



**TAYLOR'S
UNIVERSITY**

Wisdom • Integrity • Excellence

IT'S IN YOU TO **OWN THE FUTURE**

Student Declaration

I declare that:

- I understand what is meant by plagiarism
- The implication of plagiarism has been explained to us by our lecturer

This project is all our work, and I have acknowledged any use of the published or unpublished works of other people

Name of the Group Members

No.	Student ID	Student Name
1.	0361754	JONATHAN TAN KIAN HOONG
2.	0350660	JEREMY LEE MING HENG
3.	0364105	ABDUL MALIK

Marking Rubric

Criteria	Weight	Your Score
Abstract	5	
Objective	3	
Problem Identification	2	
Dataset Description	5	
Model	5	
Evaluation	5	
Presentation	5	
Total (30)		

Table of contents

Marking Rubric	2
Abstract	3
Literature Review	4
Objective.....	4
Problem statement	5
Dataset Description	5
Key Attributes	5
Relevance to the Problem	6
Model Training and Testing, and Problem-Solving Approach	7
Pre-processing	7
SVM for Amazon kindle reviews	9
Roberta Pre-trained weights.....	12
Roberta Pre-trained weights with fine tuning on our dataset	14
Comparison and key takeaway.....	15
References	16

Abstract

Consumer feedback is essential for establishing consumer preferences and pinpointing areas in need of development. The issue of properly determining whether Amazon Kindle reviews of books are neutral, negative, or positive according to unstructured language is the focus of this study. Our group use a modern neural network model, RoBERTa, to evaluate the review content while taking contextual subtleties taken into consideration, leveraging the "[Amazon Kindle Book Reviews for Sentiment Analysis](#)" Dataset. The classification of sentiments becomes challenging due to factors like clutter and class disparities in the dataset, since it is packed with textual feedback and data like users' star ratings and IDs. The model's ultimate score for accuracy was is 66.83% leveraging advanced Natural Language Processing techniques like stop-word removal, POS-Tagging, lemmatization, and tokenization. The model is capable of matching textual data using sentiment classes thanks to star ratings, which act as supervised learning markers. RoBERTa's ability to record context of words became the major focus to overcome the difficulties caused by unstructured text, such as typos and slangs. The precision of sentiment classification was substantially boosted by integrating textual data using star-based ratings, demonstrating its ability to detect trends in a broad spectrum of sentiments. This will only become more effective after future model refining and also expanding the analysis to additional domains. Besides refining sentiment analysis methods, this report shows how crucial it is for businesses to fully understand consumer preferences and improve their goods and services.

Literature Review

Sentiment analysis has become increasingly crucial for analysing the rapidly expanding viewpoints on social media along with other websites where users can share their opinions, emotions, and comments. Sentiment analysis is employed to ascertain if the text's sentiment represents neutral, positive, negative, or some other subjective emotion (Tan et al., 2022). Preprocessing data constitutes a key element, particularly for tasks like sentiment analysis. Lemmatization, tokenization, lowercasing, and stop-word removal are the fundamental preprocessing methods used to assist in data cleaning and boost model performance (Liao, 2024). By using a transformer model like RoBERTa, that converts words into an embedding space that is both concise and comprehensible, have shown significant improvements in terms of accuracy compared to other models such as LSTMS (Uddagiri Sirisha and Bolem Sai Chandana, 2022). The model's ability to

effectively categorize sentiments can be significantly improved by integrating metadata such as star ratings into the sentiment analysis (Wida Sofiya and Setiawan, 2023). Challenges such as interpretability, and overfitting are still common issues that remains due to the complicated nature of classes and noisy data. This can be resolved by the data preprocessing steps and utilizing the data-expansion strategy, a form of fine-tuning, which utilize extensive amount of data to refine the hyper-parameter sets (Ying, 2019). Even though preprocessing methods alongside with transformer models such as RoBERTa have substantially improved sentiment analysis, issues including interpretability and overfitting nevertheless exist. Fortunately, incorporating metadata and applying data-expansion methods offer viable options to improve the model's efficiency.

Objective

The aim of this assignment is to carry out sentimental analysis on Amazon Kindle review dataset to determine whether a review is positive, negative or neutral based on the review text given. Most of the time, reviews are written by humans and determining whether a review is positive or not solely relies on the context of the review, thus we will be training a neural network model called Roberta which will account for words and also the context related to the words.

Problem statement

In today's competitive market, customer reviews play a crucial role in determining customer preferences and also determining improvement for the product or services. Sentiment analysis of these reviews will provide valuable insights for companies in customer satisfaction and identify areas that requires improvement. The primary problem we will be addressing is how can we accurately classify the sentiment of Amazon Kindle book reviews using the provided text and associated metadata?

Dataset Description

The dataset used, titled "[Amazon Kindle Book Reviews for Sentiment Analysis](#)" Dataset consist of a plethora of textual feedback that, when analysed effectively, can reveal to us customer sentiment. Other fields like review titles, helpful votes, reviewer IDs, and review dates add additional context for the analysis. However, due to the unstructured nature of the dataset such as noise and data imbalances in sentiment classes, identifying and classifying sentiment accurately will pose a significant

challenge. Although the dataset contains noise in the data, the dataset is well suited for natural language processing (NLP) tasks such as sentiment classification, where the review text serves as input and the star ratings serve as ground truth labels. Its diverse content makes it a valuable resource for training models to understand consumers preferences and trends.

Key Attributes

1. reviewText
Main content of user's review which is the core unstructured data used to determine sentiment whether its positive, negative, or neutral.
2. rating
Numerical score (typically between 1 and 5) given by the reviewer to imply the satisfaction of the reviewer. Star Rating serves as a labelled data for supervised learning tasks like sentiment classification.
3. reviewerID
Consist of numbers and alphabets which is generated for each reviewer. Its primary function is to create a unique reference for each individual reviewer, which can be leveraged for various analytical and processing purposes. On the other hand, reviewer ID facilitates the identification of patterns or biases in reviewer behaviour such as frequent reviewers, reviewers with consistently high or low ratings.

Relevance to the Problem

The primary input of sentiment analysis is unstructured textual data which contains the reviewer's subjective expressions, opinions, and emotions. Textual data can range from short concise text such as "Loved it!" to detailed reviews explaining the strength and weakness of a book. This text often includes noise such as typos, slang, emojis, or varying sentence structure which can be challenging to train the machine learning model. In order to reduce the challenges, we are going to use advanced language processing (NLP) techniques like tokenization, stop word removal, and stemming/lemmatization to preprocess and clean the data to prepare the text for further analysis. This dataset provides an abundance of reviews allowing us to train a model to learn and generalize patterns effectively across diverse sentiments.

Secondly, Star ratings serve as labelled outputs when it comes to supervised sentiment analysis tasks. For example, 1-to-2-star rating correspond to negative sentiment, 3 stars rating correspond to neutral sentiment, and 4 to 5 stars to positive sentiment. These labels allow the model to build a solid relationship between review

text and the root sentiment during training process. By leveraging star ratings in our model, our model will have a better consistency in defining sentiment categories, which might otherwise be subjective. By incorporating star ratings with textual reviews, our trained model will be able to gain a comprehensive understanding of how specific phrases, words, or patterns align with user satisfaction level. This stacking of both textual data with star-based labels forms a solid foundation for building an effective sentiment analysis model.

Model Training and Testing, and Problem-Solving Approach

Pre-processing

We first started by pre-processing our dataset to convert capital letters to lower cases, removing punctuations, stop words, frequent and rare words, and special characters to clean our dataset for tokenization, lemmatization and POS-tagging.

The code sections below display all of the pre-processing steps in detail.

▼ Converting to Lower Case

```
[ ] df['cleaned-review'] = df['reviewText'].str.lower()
df.head()
```

▼ Removing Punctuation

```
[ ] import string
string.punctuation
```

```
↗ '!"#%&'()*+,-./:;<=>@[\\]^_`{|}~'
```

```
[ ] def remove_punctuations(text):
    # Ensure the text is a string (convert NaN or other non-string values to empty string)
    if not isinstance(text, str):
        text = ""

    punctuations = string.punctuation
    return text.translate(str.maketrans('', '', punctuations))
```

```
[ ] df['cleaned-review'] = df['cleaned-review'].apply(lambda x: remove_punctuations(x))
df.head(10)
```

▼ Removing Stopwords

```
[ ] import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
↗ [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
[ ] from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

```
↗ 'i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's,
her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, ar
e, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, w
ith, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, the
n, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, ver
y, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't,
hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, must...'
```

```
[ ] Stopwords = set(stopwords.words('english'))
def remove_stopwords(text):
    # Ensure the text is a string (convert NaN or other non-string values to empty string)
    if not isinstance(text, str):
        text = ""
    return " ".join([word for word in text.split() if word not in Stopwords])
```

```
[ ] df['cleaned-review'] = df['cleaned-review'].apply(lambda x: remove_stopwords(x))
df.head()
```

▼ Removing Frequent Words

```
[ ] from collections import Counter
word_count = Counter()
for text in df['cleaned-review']:
    for word in text.split():
        word_count[word] += 1

word_count.most_common(10)
```

▼ Removing Rare Words

```
[ ] Rare_words = set((word, wc) for (word, wc) in word_count.most_common()[::-10:-1])
Rare_words
```


▼ Removing Special Characters

```
[ ] import re
def remove_spl_chars(text):
    text = re.sub('[^a-zA-Z0-9]', ' ', text)
    text = re.sub('\s+', ' ', text)
    return text

[ ] df['cleaned-review'] = df['cleaned-review'].apply(lambda x: remove_spl_chars(x))
df.head()
```

▼ Lemmatization & POS-Tagging

```
[ ] from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
wordnet_map = {"N": wordnet.NOUN, "V": wordnet.VERB, "J": wordnet.ADJ, "R": wordnet.ADV}

def lemmatize_words(text):
    # Splits text into words and find POS tags
    pos_text = pos_tag(text.split())
    # Lemmatize each word based on its POS tags
    return " ".join([lemmatizer.lemmatize(word, wordnet_map.get(pos[0], wordnet.NOUN)) for word, pos in pos_text])

[ ] wordnet.NOUN

[ ] 'n'

[ ] # Applying the function to the 'cleaned-review' column
df['lemmatize_text'] = df['cleaned-review'].apply(lambda x: lemmatize_words(x))
df.head()
```

This is our dataset after all of the pre-processing steps.

	rating	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime	cleaned-review	stemmed_text	lemmatize_text
0	3	Jace Rankin may be short, but he's nothing to ...	09 2, 2010	A3HHXRELK8BHQG	Ridley	Entertaining But Average	1283385600	jace rankin may short hes nothing mess man hau...	jace rankin may short he noth mess man haul sa...	jace rankin may short he nothing mess man haul...
1	5	Great short read. I didn't want to put it dow...	10 8, 2013	A2RGNZ0TRF578I	Holly Butler	Terrific menage scenes!	1381190400	great short read didnt want put read one sitti...	great short read didnt want put read one sit s...	great short read didnt want put read one sit s...
2	3	I'll start by saying this is the first of four...	04 11, 2014	A3S0H2HV6U1I7F	Merissa	Snapdragon Alley	1397174400	ill start saying first four books wasnt expect...	ill start say first four book wasnt expect 34c...	ill start say first four book wasnt expect 34c...
3	3	Aggie is Angela Lansbury who carries pocketboo...	07 5, 2014	AC4OQW3GZ919J	Cleargrace	very light murder cozy	1404518400	aggie angela lansbury carries pocketbooks inst...	aggi angela lansburi carri pocketbook instead ...	aggie angela lansbury carry pocketbook instead...
4	4	I did not expect this type of book to be in li...	12 31, 2012	A3C9V987IQHOQD	Rjostler	Book	1356912000	expect type book library pleas find price right	expect type book librari pleas find price right	expect type book library please find price right

SVM for Amazon kindle reviews

We first trained SVM to get an understanding at how a non-neural network performs on sentiment analysis task for our dataset before training Roberta. This model will be used for comparisons later with Roberta.

Before training, testing, and evaluating our SVM, we first checked the size of our dataset and more particularly the size of each sentiment category; positive, neutral, negative.

Here we found out the sentiment labels were unbalanced as there were only 2000 neutral while negatives and positives were 4000 and 6000 respectively making up to a total of 12000 rows of data. So, we ended up only using 2000 rows for each of the label to ensure a fairer training and no overfitting of one sentiment category as positive labels will dominate over negative and neutral in that case.

```
[ ] len(df)
↗ 12000

[ ] df_neg = df.loc[df['rating'] < 3]
df_neg = df_neg.reset_index(drop=True) # Reset the index of the resulting DataFrame

[ ] df_post = df.loc[df['rating'] > 3]
df_post = df_post.reset_index(drop = True)

[ ] df_neut = df.loc[df['rating'] == 3]
df_neut = df_neut.reset_index(drop = True)

[ ] print(len(df_neg))
print(len(df_post))
print(len(df_neut))
↗ 4000
6000
2000

[ ] df_post = df_post.iloc[:len(df_neut)]
df_neg = df_neg.iloc[:len(df_neut)]

[ ] print(len(df_neg))
print(len(df_post))
print(len(df_neut))
↗ 2000
2000
2000

[ ] df_all = pd.concat([df_neg, df_post, df_neut], axis=0)
df_all = df_all.reset_index(drop=True)

[ ] len(df_all)
↗ 6000
```

Additionally, we also set up the sentiment column to display if each review was positive, negative, or neutral based on ratings. Positives will be rating of 3 and above, neutral will be exactly 3 and negative will be below 3.

✓ Setting up sentiment column to classify reviews as positive, negative, or neutral

```
[ ] df_all["Sentiment"] = np.where(
    df_all['rating'] > 3, "Positive",
    np.where(df_all['rating'] == 3, "Neutral", "Negative")
)

[ ] df_all = df_all.sample(frac = 1)
df_all = df_all.reset_index(drop = True)

[ ] df_all.head()
```

The first 5 rows of our dataset with Sentiment column.

df_all.head()

	rating	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime	cleaned-review	stemmed_text	lemmatize_text	Sentiment
0	2	I enjoyed the novella, it follows her standard...	06 8, 2014	ARTIXYNNNGHT2M	B. Brockway "Avid Reader"	If its not free, you overpaid.	1402185600	enjoyed novella follows standard felt ripped t...	enjoy novella follow standard felt rip though ...	enjoy novella follow standard felt rip though ...	Negative
1	4	Love the author. Unique in that combine scienc...	01 19, 2013	A3BSDITWB4NOMN	Susan Landowski "SUEMLANDO"	Combines Menage with Science fiction	1358553600	love author unique combine science fiction ero...	love author uniqu combin scienc fiction erotic...	love author unique combine science fiction ero...	Positive
2	3	I have not finished this book yet. Yes, I am ...	04 15, 2013	A2DM065PBHMGY	Kimberly Sansone	WTW Batman... I mean, Mr. Smoke.	1365984000	finished book yet yes almost finished reading ...	finish book yet ye almost finish read crazy as...	finish book yet yes almost finish reading crazy...	Neutral
3	3	Cloning Jesus Christ seems like a good place t...	01 31, 2012	A2YHQ1K8U290GN	Saltraker "Saltraker"	Interesting plot but poorly executed	1327968000	cloning jesus christ seems like good place sta...	clone jesu christ seem like good place start ...	clone jesus christ seem like good place start ...	Neutral
4	5	The heroine was spunky and didn't take any cra...	07 14, 2014	AP92OPNW68BM6	NaN	Different, passionate!	1405296000	heroine spunky didnt take crap especially phil...	heroin spunki didnt take crap especi philli ac...	heroine spunky didnt take crap especially phil...	Positive

Next, we moved onto training and testing our data using a 70/30 split.

Training and testing

```
[49] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_all.lemmatize_text, df_all.Sentiment, test_size=0.3, random_state=42 )

[50] from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
x_train_vector = v.fit_transform(x_train)
x_test_vector = v.transform(x_test)
```

We used linear kernels for our SVM since we assume there is approximately linear relationship between sentiment and text.

Classification model (SVM)

```
[51] from sklearn.feature_extraction.text import TfidfVectorizer

# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000)
x_train_tfidf = tfidf.fit_transform(x_train)
x_test_tfidf = tfidf.transform(x_test)
```

```
[52] from sklearn.svm import SVC
# Train SVM model
clf = SVC(kernel='linear', class_weight='balanced')
clf.fit(x_train_tfidf, y_train)
```

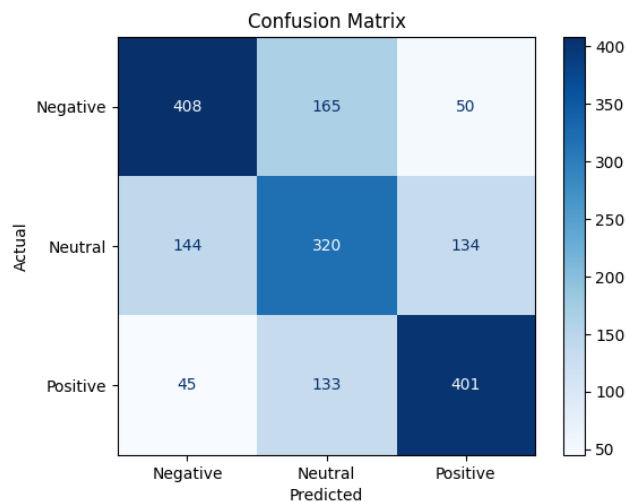
```
SVC
SVC(class_weight='balanced', kernel='linear')
```

```
[57] # Predict and evaluate
y_pred = clf.predict(x_test_tfidf)
```

Finally, we evaluated our SVM model to see its accuracy score, precision, recall, f1-score, support, and confusion matrix

	precision	recall	f1-score	support
Negative	0.68	0.65	0.67	623
Neutral	0.52	0.54	0.53	598
Positive	0.69	0.69	0.69	579
accuracy			0.63	1800
macro avg	0.63	0.63	0.63	1800
weighted avg	0.63	0.63	0.63	1800

Confusion Matrix:
[[408 165 50]
[144 320 134]
[45 133 401]]



The accuracy score we got was 63%.

Roberta Pre-trained weights

For our project, we chose the Roberta model as the neural network model for sentiment analysis of our Amazon Kindle reviews dataset. We loaded Roberta with the pre-trained weights that has been trained on Twitter tweet that was fine-tuned for sentiment analysis and been trained on informal languages which is applicable to the reviews we are using.

Loading pre-trained Roberta model

```
[ ] # Load pre-trained RoBERTa model and tokenizer
MODEL = "cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL, num_labels=3)

# Example review text
example = "This book is incredibly slow paced, the characters lack development and most importantly, the ending sucks!"

# Function to get RoBERTa sentiment scores (used for evaluation)
def polarity_scores_roberta(text):
    try:
        # Tokenize the input text (truncate to max length of 512 tokens)
        encoded_text = tokenizer(text, return_tensors='pt', truncation=True, max_length=512)

        # Perform inference
        output = model(**encoded_text)
        scores = output[0][0].detach().numpy()
        scores = softmax(scores)

        # Return scores as a dictionary
        return {
            'roberta_neg': scores[0],
            'roberta_neu': scores[1],
            'roberta_pos': scores[2]
        }
    except Exception as e:
        print(f"Error processing text: {text[:50]}... - {str(e)}")
        return None # Return None for problematic texts

# Split the dataset into train and test
train_df, test_df = train_test_split(df_all, test_size=0.2, random_state=42)
```

config.json: 100%  747/747 [00:00<00:00, 43.0kB/s]

vocab.json: 100%  899k/899k [00:00<00:00, 1.69MB/s]

merges.txt: 100%  456k/456k [00:00<00:00, 883kB/s]

We then ran the pre-trained model on our test set of 1200 rows.

Run pre-trained model on test set

```
[ ] # Initialize result storage
res = {}
true_labels = [] # Store true labels for the test set
predictions = [] # Store predicted labels for the test set

# Iterate through the test set (now using only the test set for evaluation)
for i, row in tqdm(test_df.iterrows(), total=len(test_df)):
    text = row['lemmatize_text'] # Extract the lemmatized text
    myid = row['reviewerID'] # Extract the reviewer ID
    true_labels.append(row['Sentiment']) # True sentiment label (append to list correctly)

    # Compute RoBERTa sentiment scores
    roberta_result = polarity_scores_roberta(text)

    if roberta_result: # Only save results if not None
        # Get the predicted sentiment label based on the max score
        predicted_label = max(roberta_result, key=roberta_result.get)
        predictions.append(predicted_label) # Store predicted label
        res[myid] = roberta_result # Store the result for each review

# Convert results to DataFrame
results_df = pd.DataFrame(res).T.reset_index().rename(columns={'index': 'reviewerID'})

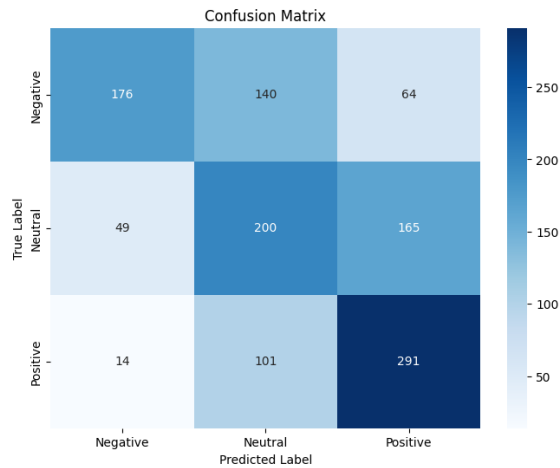
# Map predicted labels to numeric values for confusion matrix
prediction_map = {'roberta_neg': 0, 'roberta_neu': 1, 'roberta_pos': 2}
predictions_numeric = [prediction_map[label] for label in predictions]

# Convert true labels to numeric values
true_labels_numeric = [label_map[label] for label in true_labels]
```

100%  1200/1200 [04:58<00:00, 4.01it/s]

Accuracy Score: 55.58%

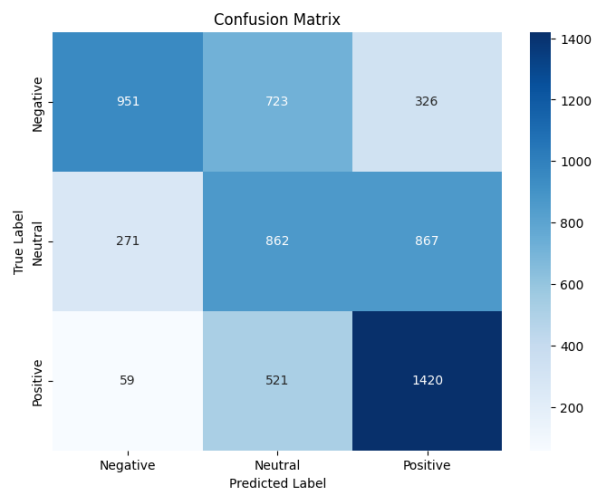
Classification Report:				
	precision	recall	f1-score	support
Negative	0.74	0.46	0.57	380
Neutral	0.45	0.48	0.47	414
Positive	0.56	0.72	0.63	406
accuracy			0.56	1200
macro avg	0.58	0.55	0.56	1200
weighted avg	0.58	0.56	0.55	1200



From the evaluation we carried out, we found that our Roberta model with pre-trained weights on the test set of 1200 had an accuracy score of 55.58%.

Accuracy Score: 53.88%

Classification Report:				
	precision	recall	f1-score	support
Negative	0.74	0.48	0.58	2000
Neutral	0.41	0.43	0.42	2000
Positive	0.54	0.71	0.62	2000
accuracy			0.54	6000
macro avg	0.57	0.54	0.54	6000
weighted avg	0.57	0.54	0.54	6000



To see if it will make any difference, we ran the model on the whole dataset df_all of 6000 rows and in turn we got a slightly lower accuracy score of 53.88%. So, now to increase the accuracy we know the best way was to fine-tune the Roberta model with pre-trained weights on our train dataset to learn the trends and classify positive, negative, and neutral reviews on our dataset better.

Roberta Pre-trained weights with fine tuning on our dataset

We carried out similar steps to the previous code but added on the epoch training and optimizers for fine-tuning process.

- ✓ Fine tuning the pre-trained model on our train set

```
[ ] from transformers import AdamW
import torch
from tqdm import tqdm

# Define optimizer
optimizer = AdamW(model.parameters(), lr=5e-5)
loss_fn = torch.nn.CrossEntropyLoss()

# Number of epochs and iterations per epoch
epochs = 3
iterations_per_epoch = 300


for epoch in range(epochs):
    total_loss = 0
    for iteration, batch in enumerate(tqdm(train_dataloader, desc=f"Training Epoch {epoch + 1}/{epochs}")):
        optimizer.zero_grad()

        # Tokenize the batch
        inputs = tokenizer(batch['text'], padding=True, truncation=True, return_tensors="pt", max_length=512).to(device)
        labels = torch.tensor(batch['label']).to(device)

        # Forward pass and compute loss
        outputs = model(**inputs)
        loss = loss_fn(outputs.logits, labels)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    # Print loss after each epoch
    print(f"Epoch {epoch + 1} Loss: {total_loss / len(train_dataloader):.4f}")
```

 /usr/local/lib/python3.10/dist-packages/transformers/optimization.py:591: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Please use the implementation in torch.nn.optim.AdamW instead. Triggered by torch.optim.AdamW.

Training Epoch 1/3: 0% | 0/300 [00:00<, ?it/s] <ipython-input-56-53bd29edf0c5>:20: UserWarning: To copy construct from a tensor, it is recommended to use source tensor.clone() as copying should not be performed. Triggered by torch.nn.functional.cross_entropy.

Training Epoch 1/3: 100% | 300/300 [03:08<00:00, 1.59it/s]

Epoch 1 Loss: 0.8378

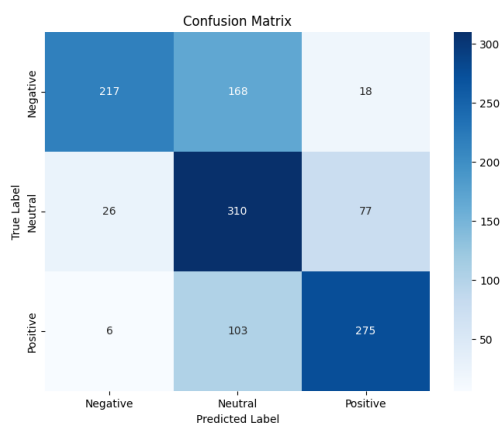
Training Epoch 2/3: 100% | 300/300 [03:19<00:00, 1.50it/s]

Epoch 2 Loss: 0.5514

Training Epoch 3/3: 100% | 300/300 [03:15<00:00, 1.53it/s] Epoch 3 Loss: 0.2968

Accuracy Score: 66.83%

Classification Report:				
	precision	recall	f1-score	support
Negative	0.87	0.54	0.67	403
Neutral	0.53	0.75	0.62	413
Positive	0.74	0.72	0.73	384
accuracy			0.67	1200
macro avg	0.72	0.67	0.67	1200
weighted avg	0.71	0.67	0.67	1200



Finally, we got an accuracy score of 66.83%

Comparison and key takeaway

To conclude our work, we will like to first mention that in the end the model with the best performance in terms of accuracy score was as expected Roberta model with pre-trained weights and fine-tuning with a score of 66.83% followed by SVM model with an accuracy of 63% and finally by Roberta model with no fine-tuning to our dataset with an accuracy of 55.58% and 53.88%.

This tells us that neural-network models due to them being built on multiple layer architecture. The deeper layers help the model understand the hierarchical structure of language which allows it to capture subtle cues about sentiment and hence be able to classify the sentiment labels better than SVM can. Though, it is important to mention in our case SVM did not fall behind but perhaps if we had adjusted the Roberta model fine-tuning with more epoch for training, the accuracy might be higher or if we had used another neural-network model like ELECTRA it could've given us a higher accuracy score.

References

- Dataset link: <https://www.kaggle.com/datasets/meetnagadia/amazon-kindle-book-review-for-sentiment-analysis>
- Mehta, J. (2023). *The impact of customer reviews on growth*. [online] abmatic.ai. Available at: <https://abmatic.ai/blog/impact-of-customer-reviews-on-growth>.
- IBM (2023). *What is sentiment analysis?* | IBM. [online] www.ibm.com. Available at: <https://www.ibm.com/topics/sentiment-analysis>.
- AWS (2023). *What is Sentiment Analysis? - Sentiment Analysis Explained* - AWS. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/what-is/sentiment-analysis/>.
- FINE-GRAINED SENTIMENT ANALYSIS IN SOCIAL MEDIA USING GATED RECURRENT UNIT WITH SUPPORT VECTOR MACHINE By Wida Sofiya, Erwin Budi Setiawan
Container: Jurnal Teknik Informatika (Jutif)
Year: 2023
Volume: 4
Issue: 3
DOI: 10.52436/1.jutif.2023.4.3.855
URL: https://www.researchgate.net/publication/372982996_FINE-GRAINED_SENTIMENT_ANALYSIS_IN_SOCIAL_MEDIA_USING_GATED_RECURRENT_UNIT_WITH_SUPPORT_VECTOR_MACHINE
- Sentiment analysis based on machine learning models By Xinya Liao
Container: Applied and Computational Engineering
Year: 2024
Volume: 54
Issue: 1
DOI: 10.54254/2755-2721/54/20241434
URL: https://www.researchgate.net/publication/379406018_Sentiment_analysis_based_on_machine_learning_models
- Aspect based sentiment & emotion analysis with roberta, LSTM By Uddagiri Sirisha, Boleem Sai Chandana
Container: International Journal of Advanced Computer Science and Applications
Year: 2022
Volume: 13
Issue: 11
DOI: 10.14569/ijacsa.2022.0131189
- (PDF) An Overview of Overfitting and its Solutions By
Container: ResearchGate
URL: https://www.researchgate.net/publication/331677125_An_Overview_of_Overfitting_and_its_Solutions
- RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network By Kian Long Tan, Chin Poo Lee, Kalaiarasi

Sonai Muthu Anbananthen, Kian Ming Lim
Container: IEEE Access
Year: 2022
Volume: 10
DOI: 10.1109/access.2022.3152828