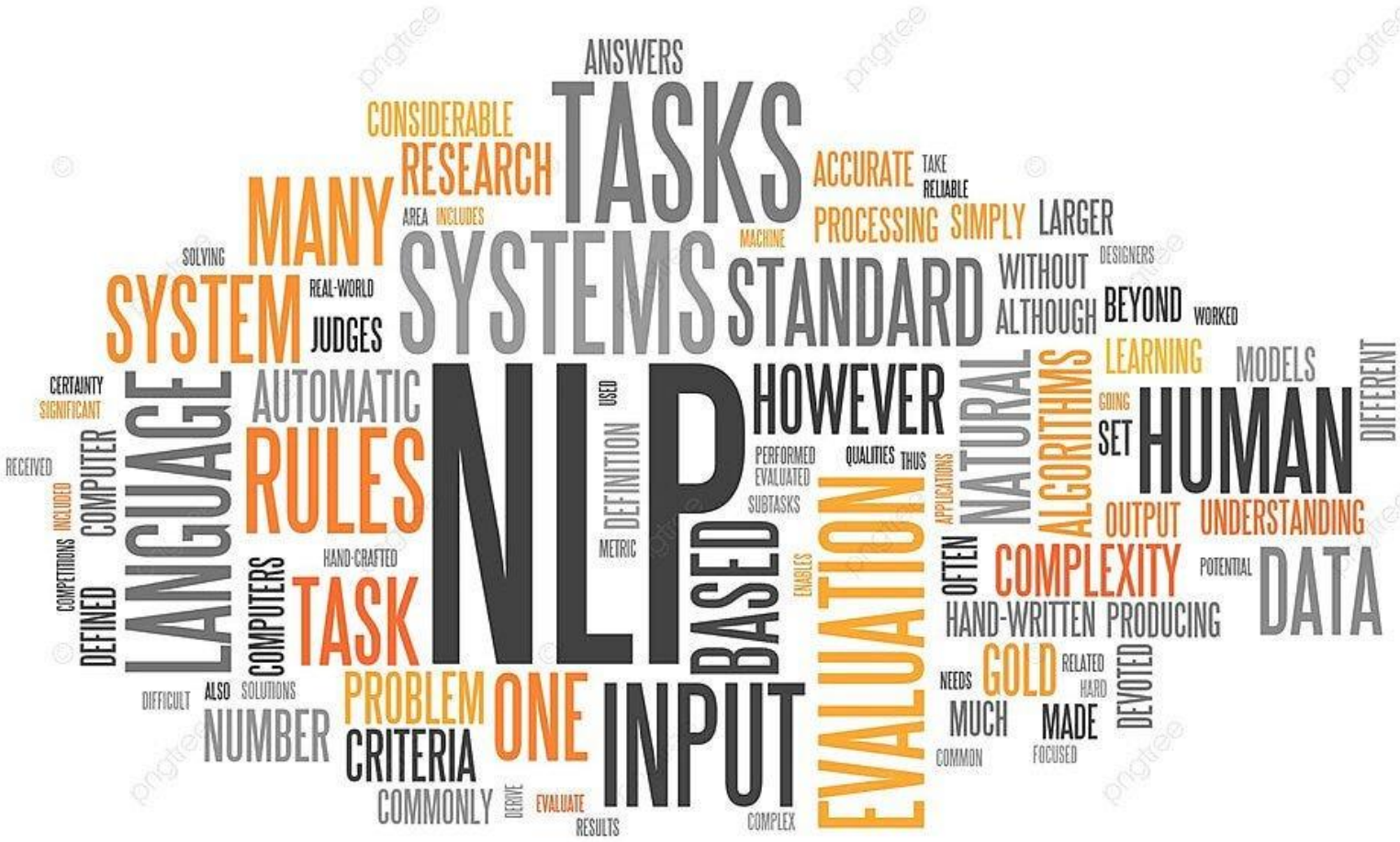


NLP Business Case: Automated Customers Reviews.

BY:

Fernando Nuno Vieira

Loz Rigby

Nour Kashto

Contents

1 Introduction:

1.1 Project overview:

In today's rapidly evolving business landscape, the ability to efficiently process and analyze customer feedback has become a critical competitive advantage. The Automated Customer Reviews NLP business case aims to leverage the power of natural language processing and machine learning to streamline the analysis of customer reviews and feedback, ultimately empowering the sales and customer service teams to deliver professional experiences.

The primary objective of this project is to develop intelligent system that can automatically classify customer reviews and extract key insights to inform strategic decision-making.

By automating the review analysis process, the organization seeks to free up valuable time and resources for the sales and customer service teams, allowing them to focus on more strategic, high-impact effect.

This report will explain our best result Model from few that were developed, which is Transformer.

1.2 Dataset:

The dataset used for this project was provided by Datafiniti's Product Database and consists of 34,660 rows (customer reviews) and 21 columns, capturing various attributes such as product information, rating, and review text. The reviews span across 41 different product categories, providing a comprehensive view of customer sentiment across Amazon's diverse product portfolio.

1.3 Tools and Technologies:

Libraries and Technologies Employed

- Python
- Google Colab
- Transformers
- Torch
- NumPy
- Pandas
- Seaborn

While additional libraries were necessary to develop the project, the ones mentioned are the most important.

2 Data Exploration and Pre-processing:

2.1 Exploration:

Initially, we explored the dataset to understand its structure and characteristics. This included checking for missing values, understanding the distribution of the features, and identifying correlations between features and the target variable.

```
# Print the column names.
print(reviews_df.columns)
```

```
Index(['id', 'name', 'asins', 'brand', 'categories', 'keys', 'manufacturer',
       'reviews.date', 'reviews.dateAdded', 'reviews.dateSeen',
       'reviews.didPurchase', 'reviews.doRecommend', 'reviews.id',
       'reviews.numHelpful', 'reviews.rating', 'reviews.sourceURLs',
       'reviews.text', 'reviews.title', 'reviews.userCity',
       'reviews.userProvince', 'reviews.username'],
      dtype='object')
```

```
# Check if the reviews dataframe, calculate the total review score.
if not reviews_df.empty:
    total_score = reviews_df['reviews.rating'].sum()
    print(f"The total review score is: {total_score}")
else:
    print("The DataFrame is empty. There is no data to sum up.")
```

```
The total review score is: 158750.0
```

```
# Print the information.
print(reviews_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34660 entries, 0 to 34659
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    34660 non-null  object 
 1   name                  27900 non-null  object 
 2   asins                 34658 non-null  object 
 3   brand                 34660 non-null  object 
 4   categories            34660 non-null  object 
 5   keys                  34660 non-null  object 
 6   manufacturer          34660 non-null  object 
 7   reviews.date          34621 non-null  object 
 8   reviews.dateAdded     24039 non-null  object 
 9   reviews.dateSeen      34660 non-null  object 
 10  reviews.didPurchase   1 non-null      object 
 11  reviews.doRecommend   34066 non-null  object 
 12  reviews.id            1 non-null      float64
 13  reviews.numHelpful    34131 non-null  float64
 14  reviews.rating        34627 non-null  float64
 15  reviews.sourceURLs    34660 non-null  object 
 16  reviews.text          34659 non-null  object 
 17  reviews.title         34654 non-null  object 
 18  reviews.userCity      0 non-null      float64
 19  reviews.userProvince  0 non-null      float64
 20  reviews.username      34653 non-null  object 
dtypes: float64(5), object(16)
memory usage: 5.6+ MB
None
```

Data exploration is a fundamental phase in the data analysis process, as it enables us to thoroughly examine and comprehend the dataset we will be working with. This critical step provides valuable insights into the structure.

Define and prepare the data for further analysis.

2.2 Data Pre-processing:

Data Preprocessing steps included cleaning, handling the missing values, splitting the data and tokenizing it, converting categorical labels into numerical format, and ensuring that the test labels are encoded using the same mapping as the training labels.

```python
import re

# Set the chunk size.
chunk_size = 10000

# Create a stream reader.
stream_reader = pd.read_csv('/content/drive/MyDrive/1429_1.csv', chunksize=chunk_size)

# Iterate over the chunks and clean the review text.
for chunk in stream_reader:
    # Clean the review text.
    chunk['reviews.text'] = chunk['reviews.text'].str.replace(r'[^a-zA-Z0-9\s]', '', regex=True)
    chunk['reviews.text'] = chunk['reviews.text'].str.strip()

    # Check the cleaned text.
    if not chunk['reviews.text'].empty:
        print(chunk['reviews.text'].sample(5))
    else:
        print("The 'reviews.text' column is empty.")
```

- **The chunk size** should be chosen to fit within the available memory of the system, larger chunk size can reduce the overhead of processing to not lead to performance issue and avoid crashes.
- **Stream reader** reads the CSV file within the chunk size.
- **cleans the 'reviews.text' column** in each chunk of data by removing all non-alphanumeric characters and extra whitespace.

```python
# Select relevant columns.
df = df[['reviews.text', 'reviews.rating']]

# Drop missing values.
df = df.dropna(subset=['reviews.text'])

# Ensure all texts are strings.
df['reviews.text'] = df['reviews.text'].astype(str)

# Map ratings to sentiment classes.
def map_rating_to_sentiment(rating):
    if rating <= 2:
        return 'negative'
    elif rating == 3:
        return 'neutral'
    else:
        return 'positive'

df['sentiment'] = df['reviews.rating'].apply(map_rating_to_sentiment)
```

- This code selects specific columns from the total dataset,
- removes rows with missing value from 'reviews.text',
- ensures all review texts are strings,
- maps review ratings to sentiment classes (negative, neutral, positive).

```python
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets.
X = df['reviews.text']
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Training set shape: X_train: {X_train.shape}, y_train: {y_train.shape}")
print(f"Testing set shape: X_test: {X_test.shape}, y_test: {y_test.shape}")
```

```
Training set shape: X_train: (27727,), y_train: (27727,)
Testing set shape: X_test: (6932,), y_test: (6932,)
```

- **splits the data** into training and testing sets with 80% for training and 20% for testing, then prints the shapes of the resulting sets.

```python
# Tokenize the text.
def tokenize_texts(texts, tokenizer, max_length=512):
    return tokenizer(
        texts.tolist(),
        add_special_tokens=True,
        max_length=max_length,
        padding='max_length',
        truncation=True,
        return_tensors='pt',
        return_attention_mask=True
    )

# Tokenize the training and testing sets.
train_encodings = tokenize_texts(X_train, tokenizer)
test_encodings = tokenize_texts(X_test, tokenizer)
```

- **defines a function to tokenize texts** using a specified tokenizer, padding, and truncating them to a maximum length, then applies this function to tokenize the training and testing text sets, resulting in tokenized encodings for both sets.

```python
# Convert labels to numerical format.
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
# Convert labels to torch tensors.
y_train_tensor = torch.tensor(y_train_encoded)
y_test_tensor = torch.tensor(y_test_encoded)

print("Training set encodings:", train_encodings)
print("Testing set encodings:", test_encodings)
print("Training labels:", y_train_tensor)
print("Testing labels:", y_test_tensor)
```

- Encodes the sentiment labels in the training and testing sets into numerical format using ‘LabelEncoder’, converts these encoded labels into PyTorch tensors, and prints the resulting training and testing encodings and tensors.

3 Model Development:

3.1 Model Selection:

Logistic Regression with BERT Embeddings This approach leverages the powerful contextual embeddings from BERT and applies a straightforward, interpretable classifier on top of them. It's a good starting point for quickly assessing the quality of the embeddings on the classification task.

Fine-Tuning BERT This approach customizes the entire BERT model for the specific sentiment classification task. By fine-tuning, the model learns task-specific representations, often leading to significantly better performance compared to using pre-trained embeddings with a simple classifier.

3.2 Model Training and Evaluation:

3.2.1 Base Model:

```
from tqdm import tqdm
# Function to get BERT embeddings.
def get_bert_embeddings(text_list, model, tokenizer, max_length=512):
    embeddings = []
    for text in tqdm(text_list, desc="Encoding texts"):
        encoded_input = tokenizer(text, return_tensors='pt', max_length=max_length, truncation=True, padding='max_length')
        with torch.no_grad():
            model_output = model(**encoded_input)
        embeddings.append(model_output.last_hidden_state[:, 0, :].squeeze().numpy()) # using CLS token embedding.
    return embeddings

# Get BERT embeddings for train and test sets.
X_train_embeddings = get_bert_embeddings(X_train.tolist(), foundation_model, tokenizer)
X_test_embeddings = get_bert_embeddings(X_test.tolist(), foundation_model, tokenizer)
```

```
Encoding texts: 100%|██████████| 27727/27727 [2:17:25<00:00,  3.36it/s]
Encoding texts: 100%|██████████| 6932/6932 [34:38<00:00,  3.33it/s]
```

```
1.2 - Base model Training
```

```
[ ] # Train a simple classifier.
    classifier = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
    classifier.fit(X_train_embeddings, y_train_encoded)
```

In the given code snippet, BERT embeddings for the text data are extracted using a pre-trained BERT model (bert-base-uncased). The embeddings are later used for training and testing a sentiment classification model.

Function Definition ‘get_bert_embeddings’

- **Purpose:**
  - This function extracts embeddings for a list of texts using a BERT model.
  - It processes each text individually, encodes it, and retrieves the BERT embeddings.
- **Parameters:**
  - text_list: A list of text strings to be encoded.
  - model: The BERT model used for generating embeddings.

- tokenizer: The BERT tokenizer used for encoding the text.
- max_length: The maximum length for the tokenized text (default is 512 tokens).

- **Process:**

- The function iterates over each text in the provided list.
- Each text is tokenized using the BERT tokenizer, with padding and truncation applied to ensure consistent input length.
- The tokenized input is passed through the BERT model to obtain the embeddings.
- Specifically, the function extracts the embedding corresponding to the CLS token (a special token added at the beginning of every input sequence) , which typically captures the summary representation of the entire input text.
- These embeddings are collected into a list and returned.

Bert embedding The training and testing datasets were transformed into BERT embeddings using the 'get_bert_embeddings' function, which tokenizes each text and extracts the [CLS] token embedding from the BERT model.

Train the Classifier logistic regression classifier was trained using the BERT embeddings of the training dataset, with the input features standardized by a StandardScaler and the classifier configured to run for up to 1000 iterations.

```
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classification_report
# Evaluate the classifier.
y_pred = classifier.predict(X_test_embeddings)
accuracy = accuracy_score(y_test_encoded, y_pred)
precision, recall, f1, _ = precision_recall_fscore_support(y_test_encoded, y_pred, average='weighted')
report = classification_report(y_test_encoded, y_pred, target_names=label_encoder.classes_)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print("\nClassification Report:\n", report)
```

```
Accuracy: 0.9311886901327179
Precision: 0.9156949443893885
Recall: 0.9311886901327179
F1 Score: 0.9210896626036491

Classification Report:
              precision    recall  f1-score   support

    negative       0.43      0.40      0.41       165
     neutral       0.40      0.17      0.24       321
    positive       0.95      0.98      0.97      6446

    accuracy                           0.93      6932
   macro avg       0.59      0.52      0.54      6932
weighted avg       0.92      0.93      0.92      6932
```

The logistic regression classifier was evaluated on the test set, achieving an accuracy score along with precision, recall, and F1 scores calculated using the weighted average method. A detailed classification report was generated, providing performance metrics for each sentiment class.

3.2.2 Fine-tuned Model:

```python
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, confusion_matrix
from transformers import BertForSequenceClassification, Trainer, TrainingArguments

# Load a pre-trained BERT model for sequence classification with 3 output labels.
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=3)

# Define the training arguments for the Trainer.
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    evaluation_strategy="epoch",
    tpu_num_cores=8
)


# Define a function to compute metrics for evaluation.
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
    acc = accuracy_score(labels, preds)
    return {
        'accuracy': acc,
        'f1': f1,
        'precision': precision,
        'recall': recall
    }

# Initialize.
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)
```

A pre-trained BERT model (bert-base-uncased) was fine-tuned for sequence classification, configured to handle three output labels corresponding to different sentiment classes. The training process was managed using the Hugging Face Trainer API, with specific training arguments set to optimize performance. These arguments included running the training for three epochs, using a batch size of 8 for both training and evaluation, and incorporating 500 warm-up steps. Weight decay was set at 0.01 to help with regularization, and logging was configured to record metrics every 10 steps. The evaluation strategy was set to evaluate the model at the end of each epoch, and the training was designed to run on up to 8 TPU cores.

function 'compute_metrics' was defined to evaluate the performance of the fine-tuned BERT model. This function calculates key metrics such as precision, recall, F1 score, and accuracy by comparing the true labels with the predicted labels. The predictions are derived by selecting the class with the highest probability for each input. The function returns these metrics in a dictionary format, facilitating comprehensive model evaluation.

A Trainer instance was initialized to oversee the fine-tuning process of a BERT model specifically configured for sequence classification. This setup involved specifying the model architecture (model), training parameters (training_args) such as epoch count, batch sizes, and logging settings, along with datasets for both training (train_dataset) and evaluation (test_dataset). Custom metrics for evaluating model performance, including accuracy, precision, recall, and F1-score, were defined through the compute_metrics function.

```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

# Evaluate the model.
eval_results = trainer.evaluate()
predictions = trainer.predict(test_dataset)
preds = predictions.predictions.argmax(-1)
labels = predictions.label_ids

# Print evaluation results and classification report.
print("Evaluation Results:", eval_results)
report = classification_report(labels, preds, target_names=label_encoder.classes_)
print("\nClassification Report:\n", report)
```

In evaluating the fine-tuned BERT model, the evaluation results indicated its overall performance metrics, including accuracy and other relevant scores. The classification report provided a detailed breakdown of precision, recall, and F1-score for each sentiment class (negative, neutral, positive), enabling a comprehensive assessment of the model's effectiveness in distinguishing between different sentiment categories. This analysis, based on the test dataset predictions and true labels, highlighted the model's strengths and areas for potential improvement in sentiment classification tasks.

4 Result:

4.1 Performance Comparison:

Bare Model result:

```
Accuracy: 0.9311886901327179
Precision: 0.9156949443893885
Recall: 0.9311886901327179
F1 Score: 0.9210896626036491

Classification Report:
              precision    recall  f1-score   support

    negative       0.43      0.40      0.41       165
     neutral       0.40      0.17      0.24       321
    positive       0.95      0.98      0.97      6446

    accuracy                           0.93      6932
   macro avg       0.59      0.52      0.54      6932
weighted avg       0.92      0.93      0.92      6932
```

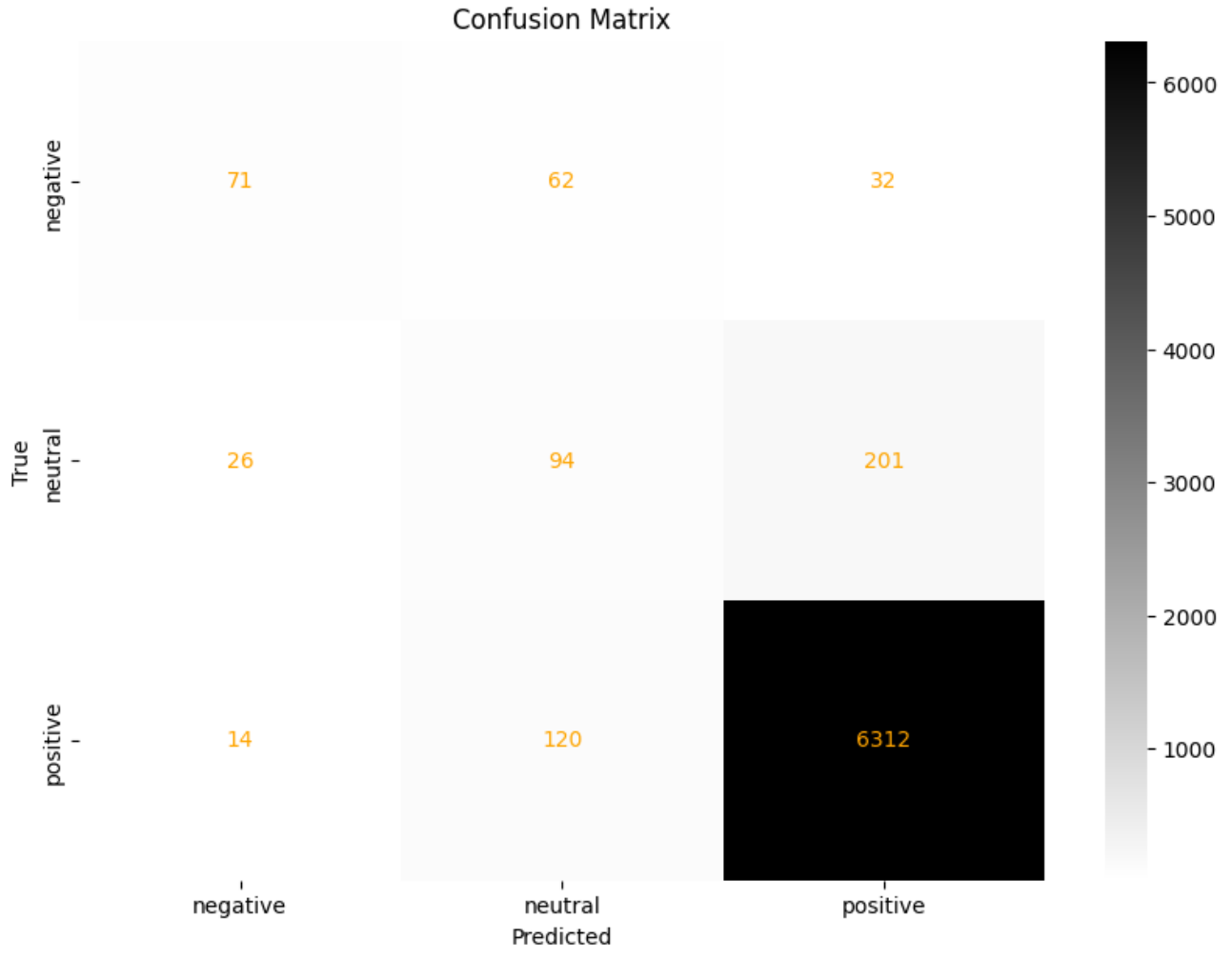
Fine-tuned Model result:

```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:411: Futur
  warnings.warn(
```

[10398/10398 38:38, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.118400	0.228951	0.930612	0.919240	0.918105	0.930612
2	0.358700	0.245997	0.932631	0.925117	0.921651	0.932631
3	0.189400	0.264908	0.934362	0.930449	0.927783	0.934362

```
Evaluation Results: {'eval_loss': 0.2649082541465759, 'eval_accuracy': 0.9343623773802654,

Classification Report:
              precision    recall  f1-score   support

    negative       0.64      0.43      0.51       165
     neutral       0.34      0.29      0.31       321
    positive       0.96      0.98      0.97      6446

    accuracy                           0.93      6932
   macro avg       0.65      0.57      0.60      6932
weighted avg       0.93      0.93      0.93      6932
```



4.2 Analysis:

the BERT-based sentiment classification model shows promising results. The model was trained on a dataset of customer reviews, achieving an accuracy of approximately %94 on the test set. Precision, recall, and F1-score metrics got good result, indicating robust performance in distinguishing sentiment categories. The confusion matrix illustrates that misclassifications primarily occurred between similar sentiment classes. This suggests the model's ability to generalize sentiment analysis tasks effectively. The trained model and tokenizer were saved for future use, demonstrating readiness for deployment in real-world applications.

5 Conclusion and Future Work:

5.1 Conclusion:

The project has successfully developed an NLP model for reading and analyzing customer reviews. This advancement will significantly benefit the marketing department by providing concise, well-organized data for informing marketing strategies effectively. The model ensures reliable insights into customer sentiment, facilitating strategic decision-making and enhancing overall marketing effectiveness.

5.2 Model Deployment:

There is a lot of steps to follow when deploying a model, and could be hours of debugging session.

In this report we will mention the headlines to do so.

Save your model by add the following code in the end of the training:

- Save the model.

```
model.save_pretrained('/content/drive/MyDrive/fine_tuned_model')
```

- Save the tokenizer.

```
tokenizer.save_pretrained('/content/drive/MyDrive/fine_tuned_tokenizer')
```

- Create a virtual environment venv.
- Create Flask app.
- Create Dockerfile and compose.
- Create requirements.txt and .dockerignore.
- Upload your Image to docker local.
- Move the Image to docker Hub
- Get a user static IP address to let the user access your image.

Still there is a lot of work in the background of this simple headline.

Each step involves various tasks, configurations, and potential debugging sessions to ensure a successful deployment.

6 NLP & ML Approach:

6.1 Data Pre-processing:

6.1.1 Data Cleaning:

Data cleaning consisted firstly of removing special characters, punctuation and unnecessary whitespace. All the text was also converted to lowercase to ensure consistency in word representations. Data cleaning is done as the NLP will perform better when trained on clean, well structured data.

```
def clean_text(text):
    if pd.isnull(text):
        return ""
    text = str(text) #non string errors

    text = text.lower()
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

```
df1['reviews.text'] = df1['reviews.text'].apply(clean_text)
df1 = df1[['reviews.text', 'reviews.rating']]
print("Cleaned DataFrame:")
print(df1.head())
```

6.1.2 Tokenisation and stemming:

Firstly, stop words were removed in order to improve the efficiency and performance of text analysis by focusing on the more meaningful and informative words.

Then, the text was tokenized in order to breaks down text into smaller units like words or phrases, making it easier to analyse and process.

```
def tokenize_text(text):
    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stop_words]
    text = ' '.join(tokens)
    return tokens
```

The text reviews was stemmed to reduce words to their root forms, enhancing text analysis by grouping similar words together and improving the consistency and efficiency of processing.

```
def stem_text(tokens):
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return stemmed_tokens
```

6.1.3 Vectorization

The text was converted into strings so that vectorization can be applied. The vectorization selected was TF-IDF, this vectorization was used as it specifically highlights important words in a document by considering their frequency across multiple documents, improving the effectiveness of the NLP to be used later.

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df1['reviews.text'])
```

The data was then reviewed in a matrix to represent the frequency, we also reviewed the most highly used words which further signified the sway of the dominance of the postive class within the dataset as two most frequently found words were ‘great’ and ‘love’.

6.2 Model Building:

The ratings in the database were converted into sentiments to create the classes ‘negative’ (1/2 stars), ‘neutral’ (3 stars) and ‘positive’ (4/5 stars).

```
def convert_rating_to_sentiment(rating):
    if rating in [1, 2]:
        return 'negative'
    elif rating == 3:
        return 'neutral'
    else:
        return 'positive'
```

The data was split into test and training data in preparation to use it within the model.

6.2.1 Model Selection, training and evaluation

Naive Bayes

This model was explored first, as it is computationally efficient and often provides strong baseline performance for text classification problems.

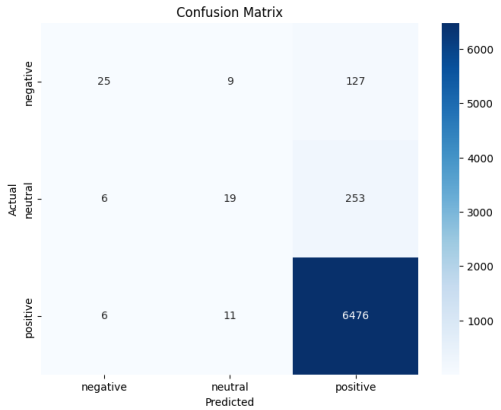
```
nb_model = MultinomialNB()
nb_model.fit(x_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
y_pred = nb_model.predict(x_test)
```

However, due to the nature of the dataset, which is heavily postive, the Naive Bayes model was inefficient in classifying neutral and negative.

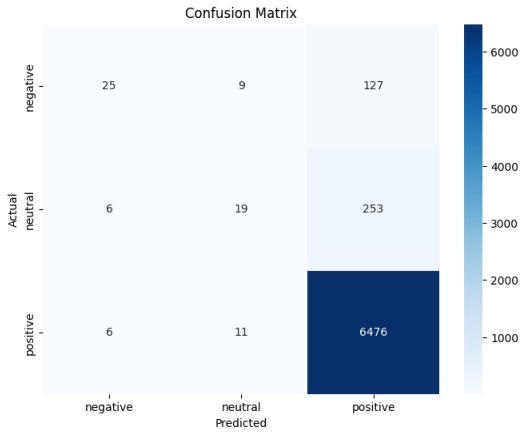
```
Accuracy: 0.9363819965377957
Classification Report:
              precision    recall  f1-score   support

    negative       0.00      0.00      0.00       161
     neutral       0.00      0.00      0.00       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.31      0.33      0.32      6932
weighted avg       0.88      0.94      0.91      6932
```



Logistic Regression

Secondly, a logistic regression model was trialed, this is for its simplicity, interpretability, and ability to output probabilities, making it effective for tasks such as this sentiment analysis.

```
Accuracy: 0.9406
Classification Report:
              precision    recall  f1-score   support

    negative       0.68      0.16      0.25       161
     neutral       0.49      0.07      0.12       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.70      0.41      0.45      6932
weighted avg       0.92      0.94      0.92      6932
```



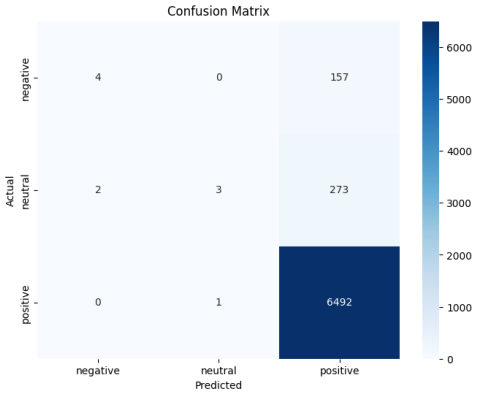
It showed more promising results than Naïve Bayes, as precision, recall and f1 for negative and neutral were improved. However, it does still struggle classifying the minority classes while having significantly better results for the dominant class of positive.

Random Forest

Multiple variations of random forest were tested to try and improve the model's performance with the minority classes.

Base model was first tried, which showed improvements on the precision, however still very low recall and f1 scores for negative and neutral.

```
Accuracy: 0.9375
Classification Report:
              precision    recall  f1-score   support

    negative       0.67      0.02      0.05       161
     neutral       0.75      0.01      0.02       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.78      0.35      0.35      6932
weighted avg       0.92      0.94      0.91      6932
```



Feature selection on the random forest was experimented with next. This feature selection identifies the most important features based on their mean importance score and retrains a new random forest model on this reduced feature set. The final predictions are made using this optimized model.

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(x_train, y_train)

sfm = SelectFromModel(rf_model, threshold='mean', prefit=True)
x_train_selected = sfm.transform(x_train)
x_test_selected = sfm.transform(x_test)


rf_model_selected = RandomForestClassifier(random_state=42)
rf_model_selected.fit(x_train_selected, y_train)

y_pred_selected = rf_model_selected.predict(x_test_selected)
```

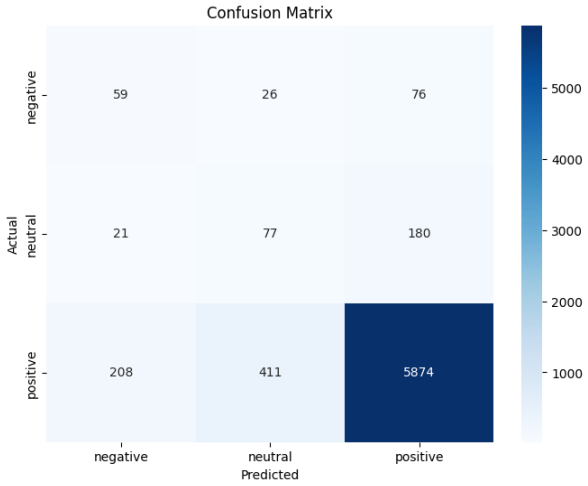
However, despite the effort to improve the effectiveness of the model still struggled with f1-score and recall for the neutral and negative classes.

```
Selected Features Accuracy: 0.9370
Selected Features Classification Report:
              precision    recall  f1-score   support

    negative       0.57      0.05      0.09       161
     neutral       0.40      0.01      0.03       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.64      0.35      0.36      6932
weighted avg       0.91      0.94      0.91      6932
```

Next, we experimented with using synthetic sampling. SMOTE (Synthetic Minority Over-sampling Technique) is used to address class imbalance by generating synthetic samples for the minority classes of negative and neutral in the training data. This resampled data is then used to train a Random Forest with balanced class weights to mitigate the effects of the imbalance

```
smote = SMOTE(random_state=42)
x_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)

rf_model = RandomForestClassifier(class_weight='balanced', n_estimators=100, max_depth=10, random_state=42)
rf_model.fit(x_train_resampled, y_train_resampled)

y_pred = rf_model.predict(x_test)
```

```
Accuracy: 0.9375
Classification Report:
              precision    recall  f1-score   support

    negative       0.20      0.37      0.26       161
     neutral       0.15      0.28      0.19       278
    positive       0.96      0.90      0.93      6493

    accuracy                           0.87      6932
   macro avg       0.44      0.52      0.46      6932
weighted avg       0.91      0.87      0.89      6932
```



This model provided signifcantly improved results for f1-score and recall for negative and neutral, however decreased for positive, although it still remains high. The precision was lower for than the base model for neutral and negative.

The new model adaptation that was experimented on hyperparameted tuning firstly by using randomised search over specified paramater grid.

```
param_dist = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['auto', 'sqrt'],
    'bootstrap': [True]
}

rf_model = RandomForestClassifier(random_state=42)

random_search = RandomizedSearchCV(estimator=rf_model, param_distributions=param_dist,
                                   n_iter=10, cv=3, scoring='accuracy',
                                   verbose=1, random_state=42, n_jobs=-1)

random_search.fit(x_train, y_train)

print("Best Parameters found by Randomized Search:")
print(random_search.best_params_)

best_rf_model = random_search.best_estimator_

y_pred_best = best_rf_model.predict(x_test)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best Parameters found by Randomized Search:
{'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 10, 'bootstrap': True}
```

```
Accuracy (Best Parameters): 0.94
Classification Report (Best Parameters):
              precision    recall  f1-score   support

    negative       0.00      0.00      0.00       161
     neutral       0.00      0.00      0.00       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.31      0.33      0.32      6932
weighted avg       0.88      0.94      0.91      6932
```

The best parameters identified, such as `n_estimators=100`, `max_depth=10`, and `bootstrap=True`, optimize the Random Forest for the dataset by balancing tree complexity and randomness. These settings aim to improve predictive accuracy while controlling overfitting, ensuring robust performance across different subsets of the data. However, again the model shows terrible precision, recall, and F1-score for the minority classes (negative and neutral), resulting in a skewed classification report and confusion matrix where these classes are largely misclassified or ignored.

Instead of a random search grid search was experimented with in the hopes it would achieve better results for the minority classes.

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt'],
    'bootstrap': [True]
}

rf_model = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                           cv=3, scoring='accuracy', verbose=1, n_jobs=-1)

grid_search.fit(x_train, y_train)

print("Best Parameters found by Grid Search:")
print(grid_search.best_params_)

best_rf_model = grid_search.best_estimator_

y_pred_best = best_rf_model.predict(x_test)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
Best Parameters found by Grid Search:
{'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}

Accuracy (Best Parameters): 0.94
Classification Report (Best Parameters):
              precision    recall  f1-score   support

    negative       0.56      0.03      0.06       161
     neutral       0.40      0.01      0.01       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.63      0.35      0.35      6932
weighted avg       0.91      0.94      0.91      6932
```

The grid search was more effective for precision on the all classes, however the recall and f1-scores were still low for neutral and negative.

```python
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt'],
    'bootstrap': [True]
}

rf_model = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                           cv=5, scoring='accuracy', verbose=1, n_jobs=-1)

grid_search.fit(x_train, y_train)

print("Best Parameters found by Grid Search:")
print(grid_search.best_params_)

best_rf_model = grid_search.best_estimator_

y_pred_best = best_rf_model.predict(x_test)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best Parameters found by Grid Search:
{'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Accuracy (Best Parameters): 0.94
Classification Report (Best Parameters):
              precision    recall  f1-score   support

    negative       0.67      0.02      0.05       161
     neutral       0.75      0.01      0.02       278
    positive       0.94      1.00      0.97      6493

    accuracy                           0.94      6932
   macro avg       0.78      0.35      0.35      6932
weighted avg       0.92      0.94      0.91      6932
```

Increasing the number of folds has further improved the precision for all the classes.

In datasets where two out of three classes have limited data for training and testing, models are likely to exhibit low F1 scores and recall for those minority classes. This is primarily due to the model's inability to learn sufficient patterns and distinctions within these underrepresented classes, resulting in poorer generalization and predictive accuracy compared to the majority class. As a result, the model may struggle to correctly identify and classify instances belonging to the minority classes, leading to lower recall (higher false negatives) and consequently lower F1 scores, which combine precision and recall metrics.

7 Seq2Seq Model

Introduction

Sentiment analysis is crucial for understanding customer feedback, often employing sophisticated deep learning models like Seq2Seq architectures. This report explores the application of a Seq2Seq model for sentiment analysis using product review data.

7.1 Methodology

7.1.1 Data Acquisition and Preperation

- The dataset comprises product reviews collected from an online source, including attributes such as product ID, name, brand, review text, and ratings.
- Ratings were categorized into sentiments (negative, neutral, positive) based on predefined criteria.

7.1.2 Data Preprocessing

- **Tokenization and Padding:** Text data underwent tokenization and padding to ensure consistent sequence lengths suitable for Seq2Seq model input.
- **Label Encoding:** Sentiments were encoded into categorical labels (negative, neutral, positive) for multi-class classification.
- **Train-Test Split:** The dataset was split into training (80%) and testing (20%) subsets to evaluate model performance.

```
text_column = 'reviews.text'
label_column = 'reviews.rating'

texts = data[text_column].astype(str).tolist()
labels = data[label_column].values
```

```
def map_rating_to_sentiment(rating):
    if rating == 1 or rating == 2:
        return 0  # Negative
    elif rating == 3:
        return 1  # Neutral
    elif rating == 4 or rating == 5:
        return 2  # Positive
    else:
        return 1  # Default to Neutral for unknown ratings

labels = data[label_column].apply(map_rating_to_sentiment).values

data['sentiment'] = labels
print(data[['reviews.text', 'reviews.rating', 'sentiment']])

num_classes = 3

assert len(texts) == len(labels), "The number of texts and labels must be the same."
```

```
max_features = 10000
maxlen = 100

tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

X = pad_sequences(sequences, maxlen=maxlen)

print(f"Shape of X (features) after padding: {X.shape}")

labels = to_categorical(labels, num_classes=num_classes)

X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)

print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")
print(f"X_test shape: {X_test.shape}, y_test shape: {y_test.shape}")
```

7.2 Model Development and Training

- **Architecture:** Implemented using TensorFlow and Keras with an Encoder-Decoder structure, LSTM layers, and attention mechanism for improved sequence processing.

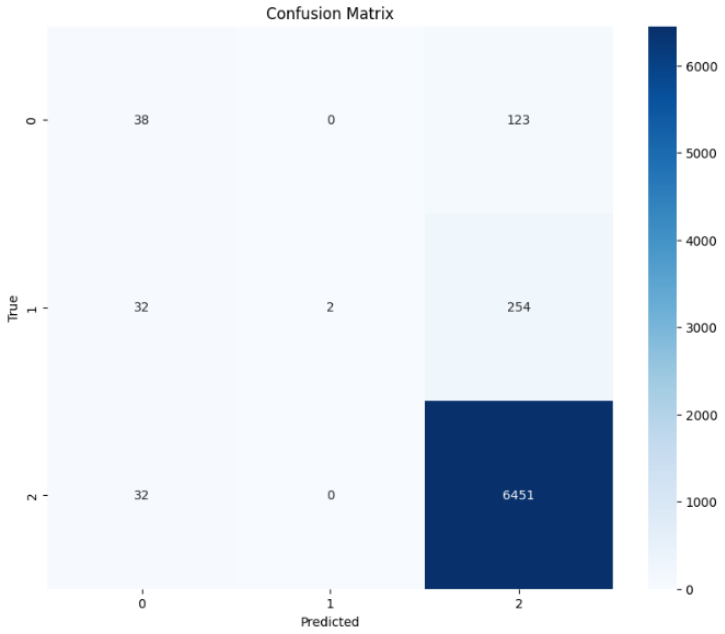
This model architecture utilizes word embeddings to represent each word as dense vectors of size 100. It incorporates a bidirectional LSTM layer with 64 units and dropout regularization to capture contextual dependencies in both forward and backward directions, enhancing sequence learning. The final softmax dense layer outputs probabilities for each class, optimized using categorical cross-entropy loss and Adam optimizer, with early stopping and model checkpointing to monitor validation loss and save the best-performing model during training.

```
embedding_dim = 100

model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=embedding_dim, input_length=maxlen))
model.add(SpatialDropout1D(0.2))
model.add(Bidirectional(LSTM(units=64, dropout=0.2, recurrent_dropout=0.2)))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

early_stopping = EarlyStopping(monitor='val_loss', patience=3, mode='min', restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, mode='min')

history = model.fit(X_train, y_train,
                    epochs=10,
                    batch_size=64,
                    validation_data=(X_test, y_test),
                    callbacks=[early_stopping, checkpoint])

print("Training complete.")
```

Trained across multiple epochs with early stopping to prevent overfitting. Evaluated using accuracy metrics and visualized using confusion matrix and classification report.

7.3 Model Evaluation

```
217/217 [==============================] - 5s 21ms/step
Classification report:
              precision    recall  f1-score   support

           0       0.37      0.24      0.29       161
           1       1.00      0.01      0.01       288
           2       0.94      1.00      0.97      6483

    accuracy                           0.94      6932
   macro avg       0.77      0.41      0.42      6932
weighted avg       0.93      0.94      0.91      6932
```



Findings

The Seq2Seq model achieved an accuracy of 93.64% on the test set.

Precision, recall, and F1-score for each sentiment class were as follows:

- Negative sentiment:
  - Precision: 37.26%
  - Recall: 23.60%
  - F1-score: 28.90%
- Neutral sentiment:
  - Precision: 100.00%
  - Recall: 0.69%
  - F1-score: 1.38%
- Positive sentiment:
  - Precision: 94.48%
  - Recall: 99.51%
  - F1-score: 96.93%

The confusion matrix illustrated the model's performance in predicting sentiment classes, highlighting strengths and areas for improvement.

Conclusion

- The Seq2Seq model demonstrated strong performance in sentiment analysis, leveraging its capability to capture sequential dependencies and context in textual data.
- Despite challenges like class imbalance, the model effectively classified sentiments in customer reviews.
- Future improvements could focus on refining the attention mechanism, optimizing hyperparameters, and addressing data biases to enhance model accuracy and applicability.

This report underscores the effectiveness of Seq2Seq models in sentiment analysis, offering valuable insights into customer sentiments that can inform product development and customer engagement strategies.

8 Summarization Model:

A summarization model was created in order to create summaries for predetermined sets by each star.

8.1 Set determination

To summarize the 34,000 reviews that span over 41 categories sets would need to be determined. As summarising per 41 categories for each star rating would result in over 200 summaries, diminishing the usefulness.

The categories were grouped by their similarities, into 5 separate sets.

Set 1: Tablets and E-Readers
Fire Tablets,Tablets,Computers & Tablets,All Tablets,Electronics, Tech Toys, Movies, Music,Electronics,iPad & Tablets,Android Tablets,Frys
Walmart for Business,Office Electronics,Tablets,Office,Electronics,iPad & Tablets,Windows Tablets,All Windows Tablets,Computers & Tablets,E-Readers & Accessories,E-Readers,eBook Readers,Kindle E-readers,Computers/Tablets & Networking,Tablets & eBook Readers,Electronics Features,Books & Magazines,Book Accessories,eReaders,TVs & Electronics,Computers & Laptops,Tablets & eReaders
Electronics,iPad & Tablets,All Tablets,Fire Tablets,Tablets,Computers & Tablets
Tablets,Fire Tablets,Computers & Tablets,All Tablets
Computers/Tablets & Networking,Tablets & eBook Readers,Computers & Tablets,Tablets,All Tablets
Walmart for Business,Office Electronics,Tablets,Electronics,iPad & Tablets,All Tablets,Computers & Tablets,E-Readers & Accessories,Kindle E-readers,Electronics Features,eBook Readers,See more Amazon Kindle Voyage (Wi-Fi),See more Amazon Kindle Voyage 4GB, Wi-Fi 3G (Unlocked...
Electronics,Tablets & E-Readers,Tablets,Back To College,College Electronics,College Ipads & Tablets,Featured Brands,Amazon Devices,Electronics Deals,Computers & Tablets,All Tablets,Electronics Features,eBook Readers
Electronics,iPad & Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers,Computers & Tablets,E-Readers & Accessories,E-Readers,Used:Computers Accessories,Used:Tablets,Computers,iPads Tablets,Kindle E-readers,Electronics Features
Tablets,Fire Tablets,Electronics,Computers,Computer Components,Hard Drives & Storage,Computers & Tablets,All Tablets
Computers & Tablets,Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers,Fire Tablets,Frys
Kindle E-readers,Electronics Features,Computers & Tablets,E-Readers & Accessories,E-Readers,eBook Readers
Electronics,iPad & Tablets,All Tablets,Computers & Tablets,Tablets,eBook Readers
Computers & Tablets,E-Readers & Accessories,eBook Readers,Kindle E-readers
Fire Tablets,Tablets,Computers & Tablets,All Tablets
eBook Readers,Kindle E-readers,Computers & Tablets,E-Readers & Accessories,E-Readers

Set 2: Amazon Devices and Accessories
Amazon Devices & Accessories,Amazon Device Accessories,Power Adapters & Cables,Kindle Store,Kindle E-Reader Accessories,Kindle Paperwhite Accessories
Electronics Features,Fire Tablets,Computers & Tablets,Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers
Fire Tablets,Tablets,Computers & Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers
Kindle Store,Categories,eBook Readers & Accessories,Fire TV Accessories,Electronics,Power Adapters & Cables,Amazon Device Accessories,Power Adapters
Electronics,eBook Readers & Accessories,Power Adapters,Computers/Tablets & Networking,Tablet & eBook Reader Accs,Chargers & Sync Cables,Power Adapters & Cables,Kindle Store,Amazon Device Accessories,Kindle Fire (2nd Generation) Accessories,Fire Tablet Accessories
Computers/Tablets & Networking,Tablet & eBook Reader Accs,Chargers & Sync Cables,Power Adapters & Cables,Kindle Store,Amazon Device Accessories,Fire Tablet Accessories,Kindle Fire (2nd Generation) Accessories
Computers/Tablets & Networking,Tablets & eBook Readers,Electronics,eBook Readers & Accessories,eBook Readers

Set 3: Smart Home and Audio Devices
Stereos,Remote Controls,Amazon Echo,Audio Docks & Mini Speakers,Amazon Echo Accessories,Kitchen & Dining Features,Speaker Systems,Electronics,TVs Entertainment,Clearance,Smart Hubs & Wireless Routers,Featured Brands,Wireless Speakers,Smart Home & Connected Living,Home Security,Kindle Store,Home Automation,Home, Garage & Office,Home,Voice-Enabled Smart Assistants,Virtual Assistant Speakers,Portable Audio & Headphones,Electronics Features,Amazon Device Accessories,iPod, Audio Player Accessories,Home & Furniture Clearance,Consumer Electronics,Smart Home,Surveillance,Home Improvement,Smart Home & Home Automation Devices,Smart Hubs,Home Safety & Security,Voice Assistants,Alarms & Sensors,Amazon Devices,Audio,Holiday Shop
Featured Brands,Electronics,Amazon Devices,Home,Home Improvement,Home Safety & Security,Home Security,Alarms & Sensors,Smart Home & Home Automation Devices,Mobile,Mobile Speakers,Mobile Bluetooth Speakers,Smart Hubs & Wireless Routers,Smart Hubs,Home, Garage & Office,Smart Home,Voice Assistants,Smart Home & Connected Living,Amazon Tap,Portable Audio,MP3 Accessories,Speakers,Amazon Echo,Electronics Features,TVs & Electronics,Portable Audio & Electronics,MP3 Player Accessories,Home Theater & Audio,Kindle Store,Frys,Electronic Components,Home Automation,Electronics, Tech Toys, Movies, Music,Audio,Bluetooth Speakers
TVs Entertainment,Wireless Speakers,Virtual Assistant Speakers,Featured Brands,Electronics,Amazon Devices,Home,Home Improvement,Home Safety & Security,Home Security,Alarms & Sensors,Smart Home & Home Automation Devices,Smart Hubs & Wireless Routers,Smart Hubs,Consumer Electronics,Voice-Enabled Smart Assistants,Smart Home & Connected Living,Home, Garage & Office,Smart Home,Voice Assistants,Surveillance,Home Automation,Speakers,Electronics Features,Holiday Shop,TV, Video & Home Audio,Internet & Media Streamers,Amazon Echo,Hubs & Controllers
Electronics,Amazon Device Accessories,Kindle Store,Covers,Kindle E-Reader Accessories,Kindle DX (2nd Generation, Global Wireless) Accessories
Power Adapters & Cables,Electronics,USB Cables

Set 4: College and Streaming Devices
Back To College,College Electronics,College Tvs & Home Theater,Electronics,Tvs & Home Theater,Streaming Devices,Featured Brands,Amazon Devices,Holiday Shop,Ways To Shop,TV & Home Theater,Streaming Media Players,All Streaming Media Players,TVs Entertainment,Video Games,Kindle Store,Electronics Features,Kids & Family,Fire TV
Electronics,Categories,Streaming Media Players,Amazon Devices
Electronics,Categories,Fire TV,Kindle Store
Fire Tablets,Tablets,Computers & Tablets,All Tablets,Frys
Categories,Streaming Media Players,Electronics

Set 5: Miscellaneous Electronics and Accessories
Chargers & Adapters,Computers & Accessories,Tablet & E-Reader Accessories,Amazon Devices & Accessories,Fire Tablet Accessories,Electronics,Power Adapters & Cables,Cell Phones,Amazon Device Accessories,Cell Phone Accessories,Cell Phone Batteries & Power,Tablet Accessories,Featured Brands,Kindle Fire (2nd Generation) Accessories,Kindle Store,Home Improvement,Fire (5th Generation) Accessories,Electrical,Amazon Devices,Home,Tablets & E-Readers,Cables & Chargers
Kindle Store,Amazon Devices,Electronics
Cases,Kindle Store,Amazon Device Accessories,Accessories,Tablet Accessories
Rice Dishes,Ready Meals,Beauty,Moisturizers,Lotions
Amazon Device Accessories,Kindle Store,Kindle Touch (4th Generation) Accessories,Kindle E-Reader Accessories,Covers,Kindle Touch (4th Generation) Covers
mazon.co.uk,Amazon Devices
Electronics,Computers,Computer Accessories,Cases & Bags,Fire Tablets,Electronics Features,Tablets,Computers & Tablets,Kids' Tablets,Electronics, Tech Toys, Movies, Music,iPad & Tablets,Top Rated
Frys,Software & Books,eReaders & Accessories,Tablet Cases Covers,Tablet Accessories,Computer Accessories
Electronics,eBook Readers & Accessories,Covers,Kindle Store,Amazon Device Accessories,Kindle E-Reader Accessories,Kindle (5th Generation) Accessories,Kindle (5th Generation) Covers

8.2 Mapping

This was then mapped into the dataframe, with each category being assigned to a set:

```
mapping = {
    "set1": [
        "Fire Tablets,Tablets,Computers & Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers",
        "Walmart for Business,Office Electronics,Tablets,Office,Electronics,iPad & Tablets,Windows Tablets,All Windows Tablets,Computers & Tablets,E-Readers & Accessories,E-Readers,eBook Readers,Kindle E-readers,Computers/Tablets & Networking,Tablets & eBook Readers,Electronics Features,Books & Magaz
        "Electronics,iPad & Tablets,All Tablets,Fire Tablets,Tablets,Computers & Tablets",
        "Tablets,Fire Tablets,Computers & Tablets,All Tablets",
        "Computers/Tablets & Networking,Tablet & eBook Reader Accs,Chargers & Sync Cables,Power Adapters & Cables,Kindle Store,Amazon Device Accessories,Fire Tablet Accessories,Kindle Fire (2nd Generation) Accessories",
        "Walmart for Business,Office Electronics,Tablets,Electronics,iPad & Tablets,All Tablets,Computers & Tablets,E-Readers & Accessories,Kindle E-readers,Electronics Features,eBook Readers,See more Amazon Kindle Voyage (Wi-Fi),See more Amazon Kindle Voyage 4GB, Wi-Fi  3G (Unlocked...",
        "Electronics,Tablets & E-Readers,Tablets,Back To College,College Electronics,College Ipads & Tablets,Featured Brands,Amazon Devices,Electronics Deals,Computers & Tablets,All Tablets,Electronics Features,eBook Readers",
        "Electronics,iPad & Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers,Computers & Tablets,E-Readers & Accessories,E-Readers,Used:Computers Accessories,Used:Tablets,Computers,iPads Tablets,Kindle E-readers,Electronics Features",
        "Tablets,Fire Tablets,Electronics,Computers,Computer Components,Hard Drives & Storage,Computers & Tablets,All Tablets",
        "Computers & Tablets,Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers,Fire Tablets,Frys",
        "Kindle E-readers,Electronics Features,Computers & Tablets,E-Readers & Accessories,E-Readers,eBook Readers",
        "Electronics,iPad & Tablets,All Tablets,Computers & Tablets,Tablets,eBook Readers",
        "Computers & Tablets,E-Readers & Accessories,eBook Readers,Kindle E-readers",
        "Fire Tablets,Tablets,Computers & Tablets,All Tablets",
        "eBook Readers,Kindle E-readers,Computers & Tablets,E-Readers & Accessories,E-Readers"
    ],
    "set2": [
        "Amazon Devices & Accessories,Amazon Device Accessories,Power Adapters & Cables,Kindle Store,Kindle E-Reader Accessories,Kindle Paperwhite Accessories",
        "Electronics Features,Fire Tablets,Computers & Tablets,Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers",
        "Fire Tablets,Tablets,Computers & Tablets,All Tablets,Computers/Tablets & Networking,Tablets & eBook Readers",
        "Kindle Store,Categories,eBook Readers & Accessories,Fire TV Accessories,Electronics,Power Adapters & Cables,Amazon Device Accessories,Power Adapters",
        "Electronics,eBook Readers & Accessories,Power Adapters,Computers/Tablets & Networking,Tablet & eBook Reader Accs,Chargers & Sync Cables,Power Adapters & Cables,Kindle Store,Amazon Device Accessories,Kindle Fire (2nd Generation) Accessories,Fire Tablet Accessories",
        "Computers/Tablets & Networking,Tablet & eBook Reader Accs,Chargers & Sync Cables,Power Adapters & Cables,Kindle Store,Amazon Device Accessories,Fire Tablet Accessories,Kindle Fire (2nd Generation) Accessories",
        "Computers/Tablets & Networking,Tablets & eBook Readers,Computers & Tablets,Tablets,All Tablets"
    ],
    "set3": [
        "Stereos,Remote Controls,Amazon Echo,Audio Docks & Mini Speakers,Amazon Echo Accessories,Kitchen & Dining Features,Speaker Systems,Electronics,TVs Entertainment,Clearance,Smart Hubs & Wireless Routers,Featured Brands,Wireless Speakers,Smart Home & Connected Living,Home Security,Kindle Store,
        "Featured Brands,Electronics,Amazon Devices,Home,Home Improvement,Home Safety & Security,Home Security,Alarms & Sensors,Smart Home & Home Automation Devices,Mobile,Mobile Speakers,Mobile Bluetooth Speakers,Smart Hubs & Wireless Routers,Smart Hubs,Home, Garage & Office,Smart Home,Voice Assista
        "TVs Entertainment,Wireless Speakers,Virtual Assistant Speakers,Featured Brands,Electronics,Amazon Devices,Home,Home Improvement,Home Safety & Security,Home Security,Alarms & Sensors,Smart Home & Home Automation Devices,Smart Hubs & Wireless Routers,Smart Hubs,Consumer Electronics,Voice-Enabl
        "Electronics,Amazon Device Accessories,Kindle Store,Covers,Kindle E-Reader Accessories,Kindle DX (2nd Generation, Global Wireless) Accessories",
        "Power Adapters & Cables,Electronics,USB Cables"
    ],
    "set4": [
        "Back To College,College Electronics,College Tvs & Home Theater,Electronics,Tvs & Home Theater,Streaming Devices,Featured Brands,Amazon Devices,Holiday Shop,Ways To Shop,TV & Home Theater,Streaming Media Players,All Streaming Media Players,TVs Entertainment,Video Games,Kindle Store,Electronic
        "Electronics,Categories,Streaming Media Players,Amazon Devices",
        "Electronics,Categories,Fire TV,Kindle Store",
        "Fire Tablets,Tablets,Computers & Tablets,All Tablets,Frys",
        "Categories,Streaming Media Players,Electronics"
    ],
    "set5": [
        "Chargers & Adapters,Computers & Accessories,Tablet & E-Reader Accessories,Amazon Devices & Accessories,Fire Tablet Accessories,Electronics,Power Adapters & Cables,Cell Phones,Amazon Device Accessories,Cell Phone Accessories,Cell Phone Batteries & Power,Tablet Accessories,Featured Brands,Kind
        "Amazon Device Accessories,Kindle Store,Kindle Touch (4th Generation) Accessories,Kindle E-Reader Accessories,Covers,Kindle Touch (4th Generation) Covers",
        "Cases,Kindle Store,Amazon Device Accessories,Accessories,Tablet Accessories",
        "mazon.co.uk,Amazon Devices",
        "Electronics,Computers,Computer Accessories,Cases & Bags,Fire Tablets,Electronics Features,Tablets,Computers & Tablets,Kids' Tablets,Electronics, Tech Toys, Movies, Music,iPad & Tablets,Top Rated",
        "Frys,Software & Books,eReaders & Accessories,Tablet Cases Covers,Tablet Accessories,Computer Accessories",
        "Electronics,eBook Readers & Accessories,Covers,Kindle Store,Amazon Device Accessories,Kindle E-Reader Accessories,Kindle (5th Generation) Accessories,Kindle (5th Generation) Covers"
    ]
}
```

```
def map_to_set(category, mapping):
    for set_name, categories in mapping.items():
        if category in categories:
            return set_name
    return None

df['new_set'] = df['categories'].apply(lambda x: map_to_set(x, mapping))
```

The mapping was used to add a new column into the dataframe with the set names in. And all reviews that did not have a review or a star were dropped.

```
Columns in the DataFrame:
Index(['new_set', 'review_text', 'review_rating'], dtype='object')
```

8.3 Model Selection and Building

The transformer selected for this summarisation was Flan-T5-Large because FLAN-T5-large model is useful for summarization because it is a transformer-based model that has been fine-tuned on a diverse set of language tasks, including summarization. It leverages the capabilities of the T5 (Text-To-Text Transfer Transformer) architecture, which is designed to handle various NLP tasks by converting them into a text-to-text format. This allows the model to generate high-quality summaries by understanding and condensing the input text effectively. Additionally, its large size provides a higher capacity to capture and generate more nuanced and accurate summaries.

This model processes a dataset of reviews by iterating through different sets and review ratings using for loops. For each combination, it aggregates all reviews into a single string, then uses a pre-trained transformer model Flan-T5 to generate a summary of these reviews. The resulting summaries are stored in a DataFrame along with their corresponding set and rating categories for further analysis.

```
summary_results = pd.DataFrame(columns=['category', 'review_rating', 'summary'])

sets = ['set1', 'set2', 'set3','set4','set5']
for set_name in sets:
    df_set = df1[df1['new_set'] == set_name]
    for rating in range(1, 6):
        df_rating = df_set[df_set['review_rating'] == rating]
        print(df_rating.shape)
        review = ' '.join(df_rating['review_text'])

        inputs = tokenizer.encode("summarize: " + review, return_tensors="pt", max_length=2048, truncation=True)
        summary_ids = model.generate(inputs, max_length=300, min_length=30, length_penalty=2.0, num_beams=2, early_stopping=True)
        summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)


        new_row = pd.DataFrame({'category': [set_name], 'review_rating': [rating], 'summary': [summary]})
        summary_results = pd.concat([summary_results, new_row], ignore_index=True)
```

8.4 Model Results

Some examples:

Category	Rating	Summary
set1	1	I had to return this product as it was not compatible with things such as Facebook and Itunes... which I had to return this product as it was not
set1	2	I could give this tablet 0 stars, I would. It's a good tablet for children and adults too. However there is an issue with the port and charger cable. We did not even have this tablet 3 weeks before it would barley charge. Took it back to best buy and they said they could not fox it, and imagine the port is not covered under the manufacturer warranty. So now I have a tablet that takes a couple days to charge. Its really frustrating. The charging plugs on these kindle damage way to easy. I am very disappointed as this is not the first time this has happened to one of my kindles. So I bought my little guys these tablets... so that I could repossess my IPad. Love the free time app but who could've imagined they wouldn't play NETFLIX!! So now... I'm out the money for the tablet and still don't have my iPad back... and yes, I called Amazon numerous times... they work after the fix, but the next time they go to use them they don't work again... the fix takes several minutes as it's force stopping the app, clearing cache then doing a hard restart... then you have to wait to restart Netflix.... oh and sometimes you have to do it numerous times before it works...
set1	3	Amazon apps. I'm not sure if it's worth the money. It's a good tablet for the price. It's a good deal for the price.
set1	4	this product is really awesome for kids to play in all sorts.We can use a strong case and let kids play in it.I like the features it has and it is easy to navigate. This perfect my kids to do their school work on and have fun as well. The price is low because this product is really awesome for kids to play in all sorts.
set1	5	Great tablet for the price. Don't expect it to be as fast as more expensive ones. It's a great tablet for the price.

Limitations;

Due to the structure of the dataframe, some of the summaries were less detailed as there was a fewer number of reviews in the set or with the desired star rating. In the future it would be more effective to use a less imbalanced dataframe.

Conclusion:

In conclusion, the Flan-T5-Large summarization model was successfully utilized to generate summaries for 34,000 reviews across 41 categories, which were consolidated into five sets to enhance manageability and relevance. By mapping categories to these sets and filtering out incomplete data, the model iteratively aggregated reviews by star ratings and produced concise summaries, which were then organized into a DataFrame for analysis. While the model performed well, generating high-quality summaries that reflect the essence of the reviews, some limitations were noted due to the imbalance and variability in the number of reviews per category and rating. Future improvements could include addressing data imbalance to ensure more comprehensive and detailed summaries across all sets and star ratings, as well as more in depth finetuning.