazon which you may find here: (https://www.kaggle.com/datasets/magdawjcicka/amazon-reviews-2018-electronics/data?select=electronics_sample.csv)
We wanted to see how some ML models and state-of-the-art LLMs would perform on a small dataset for sentiment analysis. We added labels to the above dataset with the following formula:

1. If the rating was below 3 stars it was negative.
2. If the rating was 3 stars it was neutral.
3. If the rating was above 3 stars it was positive.

and used it for our models.
In order to run our jupyter notebooks we used Kaggle's resources to take advantage of the free CPUs and the GPUs it provides for the LSTM and BERT/DistilBERT models.

Lastly, we created visualizations to highlight the distribution of the reviews and commonly used words in the reviews. Below you will find details for each model, the pre-processing techniques used and the results of the models.

Disclaimer: due to the small amount of neutral reviews and the small size in general of the dataset , the performance of the models is above average but it's still pretty low for their capabilities.

Data Pre-processing

For preprocessing we used the following commonly used NLP techniques:

- Removal of stop words
- Removal of special characters (such as exclamation marks , commas etc)
- Lower-casing of the text data
- Lemmatization
- Removal of numbers

We also cleaned the dataset from the so called NaN values which cause harm to the training of our models.

Data splitting for cross-validation was 80% training data and 20% testing data for our dataset.

<u>Models</u>

<u>Logistic Regression</u>

In our analysis, we employed a Logistic Regression model to classify the sentiment of reviews in our dataset. We transformed the review text into a high-dimensional vector space using a HashingVectorizer, and then split the dataset into training and testing sets. The Logistic Regression model was trained on the training set with the Stochastic Average Gradient Accelerated solver (saga) , set to handle a multinomial loss, and allowed a maximum of 1000 iterations for the solver to converge. The performance of the model was evaluated on the test set. The confusion matrix shows that the model made a total of 1248, 175, and 1253 correct predictions for ‘Negative’, ‘Neutral’, and ‘Positive’ sentiments. The classification report shows us a more detailed view of the model’s performance. The model achieved an overall accuracy of **<u>67%</u>**. The precision, recall, and F1-score for each class tells us how well the model performed for each sentiment category. The model had a high recall of 79% for both ‘Negative’ and ‘Positive’ sentiments and it was able to correctly identify a high proportion of these sentiments. However, it had a lower recall of 22% for ‘Neutral’ sentiments, that tells us it had difficulty in correctly identifying neutral reviews.

<u>SVM Model</u>

Support Vector Classifiers are known for their incredibly good performance on classification tasks such as sentiment analysis hence this model could not have been left out from our experiments. Similarly as above, we used the Hashing vectorizer to turn the text data into numerical inputs for the model training part. We used the Linear SVC variant of SVMs which uses a Linear kernel and regularization to combat overfitting, providing the best performance out of all other SVMs. Due to the fact that we had few neutral reviews the SVM underperformed on our case and produced an overall accuracy of **<u>66%</u>**.

<u>K-NN model</u>

We also used the k-Nearest Neighbors (k-NN) algorithm to classify the sentiment of reviews in the dataset. The model was trained on the training set with k set to 90, meaning that the algorithm considered the 90 nearest neighbors in the feature space when making a prediction. The performance of this model was also evaluated on the test set. The confusion matrix shows that the model made a total of 1265, 84, and 1100 correct predictions for ‘Negative’, ‘Neutral’, and ‘Positive’ sentiments. The classification report gives us a more detailed view of the model’s performance. The model made an overall accuracy of **<u>62%</u>**. The precision, recall, and F1-score for each class tells us how well the model performed for each sentiment category. The model had a high recall of 80% for ‘Negative’ sentiments, telling us that it was able to correctly identify more of these sentiments. However, it had a lower recall of 10% for ‘Neutral’ sentiments, making it more difficult in correctly identifying neutral reviews.

<u>LSTM model</u>

After the classical ML algorithms we went to more “heavy” methods such as Deep Learning in this case and LLMs. Starting off with the deep learning part, we used a Long-Short Term Memory (LSTM) neural network that has the ability to retain information from previous neurons and pass them forward to the next ones for better sequence classification.
Here, to make things a bit easier for the model we used word2vec word vectors as input data which are pre-trained, we also used a label encoder to encode our labels to numerical form so we can match it later on with the predictions from the LSTM and we created the architecture of the Neural Network using the PyTorch library.
Later on, we created a one-epoch iteration to create a learning curve for the LSTM and finally we performed a 50-epoch run to fine-tune our model to the training data and obtained the predictions which gave a result of **<u>60%</u>**, being pretty underwhelming for a neural network.

<u>BERT model</u>

Furthermore, we used two state-of-the-art LLMs to measure their performance in this small dataset. Despite being pre-trained in English and us fine-tuning the LLMs for our dataset they surpassed the other models but they still had underwhelming performance in general.
To begin with, we used BERT initially to observe its performance on the dataset. We used the BertTokenizer and the bert-base-uncased transformer from HuggingFace , we encoded our data to be supported by BERT by using PyTorch and fine-tuned the model in 2 epochs. BERT had a slow training time, it took 24-25 minutes for each epoch to finish , leading to a 55 minutes total time for training. This model had the best performance of all , reaching **<u>69%</u>** in overall accuracy.

<u>DistilBERT model</u>

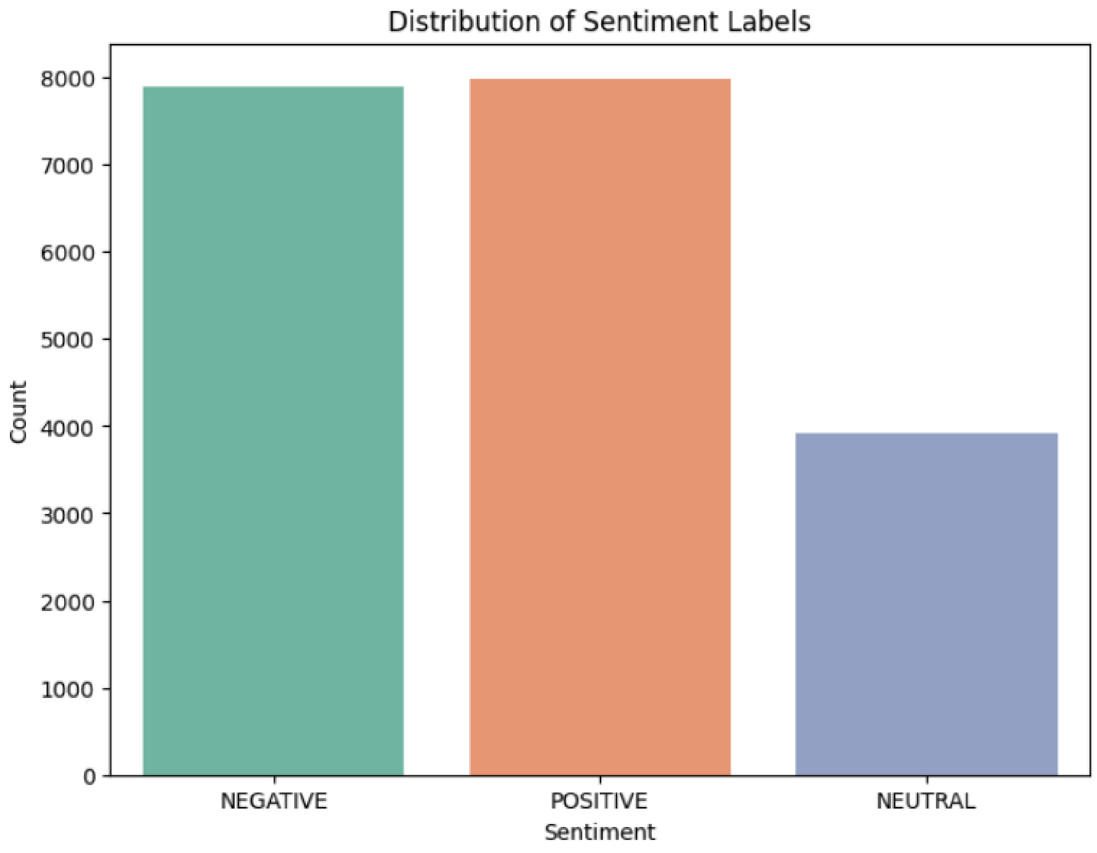
Lastly, we used DistilBERT from HuggingFace to also test its performance on the dataset. DistilBERT is a distilled version of BERT, offering a smaller and faster alternative while retaining much of BERT's performance. It's suitable for sentiment analysis applications where computational resources are limited such as in our case. We followed the same procedure as above and this model had faster training times , 12-13 minutes to be fair, literally half the time of BERT, but as expected the performance was worse, reaching an overall accuracy of **<u>66%</u>**.

<u>Results Table</u>

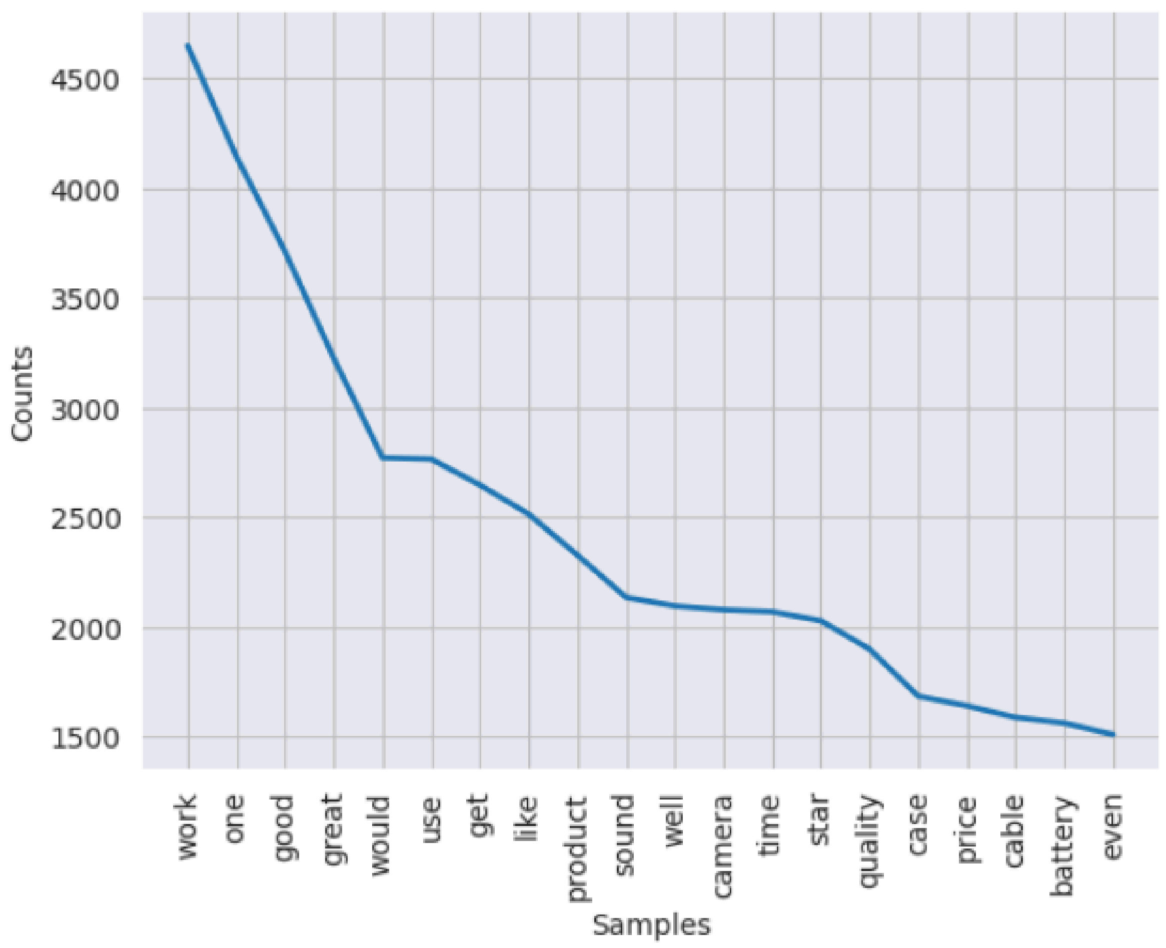
<u>Model</u>	**<u>Precision</u>**	**<u>Recall</u>**	**<u>F1-score</u>**	**<u>Accuracy</u>**
Logistic Regression	65%	67%	67%	67%
SVM	63%	66%	66%	66%
k-NN	64%	61%	61%	61%
LSTM	60%	60%	60%	60%
BERT	**67%**	**69%**	**69%**	**69%**
DistilBERT	66%	67%	67%	67%

Visualizations

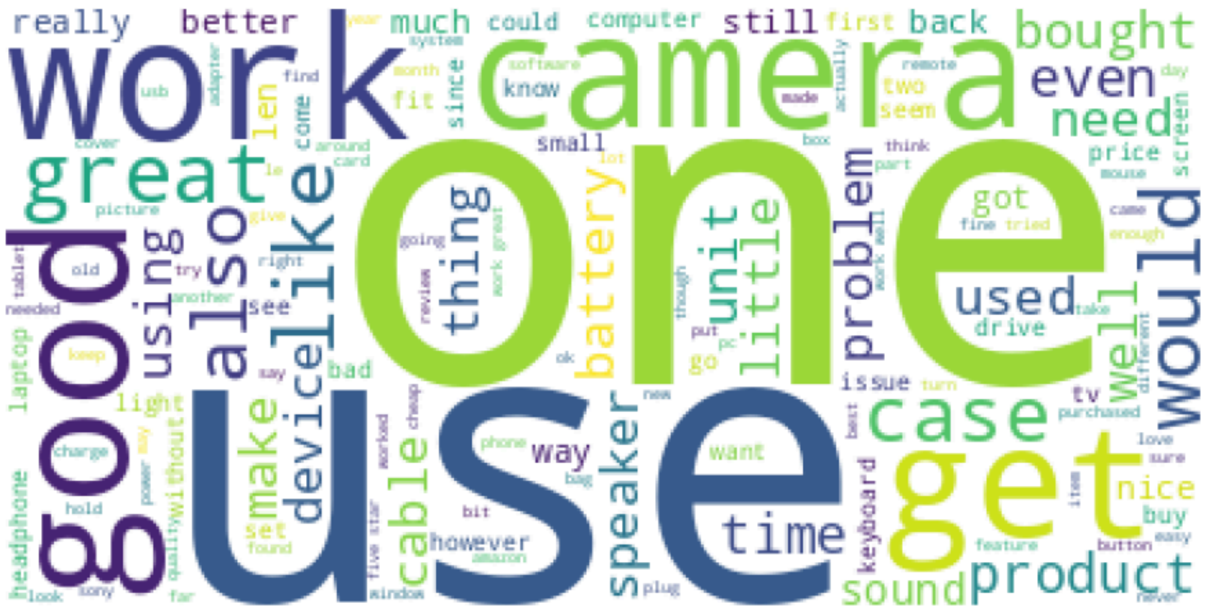
Below you will find our visualizations for this dataset:



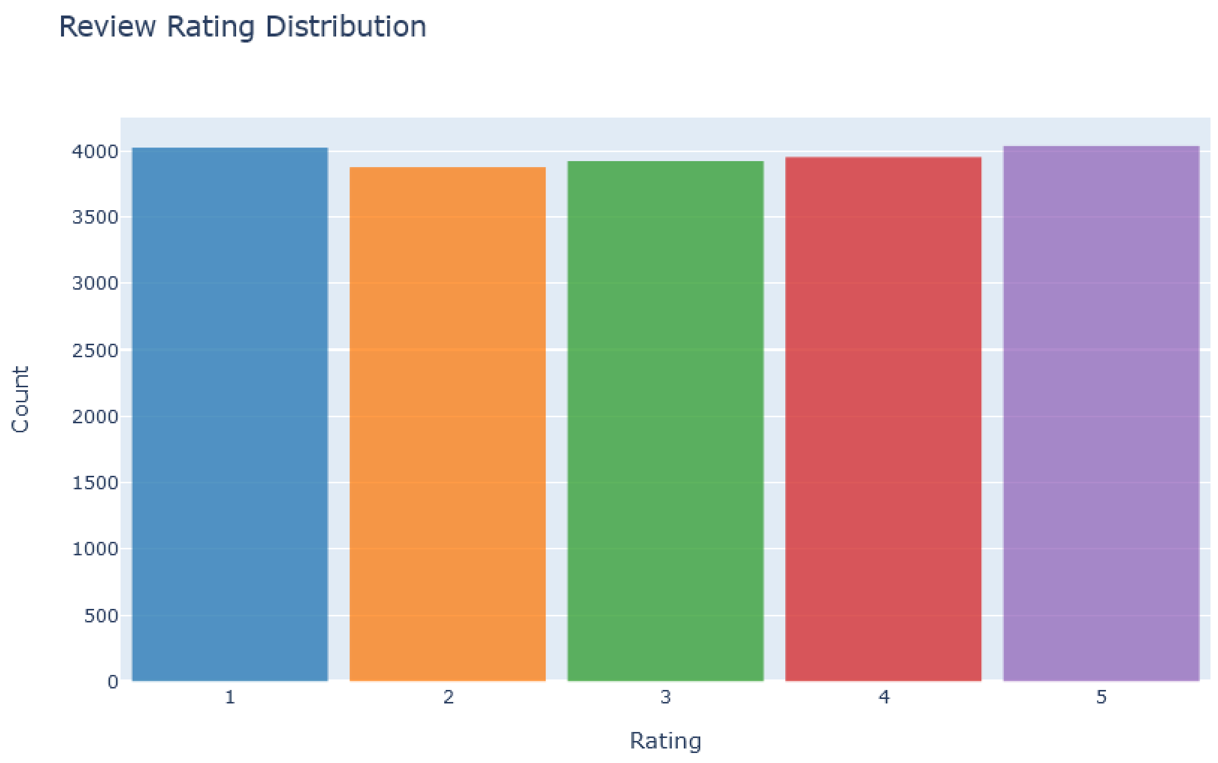
Here we can see that there is an uneven distribution of neutral reviews compared to the other two labels, which caused problems to our models.



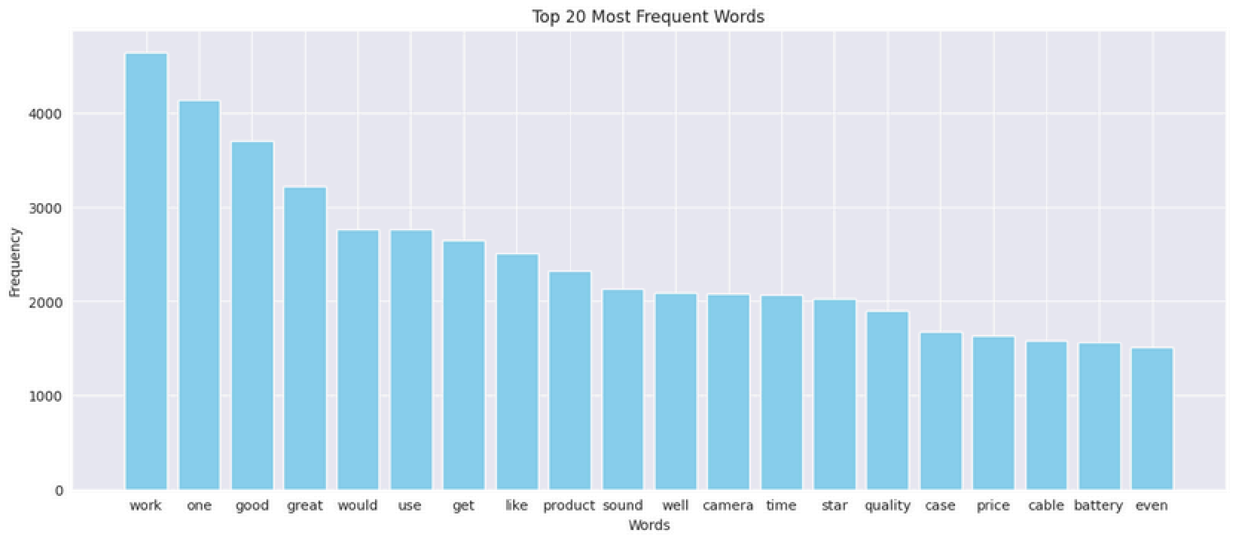
Here we observe that words like “good, great” etc for the positive sentiment, and words like camera, price, cable, battery etc for electronic products are prevalent in the dataset.



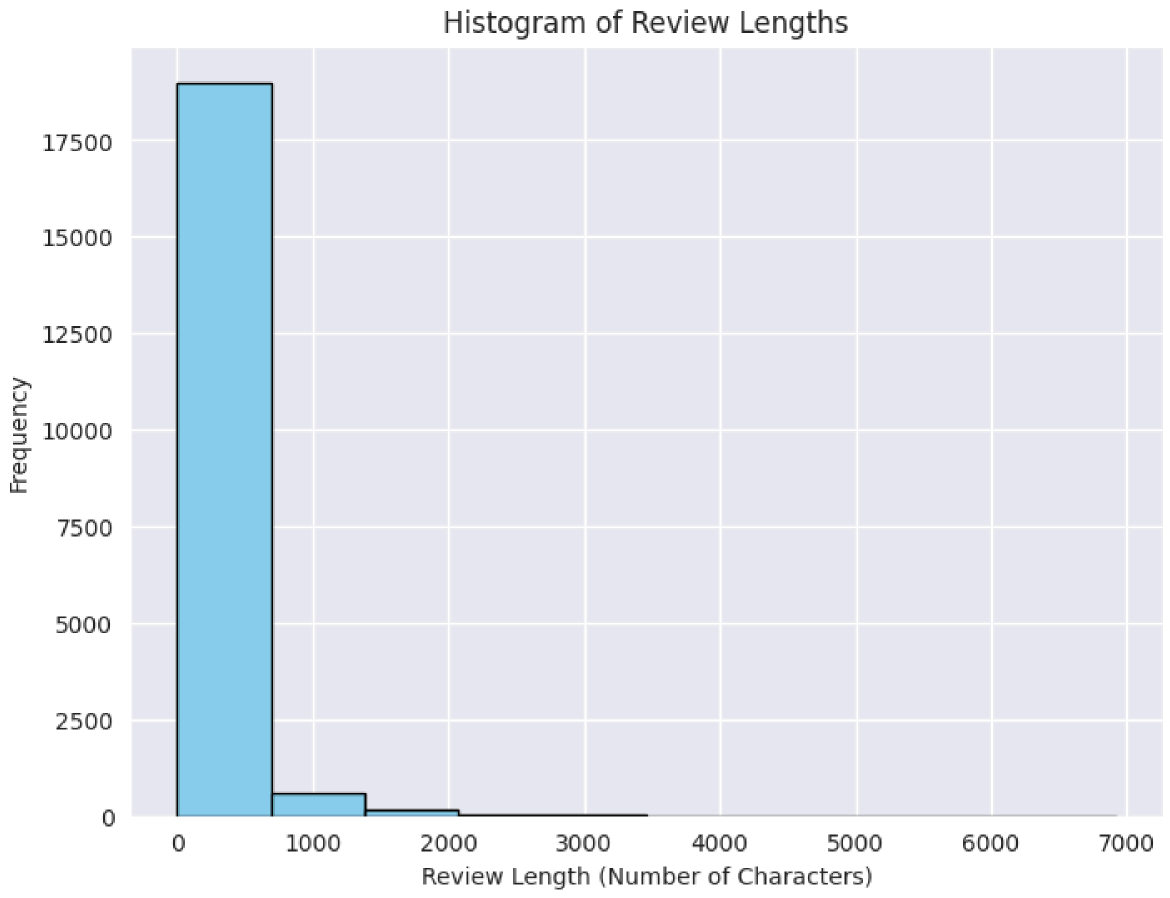
Same as above , lots of words used for reviews and electronic products.



Here we can see the number of reviews by rating, neutral reviews had a number of 3000 while negative and positive had a number of 7998 and 7889 respectively.



Here we can see the top 20 most frequent words in bar chart form.



Lastly, we can see that the length of the reviews wasn’t all that high after all which is to be expected from review text data.