

ML FINAL --- Chinese TV Dialogue Multiple-choice Problem

隊名: NTU_r06522828_?????

機械碩一 r06522828 王榆昇、機械碩一 r06522825 董士豪

電機碩一 r06945032 徐瑋廷、物理碩一 r06222023 王宇昕

1. Introduction & motivation

本次的 final project 為 Chinese TV Dialogue Multiple-choice Problem，任務是輸入一段話，接下來有六個句子作為選項，挑出哪個選項最適合接為下一句。而 training data 是五個劇本，並沒有給定 label。

一開始認為需要的是 sequence to sequence 模型，但經過討論過後了解因為有了選項，有明確的句子，所以可以去計算 similarity，就能選出最佳的答案，但由於對話有時序性，因此會需要 RNN 的模型。另一方面，由於 training data 是連續對話的劇本，也沒有標出 label，對於如何進行 supervised learning 並沒有直覺的想法，我們認為處理文字是最具有挑戰性的任務，且對於 RNN 時序模型很有興趣，希望能從中學習到更多處理文字相關的知識，以及能自主學習以往較少接觸的任務，因此選擇這道題目作為我們的專題。

2. Data Preprocessing

我們使用 jieba 分詞，以較適合繁體中文的 dict.txt.big 為詞庫。jieba 有 3 種分詞模式，全模式、精確模式，以及搜尋引擎模式，選擇最適合文本分析的精確模式。過程中有使用只抽取關鍵字，但是切出來的結果是依照重要性輸出，失去了時序的特性，因此就不採用，結果如下。

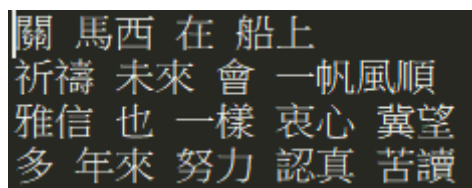


Fig1：切詞示意圖

切完後的句子就能得到很多詞，把這些中文詞就視為英文的單字一樣，採用 word2vec 建字典，並且將字詞轉成向量，並以此 word2vec 的 model 作為之後 embedding 層的參數。我們使用 gensim word2vector 的函式，在製作文本時會刪掉無用的標點符號，但保留“?”、...、“!”等能提供一些語意訊息的符號，經過不斷測試調整參數，發現大約 150~200 維的向量 testing 結果較好。我們曾嘗試用 fasttext 函式來處理 out of vocabulary 的問題，可以解決沒看過不過和字典有相似的詞，但發現字典的大小並沒有明顯增加，而且結果並沒有比 word2vec 好，或

許比起英文，中文的相似詞較不容易取出。除此之外，我們也在網路找了 pre-train 的中文 word2vec model，但是發現此次任務所用到的詞，很多都沒出現在這個 model 裡，所以不太適用。

在選擇句子大小時，發現 training 的劇本，每句都比 testing data 的句子短很多，因此我們直接將劇本連續 3 句串起來作為我們的 question，下一句為 good answer，並隨機抽取一句作為 bad answer，至於為何要抽取 good 和 bad answer 作為 training data，在描述 model 的段落會再次提及。除了句子長度有進行處理，我們也將標點符號去除掉，並且嘗試處理被 jieba 切出來只有一個字的情況，將一個字串到前面的詞，結果有稍微提升。

3. Model Description

模型 reference: <https://github.com/sachinbiradar9/Question-Answer-Selection>

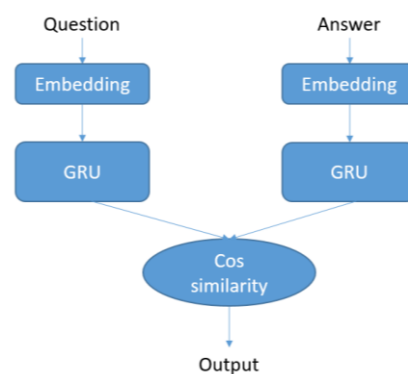


Fig2: 基本模型

Fig 2. 是最一開始使用的模型，我們將經過 preprocessing 的劇本，用 gensim 的 word2vec 函式把字詞轉成向量，且能表達出語意的相似度。將 word2vec model 的參數作為 embedding 層的參數，並且建立出字詞的字典，有著對應的數字。我們將 training data 每連續 3 句串在一起作為 question，下一句作為 good answer，並在劇本隨便抽一句作為 bad answer。要丟入模型前，將 question 和 answer 句子中的字詞用字典轉成數字 id，設定固定句子長度，倘若不足則在前方補零，過長則從前方截斷句子，作為 input 進行 training。

再來我們參考[3]的做法，在 training 時每個問句都會同步訓練 good answer 和 bad answer，而不是分開訓練，對此我們的想法是對於同樣的一個問句，在 training 的一個 epoch 中不應該重複出現，而是應該同時讓機器理解對於這個問句答案的好壞，因此在輸入一個問句時，我們會同時輸入 good answer 和 bad answer，經過相同的 hidden layer 並計算對於問句的 cosine similarity 後，使用自定義的 loss function 將兩者結果合併。若是 good answer 作為 input，會希望 cosine similarity 趨近於 1，若採用 bad answer 作為 input，則希望 cosine similarity 趨近

於-1，因此我們的 loss function 如下：

$$\text{loss} = \text{ReLU}(\text{margin} - x_0 + x_1)$$

其中，margin 取 0.05， x_0 為 good answer 的輸出， x_1 為 bad answer 的輸出，若 $x_0 \rightarrow 1, x_1 \rightarrow -1$ ，則 loss function 會不斷趨近-1.95，經過 relu function 後也就會趨近於 0，因此訓練的過程將會使 loss 越來越趨向於 0。在 predict testing data 時，將六個選項的 answer 依序丟入 question、good answer 的 pair，最後 output 會輸出 question 和該選項的 good similarity，取 output 最大的為最後答案。

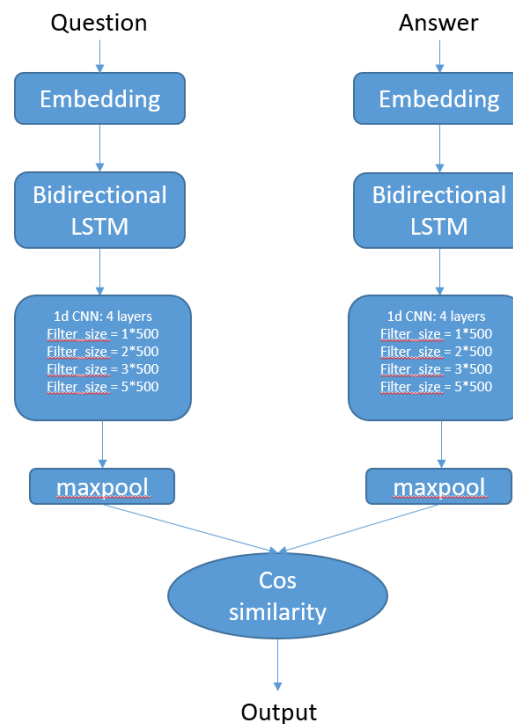


Fig3: BILSTM - CNN 模型

Fig 3. 此模型將 GRU 改成 Bidirectional LSTM，然後再接上 1D CNN。CNN 的優點是能將 LSTM 結果有效率的抽出 features，並且有 share weights 特性，相比於一般的 NN 能減少大量參數，除了能增加 accuracy，也減少 overfitting 的情況。

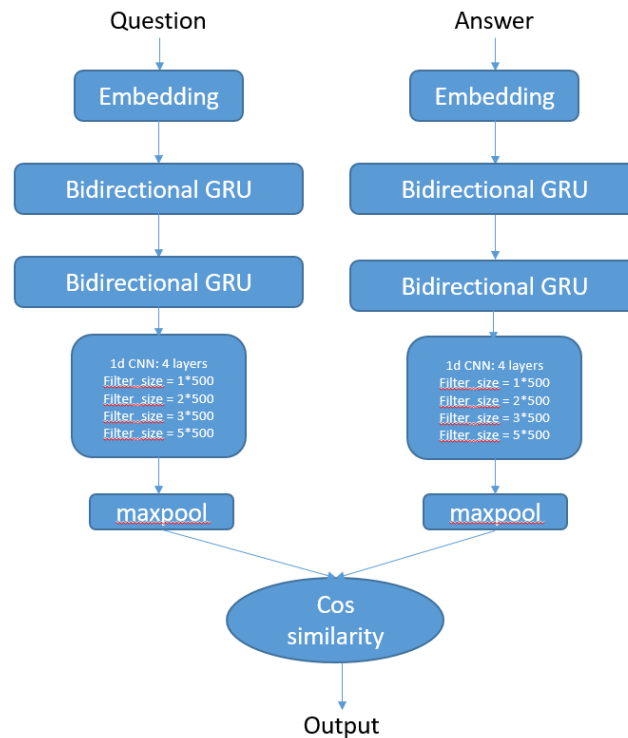


Fig4: 雙層 BIGRU - CNN 模型

Fig 4.的模型是利用了兩個 Bidirectional GRU 取代了 BILSTM 的模型。我們嘗試著增加 BILSTM 層的深度，但 LSTM 的參數實在太多且結構太過複雜，非常難訓練且訓練時間很長，因此將 Bidirectional LSTM 換成參數較少的 Bidirectional GRU。

在訓練時通常會把 batch_size 固定在 512，optimizer 為 Adam(lr = 0.001)，epochs = 10，但有時會遇到較難訓練的模型，藉會需要降低 learning rate 到 0.0005，或是將 batch_size 調整至 256。此外，我們會切出 10% 的資料為 validation set，作為 model check point 的依據存下最好的 model。

4. Experiment and Discussion

在 word embedding 方面，我們都是利用 gensim word2vector function，總共試過了五種不同的參數，並在 lstmenn 的模型上測試：

Vector size	Iter	Min_count	Windows	Public	Private
90	10	2	7	0.49762	0.49960
100	10	1	8	0.49604	0.48695
120	11	2	8	0.50869	0.49051
150	12	2	8	0.51225	0.49723
180	13	2	8	0.51739	0.49723

上表參數調整的基準是當 vector size 越大時，就給予更多的 iteration，測試後可

以發現當 vector size 越大，各詞間 cosine similarity 的值就會越小。一開始時我們都使用較小的 embedding vector size，public score 一直無法突破，但逐漸加大 vector size 後，public score 也跟著增加，因此最後固定使用 vector size = 150 或 180 的 w2v 模型。

在處理資料時，由於 min_count=2，遇到不在 w2v 模型裡的詞我們都指定該詞 id 為 0，在 test data 中遇到不認識的詞也是指定 id 為 0，我們嘗試令 embedding 層參數的 trainable = True，讓模型在 training 過程中也可以對於認識、不認識的 word embedding vector 再更進一步訓練：

Trainable	Vector size	Hidden dim	Public	Private
False	180	128	0.51739	0.49723
True	180	128	0.49723	0.49288

無論在 public、private 的分數都下降了，可見這樣的嘗試是失敗的，還是保持 gensim word2vector 訓練出來的詞向量能得到較好的結果。

接著比較有無 CNN 層：

CNN	Vector size	Hidden dim	Public	Private
Without	150	100	0.47114	0.46245
with	150	136	0.51225	0.49723

加入了 CNN 層後，在 kaggle 上的成績獲得了很大的進步，也印證了從[1]中提到的，RNN 層後加入 CNN 層並 pooling，能幫助我們提取 question、answer encoded 後的重點。對於這個情況我們也推論，若沒有 CNN 層則只能使用 RNN 層最後的輸出做內積，但一個問題或答句的重點並不總是會集中在最後一個字講完後的輸出，相反的重點更有可能分散在在 RNN time step 的某幾次輸出中，利用了 1D CNN 我們就能考量到每個時間點的輸出，並讓機器學習如何提取重點，也更能真實的反應問句、答句之間的關係。

對於 LSTMCNN 的模型，我們嘗試了不同的 Bidirectional GRU、2 層

Bidirectional GRU、2 層的 lstm 替換掉原本的 BiLSTM：

	Vector size	Hidden dim	Public	Private
BIGRU	150	128	0.50790	0.48458
2 層 BIGRU	150	128	0.50869	0.49130
BiLSTM	150	136	0.51225	0.49723
2 層 BiLSTM	150	100	0.49723	0.49723

若在 RNN 層使用 GRU，兩層的 GRU 會比一層的 GRU 還要好，發揮了”deep”的效用，但不論是一層或是兩層，效果都比原本的 LSTM 還差；從堆疊 GRU 的情況為啟發，我們嘗試也使用兩層的 LSTM，卻發現在 Public 上準確率掉了不少，我們認為是兩層的 LSTM 的參數過多，除了使模型不好 train 以外，也可

能太過 overfitting training data 的關係。

在對 BILSTM 層做的調整失敗後，我們嘗試改動 CNN 層，分別測試了擴展 CNN 層的 filter 數、使用 2 層 CNN、將 activation function 改成 relu:

	Vector size	Nb_fiter	Public	Private
CNN	150	500*4	0.51225	0.49723
CNN	150	1000*4	0.50355	0.48853
2 層 CNN	150	500*4	0.47193	0.47430
Relu	180	500*4	0.48613	0.47351

由於我們使用了 4 種不同的 filter 去擷取特徵，因此每個 nb_filter 都須乘以 4 才是一個模型裡真正的 feature map 數。而以上表來看，這兩個嘗試都是失敗的，我們推論若是增加 filter 數，雖然能讓偵測出更多的特徵但也會使得模型更複雜，對於 encoded size vector 大約在 100~150 間也可能偵測到許多重複的特徵而導致效果變差；若是多加了一層 CNN 後，模型變得太複雜且參數太多，在 kaggle 上的分數大幅地掉了 0.03 左右，考量到 CNN 容易 overfitting 的情況，推論極有可能是因為對 training data overfitting，造成準確率大幅下降。

以上的實驗都存在著一個共通點，public 和 private 之間的分數差異過大，對此我們嘗試著進行修正，作法是在 CNN 層加上 dropout。

dropout	Vector size	Hidden dim	Public	Private
no	150	136	0.51225	0.49723
0.2	150	150	0.50592	0.49328
no	180	128	0.51739	0.49723
0.2	180	150	0.51660	0.50316

有趣的是，若加上 dropout，在 Public 的表現都會不約而同地下降，下降幅度甚至會大到令我們認為這是無效的嘗試，但對比 Private 分數卻發現，加入了 dropout 的模型雖然 private 分數還是會低於 public 分數 0.01 以上，但兩者之間的差距卻被縮小了。以上述四個模型為例，加入 dropout 前 public 與 private 的分數平均差距 0.01759，但加入 dropout 後兩者分數平均差距縮小至 0.01304，而更有趣的還在 ensemble 的結果。

在探討 ensemble 前，先簡單介紹將會使用的模型，以及組合的方法。

使用模型：

Model	Layer type	Hidden dim	Vector size	Dropout
1	lstm + cnn	128	180	No
2	lstm + cnn	150	180	Yes
3	lstm + cnn	136	150	No
4	lstm + cnn	150	150	yes

5	Gru	128	150	No
6	lstm + cnn	100	180	yes

ensemble 方法：

No.	Models	Method	Public	Private
1	1 + 3 + 5	Sum	0.53478	0.52134
2	2 + 4 + 5	Sum	0.53754	0.52252
3	2 + 4 + 5 + 6	Sum	0.53478	0.52411
4	2 + 4 + 5 + 6	Vote	0.52411	0.51778
5	2 + 4 + 6	Sum	0.53043	0.52252

ensemble1、2 比較：延續上面對 dropout 的討論，雖然在個別模型中不加入 dropout 獲得的 public 分數較高，但在 ensemble 時，使用 dropout 模型卻能在 public、private 都得到較高的準確率。對於這點我們的討論是，在 CNN 層若不使用 dropout，會使模型非常容易 overfitting，只是恰巧得到的結果能剛好符合 public data 的分布與結果，因此 public 的分數較高。但到了 ensemble 的階段，兩個較高分的模型都帶有一定程度的 overfitting，而這個 overfitting 造成的偏見限制了 ensemble 後的效果；相反來說，加入了 dropout 後雖然單一模型分數較低，但減緩 overfitting 的原因下保護了模型的輸出不過於極端，也因此 ensemble 能得到更好的結果。從以上來看，dropout 層對於一個模型真的非常重要，能使該模型面對各種情況時更有彈性，才不會 overfit training 和 public testing data。

ensemble3：從上述兩個模型中，我們認為 private 應有在提高的可能，因此我們再加了 lstm cnn with dropout 模型一同 ensemble，希望帶有 dropout 模型減緩 overfitting 的效果能發揮所用，結果雖然 public 下降，但 private 分數的確如我們所想而上升。

ensemble4：除了加總以外，我們也嘗試用投票，但效果非常不好，我們認為是參與決策的 model 不夠多的關係。

ensemble5：這裡我們嘗試把較低分 GRU model 拿掉，只讓三個高分的 LSTMCNN model 參與決策，結果在 public、private 的分數都降低，推斷是太過相似的架構有可能存在著相同的盲點，而增加 model 多樣性能幫助 ensemble 後的結果更加全面。

5. Conclusion

對於這次的題目我們做了許多不同的嘗試，除了一般的參數調整之外，也是過了直接的問答 cosine similarity 比對、考量到時序的 RNN 模型和加入了 CNN

機制的模型。從中可以發現調整參數固然重要，最好與最壞可能會讓 kaggle 上的分數差距 0.02~0.03，但每次能大幅躍進都是在嘗試了新的模型之後，說明了每個 model 都有其極限，在怎樣調整參數都無法大幅突破，而找到一個合適的 model 才是更好的做法。除了 RNN model 外，word embedding 也是非常重要的一環，隨 vector size 改變同樣的 model 差距也會到 0.02 左右，若 vector size 太小會使電腦無法明確分辨每個詞的相似度，太大則會令每個詞都太不相關，調整至適度的大小才能正確地表示詞語間的關係，也才能得到好的結果。

對比其他人的成果後，我們檢討自己的模型為何表現不好，最後歸納出了兩點。首先是 data preprocessing 的部分，在面對 training data 時其他組會將每一句不斷加下一句並計算字數，利用機率等方法讓 training data 的字數分佈能趨近 testing data 的分佈，而我們只是單純的以前三句做為問句、下一句做為答句，如此 training 的字數分佈勢必和 testing 的字數差異過大，在面對 testing data 時自然也不會有好表現。第二點是 loss function 的部分，依我們的定義方法有個缺點，也就是 good answer 和 bad answer 的比例永遠是 1:1，除此之外我們 ensemble 後能提升的幅度相比於其他組非常小，推論是因為我們模型的輸出介於 -1~1 之間，當有一個模型犯了嚴重的錯誤時，得到的 output 會趨近於 -1，就算其他的模型的預測是正確的也難以修正這樣的錯誤，也造成了準確度無法提升，換而言之我們的做法沒有預留給每個模型犯錯的空間。

總而言之，這次的 final project 是個有趣的體驗，學到了一些 NLP 相關的知識，還有對於這種沒有 label 的資料該如何處理、如何訓練。

6. References

- [1] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou, “LSTM-Based Deep Learning Models for Non-factoid Answer Selection,” *IEEE First International Conference on Data Science in Cyberspace*, Changsha China, 13-16, June, 2016.
- [2] Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, Bowen Zhou, “APPLYING DEEP LEARNING TO ANSWER SELECTION: A STUDY AND AN OPEN TASK,” *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Scottsdale, AZ, USA, 13-17, Dec., 2015.
- [3] sachinbiradar9, “Question-Answer-Selection,” *github*.
< <https://github.com/sachinbiradar9/Question-Answer-Selection> >