Homework 3: Text Classification

AI二 B1228005 胡樂麒

1. Which (pre-trained) models do you use? Why to choose the models?

BERT-base-uncased

模型介紹:

- Google在2018年提出的一種自然語言處理(NLP)模型。
- 採用雙向編碼器(Bidirectional Encoder)學習上下文資訊, 能更準確理解句子語意。
- "uncased" 表示在訓練時忽略大小寫, 因此 "Happy" 與 "happy" 被視為相同。

選擇原因:

- BERT 具備強大的語意理解能力, 適合文本分類任務如情緒分析。
- 適用於需要高準確率與語義精度的任務。

DistilBERT-base-uncased

模型介紹:

- 由 Hugging Face 提出, 屬於 BERT 的「輕量版本」, 使用知識蒸餾(Knowledge Distillation)技術從 BERT 學習而來。
- 運算速度較快, 模型大小也更小。
- 同樣為 uncased 版本, 忽略單字大小寫。

選擇原因:

- 適合在計算資源有限、需要快速訓練的情況。

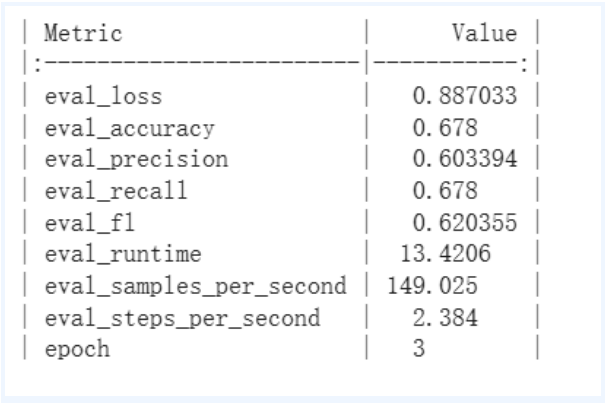
2. Please describe your model performance.

結果分析:

由下方結果可以看出, BERT-base-uncased 的整體表現明顯優於 DistilBERT。
不管是動態填充或靜態填充的方式, BERT-base-uncased的準確率都高於DistilBERT約0.2~0.3左右。此外, 透過 loss值, 可以看到 BERT 的 loss 更低, 表示在模型預測與實際標籤之間的誤差較小。不過, distilbert-base-uncased 雖然表現較差, 但這個模型有著更快的處理效率。從steps_per_second和samples_per_second可以看出DistilBERT 在速度上明顯較快, 平均只需約一半的時間便能完成一輪評估, 符合模型設計的初衷。

模型名稱	Padding類型	precision	accuracy	f1	runtime	loss
bert-base-uncased	dynamic	0.927278	0.9275	0.926786	6.0226	0.165024
bert-base-uncased	static	0.934218	0.9345	0.933954	13.9921	0.157639
distilbert-base	dynamic	0.609048	0.6785	0.628391	5.9388	0.899087
distilbert-base	static	0.603394	0.678	0.620355	13.4206	0.887033

bert-base-uncased-dynamic		Metric	Value | |:---|---:| | eval_loss | 0.165024 | | eval_accuracy | 0.9275 | | eval_precision | 0.927278 | | eval_recall | 0.9275 | | eval_f1 | 0.926786 | | eval_runtime | 6.0226 | | eval_samples_per_second | 332.082 | | eval_steps_per_second | 5.313 | | epoch | 3 |
distilbert-base-uncased-dynamic		Metric	Value | |:---|---:| | eval_loss | 0.899087 | | eval_accuracy | 0.6785 | | eval_precision | 0.609048 | | eval_recall | 0.6785 | | eval_f1 | 0.628391 | | eval_runtime | 5.9388 | | eval_samples_per_second | 336.766 | | eval_steps_per_second | 5.388 | | epoch | 3 |
bert-base-uncased-static		Metric	Value | |:---|---:| | eval_loss | 0.157639 | | eval_accuracy | 0.9345 | | eval_precision | 0.934218 | | eval_recall | 0.9345 | | eval_f1 | 0.933954 | | eval_runtime | 13.9921 | | eval_samples_per_second | 142.938 | | eval_steps_per_second | 2.287 | | epoch | 3 |

distilbert-base-uncased-static	

Metric	Value
eval_loss	0.887033
eval_accuracy	0.678
eval_precision	0.603394
eval_recall	0.678
eval_f1	0.620355
eval_runtime	13.4206
eval_samples_per_second	149.025
eval_steps_per_second	2.384
epoch	3

3. What problems do you encounter when doing this homework?

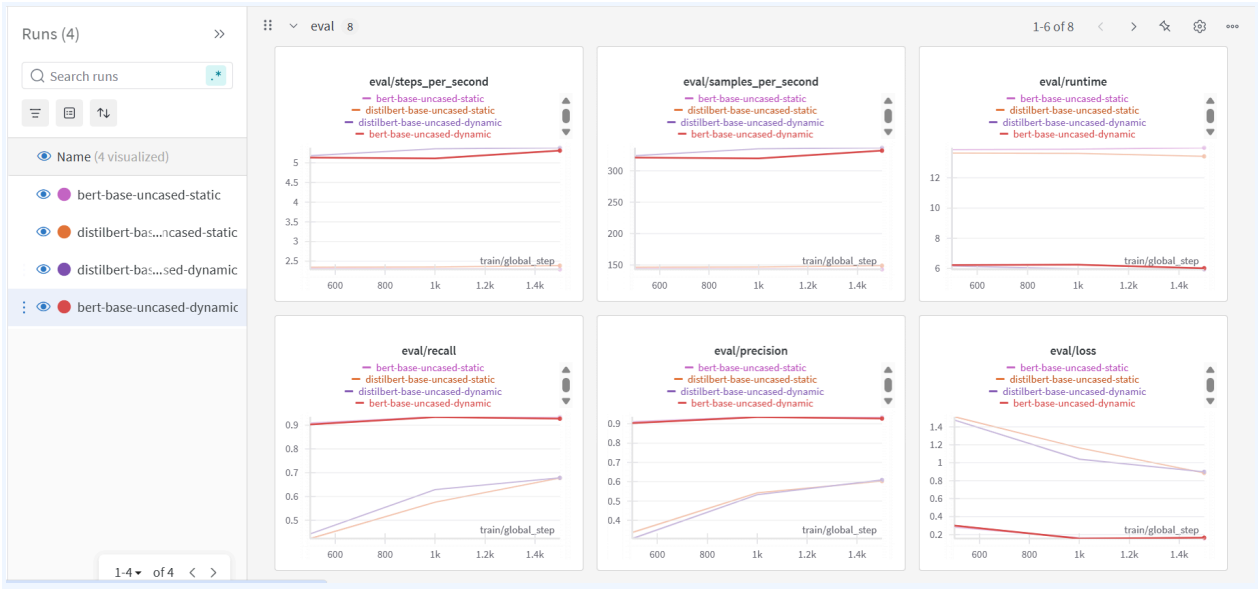
因為程式碼大部分都有出現在老師的影片中，所以我在過程中只有遇到兩個問題。
1.因為作業中有使用動態填充，但是我忘記在trainer中加入data_collator=data_collator，所以程式碼報錯，靜態填充不用加入這一行是因為他在資料預處理時就已經完成了，但是動態填充要根據每個不同的長度來進行填充所以要加入這一行。
2.在使用tensorboard畫出結果的時候，因為展示圖片頁面的專案名稱設定一直有問題，不管怎麼命名名稱都是".”，但是旁邊資料夾中卻有正確顯示檔名，所以最後我選擇使用wandb來呈現結果。

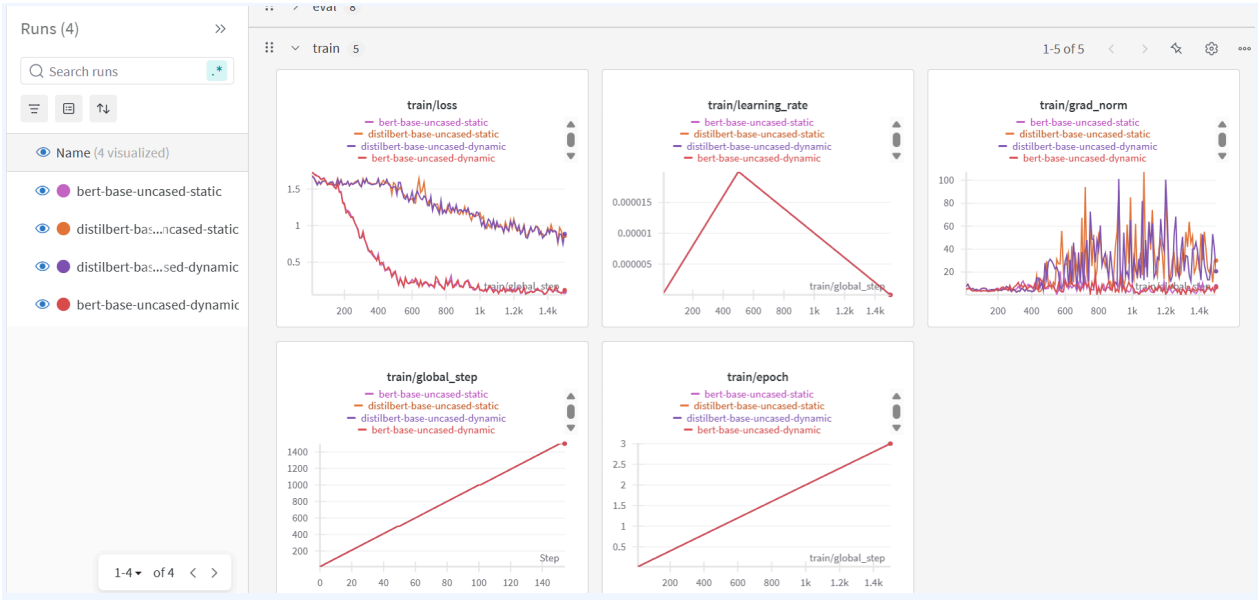
4. Please try static padding and compare the results with the one trained from the dynamic padding approach.

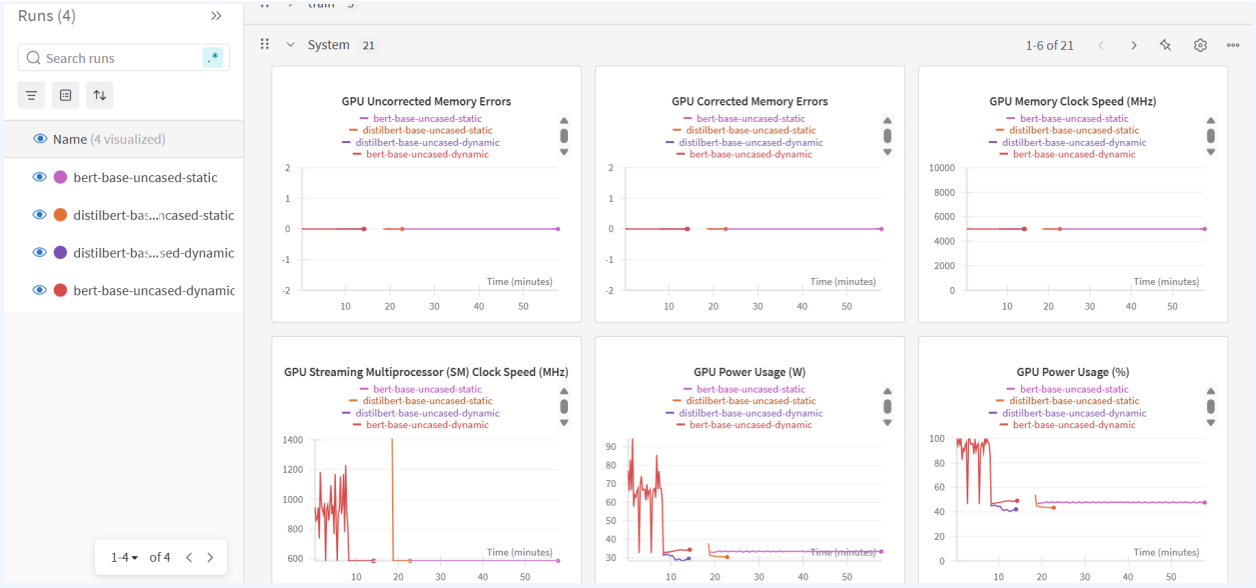
在本次情緒分類任務中，我分別使用了 靜態 padding（static padding） 與 動態 padding（dynamic padding） 兩種方式來處理輸入的序列長度，並比較其對模型效能的影響。靜態 padding 是在資料預處理階段就統一所有輸入序列的長度至一個固定值（例如最大長度），而動態 padding 則會根據當前 batch 中最長的句子動態調整填充長度。

結果顯示，對於 bert-base-uncased 模型而言，靜態 padding 的準確率（0.9345）略高於動態 padding（0.9275），在 precision 與 f1-score 指標上也有小幅度提升，代表統一長度可能對模型的穩定性有幫助。不過執行時間變長了許多，runtime 從 6 分鐘提升至 13.99 分鐘，花了約兩倍的時間。原因可能在於 static padding 導致每個 batch 中需要處理較多的 padding token，模型計算量隨之上升。準確率也只有微小的改變，distilbert-base-uncased亦然，所以我認為在這次的作業中比較適合使用動態填充。

5. Anything that can strengthen your report.(使用WANDB)






Runs (4)
Search runs
Name (4 visualized)
bert-base-uncased-static
distilbert-bas...ncased-static
distilbert-bas...sed-dynamic
bert-base-uncased-dynamic
1-4 of 4
System 21
1-6 of 21
GPU Uncorrected Memory Errors
GPU Corrected Memory Errors
GPU Memory Clock Speed (MHz)
GPU Streaming Multiprocessor (SM) Clock Speed (MHz)
GPU Power Usage (W)
GPU Power Usage (%)
bert-base-uncased-static
distilbert-base-uncased-static
distilbert-base-uncased-dynamic
bert-base-uncased-dynamic
Time (minutes)