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

1. Introduction

Sentiment analysis, the process of determining the emotional tone behind a series of words, is a critical tool in understanding and interpreting textual data. With the exponential growth of user-generated content on social media, reviews, and other online platforms, the ability to automatically analyze sentiment has become increasingly valuable for businesses, researchers, and policymakers. This report delves into an experimental comparison of several machine learning (ML) models to evaluate their effectiveness in sentiment analysis and particularly on electronic product reviews.

In this experiment, we tested a variety of models, including traditional approaches such as logistic regression and support vector machines, k-Nearest Neighbors and state-of-the-art deep learning techniques, specifically Long Short-Term Memory (LSTM) networks and Bidirectional Encoder Representations from Transformers (BERT) along with its distilled version, which is faster at the cost of performance, DistilBERT. Our objective was to identify which model offers the best performance in terms of accuracy, precision, recall, and overall robustness.

The analysis involved comprehensive preprocessing of text data, model training and evaluation, and the use of visualizations to interpret data distribution and the most frequent words. By leveraging tools like distribution plots, token frequency diagrams and word clouds, we aimed to gain a deeper understanding of the dataset and how our models would perform on it.

This introduction sets the stage for a detailed exploration of the methodologies employed, the experimental setup, and the results obtained, providing a clear pathway to the insights and conclusions drawn from this comparative study of sentiment analysis models. We also “pave the way” for newer LLMs to be used for this task, such as GPT and Llama, on electronic product review datasets.

2. Theoretical Framework

This project will be used to study how human emotions are expressed through language choices. As mentioned in the book of Jurafsky and Martin [1] emotion is a relatively brief episode of response to the evaluation of an external or internal event as being of major significance. Emotions can be expressed in several ways, sometimes with unclear words that make it very hard for sentiment analysis models to detect emotion properly. Taken from [2] , sentiment analysis also helps businesses identify trends from the data obtained from customer surveys and reviews for their products.

Through technological advancements, Large Language Models such as BERT and RoBERTa have made big leaps in discovering sentiment in a multitude of texts but they require a very large amount of data to be able to provide researchers with good results. Newer LLMs such as GPT and Llama also provide good results but the usage of the better fine-tuned models is limited behind a pay-wall.

2.1 Computational Linguistics

Computational linguistics is an interdisciplinary field that combines insights from computer science and linguistics to study how language can be modeled and processed using computers. In the book of Jurafsky and Martin [1] several fields and techniques are mentioned, including the one we're studying in this project, sentiment analysis, and the following: Machine Translation, Dialogue systems/Chat bots, Information retrieval and Speech recognition. Techniques include using Machine Learning algorithms to obtain results (such as classification algorithms like Naive Bayes and Support Vector Machines for sentiment analysis), Deep Learning methods (RNNs, LSTMs) and the state-of-the-art LLMs (BERT, RoBERTa, GPT, Llama etc.).

2.2 Application Domain

The domain of application from Computational Linguistics for this project will be sentiment analysis. Sentiment analysis, also known as opinion mining, is a field within natural language processing (NLP) that focuses on determining the sentiment expressed in text. This can range from identifying simple binary sentiments (positive or negative) to more nuanced feelings such as joy, anger, sadness, and others. We'll be trying to identify Positive, Neutral and Negative sentiment in our case.

2.2.1 Linking with Semantics

Sentiment analysis is closely linked with semantics, as understanding the meaning and context of words, phrases, and sentences is crucial for accurately determining sentiment. Here are several ways in which sentiment analysis and semantics intersect, some of which are also mentioned in the book of Jurafsky and Martin [1]:

Contextual understanding: which involves recognizing how words and phrases interact within a sentence or larger text to convey specific meanings.

Semantic Role Labeling (SRL): SRL assigns roles to words in a sentence (e.g., who did what to whom). An example for an electronics product review would be: “I really like this phone” meaning I as the subject/agent express positive sentiment towards the phone (object).

Sarcasm and Irony Detection: Sarcasm and irony involve using words to convey a meaning opposite to their literal interpretation. An example for electronic product reviews would be: “Oh wow, this phone is great!! Top tier battery life!!” and having 0/5 stars which would indicate sarcasm.

Negation and Modifier handling: Negations and modifiers change the meaning of a sentiment. For example, combinations like “Not good” (negation) or “very good” (modifier) can define if a text is negative or positive.

2.3 Description of the models

For this project, we’ll be using several classic Machine Learning algorithms, Deep Learning methods and state of the art LLMs to evaluate their performance on our dataset for Sentiment analysis. Models used are: Logistic Regression, k-NN, SVM, LSTM, BERT and DistilBERT. They will be evaluated on a test set and their performance will be measured using 4 metrics: Precision, Recall, F1-score and Accuracy , all of which are used in Machine Learning for this purpose.

2.4 Deep Learning

While classical Machine Learning models provide substantial results most of the time for sentiment analysis the newer deep learning models come into play and make the difference in performance , at the cost of computational resources, due to their ability to understand context better, retain words and they're also pre-trained, usually on large corpora of words that allow them to surpass classic models even without fine-tuning. Deep learning is a subfield of Machine Learning, and its name stems from the fact that neural networks are usually comprised of an input layer with input neurons, multiple hidden layers with neurons that process the input (hence the name deep) and one output layer with neurons that provide the output. All of machine learning can be characterized as learning to make predictions based on past observations, deep learning approaches work by learning to not only predict but also to correctly represent the data, such that it is suitable for prediction as mentioned in [3] Chapter 1.2.

2.4.1 LSTM neural networks

It is important to understand the underlying structure of the models used for Deep Learning since they're the most used for text classification problems such as Sentiment Analysis. LSTMs (or Long-Short-Term-Memory neural networks) are the most successful neural network architecture for NLP currently. They are designed to remember long-term dependencies in sequential data, overcoming the limitations of traditional RNNs that struggle with long-term dependencies due to vanishing and exploding gradients. They contain so called "memory cells" which hold data (sequences from text in our case) and making it ideal for sentiment analysis. ([3] Chapter 15.3.1)

2.4.2 BERT and DistilBERT

BERT is undoubtedly one of the most important inventions in Deep Learning since it is the predecessor of the GPT architecture used in LLMs founded in 2023 such as ChatGPT, Llama and Gemini. Founded by 4 Google AI researchers in 2019 [6] This model is based on the Transformer architecture which is comprised of an encoder which transforms (hence the name) the text data into “readable” data. More specifically, the encoder in BERT is responsible for generating contextualized representations of input text. These representations capture the meaning of words in context, allowing BERT to understand the nuances of language. The final output is a set of contextualized token embeddings, where each token’s embedding captures its context within the entire input sequence. BERT also brought to light the techniques of pre-training (using large text corpora to feed the model before actually training it on the dataset) and fine-tuning (training the model on the experiment’s dataset to get better results). DistilBERT is a “distilled” version of BERT. Distilled in Machine Learning means that a smaller model with fewer parameters and less required computational power tries to mimic the larger model.

3. Domain

The domain for this project will be electronic product review text data. We have found a dataset from Kaggle containing 19000 reviews from Amazon.

3.1 Text description

The texts in this dataset tend to be short and to the point , with users stating if they liked the product or not. Some reviews also express positive, negative or neutral sentiment with a plethora of words and different ways for example, for mobile phones and digital cameras users comment on the battery life, camera quality, price etc. Review length can be also seen from **Figure 9.**
Furthermore, as in every dataset, there are outliers in the dataset namely very long reviews with users expressing their likeness or hatred towards a product by describing every single detail including pros and cons, amount of usage (daily, weekly) etc and reviews with very few words like “Two stars”, “Five stars” , “crap”etc. Most prevalent words as shown in **Figures 5 and 8** include positive sentiment words such as “good”, “great” and words that relate to electronic products such as “battery”, “camera”, “price”, “case” and “sound”.

3.2 User description

Users in this dataset , on average, are people who simply want to help other customers by sharing their opinions on the products. There are very few outliers on this part too, with people writing incomprehensive words or words that provide no information whatsoever for the quality of a product , making their reviews useless for the models.

4. Methodology - Approach

4.1 Development problems

While testing our dataset we noticed that the distribution was not even, because we had less neutral reviews than the other categories as mentioned above. This led us to obtain more data in order to create a more concise dataset which contained 60000 reviews in an even distribution (20000 from each label), however even with this dataset, the models still showed inadequate performance and the BERT models could not run due to the hardware limitations. Therefore, we stuck to the initial plan and used the first dataset despite its weaknesses.

4.2 Data Pre-processing

For preprocessing we used the following commonly used NLP preprocessing 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

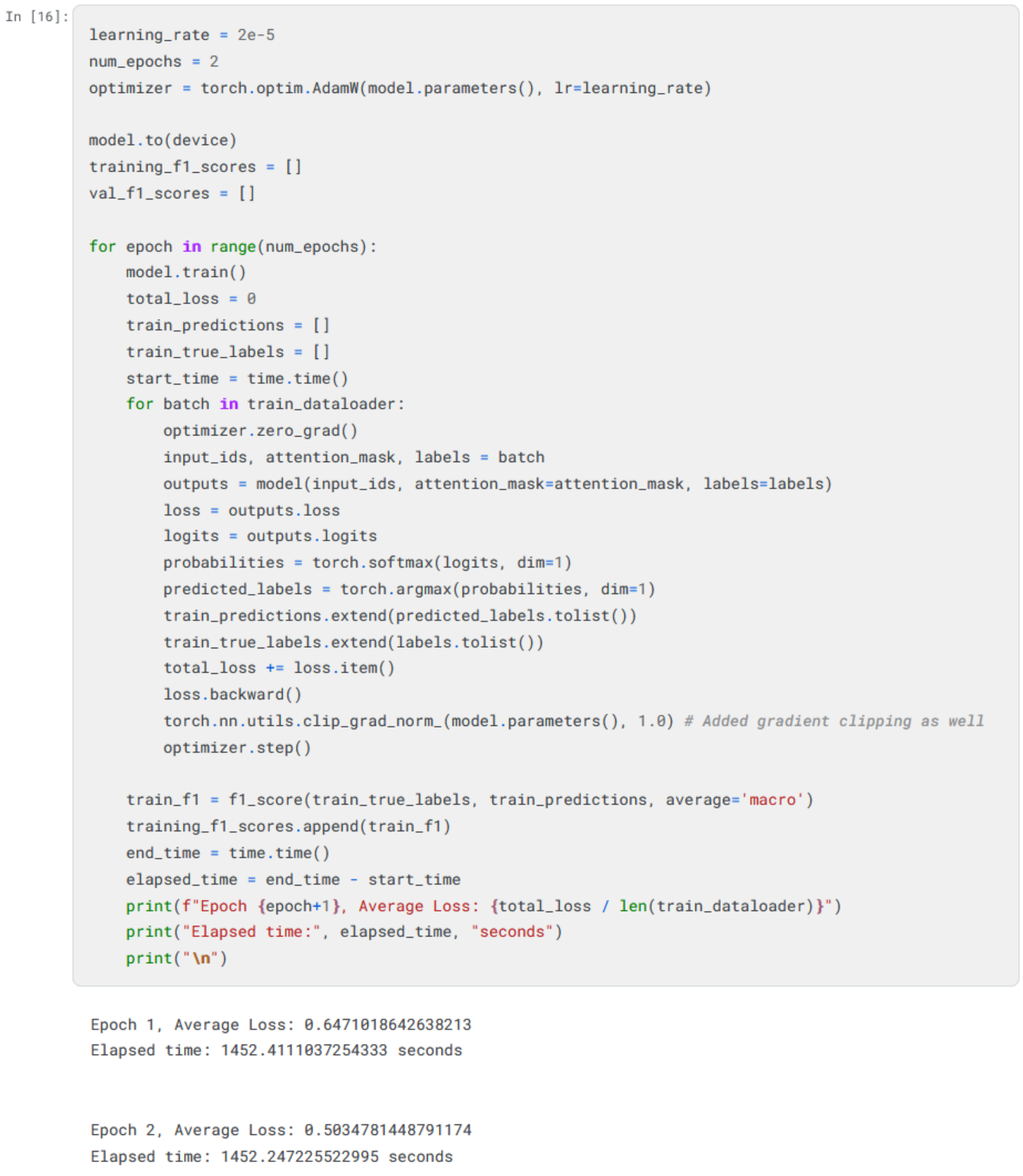
We also cleaned the dataset from the so called NaN values which cause harm to the training of our models.

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

Further knowledge on these techniques can be found at [4] and [5].

4.3 Interaction and Coding

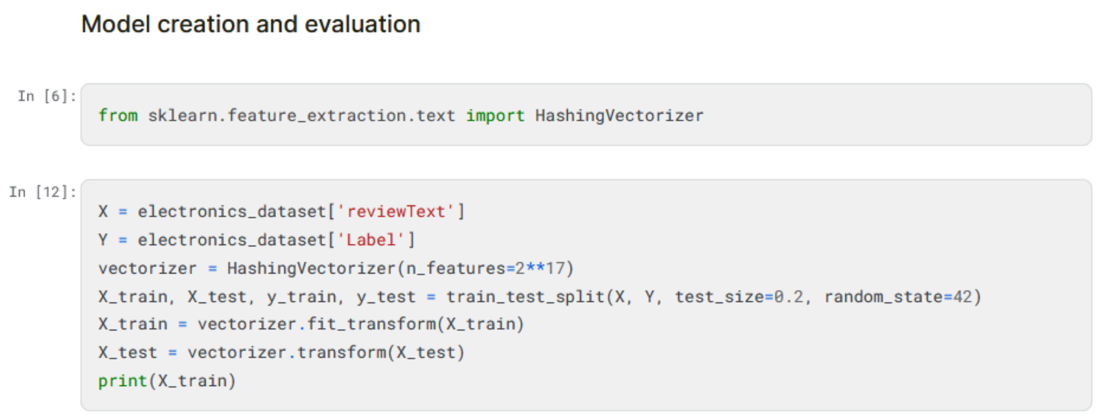
Below we'll be showcasing some screenshots of the code along with the results:



```
In [16]:
learning_rate = 2e-5
num_epochs = 2
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)

model.to(device)
training_f1_scores = []
val_f1_scores = []

for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    train_predictions = []
    train_true_labels = []
    start_time = time.time()
    for batch in train_dataloader:
        optimizer.zero_grad()
        input_ids, attention_mask, labels = batch
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        logits = outputs.logits
        probabilities = torch.softmax(logits, dim=1)
        predicted_labels = torch.argmax(probabilities, dim=1)
        train_predictions.extend(predicted_labels.tolist())
        train_true_labels.extend(labels.tolist())
        total_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0) # Added gradient clipping as well
        optimizer.step()

    train_f1 = f1_score(train_true_labels, train_predictions, average='macro')
    training_f1_scores.append(train_f1)
    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"Epoch {epoch+1}, Average Loss: {total_loss / len(train_dataloader)}")
    print("Elapsed time:", elapsed_time, "seconds")
    print("\n")
```

```
Epoch 1, Average Loss: 0.6471018642638213
Elapsed time: 1452.4111037254333 seconds


Epoch 2, Average Loss: 0.5034781448791174
Elapsed time: 1452.247225522995 seconds
```

Figure 1: Execution of BERT's training code. Here you can see the big amount of time it took for 19000 reviews. If any more were added training time would dramatically increase.

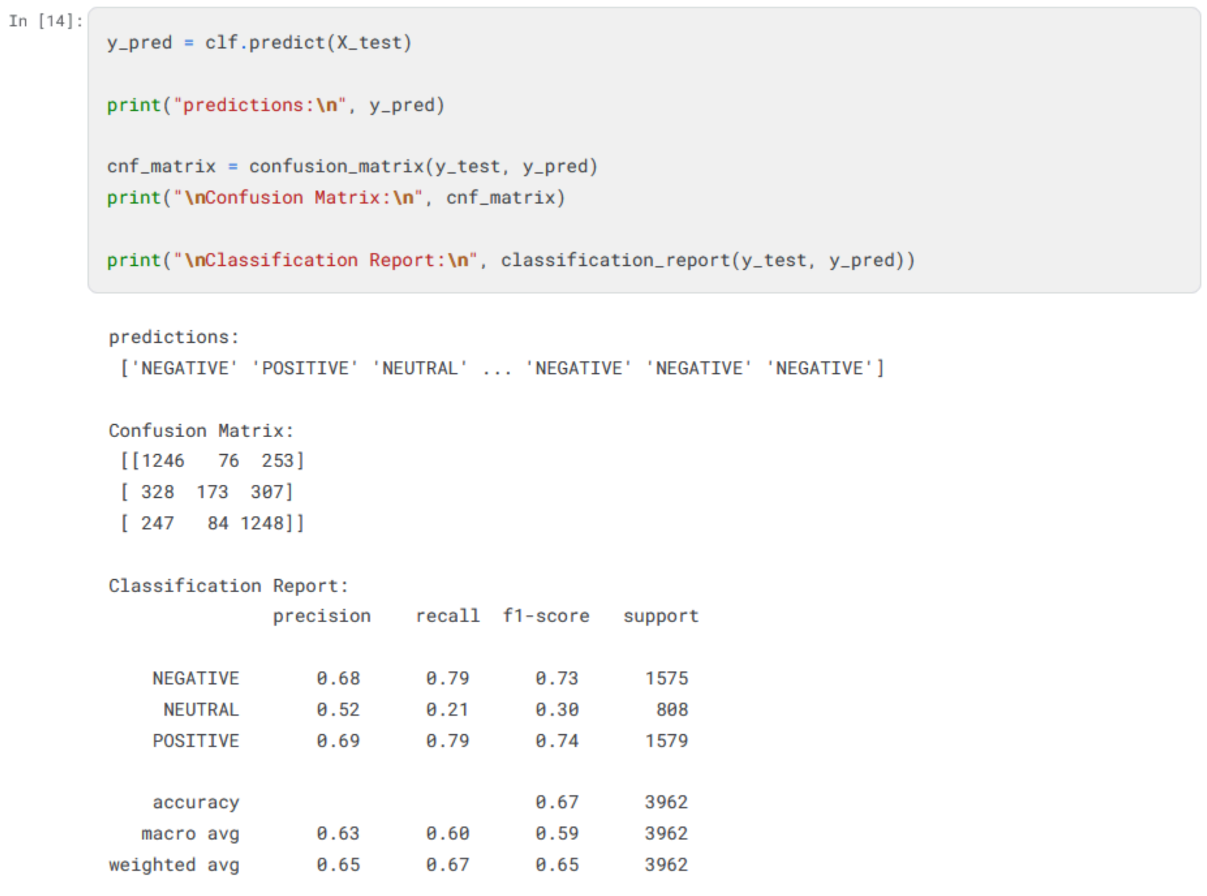


Model creation and evaluation

In [6]:
```
from sklearn.feature_extraction.text import HashingVectorizer
```

In [12]:
```
X = electronics_dataset['reviewText']
Y = electronics_dataset['Label']
vectorizer = HashingVectorizer(n_features=2**17)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
print(X_train)
```

Figure 2: Usage of Hashing Vectorizer on Logistic Regression code, providing us with the best numerical inputs for our text.



In [14]:
```
y_pred = clf.predict(X_test)

print("predictions:\n", y_pred)

cnf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", cnf_matrix)

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
predictions:
 ['NEGATIVE' 'POSITIVE' 'NEUTRAL' ... 'NEGATIVE' 'NEGATIVE' 'NEGATIVE']

Confusion Matrix:
 [[1246   76  253]
 [ 328  173  307]
 [ 247   84 1248]]

Classification Report:
               precision    recall  f1-score   support

    NEGATIVE       0.68      0.79      0.73      1575
     NEUTRAL       0.52      0.21      0.30       808
    POSITIVE       0.69      0.79      0.74      1579

    accuracy                           0.67      3962
   macro avg       0.63      0.60      0.59      3962
weighted avg       0.65      0.67      0.65      3962
```

Figure 3: Results of Logistic Regression, the apparent misinterpretation of neutral review data is apparent in the classification report.

5. Analysis - Implementation

5.1 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. This can be seen in **Figures 4 and 7**

5.2 Models

5.2.1 Logistic Regression

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 **67%** as shown in **Table 1**. 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.

5.2.2 SVM Model

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 **66%** as shown in **Table 1**.

5.2.3 K-NN model

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>** as shown in **Table 1**. 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.

5.2.4 LSTM model

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>** as shown in **Table 1**, being pretty underwhelming for a neural network.

5.2.5 BERT model

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 **69%** in overall accuracy as shown in **Table 1**.

5.2.6 DistilBERT model

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 **66%** as shown in **Table 1**.

5.3 Results - Visualizations

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
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%

Table 1: Experiment results for each model.

Below you will find our visualizations for this dataset:

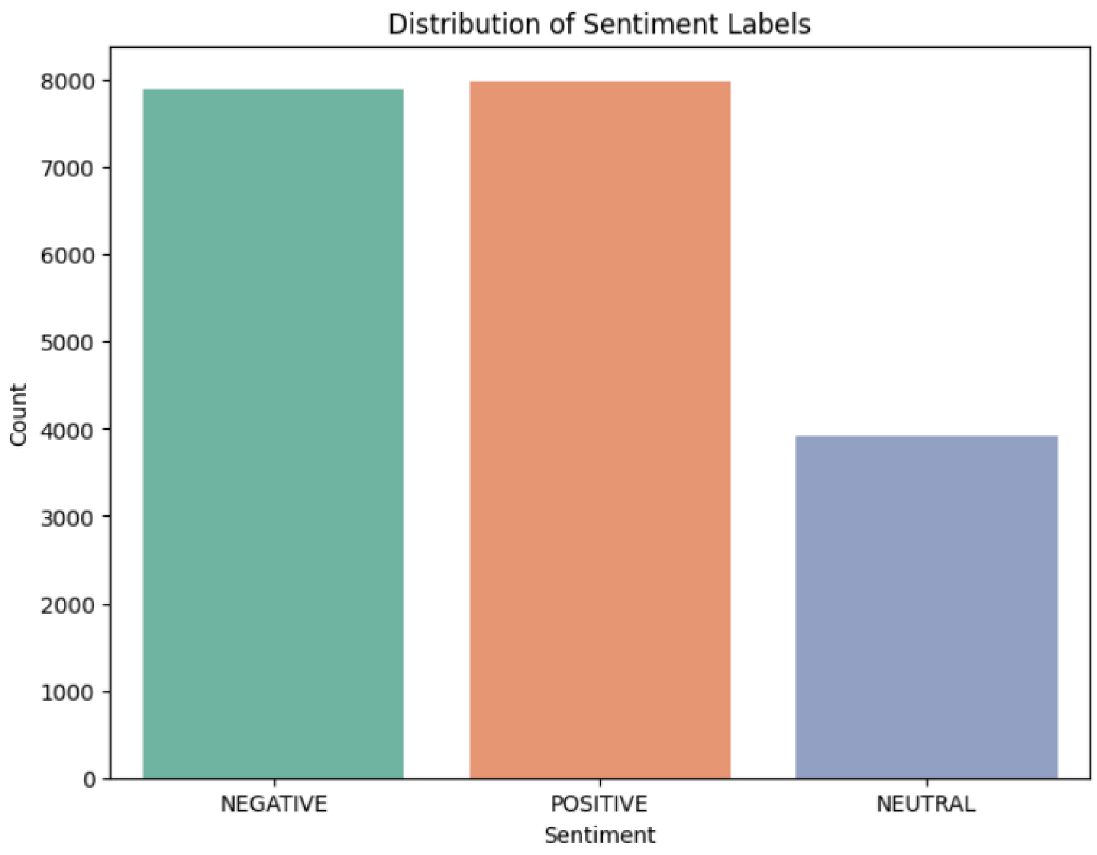
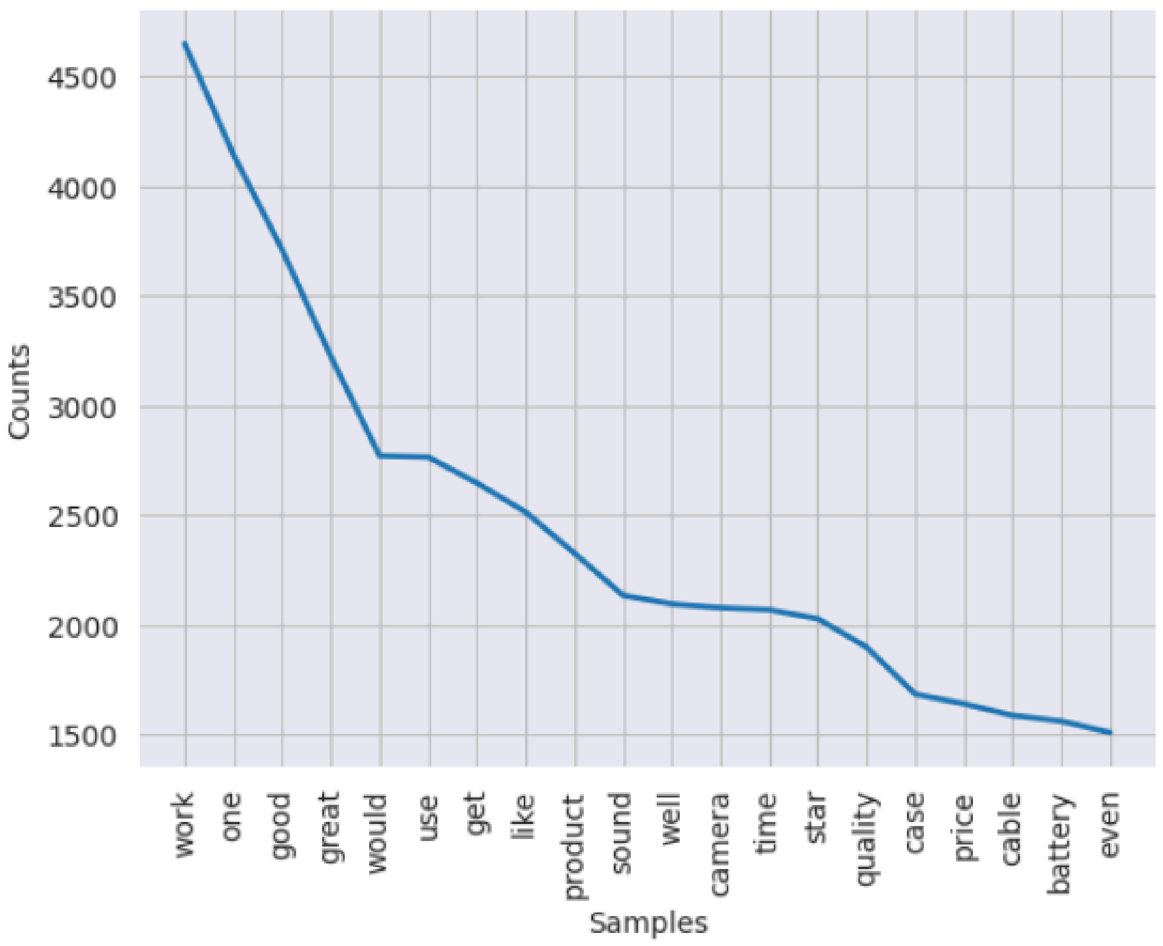


Figure 4: Label distribution in the dataset.



<u>Figure 5:</u> Token frequency diagram showing the top 20 most frequent words.

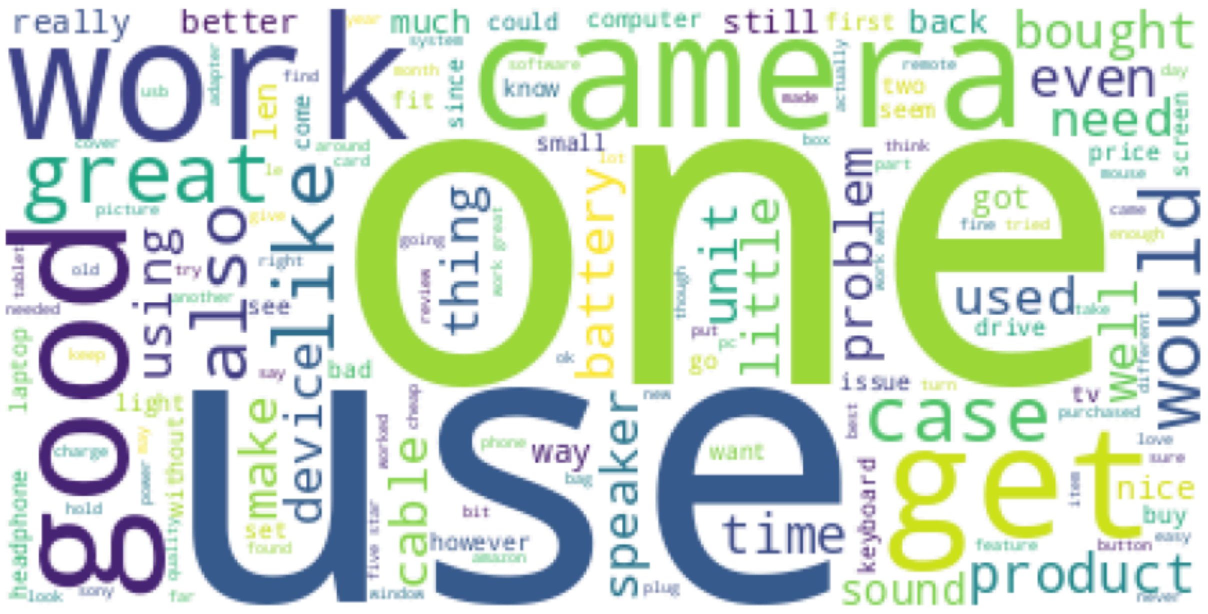


Figure 6: Word cloud with the most frequent words found in the dataset.

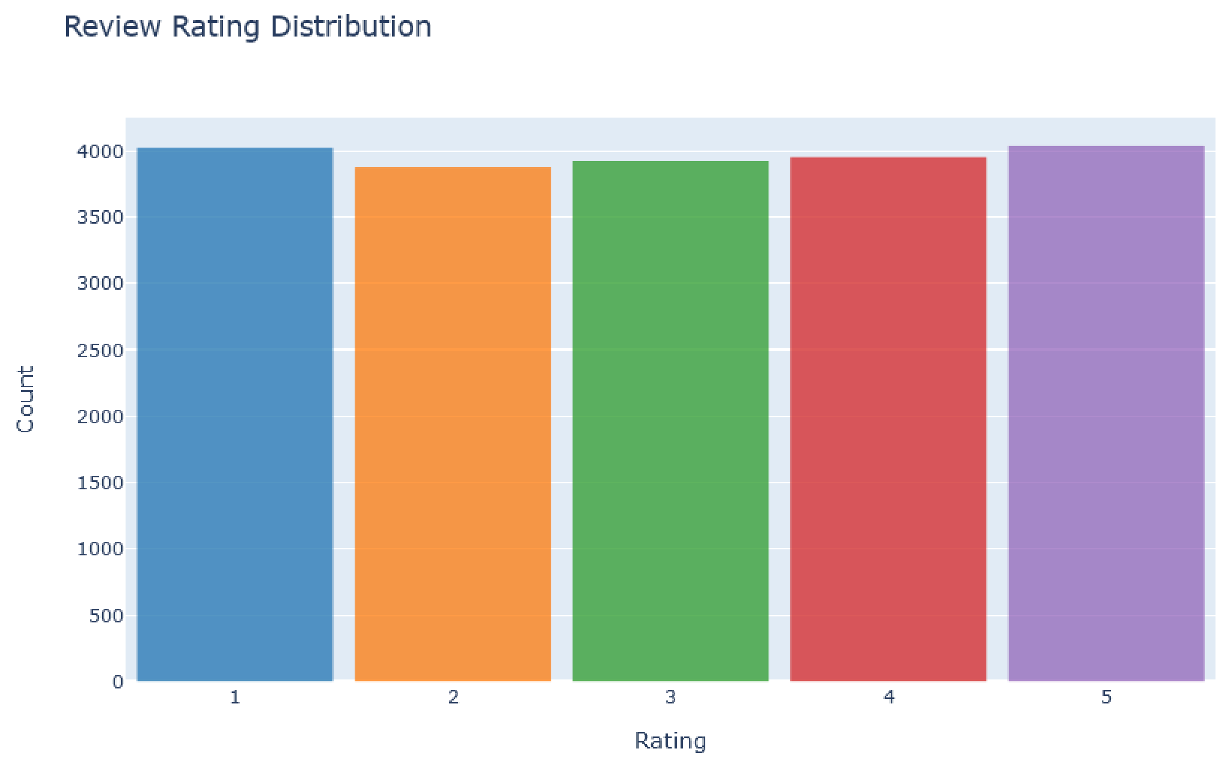
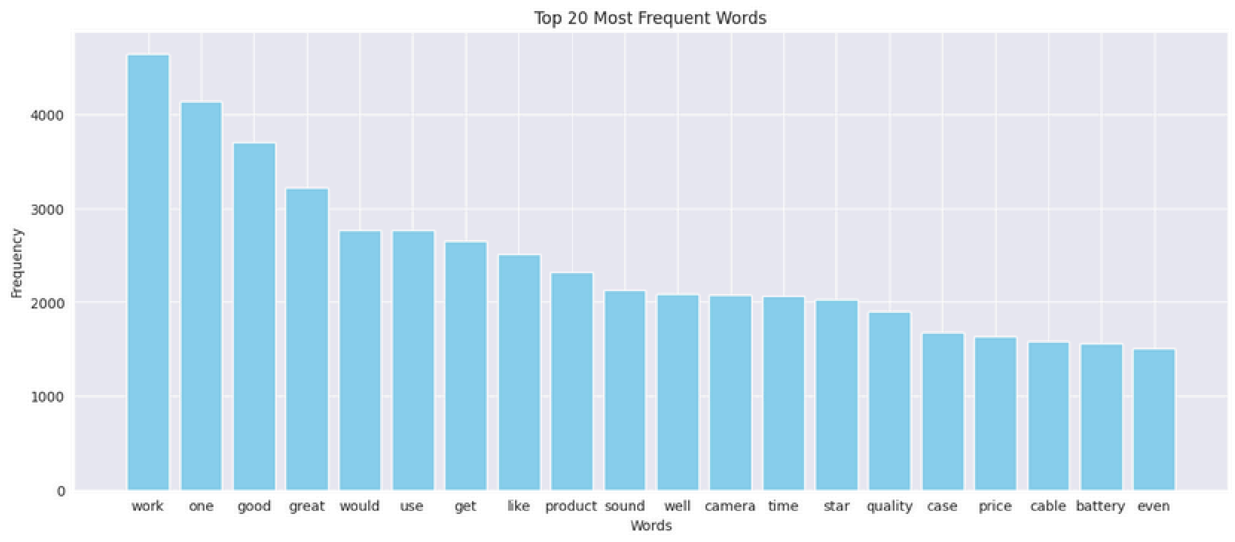
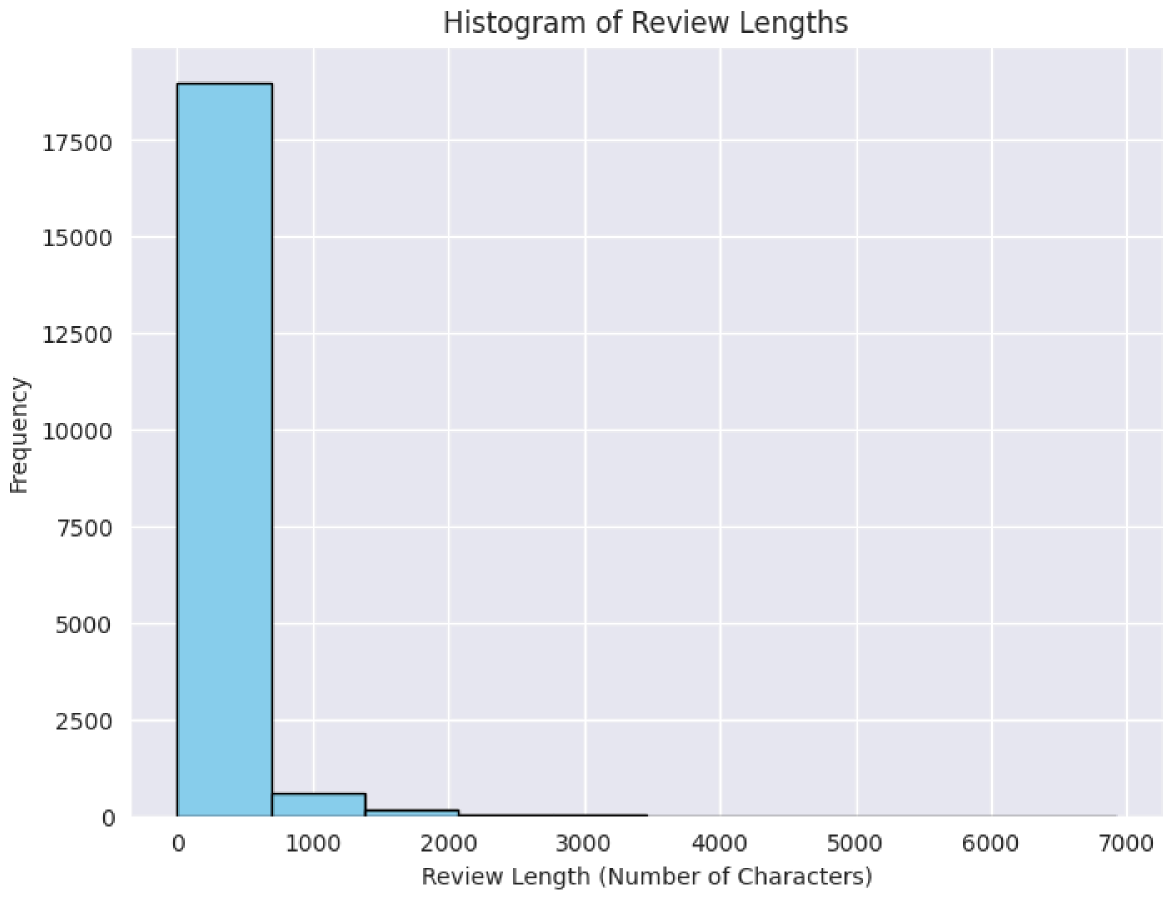


Figure 7: Review rating distribution in bar chart form.



<u>Figure 8:</u> Top 20 most frequent words in bar chart form.



<u>Figure 9:</u> Length of reviews by number of characters.

6. Conclusions and future work

Through this experiment, we explored sentiment analysis on a dataset with electronic product reviews. After analyzing Computational Linguistics and sentiment analysis on a theoretical level, we began experimenting on our dataset. We pre-processed the data through various ways and removed all values that could cause issues with training. The models tested included logistic regression, support vector machine, k-Nearest Neighbors, and deep learning approaches such as LSTM, BERT and DistilBERT. The classic ML models provide robust performance while models leveraging Deep Learning have a higher accuracy. BERT showed the best performance of all, scoring 69% accuracy, due to its ability to capture contextual nuances in language, thereby outperforming other models in both precision and recall metrics.

Due to this, BERT is suggested for datasets with complex sentiment variations, including product reviews for electronics such as this one. Visualizations showed that the distribution of data was uneven and lead to the results we obtained from the experiments. Using a dataset with better distribution among labels , however, did not yield better results and the models' performance remained at the same levels.

Future work may focus on improving the data pre-processing strategy with more or different techniques like Stemming instead of Lemmatization, bi-grams etc., further optimization of the models used in the experiments with techniques like Gradient clipping or Early stopping for the LSTM, more epochs for the BERT models and testing them on other product-review-related datasets.

Lastly, the latest state-of-the-art LLMs could be used to see their performance on this dataset (example: GPT, Llama or Gemini) , fine tune them on it and see if they can identify sentiment with better performance.

7. References

[1]
Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Daniel Jurafsky, James H. Martin Third edition, draft of 2023

[2]
https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17, Sentiment Analysis: Concept, Analysis and Applications, Shashank Gupta, Jan 7, 2018

[3]
https://github.com/Michael2Tang/ML_Doc/blob/master/Neural%20Network%20Methods%20in%20Natural%20Language%20Processing-Morgan%20%26%20Claypool%20Publishers%20(2017)%20-%20Yoav%20Goldberg%2C%20Graeme%20Hirst.pdf , committed on Github by user Michael2Tang, used in the AI 2 course taught on DIT UoA. Download it. Mar 15, 2018

[4]
https://exchange.scale.com/public/blogs/preprocessing-techniques-in-nlp-a-guide , A Guide to Text Preprocessing Techniques for NLP, Mehreen Saeed, February 2, 2022

[5]
https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-language-processing-nlp-a-beginners-guide-d6d9bf7689c9 , Preprocessing Steps for Natural Language Processing (NLP): A Beginner’s Guide, Maleesha De Silva, Apr 30, 2023

[6]
https://arxiv.org/abs/1810.04805 , BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 24 May 2019