Natural Language Inference for Fake News Detection

Member and Reputation

- B05902046 林昱賢 (停修)
- B05902091 黎峻碩 (停修)
- B05902109 柯上優: data preprocessing, coding, experiment, reporting

Summary

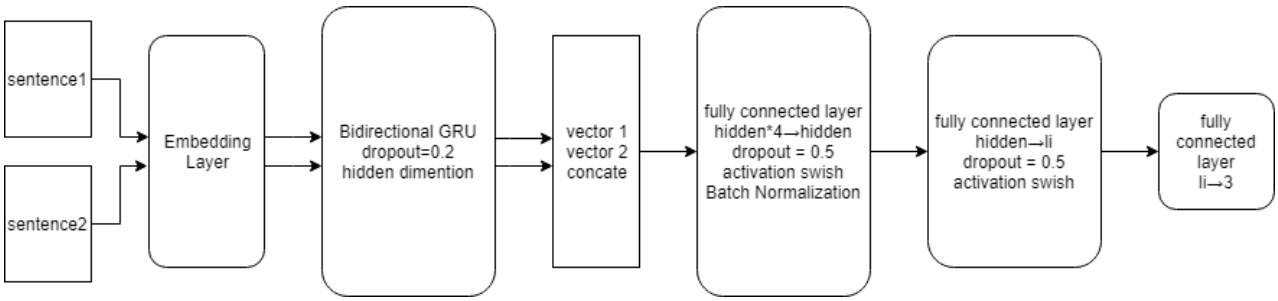
- 使用自己撰寫的 RNN 模型，並未使用 Bert，因為以下問題
  - 我第一次寫 pytorch，希望體驗更真實更重要，架設類神經網路模型的過程。
  - 若使用Bert，則重點將只剩下「資料處理」、「上網找更多資料庫來訓練」、「ensemble 幾個差不多的模型」，較為無趣。
  - 人力不足，原本預計將此部分給其他人先處理，完成 RNN 後再研究 Bert，結果組員只剩我。
- 自己的 RNN 模型
  - 在所有實驗，public score 約在 0.66478~0.74260，private score 落在 0.61147~0.71305
  - 最終 ensemble answer 的 public score = 0.74769，private score = 0.72764。

Data Preprocess

- 並未使用更多的資料，僅使用 kaggle 比賽提供的
- 為了方便訓練與增加精準度，在英文與中文的資料有不同的處理方法
  - 中文資料將半形標點符號全數改為全形，包括驚嘆號、問號、逗號。
  - 英文資料為了配合 google pretrain word2vec 的轉換，將沒有對應向量的組合字（如 old-age）拆開（old age），各自對應一個向量。
  - 共通的處理方法是將單引號與雙引號前後加上空白，以避免發生同樣的詞，被不同方式斷詞。
- 斷字的方面
  - 中文使用 jieba 的斷字函數
  - 英文使用空白當斷字的切點
- Validation
  - 使用訓練資料的後 $\frac{1}{10}$ 作為 validation dataset

Model

- 我使用的模型主要架構是如以下



- 其中 Embedding 有兩種方法
  - 使用 gensim 的 word2vec，下面稱之為 Word2Vec
    - 中文的訓練使用 train.csv + test.csv 內的所有句子
    - 英文的訓練使用 google word2vec english pretrain corpus，去官方網站下載的
    - 送入時，一個斷詞對應的該向量，$input = [batch \times paddingLen \times embeddingDimension]$
  - 使用 torch.nn.Embedding，下面稱之為 WordDict
    - 做 end-to-end 的訓練，讓模型在訓練過程自己找到向量
    - 先建立一個字典，key 為一個斷詞，value 為某個 integer
    - 送入時，一個斷詞對應的 value，$input = [batch \times paddingLen \times integer]$
  - Swish()
    - 是一種我在論文讀到的 activation function，由於 pytorch 並沒有函數可以呼叫，所以我自己額外寫的，公式為 $f(X) = X \times sigmoid(X)$。
    - 研究指出，swish 比起 ReLu，有更好的表現，由於 ReLu 在梯度遞減等於 0 時，會停止更新，相關研究指出神經網路內有高達 40% 的神經元都是 0，這樣可能會太快失去模型的能性。而 swish 會在較深的神經網路中發揮較好的功能，此外也有研究 (Hochreiter & Schmidhuber, 1997) 指出 LSTM 適合使用 swish。
  - 模型詳細的參數會在下面的實驗結果中解釋，不同的參數有不同的結果

Training

- 使用加權過後的 CrossEntropyLoss，加權為 $[\frac{1}{15}, \frac{1}{5}, \frac{1}{16}]$
- batch size = 32, epoch = 30
- Adam, lr = 1e-4

Experiment

Embedding	Zh/En	Embedding dimension	hidden state	public score	private score
Word2Vec	Zh	100	32	0.67134	0.62815
Word2Vec	Zh	100	64	0.66478	0.61147
Word2Vec	En	100	32	0.67343	0.61575
Word2Vec	En	100	64	0.67982	0.63516
WordDict	Zh	100	32	0.72892	**0.71305**
WordDict	Zh	100	64	0.73596	0.69380
WordDict	En	100	32	0.69555	0.64349
WordDict	En	100	64	0.69592	0.65801
WordDict	Zh	200	32	0.73396	0.70357
WordDict	Zh	200	64	**0.74260**	0.70774
ensemble				**0.74769**	**0.72764**

- ensemble 是使用以下幾種做投票
  - WordDict/Zh/100/32
  - WordDict/Zh/100/64
  - WordDict/Zh/200/32
  - WordDict/Zh/200/64

Observation

- 實驗中得到很多差異明顯的結果，和值得探討的差異，以下分幾點說明
- End-to-end 比起 Word2Vec 還要好
  - 表格中可以發現上面四個的實驗結果，不論是 public score 還是 private score 都低於下面六個
  - 我認為可能的原因
    - 新聞標題的用字和可能和 google pretrain 的用字有差異。
    - training set 的樣本數量實際上很少，只是使用任意兩兩組合，純用有限的 embedding 功能反而類似 bag-of-words，直接用相同的字做比較。
  - 可能的研究方向
    - 增加更多 training data
- 中文比英文還有效果
  - 可能的原因
    - 這或許和資料來源有關，資料很明顯是來自中國的報導，英文標題是使用翻譯間接得到的。
  - 可能的研究方向
    - 或許使用更多原始是英文的新聞標題，才能讓英文訓練變好。
    - 但是若只是想讓這份 testing data 有好的成果，其實可以完全無視掉英文標題，因為只是二手資料，並沒有訓練的幫助。

- Embedding dimension 和 hidden state 其實沒有顯著的影響
    - 不同的參數，誤差並沒有前面兩種差別那樣的差距
    - 可能的原因
        - 因為資料的有限，用 100 維度的向量就足夠了，200 維沒有提供更多資訊。
    - 可能的研究方向
        - 要從更大的地方開始改變，例如疊的層數，或是架構層面。
- ensemble
    - 毫不懷疑的，投票讓我們有更好的準確度

Agreed	Disagreed	Unrelated	Counts
0	0	4	48788
1	0	3	10302
4	0	0	8153
2	0	2	6771
3	0	1	5941
0	1	3	156
1	1	2	7
2	1	1	5
3	1	0	3

- 此外以上是我對於 ensemble 投票時的狀況做統計，觀察四票的分布
    - 發現完全沒有 Disagreed 的選項，也就是模型完全沒有學習到這個 class。我認為是因為 Agreed、Disagreed 對於我的模型來說太過困難，語意的層面終究沒有學習到。取而代之的是，我的模型變成了純粹的重疊字判斷器，只要是相同的字很多，就會直接判斷 Agreed。
    - 之所以都判斷 Agreed 而不是 Disagreed，我認為是因為訓練資料裡 Agreed 資料是比較多的，而為了降低 Loss，直接將它們都判斷成 Agreed 是最佳的選擇，而有了如此的結果。
    - 可以得知這個模型沒辦法處理字義的部分，我們需要更強大的模型。

Conclusion

- 和高達 0.8 的 Bert 相比，傳統的 RNN 是有極限的，我們需要一個能真正學習到字義的模型，而很明顯的 bidirectional RNN 無法。
- 資料的不足夠也是一個很大的原因，我們需要更多的資料來學習兩句話間的同意與否定，太多的無關資料湊在一起給我們的幫助有限。資料量更多元也能讓 pretrain word2vec 更有用。
- 純粹將一種語言翻譯成另一種語言，並不是增加資料的方法，也無法抽出更有用的資料。