NLP Final Project - team17

Method

Preprocess

- **Augmentation**
  augmentation的方法為基本的**字詞替換** (https://neptune.ai/blog/data-augmentation-nlp)，如'一個'→'兩個'，'很'→'非常'...等，總共使用12種替換方法，隨機性的選取包含上述字詞的句子隨機的進行替換，其中需要特別注意的是如果依照上述方法可以產生20000+~40000+新的句子，但由於在無法實際確定該augentation的效果多大，因此保守替換8000+句train dataset中的句子，以避免實際效用較小造成時間浪費、過多相似的資料反而導致augentation效果不佳等風險；id部分使用49800~80000 random pick，以避免與原始資料有相同id重複；accept則維持原句accept確保不會有錯誤評估的風險；同時過程中有考慮使用[mask]、Translate的方法進行augmentation，但由於有論文提到錯誤的使用[mask]、Translation可能會反而下降訓練成效，因此僅使用最為單純的字詞替換作為Augmentation。
- **Prepare Dataset**
  - 為了方便訓練，將原本的資料拆成許多(review, aspect, polarity) 的 pair，當成 dataset 中的一筆訓練

資料。亦即原本 csv 檔案中的一個 row 會產生 18 筆訓練資料。

- **Balance Dataloader**
  觀察我們所準備的 dataset 過後，會發現 polarity 欄位是 “not mention” 的比例極高，data 並不平衡，這有可能造成 model 偏好選擇 “not mention” 作為輸出。因此我們選擇做一些預處理，讓 model 看到的資料更平衡。以下是兩種我們嘗試過的方式。 (皆在 batch size 為 4 的倍數的情況下)
  - 在一個 batch 中，放同樣多個的 polarity 為 “not mentation”, “negative”, “neutral” 和 “positive” 的 review
  - 在一個 batch 中，放 **對於同一個 aspect** 同樣多個的 polarity 為 “not mentation”, “negative”, “neutral” 和 “positive” 的 review
    實驗發現，實作data balancing後，效果會比較好。

Model

- **Pretrained Model**
  - “hfl/chinese-macbert-base”
    因為GPU及記憶體資源限制，採用macbert-large會造成out of memory，所以改採用macbert-base。
- **Model Architecture**
  我們嘗試以二種不同的方式，來處理這個任務。
  - 第一種方式，參考Mohammed等人 [1] 的做法，input除了review之外，還加上要預測的aspect，二者皆以文字token形式，搭配[CLS]、[SEP]、[PAD]串在一起，形成[aspect, review]的pair；model採用BertForSequenceClassification；output有4個類別，除了3個polarity外，再加上1個，代表此aspect未出現在review中，形成[not mention, negative, neutral, positive]的結果。
    藉由這種設計方式，我們期待aspect做為額外input，能引導model注意review對應到特定aspect的區域，同時文字形式的aspect，能將不同aspect之間coarse-grained的階層關係納入考慮。

- 第二種方式，參考Jiahao等人 [2] 的做法，inpput只有review；model包含2個部分，第1部分採用BertModel，作為feature extractor，第2部分設計18個branch，分別對應18種aspect，每個branch包含attention pooling以及classifier，attention pooling用於引導model注意review對應到特定aspect的區域，classifier的output有4個類別，除了3個polarity外，再加上1個類別，代表此aspect未出現在review中，形成[not mention, negative, neutral, positive]的結果。review先通過feature extractor，得到的hidden states，再分別通過18個branch，預測出18個aspect的4種類別；計算predict或loss時，則分別取用對應aspect的output。
- 實驗發現，第二種方式會比第一種方式好 (如Figure 1、Figure2及Table 1所示)，推測model設計時，每個環節的學習目標若越明確，效果就會越好。如果要進一步改善，設計第三種方式，將not mention提出去，額外準備aspect detector，跟aspect polarity classifier分開，結果可能會更好，不過由於對此領域不夠熟練，程式實作時，花了較長時間摸索，因此來不及在deadline前，確認第三種方式的效果如何。

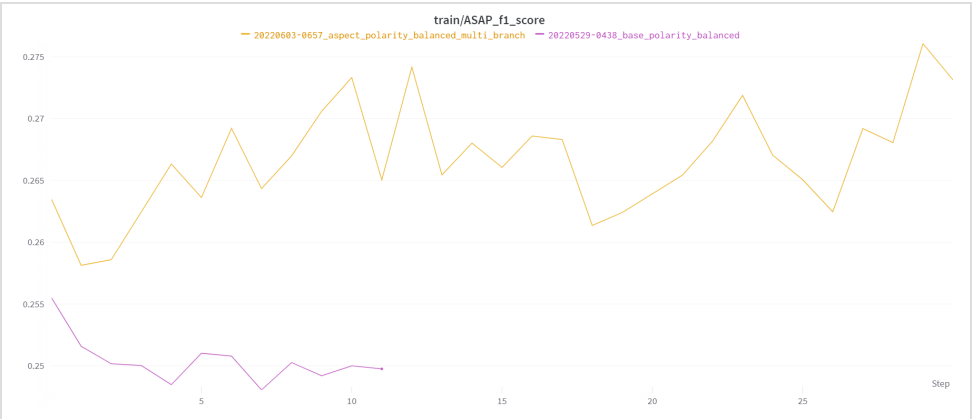

Figure 1 二種方式training過程F1-score趨勢比較

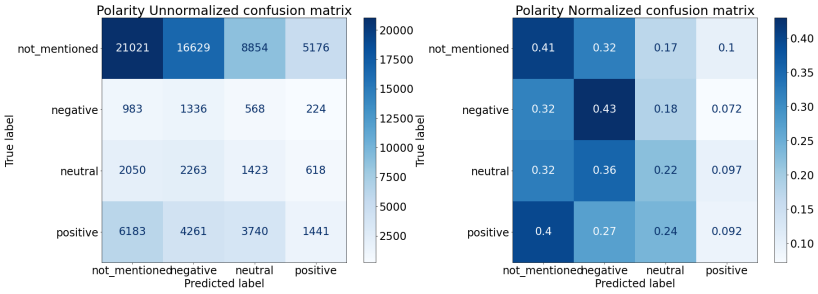

Figure 2 第二種方式validation的confusion matrix

Table 1 二種方式training結果比較

Training Result	第一種方式	第二種方式
Accuracy	0.249	**0.273**
Mean F1-score	0.249	**0.271**
F1-score not mentioned	0.252	**0.313**
F1-score not negative	0.249	**0.265**
F1-score not nertral	0.252	**0.269**
F1-score not positive	0.244	**0.238**

Training

- **Optimizer Scheduler**
  - `get_cosine_with_hard_restarts_schedule_with_warmup from transformer`
  - warmup 的 step 數量為 total step 數量的 $\frac{1}{10}$
- **Gradient Accumulation**
  - 經過 4 個 step 再更新一次 optimizer
  - 由於記憶體的限制，batch size 只有 4 ，因此採用 Gradient Accumulation 的方式，期望能達到大 batch size的效果

Ensemble

使用Ensemble average的方式進行Ensemble，方法為簡單的將複數model的變數取平均，來達到融合model的目的；有許多不同的ensemble方法可以選擇，如Voting、Bagging、Boosting、...etc.；但考慮到由於Ensemble的使用時機為model訓練完後，可能沒有時間調整參數、嘗試不同方法等原因，使用了最為單純、變因較少的Averaging方法作為本次project的Ensemble方法，其中averaging也因前述原因，以最為單純的simple averaging進行實作(將所有model的變數一視同仁，權重皆為1)，而非Weighting averaging(每個model有各自的權重占比)。

Experiment

(environment setting, hyperparameters, result, etc.)

- **Enviroment**
  - google colab with GPU Tesla PCIE P100
- **Hyperparameters**
  - batch_size: 4
  - learning_rate: $1e-5$
  - total epoch number: 20
  - epoch number to early stop: 20
  - maximum length of sequence: 512
  - doc_stride: 45
- **Result**
  - task1 : 0.42150
  - task2 : 0.20055

Future Works

由於此次成績不盡理想，所以這裡討論一些可能會提高訓練結果的想法

1. 分開訓練 task1 及 task2
   - 本次的想法是將 task1 與 task2 同時 training ，並且將 “not_mention” 視為一種 label
   - 若先訓練 task1 之後，再將其結果應用在 task2 訓練

上，也許可以縮小 “not_mention” 這個類別對 model 的影響

2. Two Stage Classifier
    - 第一個 stage 的 classifier 先 detect 該 aspect 是否有被提及，若有被提及的話，再由第二個 stage 的 classfier 去分辨是 “positive”、“negative” 還是 “neutral”
3. 單獨訓練 classifier
    - bert 部分有 pretrain 的基礎，如果想要在其上套用自己比較複雜的 classifier 的話，可能必須單獨訓練 classfier

Reference

[1] Arabic aspect based sentiment classification using BERT **https://aclanthology.org/W19-6120.pdf** (https://aclanthology.org/W19-6120.pdf)

[2] ASAP: A Chinese Review Dataset Towards Aspect Category Sentiment Analysis and Rating Prediction **https://arxiv.org/abs/2103.06605** (https://arxiv.org/abs/2103.06605)

Workload Distribution

學號姓名	
p09922002 黃寅	程式實作, report
r10525104 周宇玄	Augmentation & Ensemble, report
b07902067 郭宗穎	Balancing Dataset & Scheduler, Gradient Accumulation, report
b07902045 呂紹齊	眼睛受傷