NLP Final Project Report

Group 1 | Github Link | Kaggle name: **Team 1**

1. Your model design and concept

1. Lesk Algorithm
    - 設計理念
      我們想要設計一個rule based的方式來非監督式的解決這個問題，就不需要訓練資料，並且也可以免去優化超參數的時間。

    - 模型說明
      首先，我們必須刪除所有stopwords和標點符號，並使Lesk演算法得到word sense。每次迴圈都會輪流將句子中的一個字換成OOV，並使用Lesk，如果對比原本的結果有word sense的改變，那這次迴圈中改變的字就是pun；然而，有些時候word sense並沒有改變，意味著他是common sense，則對於句子中每個word，查看這個word的其他sense(Si')與句子中其他word的sense(Sj)的sense similarity，有最高的similarity的word就是pun。

      $$s'_i = \underset{s_i^k \in w_i | s_i^k \neq s_i}{\operatorname{argmax}} \max_{j|j \neq i} Sim(s_i^k, s_j)$$

2. BERT
    - 設計理念
      BERT base的模型在許多語言任務中表現十分傑出，像是cloze task、question answering等等。前者可以透過前後文語意推敲，後者可以輕易地找出答案在文章中的對應位置，因此相信在這次作業的任務中，可以透過前後文語意學到pun的對應位置。

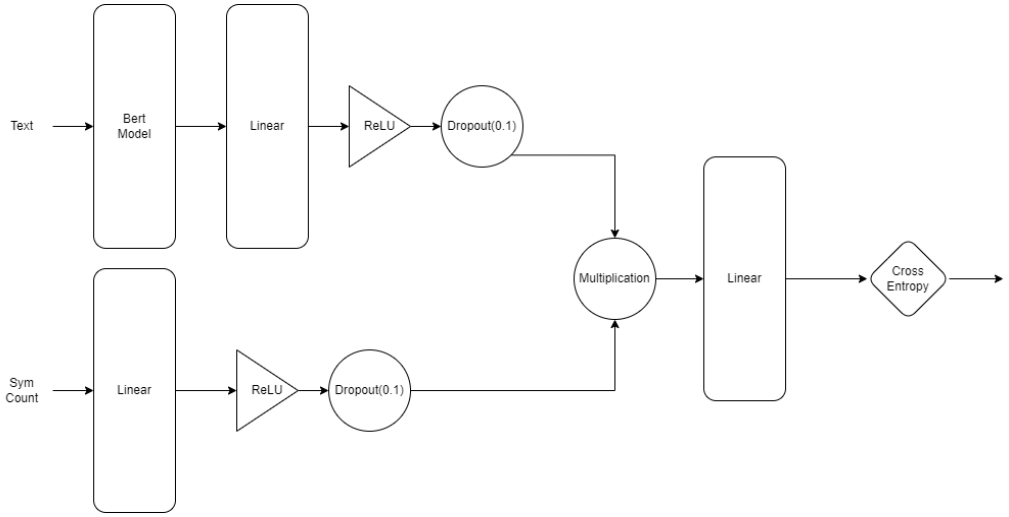
      模型說明
      將句子中的m個字做一次tokenize並給BERT做embedding，每個字變成n維的向量，並將m個n維向量放進分類器，輸出是m個二維向量(是pun與否的機率)，並對於這m個二維向量對其是pun的機率(m維)取argmax找到最可能為pun的位置。

3. RoBERTa
    - 設計理念
      同BERT的想法，不過研究資料指出某些任務中RoBERTa似乎比BERT還要優秀。在BERT中，他的random masking只在資料準備時執行一次，並在很多epoch中，複製此行為產生重複的masking，此為靜態masking；而在RoBERTa中使用動態masking，是在資料訓練時才做masking，此行為使得模型在需要更多回合或是資料量更大的時候會有更好的訓練結果。

    - 模型說明
      將句子中的m個字做一次tokenize並給RoBERTa做embedding，每個字變成n維的向量，並將m個n維向量放進分類器，輸出是m個二維向量(是pun與否的機率)，並對於這m個二維向量對其是pun的機率(m維)取argmax找到最可能為pun的位置。

4. DualAttention BERT(DAB 自己命名的)
    - 設計理念
      一般預訓練的BERT或是其他類似的模型, 在訓練時期只能取得前後文關係, 但是無法知道每個字的 定義或是相近辭有哪些。而我們認為每個字的相近詞數量, 與他是PUN的可能性正向相關, 所以我們自行設計模型, 使用相近詞數量作為一個vector, 輸入神經網路的線性層並與BERT的輸出進行融合。

    - 模型說明
      我們使用一個線性層作為近義詞數量的編碼(encode)函數, 將近義詞數量的vector投影到更高維度, 並且選擇以ReLU作為激勵函數, 添加dropout以避免過擬合。在loss function選擇上面我們使用分類問題常用的cross_entropy, 輸出每個字為PUN的可能性。屆時我們將分別展示出使用元素加法以及元素乘法、使用ReLU與使用tanh所帶來的效益, 並說明此設計帶來的public score的提升。



2. What kind of word sense representation used and experimented in your model

- Lesk Algorithm
  使用nltk內建的synsets去得到近義詞的意思，並且使用wup_similarity去衡量兩個語句之間的相似度。

- BERT、RoBERTa、DualAttention BERT
  將句子中的m個字做一次tokenize接著給BERT做encoding, 並取其hidden layer最後一層去除頭尾(CLS、SEP)作為embedding, 每個token會對應輸出一個n維的向量, 這代表一個字的sense。

3. What problem did you face during the homework and how you solved

- **Tokenizer**添加特殊識別字導致計算答案的**index**偏移
  BERT和RoBERTa的tokenize方法會使得句子出現特殊字或是出現sub words, 假設原本句子有m個字, 如果原本答案是第p個字, 經過tokenize之後可能變成n+k個字, 這時候答案的索引會產生偏移，因此我在tokenize前後利用他tokenize產生的特殊字(##)去辨識sub words並產生label偏移的索引與原本的對照，這讓我在最後產生test的時候部會選到錯誤的索引。

- **Tanh vs ReLU**
  在Activation Functions: Comparison of Trends in Practice and Research for Deep Learning中說到，激勵函數因模型不同表現也不一樣，一般而言，ReLU在大部分的模型表現較為突出, 但是在模型非常長的時候, tanh有可能表現的較好。由於我們的模型有經過BERT, 是個較深的模型, 所以我試著用tanh來進行訓練, 然而validation score和public score都出現明顯下降。最終我們仍然使用ReLU作為主要的激勵函數。

- 元素加法與乘法
  在強化學習的論文Implicit Quantile Networks for Distributional Reinforcement Learning中提到, 想要表示兩個不同長度的feature vector具有相關性, 並通過線性層擬和, 可以利用元素乘法與加法去生成並輸入下一層神經元。我們的實驗結果無論是在validation score和public score有使用加法/乘法的成績都比沒有使用來的高, 而其中加法的效果較為顯著(0.91250)。

- 版本控制和**reproduction**
  由於這是多人的小組作業, 有時候交出去得到最高成績也很容易忘記所有的參數設定, 其他組員也要花大量時間再去重測或重現這個成績, 所以我將所有的參數控制與原本的程式碼分開, 寫在yaml檔中, 使用GitOps將yaml檔列入版本控制, 並且固定所有random seed以便重現這個成績。

4. Error Analysis and Discussion

- 80% train 代表使用訓練資料的80%訓練, 20%驗證
- multiplication, addition 分別表示 DAB的元素乘法與加法
- clip=4, 代表將同義詞數量小於等於4的視為0個。
- 如果沒有標示activation function(AF), 則為ReLU

使用模型	說明	public score
Lesk	Rule based算法	0.21375
BERT	沒有發現Tokenizer加特殊字導致計算index時發生偏移	0.19375
BERT	修正index錯誤。epoch=3 , 80%train	0.86875
BERT	epoch=7 (過擬合), 80%train	0.86250
BERT	epoch=5, 80%train	0.88125
BERT	epoch=5, 100% train, no validation	0.88625
RoBERTa	epoch=7, 80% train	0.75625
DAB	epoch=8, clip=0, 80% train, multiplication	0.87500
DAB	epoch=8, clip=0, 80% train, addition	0.88750
DAB	epoch=8, clip=4, 80% train, multiplication	0.90625
DAB	epoch=8, clip=4, 80% train, addition, AF=tanh	0.89375
DAB	epoch=8, clip=4, 85% train, addition	0.91250

- Discussion
  - Why clipping ?
    以這句作為範例:
    I used to be a banker, but I lost my interest.
    由於我們使用WordNet的synset計算同義詞數量, 而我們的假設是同義詞數量和雙關語的可能性是正相關, 這時候 I 在 WordNet的同義詞數量有4個, 但如果把這四個計入則會讓模型的準確度下降, 所以才使用clip, 將同義詞數量過少的字剔除, 降低模型訓練時的噪音。

  - Why BERT is better than RoBERTa in this task?
    就如同前面所提到的RoBERTa使用動態的masking, 這使得他在資料量和回合數較多時, 可以學到更多資訊但也需要更多時間訓練, 不過這本次的任務中的資料量沒有很多, 因此可能在此方面沒有進步, 反而讓他更難收斂。

5. Compare and implement unsupervised method and supervised method

我們使用Lesk Algorithm作為unsupervised method，並且以DualAttention BERT作為supervised method。

- Lesk Algorithm
  我們一開始猜想這個方法應該會比使用神經網路的方法來的快很多，因為參數明顯較少，記憶體用量也很低，然而花費的時間卻和使用神經網路的時間不相上下。因為無法使用GPU平行處理還有其中也有三層以上的for loop，並且也使用了synsets和wup_similarity導致效能沒有預期中的高。另外，我們也發現wup_similarity在同一個字的相似度分數會不一樣，這是因為NLTK會去遍歷WordNet hierachies去取得lowest_common_hypernym()所導致的，所以其實此種方法並不是很可靠，最終的F1-score只有0.21。

- DualAttention BERT
  根據前面所述，BERT在語言任務上表現亮眼，這是因為他採用了大量的預訓練資料，能夠在訓練上展現較好的訓練成果，而我們又將similarity count加入了此種模型，使其獲得更多side info，public score 和 validation score 比普通的BERT還要好，不過可以在第四題的圖表中發現，訓練資料量會影響到模型結果，而且不一定是越多越好，有可能的原因是因為記憶體不夠，batch size都設的比較小(16)，可能導致梯度估計的不準確。