Natural Language Processing

Project 2

Discourse Relation

Team information

Team name : AirNLP

R05521608 任精瑋 r05521608@ntu.edu.tw

R04943174 王嚴徵 r04943174@ntu.edu.tw

PS.由於 Kaggle 系統在最後一天好像沒辦法併組!所以在 report 中跟助教說明組別(在 Kaggle 上是 QaQ 跟 Scott)。

Methodology, Experiment and Discussion

●I. TF-IDF + Random Forest:

●II. Deep Learning

●***Method. I:*** TF-IDF + Random Forest

首先將測試集資料(training data)切成前後兩句。之後利用結巴(jieba)套件分別對兩句話進行斷詞，再利用 doc2vec 以及 TF-IDF 兩種模型針對中文句子進行特徵向量轉換。

Doc2vec 使用 python NLTK 套件內建的中文文本進行訓練，並用訓練好的模型將前後兩個句子，各自轉成 100 維的向量，再合併(concatenate)起來成為 200 維的向量，以方便使用機器學習方法進行訓練。而 TD-IDF 的模型，設定成過濾掉出現次數(min count)小於 5 次，以及出現次數在前 5%的詞，轉換出來的向量維度約在 900 左右。

將轉為特徵向量的句子使用 Scikit learn 進行訓練，在訓練的過程中發現無論使用何種的訓練模型以及調整參數(Random Forest, Gradient boost decision tree, KNN, SVM…等)，Doc2vec 模型轉出來的向量雖然 Ein 最好可以做到 20%左右(因為測試集資料相當 unbalanced，Expansion 的數量佔了一半以上，所以很容易在測試集資料上 overfitting，Ein 也很容易做的很低)，但在 Validation error (40~50%)以及測試集資料 (Public score < 40%)的表現都很差，估計可能是因為測試集資料不夠大，所以不容易透過 doc2vec 的方式抽取出正確的向量特徵來進行訓練。

另外，也將 TF-IDF 轉出來的句子向量使用隨機森林(Random Forest)的演算法進行訓練，個別決策樹的最大深度(max depth)範圍在 3~7 之間，決策樹數目(n_estimators)分別為 100、150、200、250，訓練的過程發現 class weight 的改變對於模型的預測結果有很大的影響，也不太容易調整，些微的變動也會使預測結果偏向某一類別，因此我們還是依據原始訓練集數據的各類別比例(677/1292/741/3928)進行些微調整(**Class Weight :** ***Temporal***-5.73, ***Contingency***-3.04, ***Comparison***-5.3, ***Expansion***-0.98)，下一頁的表一為部份訓練參數與結果。

表一、隨機森林模型的參數調控與表現情形

Class Weight : ***Temporal***-5.73, ***Contingency***-3.04, ***Comparison***-5.3, ***Expansion***-0.98

Max Depth	N_estimator	Ein(%)	Out of bag score(%)
4	**200**	**65.3**	**38.3**
5	**100**	**37.3**	**50.5**
	200	**37.7**	**54.8**
6	**100**	**36.4**	**56.1**
	200	**36.8**	**56.2**
7	**100**	**36.8**	**58.6**
	200	**35.3**	**56.4**

藉由記錄各個模型對於 1000 筆測試資料的預測分布，發現表現最好的模型是最大深度為 7 的模型，但事實上，最大深度超過 5 的決策樹在最後的 1000 筆測試資料集(Testing data)會將超過一半的數據預測為 Expansion，且不容易透過 class weight 的調整來進行改善，因此我們主要選擇最大深度為 4 與 5 的決策樹模型，使用 1000 棵樹進行測試集資料預測，以下為預測結果。

Max Depth	N_estimator	Ein (%)	Out of bag score(%)	Public score (%)
4	1000	40.2	47.2	45.8
5	1000	34.9	46.1	49.4 (private:51)

最大深度大於 5 的模型，在最後的 public score 都略低於 45%，主要的原因可能還是因為訓練集資料分布相當不平均，Out of bag 資料的分布因為同樣來自測試資料集，分布應該與測試資料集很接近，所以直接拿 Out of bag score 來看還是會有 overfitting 的疑慮，所以選擇最大深度較小的決策樹來進行 Random Forest 演算法還是比較好的選擇。另外有試過使用 Gradient Boost Decision Tree 但因為測試資料集非常不均勻，所以訓練出來的模型都會 overfitting 在 Expansion 的數據集上，權重的調整也不容易；也有考慮過利用 boot strap 的方式來集成較均勻的數據(透過重複抽取來限制各 class 的 sample 數都落在 1/4 左右)，但訓練結果仍然沒有高於 50%，因此另一個可能的原因是，將文本進行向量轉換的過程所取得的 feature 還不夠強，造成 performance 的提升有瓶頸，可能需要再調整更多的參數或是增加測試集資料才能達到比較好的效果。

•***Method.II:*** Deep Learning

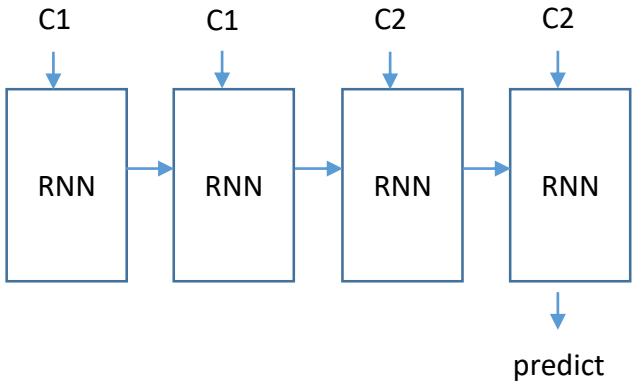
1. 使用工具：

   tensorflow

   pandas

   jieba

2. 模型建構：

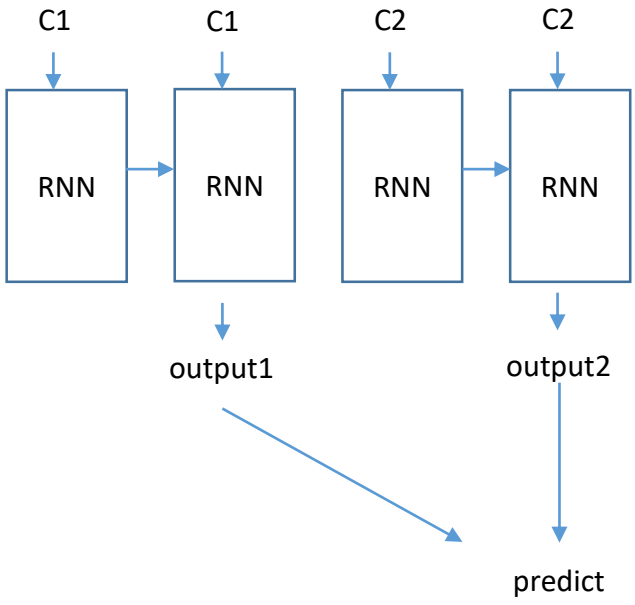


(1) single RNN

單一條 RNN 直接跑過 Clause1, Clause2 產出 predict

(2) dual RNN

用兩條 RNN 來分別跑 Clause1、Clause2，在將兩條 RNN 的 Output concatenate 在一起跑 fully connected network 最後產出 predict。



3. 結果：

(1) ***single RNN***

一開始使用單一條的 RNN 結果 public score 是 49%，效果不是很好，推測是因為有些句子含有不只一種 Relation，可能在 Clause1 內部就有一種 Relation，而 Clause1 對 Clause2 時又有另一種 Relation 所以會有衝突。

(2) ***dual RNN***

跑出來的結果是 57%，比單一條 RNN 好一點，但還是不如預期。

(3) ***overfitting***

在跑 single RNN 與 dual RNN 都會遇到 overfitting 的問題，發現有一部分是因為沒有控制字典大小的問題，model 可能會依據只出現一兩次的字來判斷 Relation，有使用 tf-idf 來調整字典，避免出現很少次數的字。調整過後 dual RNN 的結果大概從 53%上升到 57%但 overfitting 的問題還是很嚴重。