Machine Learning Final Project

r08725024 羅良瑋 r08725021 王鼎元 b05705006 李和維

Introduction and Motivation

NLP 的世界在 ELMo, BERT 等強大的 Language Model 被訓練出來後，我們得以以此為根基經過再透過 fine tuning 來 train 出下游想要的預測模型。本次 Project 我們將利用 BERT 與 XLNet 等 LM 來完成 Dialog Modeling 的任務。

Data Preprocessing and Feature Engineering

首先我們先將資料集中的特殊符號 (包括標點符號、亂碼等等) 換成空白，並將所有字母轉成小寫，並且將每筆資料都加上 BERT 訓練所需的符號 (如 [CLS]、[SEP] 等等)。接著建立正確答案的字串與其對應的 id 之間的映射表，方便模型訓練時目標對話與解答代號的對照。

```
## 將特殊符號去除後，將剩下的英文字母轉為小寫
def replace_spe_token(data, col=''):
    special_token = "['-\.\!\/_,$%^*()+\"\<>?:-=]+|[+——！，。？?、~@#￥%……&* （） ]+"
    if col:
        data = re.sub(special_token, " ", data[col])
    else:
        data = re.sub(special_token, " ", data)
    return " ".join(data.split()).lower()
```

```
## 將資料帶入模型前，將其轉換成 BERT 所需的格式
# 將第一句 tokenize 後加入，並在最後補上分隔符號 [SEP]
word_pieces = ["[CLS]"]
tokens_a = self.tokenizer.tokenize(text_a)
word_pieces += token_a + ["SEP"]
len_a = len(word_pieces)

# 第二個句子的 BERT tokens
tokens_b = self.tokenizer.tokenize(text_b)
word_pieces += token_b + ["SEP"]
len_b = len(word_pieces) - len_a
```

在選擇使用的特徵時，由於考量到兩句對話有關連的部分大多分佈於第一句的後半段，以及第二句的前半段，故我們在兩個面向中刪去了部分過於龐大的資料量：

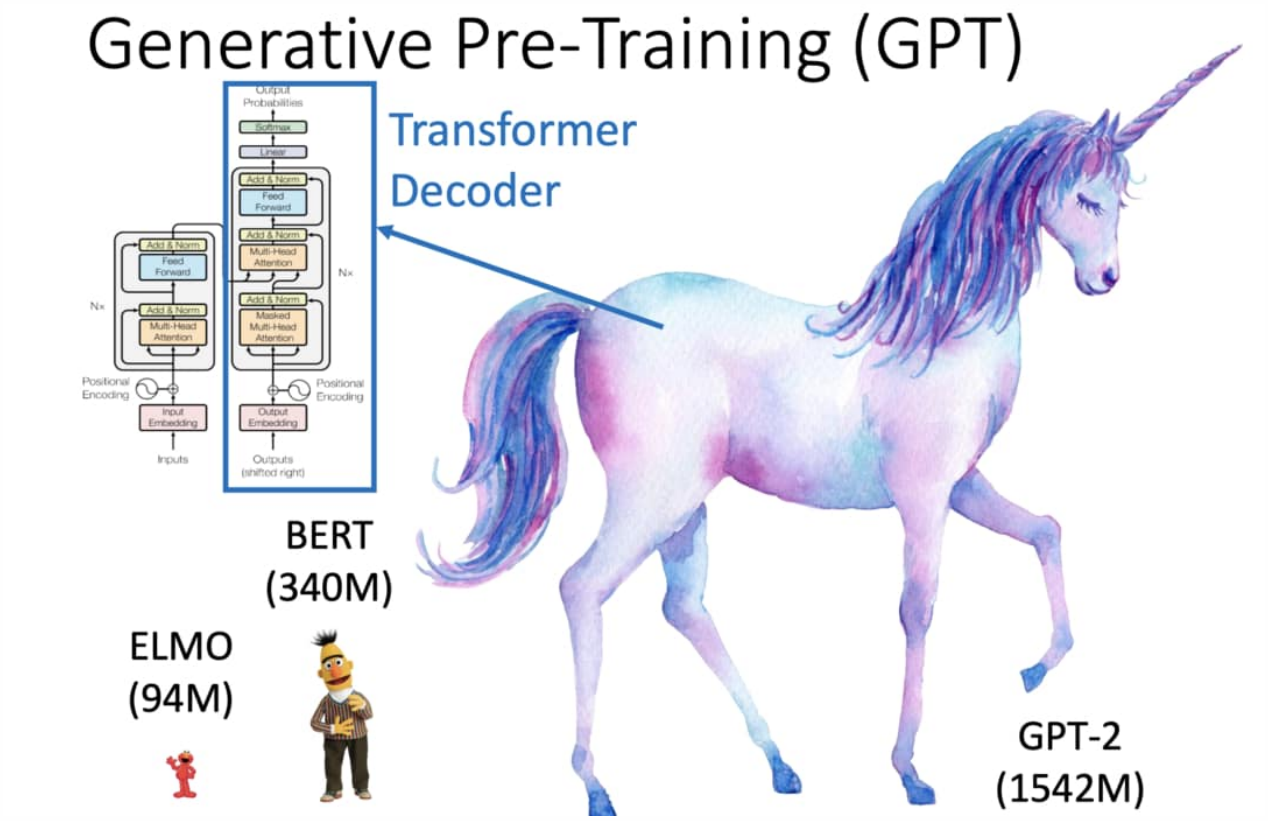
- 第一個面向是只取前文最後 n 個字來當作訓練資料，以減少模型的訓練量與訓練時間，最後以 n =300 時準確率最高。
- 第二個是面向只從 99 個錯誤答案中挑其中 m 個出來訓練以減少訓練量，最後以 m = 3 時準確率最高

Model Description

BERT Model

由於近年來在各種自然語言處理的競賽中，使用 pre-trained 模型架構訓練出的模型成績越來越好，所以我們一開始從最近推出的 pre-trained model 中挑選較為熱門的模型作為我們期末專案可能的架構。

Pre-trained 模型中，大致分為 feature-based 和 fine-tuning 類型，如：ELMo 則是 feature-based 的代表，透過將 RNN-based 各層的 hidden layer 做 weighted sum 來增加額外可能的 feature，而 fine-tuning 類型的方式則包含 GPT (generative pre-trained transformers) 和 BERT (Bidirectional Encoder Representations from Transformer) 等等，最後我們小組討論後，決定以近年頗熱門的 bert 來試試看，以下會分幾點討論。



上圖為三種不同 pre-trained model 的參數量，其中以 GPT-2 為最大，其次依序是 BERT 以及 ELMO。而 GPT-2 和 BERT 分別為 transformer 的 encoder 與 decoder。

圖片來源

選擇 BERT 的原因

我們討論後選擇是考量到 BERT 有以下幾點優點：

1. Self-attention 的機制

   一般的語言模型都只能考慮到單向的文字，忽略了一個字詞也有可能被後面的詞彙影響，即使如 ELMO 使用雙向 RNN-base model 來產生兩個方向的資訊，但其仍然是將兩個單向的資訊串接起來，並沒有同時考慮到前後文，而同為 fine-tuning 方法的 GPT-2 model，則是因為是 transformer 中 decoder 的部分，在處理當前的詞彙時，其 self attension 只能由左至右查看，而沒辦法看到右邊的詞彙，而 BERT 則能解決這類的問題。

2. Next Sentence Prediction (NSP)

   在 BERT pretrain 的過程中，google 有同時訓練兩個任務，其中一個任務為 Next Sentence Predcition ，即為讓 model 能夠判斷第二個輸入的句子是否跟第一個輸入的句子有關聯，而這樣的 pre-train 過程與我們期末專案的任務類似，因此我們認為透過 fine-tuning BERT 能夠獲得不錯的結果。

模型的架構

BERT 的全名為 Bidirectional Encoder Representations from Transformer ，是 google 於去年提出的 pre-trained 模型 ，模型中共有一層的 bert embedding layer 、 12 層 encoder layer 以及最後一層的 output layer，其中每層的 encoder layer 為 Transformer 中 encoder 的架構，詳細的 model 架構如下圖所示。

1. Embedding layer

```
(embeddings): BertEmbeddings(
  (word_embeddings): Embedding(30522, 768, padding_idx=0)
  (position_embeddings): Embedding(512, 768)
  (token_type_embeddings): Embedding(2, 768)
  (LayerNorm): BertLayerNorm()
  (dropout): Dropout(p=0.1)
)
```

2. Encoder layer (下圖為 1 層的架構，BERT 中共有 12 層)

```
(encoder): BertEncoder(
  (layer): ModuleList(
    (0): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): BertLayerNorm()
          (dropout): Dropout(p=0.1)
```

```
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): BertLayerNorm()
        (dropout): Dropout(p=0.1)
      )
    )
  )
)
```

3. Output layer

```
(pooler): BertPooler(
  (dense): Linear(in_features=768, out_features=768, bias=True)
  (activation): Tanh()
)
```

最後我們是使用 pytorch transformer 中的 BertForSequenceClassification 的模型進行實作，在上述的架構的最後接了一層輸出維度為 2 的 output layer。

```
model = BertForSequenceClassification.from_pretrained(pre_trained_model_name,
NUMS_LABELS)
```

XLNet Model

XLNet 是 Google Brain 與 CMU 聯手推出的 BERT 升級版，它採用Autoregressive model 來替代 BERT 的 Autoencoding model，解決了 BERT 在 Fine-tune 時因為沒有 [Mask] 而與預訓練不統一導致產生誤差的問題。而由於 AR 是單向的模型，作者額外提出了 Permutation Language Modeling ，把一個序列所有可能的排列都拿來作為 LM 的輸入，使得每一個位置都能利用到其他位置的訊息，藉此解決傳統 AR model 無法捕獲上下文的問題。

$$\max_{\theta} \ \mathbb{E}_{z\sim Z_T}\left[\sum_{t=1}^{T}\log p_\theta(x_{z_t}|X_{Z_{<t}})\right] \tag{1}$$

式 (1) 為 XLNet 的目標函數，其中 Z 代表長度 T 序列所有可能的排列。
除此之外，XLNet 使用了 Two-Stream Self-Attention 的機制來解決使用 Permutation Language Modeling 時可能對於不一樣的目標詞產出一模一樣 context 的問題，並且採用 Transformer-XL 中的 Segment Recurrence 機制，使在計算非常長的序列時可以透過 hidden layer 暫存而不用丟失訊息。

模型中共有一層 embedding layer 、 12 層 XLNet layer 、一層 sequence summary 以及最後一層的 output layer，詳細的 model 架構如下圖所示。

1. Embedding layer

```
(word_embedding): Embedding(32000, 768)
```

2. XLNet layer

```
(layer): ModuleList(
  (0): XLNetLayer(
    (rel_attn): XLNetRelativeAttention(
      (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (ff): XLNetFeedForward(
      (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (layer_1): Linear(in_features=768, out_features=3072, bias=True)
      (layer_2): Linear(in_features=3072, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
```

3. Sequence summary & output

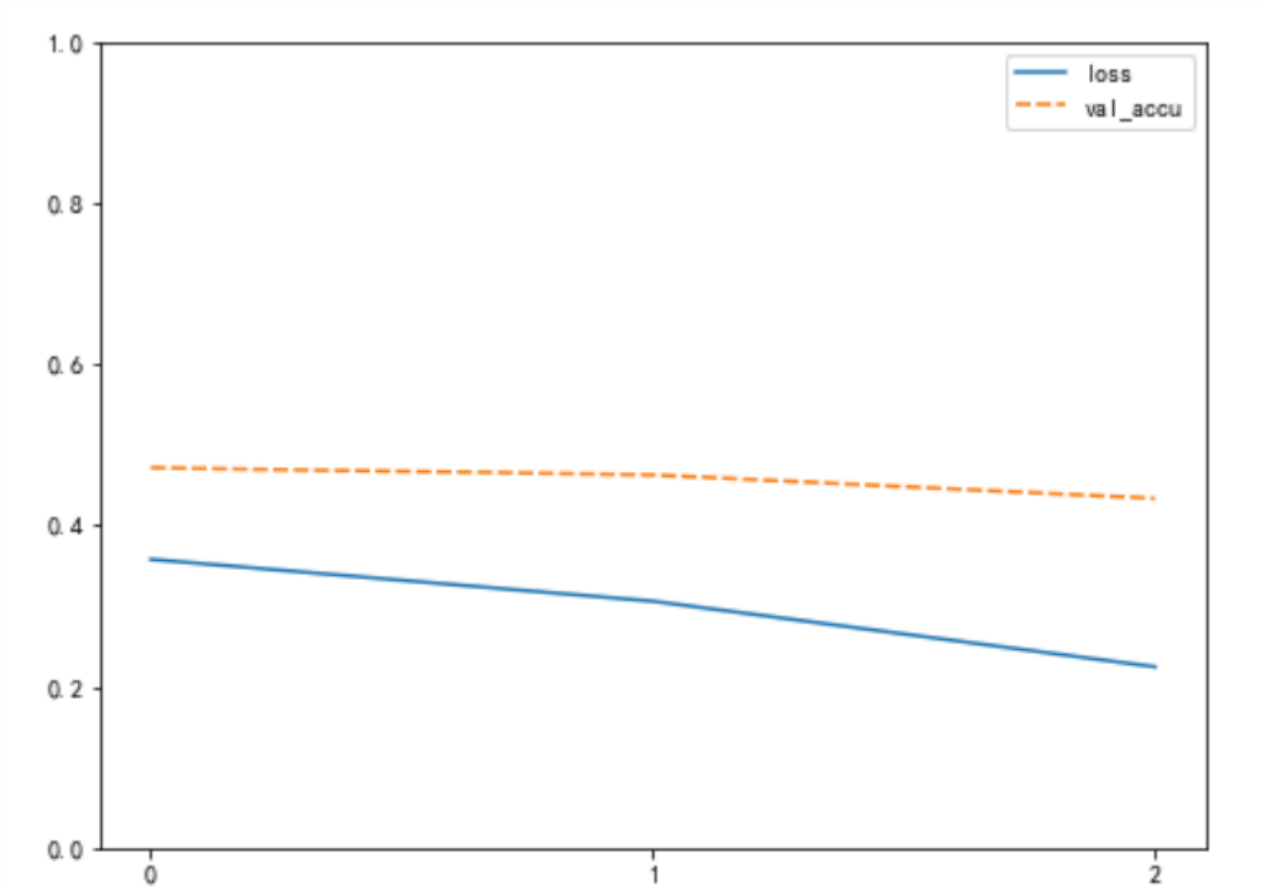
```
(sequence_summary): SequenceSummary(
  (summary): Linear(in_features=768, out_features=768, bias=True)
  (activation): Tanh()
  (first_dropout): Idnetity()
  (last_dropout): Dropout(p=0.1, inplace=False)
)
(logits_proj): Linear(in_features=768, out_features=2, bias=True)
```

Experiment and Discussion

Model	False number	First Sentence length	Learning rate	Public score / Private score
BERT	3	250	1e-5	0.48375 / 0.45833
	3	250	2e-5	0.48000 / 0.46333
	3	300	1e-5	0.49625 / 0.47833
	3	300	2e-5	0.48875 / 0.46750
	3	350	1e-5	0.48125 / 0.46666
	3	350	2e-5	0.48875 / 0.46833
	4	250	1e-5	0.47750 / 0.46166
	4	250	2e-5	0.48250 / 0.46416
	4	300	1e-5	0.48375 / 0.47166
	4	300	2e-5	0.47750 / 0.46666
	4	350	1e-5	0.49000 / 0.47333
	4	350	2e-5	0.47875 / 0.46666
XLNet	3	250	1e-5	0.43125 / 0.40333
	3	250	2e-5	0.40500 / 0.37000
	3	300	1e-5	0.44125 / 0.40500
	3	300	2e-5	0.41375 / 0.38250
	4	250	1e-5	0.44500 / 0.40583
	4	250	2e-5	0.41825 / 0.39833
	4	300	1e-5	0.43750 / 0.41666
	4	300	2e-5	0.40775 / 0.38875

在 fine-tuning 的過程中，我們發現在使用 BERT model 和 XLNet 進行 fine-tuning 時，通常在第一個或第二個 epoch 過後就會開始 overfitting ，因此我們設立 early stopping ，並取出 validation accuracy 最高的 model 作為訓練完的結果，最後在使用 validtion set 做最後的訓練。

接著我們調整我們認為可能影響到模型表現的參數，如： 第一個句子的長度、錯誤答案的訓練比例等等，來比較各個參數組合帶來的影響，最後以錯誤答案訓練數為 3 、首句長度為 300 以及學習率為 1e-5 的組合在 kaggle 的 public / private score 最高。



圖為錯誤答案訓練樹為 3 、首句長度為 300 以及學習率為 1e-5 的組合時
產生的 training loss 和 validation accuracy ， 橫軸為 epoch 數。

Conclusion

最終我們在 BertForSequenceClassification model 中以 (false_number, first_sentence_length, learning rate) = (3, 300, 1e-5) 這個 hyper parameter 配置中取得最高的 public 與 private score。

而 XLNet 雖然在許多 Task 中已經贏過 BERT，但在這次 Dialog Modeling 中我們嘗試了各種 hyper parameter 組合，在 private score 上都無法超過 BERT。推測這可能與 XLNet 在預訓練階段中去除了 Next Sentence Prediction 有關係。

Peer Review

Tasks	Contributers
Data preprocessing	羅良瑋
Feature engineering	羅良瑋、王鼎元、李和維
Research model	羅良瑋、王鼎元、李和維
Report	羅良瑋、王鼎元、李和維
Oral presentation	王鼎元、李和維

Reference

- *爱编程真是太好了*. 2019. *最通俗易懂的XLNET详解* (https://blog.csdn.net/u012526436/article/details/93196139)
- LeeMeng. 2019. *進擊的 BERT : NLP 界的巨人之力與遷移學習* (https://leemeng.tw/attack_on_bert_transfer_learning_in_nlp.html)
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (https://arxiv.org/pdf/1810.04805.pdf)
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le. 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding* (https://arxiv.org/pdf/1906.08237.pdf)
- WenWei Kang. 2019. *2019-NLP最強模型: XLNet* (https://medium.com/ai-academy-taiwan/2019-nlp%E6%9C%80%E5%BC%B7%E6%A8%A1%E5%9E%8B-xlnet-ac728b400de3)
- 李宏毅. 2019. *Transformer* (https://www.youtube.com/watch?v=ugWDIIOHtPA)
- 李宏毅. 2019. *ELMO, BERT, GPT*. (https://www.youtube.com/watch?v=UYPa347-DdE)