Google Play Application Reviews Sentiment Analysis

問題定義

這次期末專題想試試看簡單的 NLP 任務，決定做 sentiment analysis，會從 google play 的網站上挑選幾個應用程式，並抓取應用程式的評論當作資料集，評論有一顆星到五顆星，我的想法是把是把 1~2 星的評論當作 negative，3 星為 neutral，4~5 星為 positive。訓練一個模型讓模型使模型能預測某評論是正向、負面，或是中立。

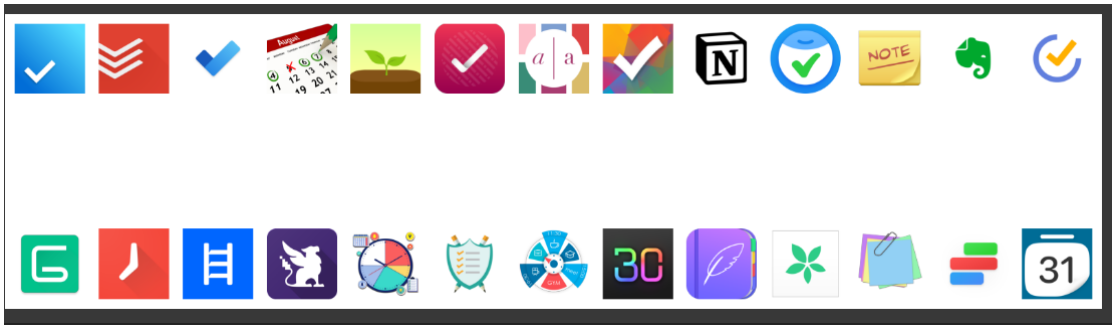
資料蒐集

Github python library:　google-play-scraper
他有提供一些方便的 API 可以將 google play 上面的應用程式的資料爬下來以 json 檔的形式呈現，這邊附上這個 library 的 github 連結:
https://github.com/JoMingyu/google-play-scraper

爬下來的資料長這樣，要使用的部分是 content 還有 score


```
{
  "appId": "com.anydo",
  "at": "2022-04-25 15:55:21",
  "content": "I used to love this app. I have used it for years. I even had (until yesterday) the paid version. I primarily use this app to sync up my alexa ta
  "repliedAt": "2022-04-26 16:53:39",
  "replyContent": "Unfortunately, Amazon has rejected our fix and will not be reopening the Any.do Skill just yet.\nAt this point in time, our efforts seem to
  "reviewCreatedVersion": "5.15.4.2",
  "reviewId": "99974541-f5b2-4b30-8053-77d120d6d6e3",
  "score": 1,
  "sortOrder": "most_relevant",
  "thumbsUpCount": 35,
  "userImage": "https://play-lh.googleusercontent.com/a/AATXAJzFdtk_ASxO4LxjZWEzWXUheo_ISStcYw10go8Z=mo",
  "userName": "Brent Schoemann"
}
```


App Category:　Productivity apps
我一開始原本打算抓遊戲類的應用程式來當作資料集，但 model performance 非常不理想，所以想說換一個 domain 蒐集 review，最後選擇 productivity 類別，這個分類主要是一些行程規劃、備忘錄、隨手筆記類型的應用程式，我總共抓取了 26 個 app 的 review。下圖為 26 個目標 app

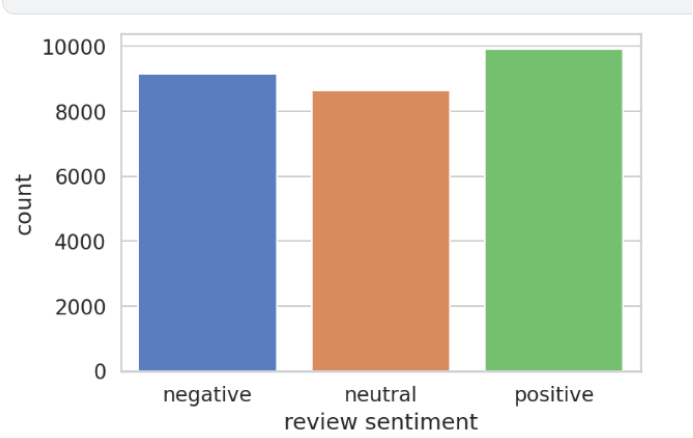


Total Collected Reviews: 27702

EDA

先觀察 1 star ~ 5 stars review 的分布，3 stars review 的數量最多

We can find out that the data distribution is actually good for training, because we have to convert the 1~2 star reviews into “Negative” class and also convert 4~5 star reviews into “positive” class, by finishing the class-converting process we will end up with a quite balanced dataset which is shown in the following graph



Data Preprocessing:

1. Text Cleaning: (Optional)
   a. Expand Contraction:
   b. Remove Emoji
   c. Remove stopwords
   d. Remove extra white

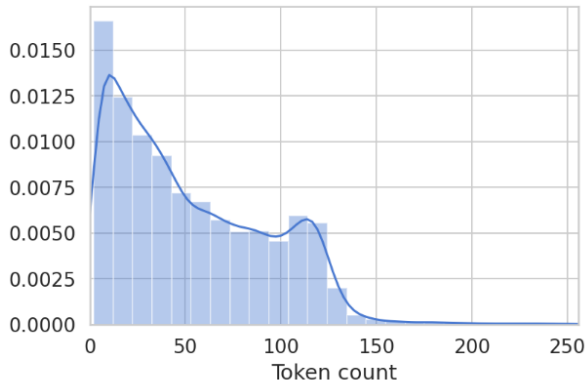
```
df["content"] = df["content"].apply(lambda x: remove_extra_white_spaces(x))
df["content"] = df["content"].apply(lambda x: expand_contractions(x))
df["content"] = df["content"].apply(lambda x: remove_emoji(x))
df["content"] = df["content"].apply(lambda x: remove_stopwords(x))
```

2. Tokenize the text

Create the corresponding tokenizers with respect to the Pretrained Models(I use bert-base-uncased, Roberta-base model in this final project)

```
tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
# tokenizer = RobertaTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

Here's a graph that shows the review word length after tokenization



Therefore, I set MAX_LENGTH = 150, after tokenization we can get the ["input_ids"] and ["attention_masks"], which will be the fed to the BERT model.

Sentiment Classification Model:

I tried Bert-base-cased and Roberta-base as training model, and add a dropout layer and a linear layer for classification

```
class SentimentClassifier(nn.Module):
    def __init__(self, n_classes):
        super(SentimentClassifier, self).__init__()
        self.bert = transformers.BertModel.from_pretrained('bert-base-cased')
#         self.bert = transformers.RobertaModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
        self.dropout = nn.Dropout(p=DROPOUT_P)
        self.linear = nn.Linear(self.bert.config.hidden_size, n_classes)
        # self.softmax = nn.Softmax(dim=1)

    def forward(self, input_ids, attention_mask):
        _, pooled_output = self.bert(input_ids, attention_mask, return_dict=False)
        output = self.dropout(pooled_output)
        output = self.linear(output)
        return output
```

Training Process and Evaluation:

1.Splitting the Data:

Train: 90% (24930) Validation:5% (1385) Test:5% (1385)

2.Training Hyperparameters:

Batch size: 16

Learning rate: 2e-5

Loss Function: CrossEntropyLoss()

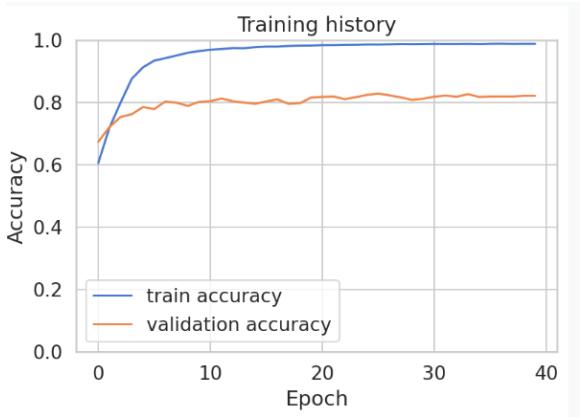
Optimizer: AdamW()

Scheduler: get_cosine_schedule_with_warmup

Experiment 1 (bert-base-cased, 40 Epochs, without text cleaning)

Training:

Best Validation Accuracy: 0.82816



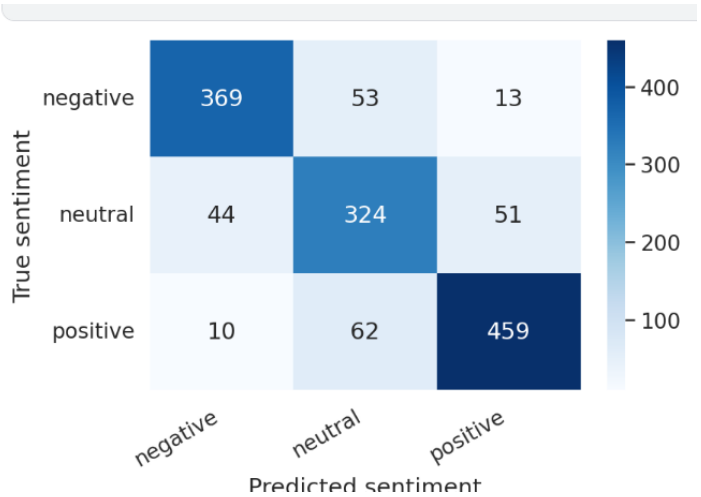
Evaluation:

Test set Accuracy: 0.83177

Precision, Recall, F1-score:

	precision	recall	f1-score	support
negative	0.87	0.85	0.86	435
neutral	0.74	0.77	0.76	419
positive	0.88	0.86	0.87	531
accuracy			0.83	1385
macro avg	0.83	0.83	0.83	1385
weighted avg	0.83	0.83	0.83	1385

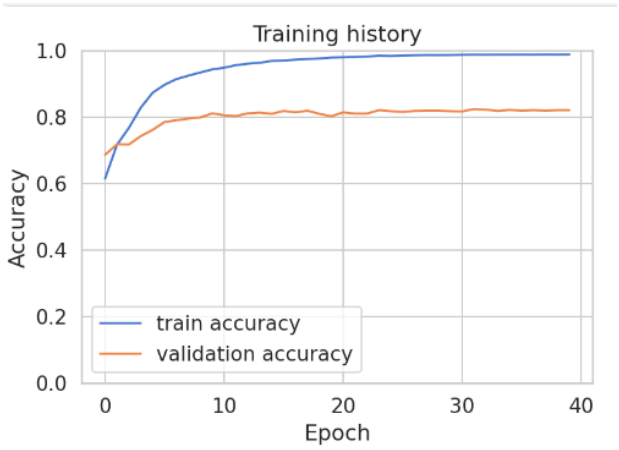
Confusion Matrix



Experiment 2 (Roberta-base, 40 epochs, without text cleaning)

Training:

Best Validation Accuracy: 0.82383



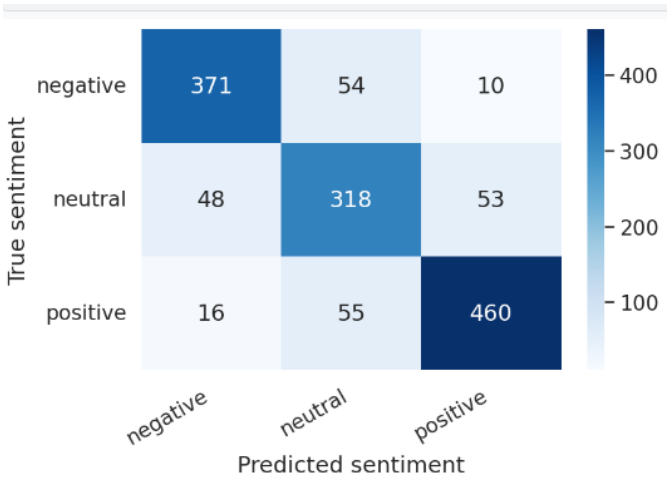
Evaluation:

Test set Accuracy: 0.82960

Precision, Recall, F1-score:

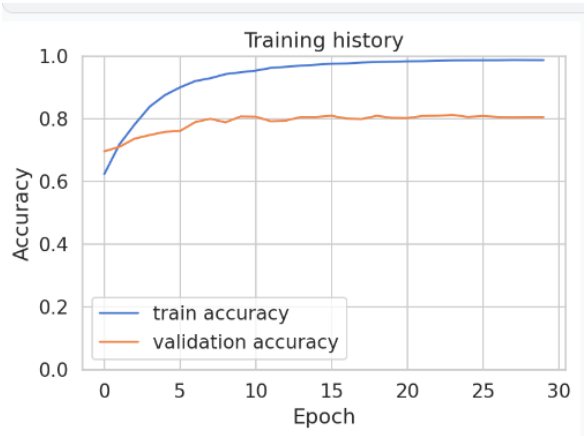
	precision	recall	f1-score	support
negative	0.85	0.85	0.85	435
neutral	0.74	0.76	0.75	419
positive	0.88	0.87	0.87	531
accuracy			0.83	1385
macro avg	0.83	0.83	0.83	1385
weighted avg	0.83	0.83	0.83	1385

Confusion Matrix



Experiment 3 (Roberta-base, 40 epochs, with text cleaning)

Best Validation Accuracy: 0.81155



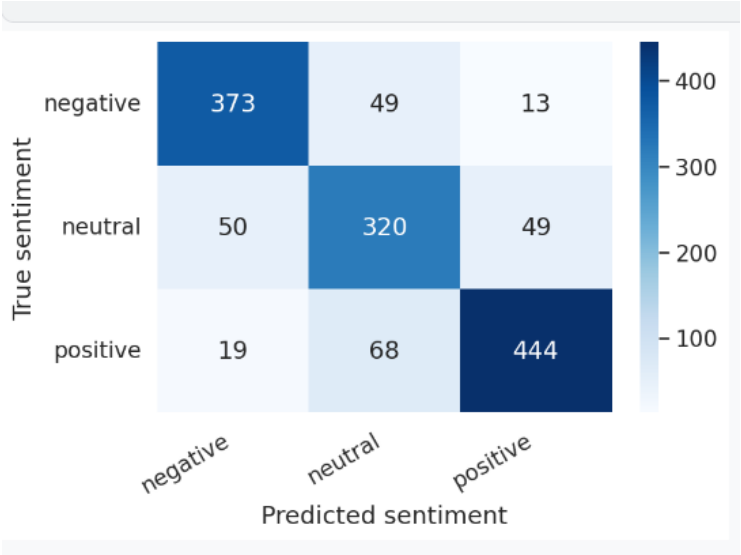
Evaluation:

Test set Accuracy: 0.82094

Precision, Recall, F1-score:

	precision	recall	f1-score	support
negative	0.84	0.86	0.85	435
neutral	0.73	0.76	0.75	419
positive	0.88	0.84	0.86	531
accuracy			0.82	1385
macro avg	0.82	0.82	0.82	1385
weighted avg	0.82	0.82	0.82	1385

Confusion Matrix



Discussion and Conclusion:

1. The best model is Experiment 1, using **bert-base-cased** and **no text cleaning**

2. Applying Text preprocessing technique didn't get any improvement, I guess I have to try other method or try to change some other parameters instead.

3. When I try using Roberta as the base model, I am anticipating that there would be a pretty good improve on the performance, however it didn't, it performed slightly worse comparing to BERT model

4. From the precision-recall-f1score chart and the confusion matrix, all three model did a great job classifying both "positive" and "negative" reviews, but can't classify "neutral" reviews correctly.
This result is expected because it is hard to define what kind of review is neutral, so the model may mis-classify some "neutral" reviews as "positive" or "negative" class, which in terms affect the total performance.

5.if I change this task into binary classification problem, that is , ignore all the neutral reviews, only use positive and negative reviews as training data, the final result will be a lot better.

6. At first, I only collect the reviews from 10 apps, and the performance is quite low, so I decide to collect more training data, and this method really help.

7. I really learned a lot from this individual final project!