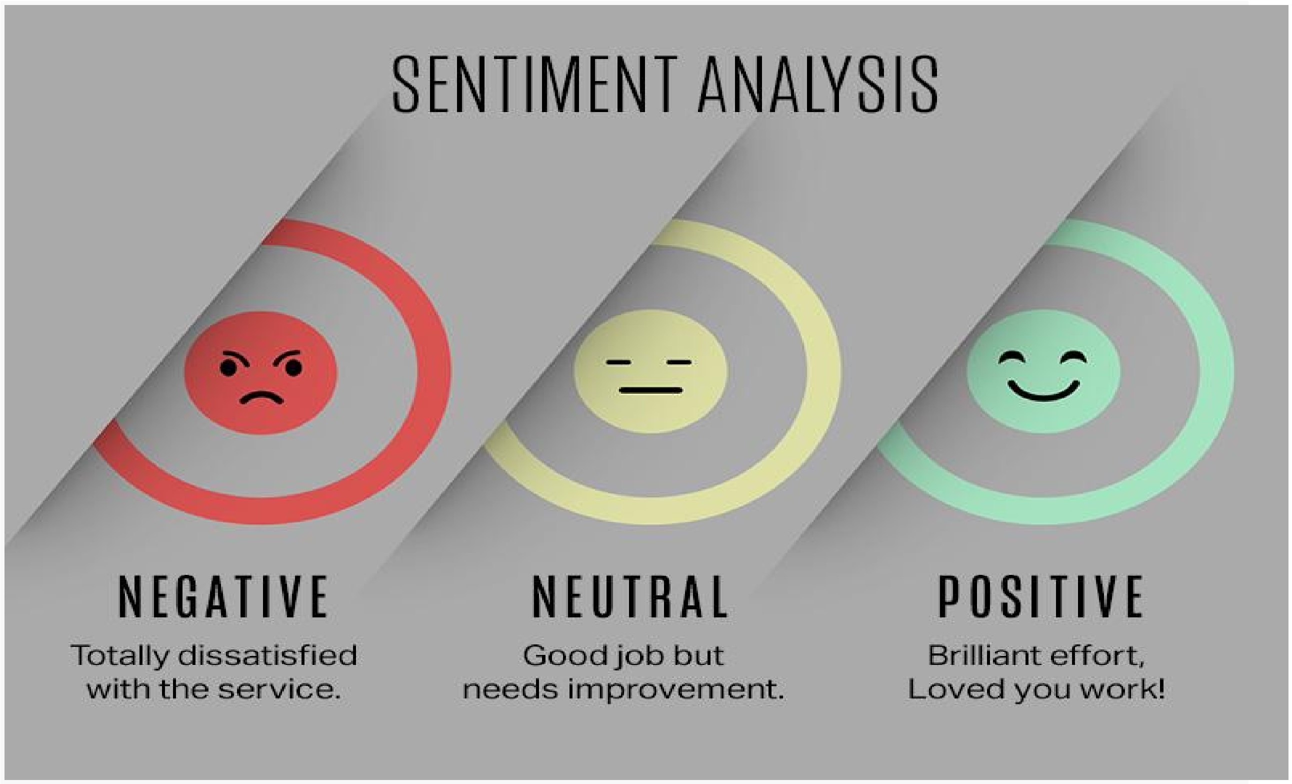
Enterprise Review Sentiment Analysis



Team: Data Divers
Team Members:
Ravi Teja Reddy(016693196)
Sanjay Bhargav(016421587)
Harika Satti (016260504)
Sai Pragna(016698552)

Enterprise Review Sentiment Analysis

By Ravi Teja Reddy, Sanjay Bhargav, Harika Satti and Sai Pragna

Acknowledgements

We would like to express our gratitude and appreciation to everyone who helped us complete this project. Special thanks go out to our professor, Mr. Vijay Eranti, for his support, motivating ideas, and encouragement throughout the fabrication process and the writing of this report. We also value the time he took to proofread and fix our mistakes. All of our classmates, especially our friends, deserve our gratitude for giving their time freely to help and support us as we developed our project.

Table of Contents

Abstract

Online shopping is one of the largest markets for businesses in the modern world. This entails the enormous responsibility of luring new clients and retaining old ones. Due to the intense competition to attract and keep customers online, businesses are being forced to use innovative strategies to improve customer experiences. Companies are increasingly examining customer feedback on websites like Amazon to learn more about how customers rate their goods and services. The purpose of this study is to examine the potential use of sentiment analysis by businesses to gather more information about the experiences of their clients. The dataset selected for this capstone includes user reviews and product ratings from Amazon. When a company reads Amazon product reviews, it can learn about how customers have felt about particular goods and services. The study's findings will enable businesses to identify the causes of both favorable and unfavorable customer feedback and implement workable solutions.

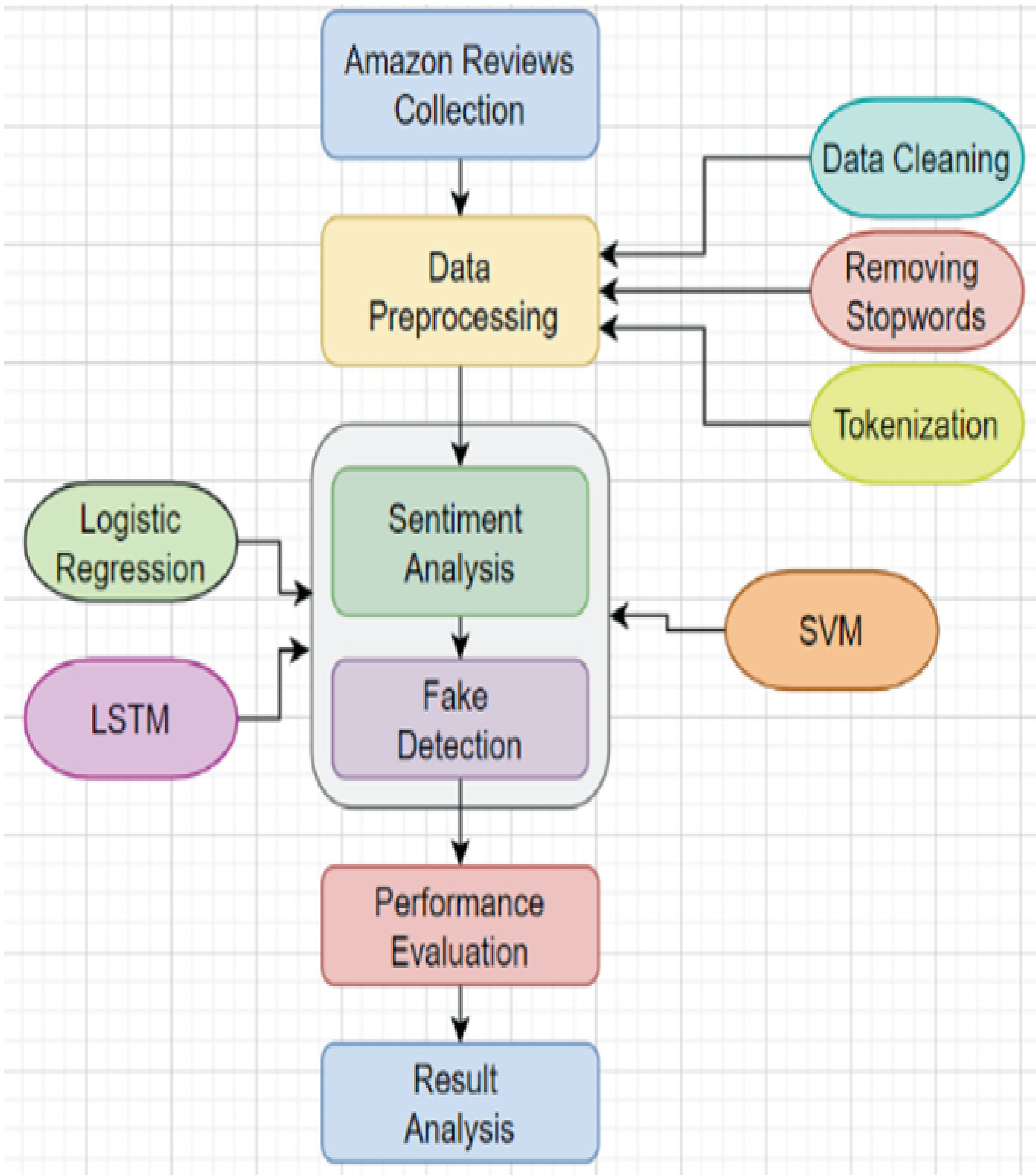
Introduction

Due to advancements in the fields of deep learning and computational control of equipment frameworks, Natural Language Processing, a sub-field of machine learning, has gained enormous notoriety over the past five years in both research and practical applications. The use of computational etymology could be one way for computers to understand how human language functions. NLP has been used in a few applications for understanding and deciphering content, sound, and video records in more recent years.

Sentiment Analysis is one of the main areas where NLP has been heavily applied. It is crucial for businesses to comprehend how customers behave and what they need from their products and services. The majority of customer reviews on products can be divided into three categories: Positive, Negative, and Neutral. It makes a difference when companies translate customer feedback through item audits to determine how satisfied customers are with their goods and services.

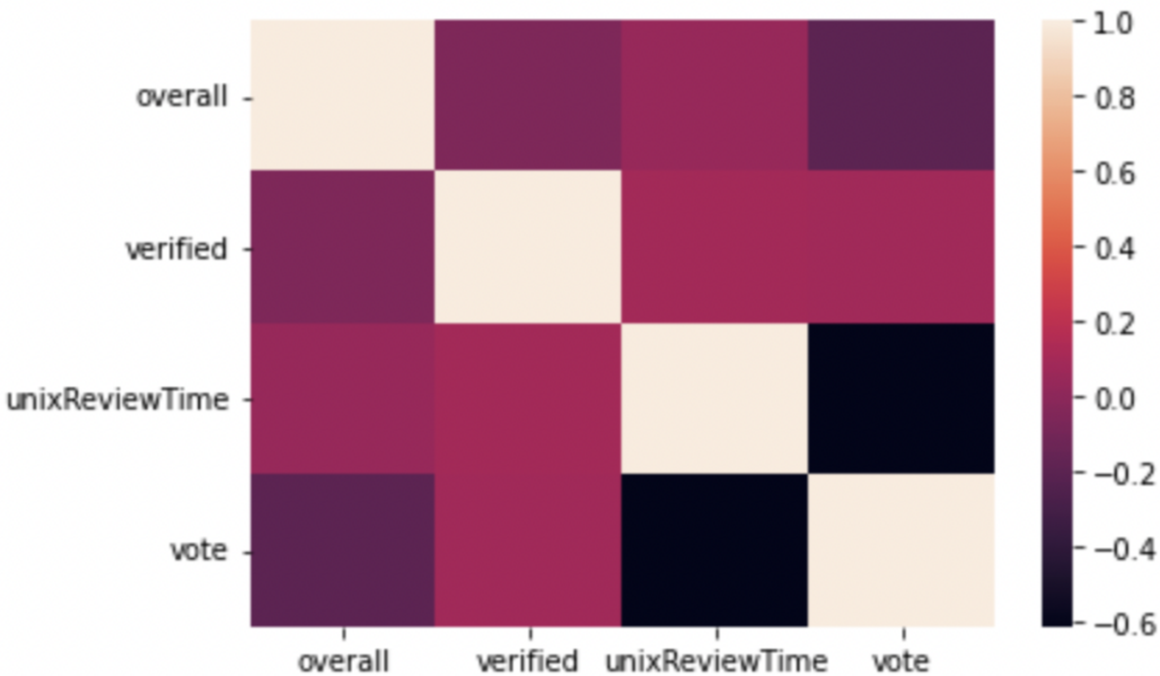
Related Work

Amazon Review Sentiment Analysis is basically related to the paper DEEP LEARNING SENTIMENT ANALYSIS OF AMAZON.COM REVIEWS AND RATINGS. The main goal of this essay is to assess how well Amazon.com reviews match the ratings that go with them. To train a recurrent neural network with a gated recurrent unit, product reviews were first converted to vectors using paragraph vectors. We developed a model using recurrent neural networks (RNN) with gated recurrent units (GRU) that learned low-dimensional vector representations of reviews using paragraph vectors and product embeddings in order to analyze the sentiment of Amazon.com reviews.


Amazon Reviews Collection
Data Cleaning
Data Preprocessing
Removing Stopwords
Tokenization
Logistic Regression
Sentiment Analysis
SVM
LSTM
Fake Detection
Performance Evaluation
Result Analysis

Data

The data used for the Amazon review sentiment analysis is gathered from UCSD EDU(https://jmcauley.ucsd.edu/data/amazon/). The data is selected from 5-core data from complete review data. This contains a subset of the data in which all users and items have at least 5 reviews. A total of 41.13 million reviews. The total size of the data utilized is 9.9GB. Some of the main attributes of this data are overall rating, verified customer, reviewText, summary, reviewerID.

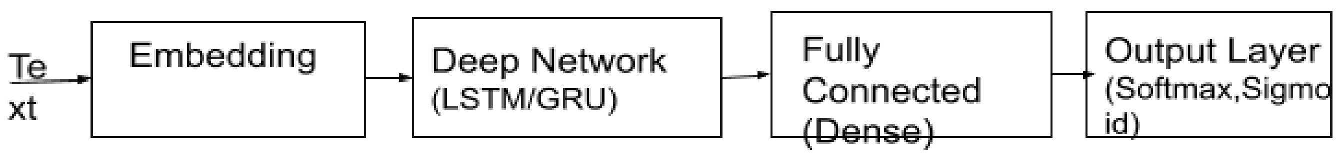


Heatmap of data

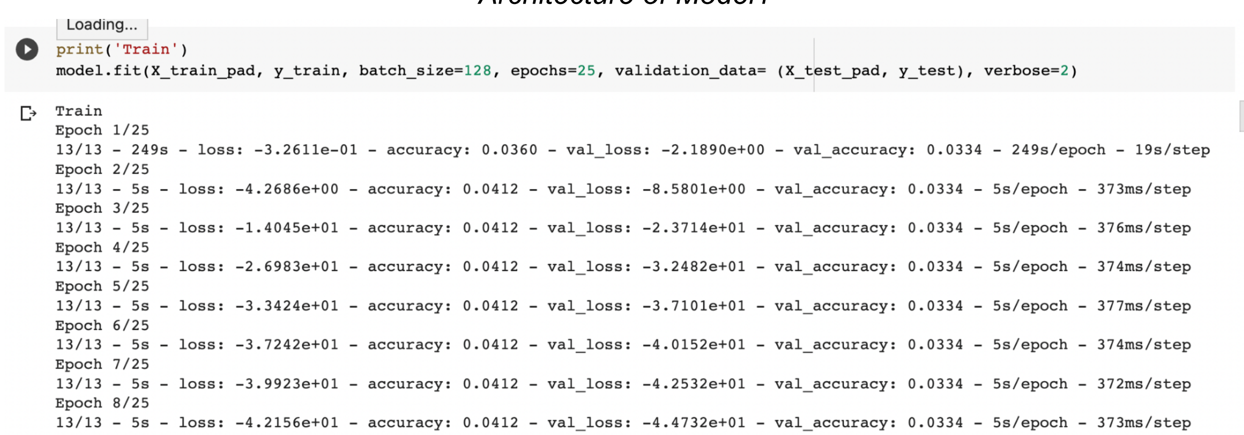
Methods

For Amazon Review Sentiment Analysis, we have implemented 3 different models. Compared each model with accuracy and loss functions and selected the best model. The models that we have implemented are

1. Sequential LSTM: Model contains the following components - Embedding, Deep Network(LSTM), Fully connected(Dense), Output layer(Sigmoid). The word embeddings of dataset can be learned whereas preparing a neural network on the classification is an issue. Some time recently it can be displayed to the network, the content information is to begin with encoded so that each word is spoken to by a unique number. This information planning step can be performed utilizing the Tokenizer API given with Keras. We include padding to create all the vectors of the same length (max_length). The model will use an Embedding layer as the first hidden layer. The Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset during training of the model. The accuracy that we obtained with this model is 41%.



Architecture of Model1



```
Loading...
print('Train')
model.fit(X_train_pad, y_train, batch_size=128, epochs=25, validation_data= (X_test_pad, y_test), verbose=2)

Train
Epoch 1/25
13/13 - 249s - loss: -3.2611e-01 - accuracy: 0.0360 - val_loss: -2.1890e+00 - val_accuracy: 0.0334 - 249s/epoch - 19s/step
Epoch 2/25
13/13 - 5s - loss: -4.2686e+00 - accuracy: 0.0412 - val_loss: -8.5801e+00 - val_accuracy: 0.0334 - 5s/epoch - 373ms/step
Epoch 3/25
13/13 - 5s - loss: -1.4045e+01 - accuracy: 0.0412 - val_loss: -2.3714e+01 - val_accuracy: 0.0334 - 5s/epoch - 376ms/step
Epoch 4/25
13/13 - 5s - loss: -2.6983e+01 - accuracy: 0.0412 - val_loss: -3.2482e+01 - val_accuracy: 0.0334 - 5s/epoch - 374ms/step
Epoch 5/25
13/13 - 5s - loss: -3.3424e+01 - accuracy: 0.0412 - val_loss: -3.7101e+01 - val_accuracy: 0.0334 - 5s/epoch - 377ms/step
Epoch 6/25
13/13 - 5s - loss: -3.7242e+01 - accuracy: 0.0412 - val_loss: -4.0152e+01 - val_accuracy: 0.0334 - 5s/epoch - 374ms/step
Epoch 7/25
13/13 - 5s - loss: -3.9923e+01 - accuracy: 0.0412 - val_loss: -4.2532e+01 - val_accuracy: 0.0334 - 5s/epoch - 372ms/step
Epoch 8/25
13/13 - 5s - loss: -4.2156e+01 - accuracy: 0.0412 - val_loss: -4.4732e+01 - val_accuracy: 0.0334 - 5s/epoch - 373ms/step
Epoch 9/25
```

Model Training

2. Pre-trained Hugging Face Model: In this method, we are using our custom dataset and training it on a pre-trained hugging face model. We have created a DataLoader class for loading and preprocessing of the data for training and inference phases. The

DataLoader class initializes a pre-trained tokenizer and this helps to encode the input sentences. We instantiate the distilbert-base-uncased model. The accuracy that we obtained with this model is 99%

```
loading configuration file https://huggingface.co/distilbert-base-uncased/resolve/main/config.json from cache at /root/.cache/huggingface/transformers/23454919
Model config DistilBertConfig {
  "_name_or_path": "distilbert-base-uncased",
  "activation": "gelu",
  "architectures": [
    "DistilBertForMaskedLM"
  ],
  "attention_dropout": 0.1,
  "dim": 768,
  "dropout": 0.1,
  "hidden_dim": 3072,
  "initializer_range": 0.02,
  "max_position_embeddings": 512,
  "model_type": "distilbert",
  "n_heads": 12,
  "n_layers": 6,
  "pad_token_id": 0,
  "qa_dropout": 0.1,
  "seq_classif_dropout": 0.2,
  "sinusoidal_pos_embds": false,
  "tie_weights_": true,
  "transformers_version": "4.19.2",
  "vocab_size": 30522
}
```

Model Configuration

3. Sentiment Analysis using BERT model: In this method, we are using unsupervised BERT model to predict the sentiment of the reviews. We will drop the overall column, which contains the ratings for each reviewText, in the training and validation dataset. Overall column will be retained in the test dataset. Labels are created using the vader algorithm. Vader algorithm returns 0 if the reviewText is negative and 1 if it's positive. Later, we use this function to predict the sentiment of the training and validation dataset. The result is outputted to the new column named 'vader_result'. In the final step, we compare these results with the actual results for the test dataset.

```
Downloading: 100% 571/571 [00:00<00:00, 21.3kB/s]
Downloading: 100% 1.25G/1.25G [00:21<00:00, 63.6MB/s]
Some weights of the model checkpoint at bert-large-uncased were not used when initializing BertForSequenceClassification: ['cls.predictions.bias', 'cls.seq_rel
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializi
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-large-uncased and are newly initialized: ['classifier.bias
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
***** Running training *****
  Num examples = 2000
  Num Epochs = 5
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 625
[175/625 25:52 < 1:07:18, 0.11 it/s, Epoch 1.39/5]
 Step  Training Loss  Validation Loss

Saving model checkpoint to ./sentiment-analysis-model2/checkpoint-125
Configuration saved in ./sentiment-analysis-model2/checkpoint-125/config.json
Model weights saved in ./sentiment-analysis-model2/checkpoint-125/pytorch_model.bin
[201/625 29:51 < 1:03:37, 0.11 it/s, Epoch 1.60/5]
 Step  Training Loss  Validation Loss
```

Model Training

Experiments

Model1 :

Vocabulary size, the size of the real-valued vector space and the maximum length of input documents should be specified to the embedding layer. The values that we have chosen for these are
EMBEDDING_DIM = 100
Input_length = max_length

```
    # x = layers.LSTM(32, recurrent_dropout=0.2, unroll=True)(inputs)
model = Sequential()
model.add(Embedding (vocab_size, EMBEDDING_DIM, input_length=max_length))
model.add(GRU(units=32,dropout=0.2, recurrent_dropout=0.2, unroll=True))
model.add(Dense(1, activation='sigmoid'))
# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Model parameters

Model2:

Model performance is evaluated at intervals during the training phase. This is achieved by a metric computation function that accepts parameters as prediction and label and returns metrics.

```
[ ] f1 = datasets.load_metric('f1')
    accuracy = datasets.load_metric('accuracy')
    precision = datasets.load_metric('precision')
    recall = datasets.load_metric('recall')
    def compute_metrics(eval_pred):
        metrics_dict = {}
        predictions, labels = eval_pred
        predictions = np.argmax(predictions, axis=1)

        metrics_dict.update(f1.compute(predictions = predictions, references = labels, average = 'macro'))
        metrics_dict.update(accuracy.compute(predictions = predictions, references = labels))
        metrics_dict.update(precision.compute(predictions = predictions, references = labels, average = 'macro'))
        metrics_dict.update(recall.compute(predictions = predictions, references = labels, average = 'macro'))
        return metrics_dict

Downloading builder script: 6.50k/? [00:00<00:00, 275kB/s]
Downloading builder script: 4.21k/? [00:00<00:00, 173kB/s]
Downloading builder script: 7.55k/? [00:00<00:00, 329kB/s]
Downloading builder script: 7.38k/? [00:00<00:00, 302kB/s]
```

Metric evaluation

Configuring distilbert-base-uncased model for pre-trained checkpoint.

```python
id2label = {idx:label for idx, label in enumerate(le.classes_)}
label2id = {label:idx for idx, label in enumerate(le.classes_)}
config = AutoConfig.from_pretrained('distilbert-base-uncased',
                                    num_labels = 5,
                                    id2label = id2label,
                                    label2id = label2id)
model = AutoModelForSequenceClassification.from_config(config)
```

Model parameters

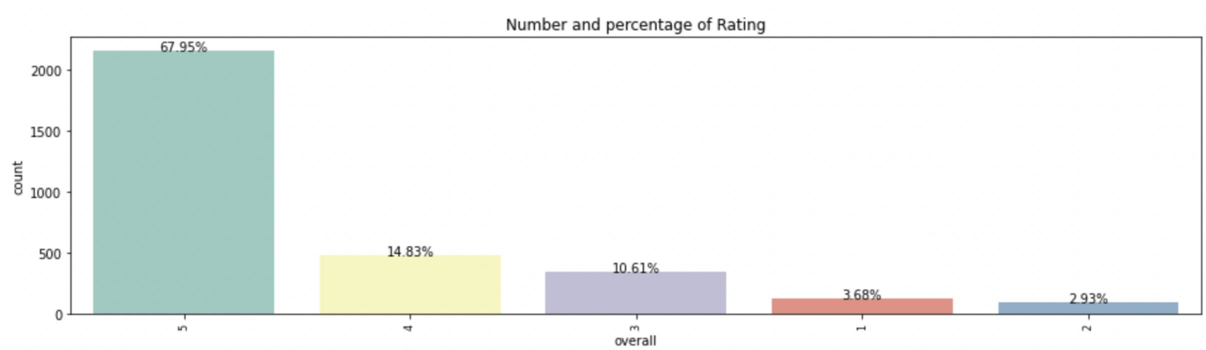
Later, we have set up the training arguments as follows
num_train_epochs=10,
per_device_train_batch_size=64,
per_device_eval_batch_size=64,
warmup_steps=500,
weight_decay=0.05

Model3:

Vader_lexicon library is used to generate labels. These labels are later compared with actual labels to calculate the accuracy of the model.

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

def vader_sentiment_result(sent):
    scores = analyzer.polarity_scores(sent)

    if scores["neg"] > scores["pos"]:
        return 0

    return 1

train_set["vader_result"] = train_set["reviewText"].apply(lambda x: vader_sentiment_result(x))
valid_set["vader_result"] = valid_set["reviewText"].apply(lambda x: vader_sentiment_result(x))
```

Label creation



Results

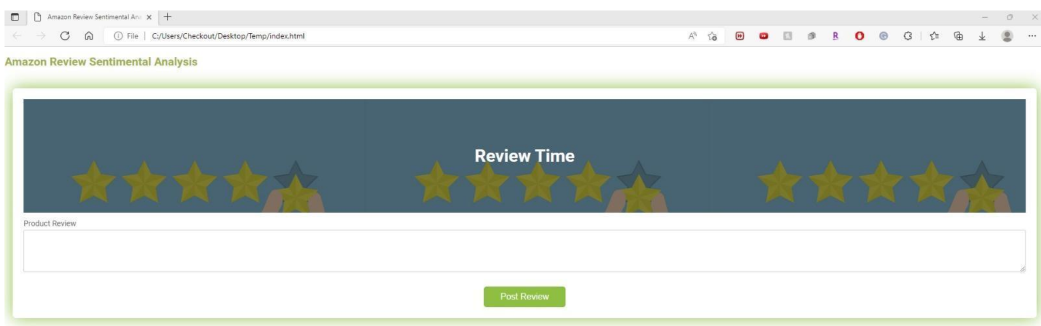
Conclusion

We have used three different models to develop this project. Among the three models we have selected the second model which gives an accuracy of 99%. The main concept of this project is to analyze the provided input review comment and perform the sentiment analysis and classify the output as the positive or negative review along with the rating.

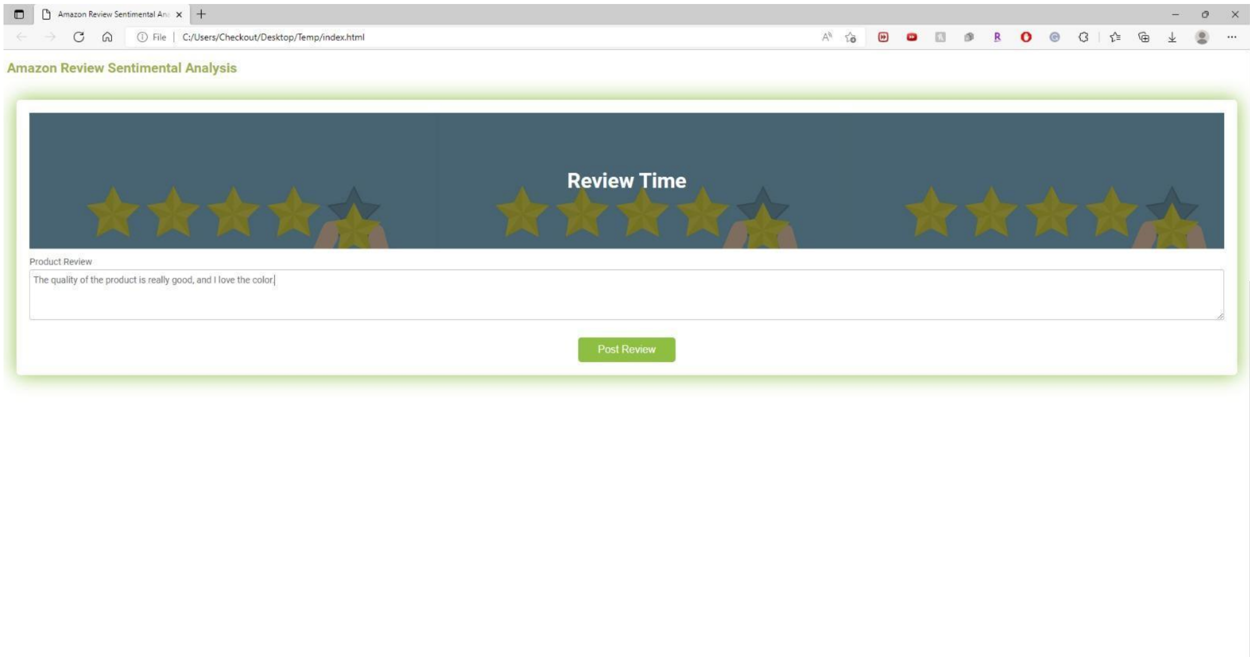
Deployed in AWS.

Front End:

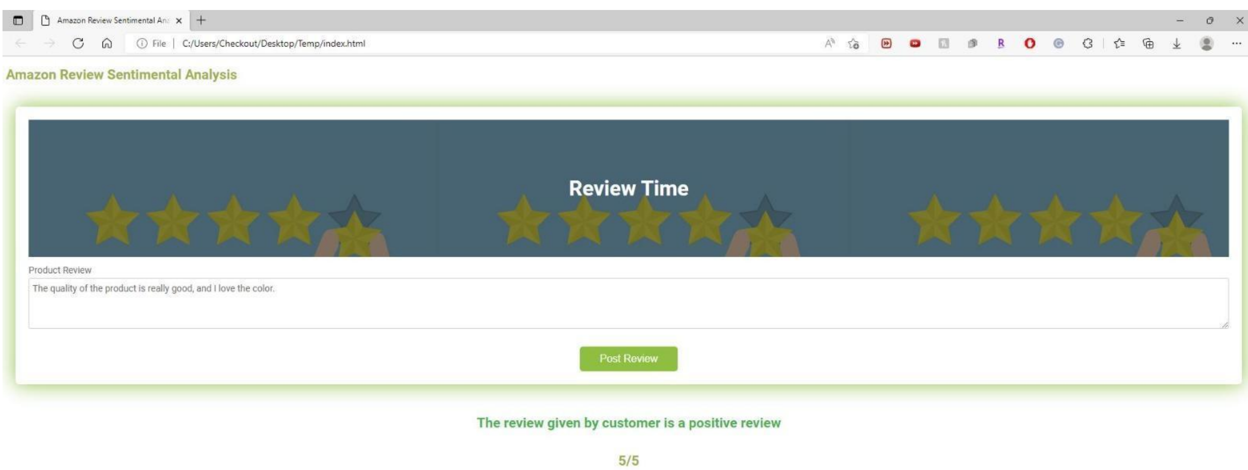
Below is the screenshot of the User Interface of Amazon Review Sentiment Analysis.



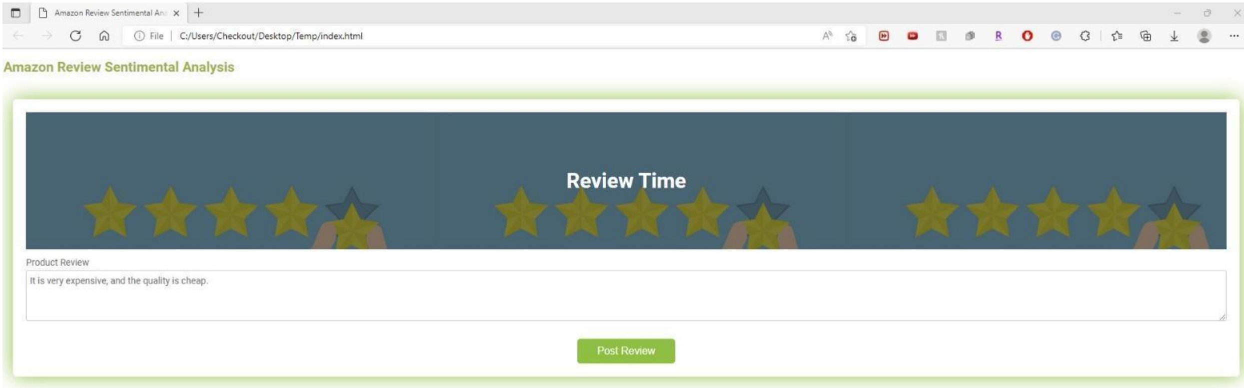
We provide the input review comment for analyzing it as a positive or negative comment.



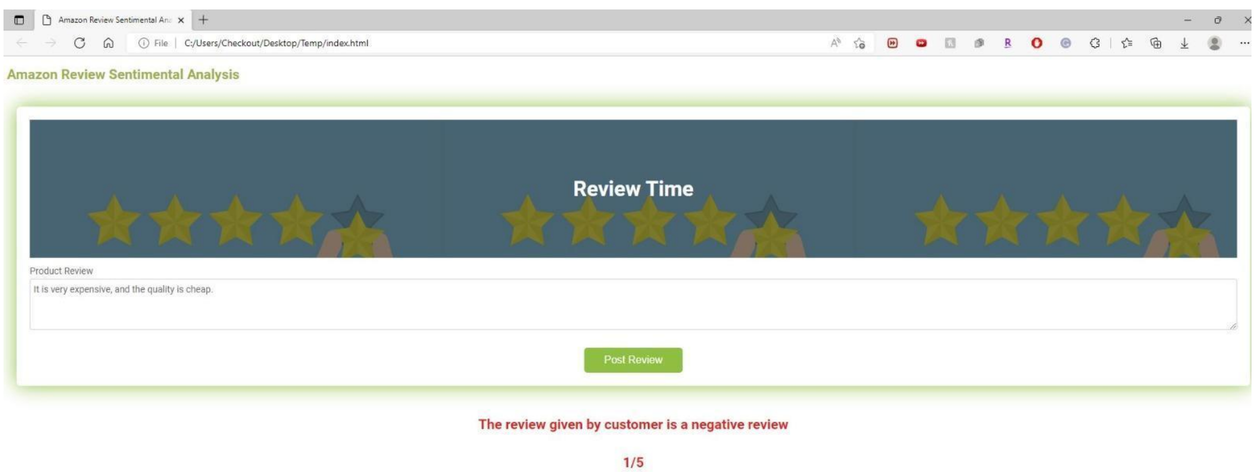
The below screen shows the output of the review after sentiment analysis by classifying the comment and also with rating.



We provide the input review comment for analyzing it as a positive or negative comment.



The below screen shows the output of the review after sentiment analysis by classifying the comment and also with rating.



Google Colab:

Testing Colab-
https://colab.research.google.com/drive/11L3Q_gm7d--c4GZTj-gUOJhYC3odx8gQ

Main Colab- https://colab.research.google.com/drive/1gcFEx3UqXkdIQ2kSqeIMa6ZXcwkPgmhy

PPT:
https://docs.google.com/presentation/d/1dSqTXfnWkAi_-Jd5yaYdcGR10QfWNZoq/edit#slide=id.p1

Github Repository:
https://github.com/ravitejareddy-dodda/258-Deep-Learning-Group/tree/main

Project Report:

Demo URL: