題目：conversations in TV shows

1. Team name, members and your work division

A. Team name:

NTU_r05525096_ICAN 我不行

B. Members:

[1] R06525055 吳伯彥

[2] R06525064 顧晨生

[3] R05525096 郭捷

C. work division:

[1] R06525055 ：前處理、model2、報告撰寫

[2] R06525064 ：前處理、model3、報告撰寫

[3] R05525096 ：前處理、model1、報告撰寫

2. Preprocessing/Feature Engineering

在前處理和特徵工程的部分，首先我們先觀察了訓練資料和測試資料。我們現在中只有極少量的符號，而在訓練資料中則有較多的符號，為了獲得更乾淨的訓練資料，我們第一步對訓練資料做了符號的過濾。

2.1. jieba 中文斷詞

使用的字典為：dict.txt.big（繁體中文）

使用方法：`words = jieba.cut(line, cut_all=False)`

cut_all = True 為 Full Mode, False 則為 Default Mode（精確模式），在此研究報告中，一律使用精確模式

(1) 比較模式設定

原例子-雪子已經拿到日本的護士執照了

經過斷詞-

Full Mode: 雪/子/已經/拿到/日本/的/護士/執照/了

Default Mode: 雪子/已經/拿到/日本/的/護士/執照/了

由上可知，採用精確模式的效果會較佳。

(2) 比較有無使用字典

原例子-窮人家的孩子比較吃得了苦

無使用字典-

Full Mode: 窮/人家/的/孩子/比/較/吃得了/苦

Default Mode: 窮/人家/的/孩子/比較/吃得了/苦

有使用字典-

Full Mode: 窮人/窮人家/人家/的/孩子/ 比較/吃得了/苦

Default Mode: 窮人家/的/孩子/比較/吃得了/苦

由上可知，有使用字典的斷詞效果會較佳。

總結(1)、(2)之結論，我們能看出有使用字典並採精確模式進行斷詞的效果是較佳的會更符合實際情況在斷詞的同時我們還對比了是否過濾停用詞，發現不過濾停用詞的效果會更好。

2.2. Word Embedding

在斷詞過後，我們使用了 gensim Word2Vec model 去訓練詞向量。並將訓練好的詞向量作為之後模型中 Embedding layer 的初始權重。

使用方法：

```
model = word2vec.Word2Vec(sentences,size=200,min_count=1,sg=1,iter=10)
```

所調整的參數有：

- sentences: 所放入訓練的句子
- size: 訓練出的詞向量設為 200 維
- sg: sg=1 表示採用 skip-gram, sg=0 表示採用 cbow 在最終我們採用 skip-gram 的形式（兩者之比較見 Experiments and Discussion）
- min_count: 設定採用出現次數大於 min_count 的詞作為訓練資料，在此為預設
- iter: 訓練的次數

2.3. 訓練資料

首先對我們發現在測試資料中的對話多以一句以上的對話為主，而訓練資料中則以短句居多。經過實驗，我們發現將訓練資料中的連續兩句話串接起來作為模型的輸入的效果會比以單句的訓練資料作為模型的輸入來得好。

在開始準備訓練資料之前，我們先通過字典將文本轉換為序列，然後再通過填充序列的方式將不同長度的訓練資料補齊。由於在本次期末專案中我們實作了多種不同的 model，而每種 model 所需要的輸入與輸出格式也不盡相同，所以我們針對不同的 model 有不同的訓練資料處理方式。

(1) seq2seq model

在 seq2seq 的 model 中我們將連續的句子的前一句作為 X1，將連續的句子的後一句作為 X2，將連續句子的後一句的下一個時間點的句子作為 Y。在每個句子的開頭會使用'\t'作為 BOS，使用'\n'作為 EOS。

(2) 相似度 model

訓練之資料格式：

X=(句子 1,句子 2) Y=(0|1) 0:表示非下一句 1:表示下一句

將訓練資料集切成正確答案對話與錯誤答案對話進行訓練

正確答案：(S1, S2)……(Sn, Sn-1) Y=1

錯誤答案：(S1, SR)……(Sn, SR) Y=0

R=隨機挑一句非正確答案的句子

舉例：

前：關馬西在船上 後：祈禱未來會一帆風順	Y = 1
前：關馬西在船上 後：其實雅信不知道	Y = 0

綠色框為正確答案集，紅色框為錯誤答案集

本研究中比較了當正確與錯誤的資料比為 1:1、1:4、1:8、1:15

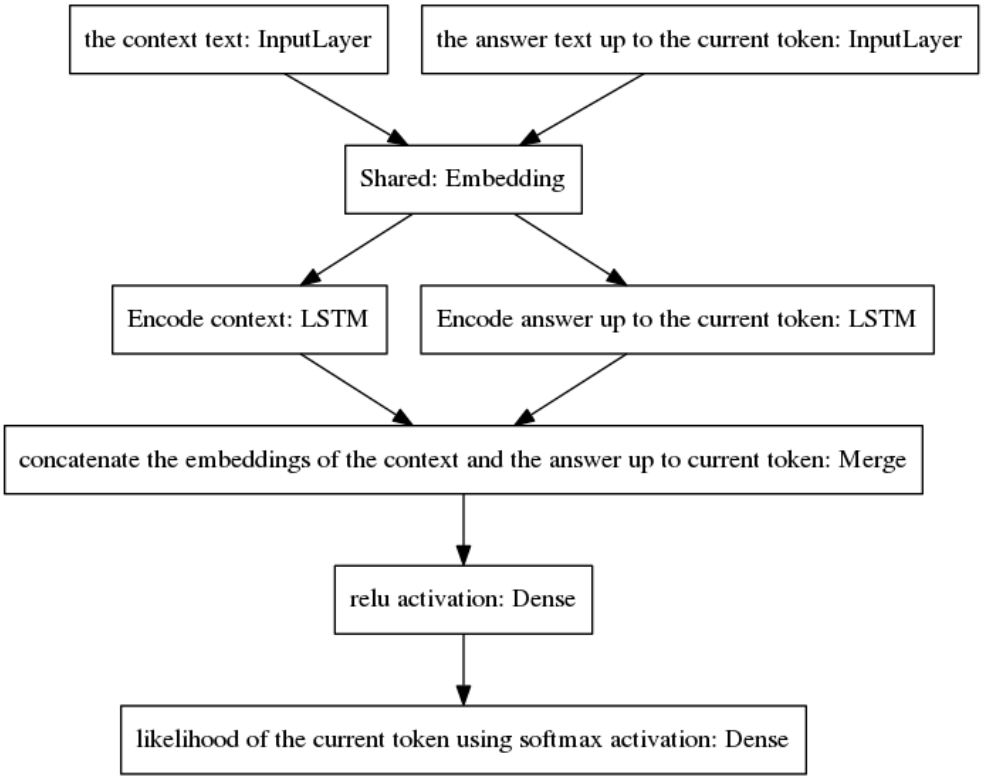
最終採用之資料比為 1:8

綜上所述，我們在本次期末專案的前處理和特徵工程的部分做了過濾符號、使用 jieba 官方提供的繁體中文斷詞字典(dict.txt.big)進行斷詞、用 gensim 的套件訓練詞向量、串接句子的動作。

3. Model Description (At least two different models)

3.1. Model 1

Model 1 我們使用的是 seq2seq 的 model，model 架構如下：



首先將連續的兩個句子的前一句和後一句分別作為 model 的 2 個 input，然後讓這兩個 input 共用相同的 embedding layer。然後再將分別通過兩個 LSTM，最後的結果再經過一個 Dense 做分類。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None)	0	
input_2 (InputLayer)	(None, None)	0	
embedding_1 (Embedding)	(None, None, 100)	4946100	input_1[0][0]
embedding_2 (Embedding)	(None, None, 100)	4946100	input_2[0][0]
lstm_1 (LSTM)	[(None, 100), (None,	80400	embedding_1[0][0]
lstm_2 (LSTM)	[(None, None, 100),	80400	embedding_2[0][0] lstm_1[0][1] lstm_1[0][2]
dense_1 (Dense)	(None, None, 49460)	4995460	lstm_2[0][0]

Total params: 15,048,460
Trainable params: 5,156,260
Non-trainable params: 9,892,200

3.2. Model2

Model2 是相似度 model，分別是兩個 embedding+lstm layer，最後 dot 起來再做 softmax

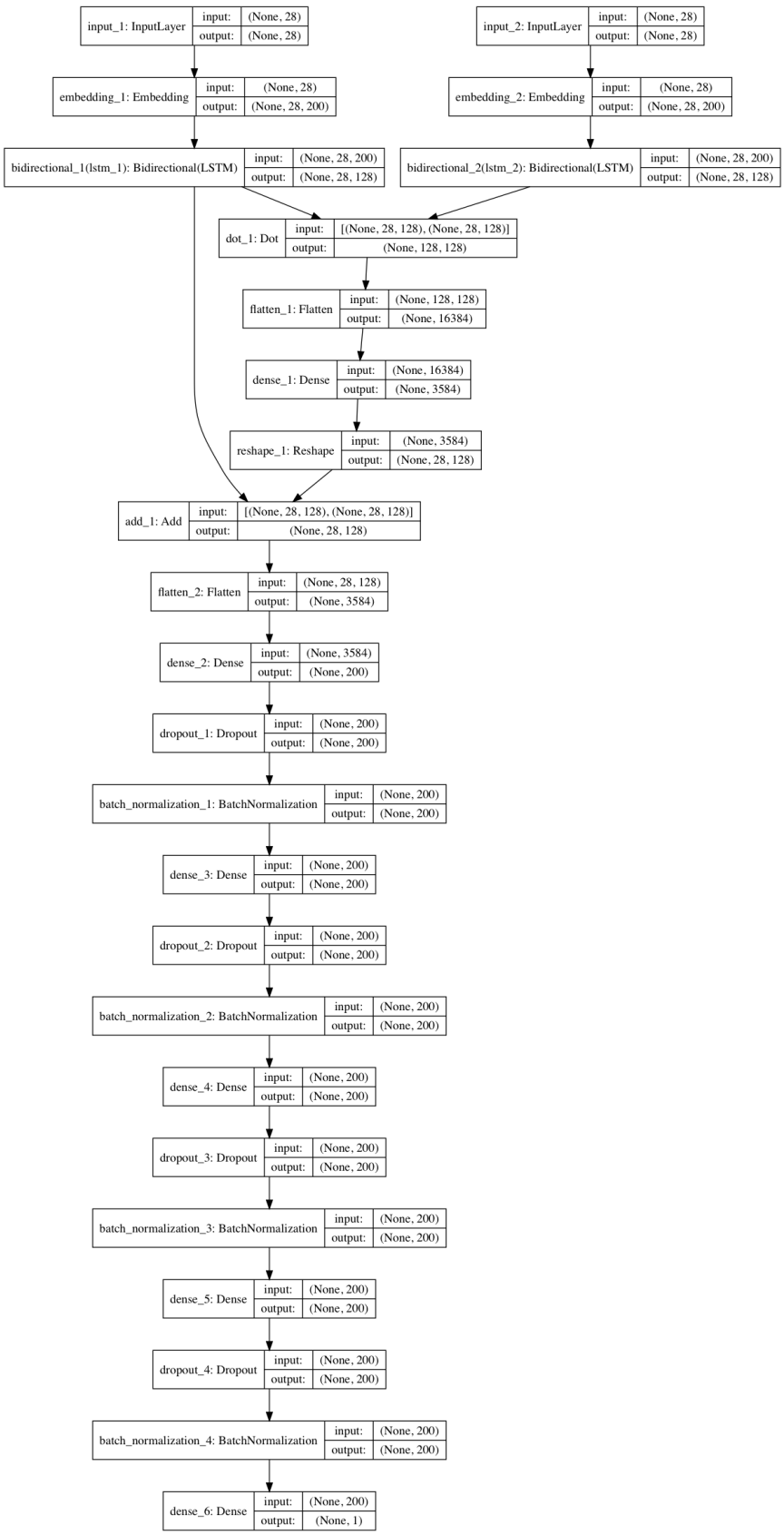
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 27)	0	
input_2 (InputLayer)	(None, 27)	0	
embedding_1 (Embedding)	(None, 27, 300)	14953500	input_1[0][0]
embedding_2 (Embedding)	(None, 27, 300)	14953500	input_2[0][0]
bidirectional_1 (Bidirectional)	(None, 27, 512)	1142784	embedding_1[0][0]
bidirectional_2 (Bidirectional)	(None, 27, 512)	1142784	embedding_2[0][0]
dropout_1 (Dropout)	(None, 27, 512)	0	bidirectional_1[0][0]
dropout_2 (Dropout)	(None, 27, 512)	0	bidirectional_2[0][0]
cu_dnnlstm_2 (CuDNNLSTM)	(None, 256)	788480	dropout_1[0][0]
cu_dnnlstm_4 (CuDNNLSTM)	(None, 256)	788480	dropout_2[0][0]
dot_1 (Dot)	(None, 1)	0	cu_dnnlstm_2[0][0] cu_dnnlstm_4[0][0]
dense_1 (Dense)	(None, 1)	2	dot_1[0][0]

Total params: 33,769,530
Trainable params: 3,862,530
Non-trainable params: 29,907,000

Train on 10900656 samples, validate on 1211184 samples

3.3. Model 3

Model3 承接於 Model2，為相似度-LSTM with Attention，模型架構如下：


input_1: InputLayer
input: (None, 28)
output: (None, 28)
input_2: InputLayer
input: (None, 28)
output: (None, 28)
embedding_1: Embedding
input: (None, 28)
output: (None, 28, 200)
embedding_2: Embedding
input: (None, 28)
output: (None, 28, 200)
bidirectional_1(lstm_1): Bidirectional(LSTM)
input: (None, 28, 200)
output: (None, 28, 128)
bidirectional_2(lstm_2): Bidirectional(LSTM)
input: (None, 28, 200)
output: (None, 28, 128)
dot_1: Dot
input: [(None, 28, 128), (None, 28, 128)]
output: (None, 128, 128)
flatten_1: Flatten
input: (None, 128, 128)
output: (None, 16384)
dense_1: Dense
input: (None, 16384)
output: (None, 3584)
reshape_1: Reshape
input: (None, 3584)
output: (None, 28, 128)
add_1: Add
input: [(None, 28, 128), (None, 28, 128)]
output: (None, 28, 128)
flatten_2: Flatten
input: (None, 28, 128)
output: (None, 3584)
dense_2: Dense
input: (None, 3584)
output: (None, 200)
dropout_1: Dropout
input: (None, 200)
output: (None, 200)
batch_normalization_1: BatchNormalization
input: (None, 200)
output: (None, 200)
dense_3: Dense
input: (None, 200)
output: (None, 200)
dropout_2: Dropout
input: (None, 200)
output: (None, 200)
batch_normalization_2: BatchNormalization
input: (None, 200)
output: (None, 200)
dense_4: Dense
input: (None, 200)
output: (None, 200)
dropout_3: Dropout
input: (None, 200)
output: (None, 200)
batch_normalization_3: BatchNormalization
input: (None, 200)
output: (None, 200)
dense_5: Dense
input: (None, 200)
output: (None, 200)
dropout_4: Dropout
input: (None, 200)
output: (None, 200)
batch_normalization_4: BatchNormalization
input: (None, 200)
output: (None, 200)
dense_6: Dense
input: (None, 200)
output: (None, 1)

模型步驟算法大致如下：

A. 將問題和答案進行預處理，input_1 為問題，input_2 為答案

B. 將問題和答案各自進行 embedding 表示。

C. 透過 BILSTM(Bi-directional LSTM)進行特徵計算。

D. 採用 BILSTM 計算的問題特徵和答案特徵，並計算 Attention 權值，用於保留答案中較為重要的詞。

E. Loss Function : cosine similarity。

4. Experiments and Discussion

4.1. Model 1

在實作 seq2seq 的 model 的時候我們遇到了許多的問題。

(1)由於 seq2seq 的輸出需要經過一個 softmax 的 Dense 做分類，所以需要把輸出的句子的每個字都做 one-hot encoding。而對 750K 的訓練資料做 one-hot encoding 所需的記憶體容量是非常大的,所以我們必須尋找其他的方法。一開始我們使用的方式是將訓練資料分別做 one-hot encoding 用 train_on_batch 的方式進行。後來我們使用 keras 提供的 sparse_categorical_crossentrop 做為 loss function 有效的解決了這個問題。因為該 loss function 是與 categorical_crossentrop 相同的 loss function ,不同在於在訓練的時候他會自動將輸出的資料做 one-hot encoding，這很好的解決了在準備訓練資料的時候做 one-hot encoding 造成的記憶體容量不足的問題。

(2)在剛開始訓練 seq2seq 的時候還遇到一個問題，如果直接 predict 會出現如下情況，輸出的結果會產生連續的相同的字。

```
<BOS>我我我我我我_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD
Epoch 1/1
256/256 [==============================] - 1s 2ms/step - loss: 2.9236
<BOS>妳廣告公司早點下班來百貨公司幫金城巡視<EOS>_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD

<BOS><BOS>我我我我我_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD
Epoch 1/1
256/256 [==============================] - 1s 2ms/step - loss: 2.8224
<BOS>妳廣告公司早點下班來百貨公司幫金城巡視<EOS>_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD

<BOS>我我我我我_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD
Epoch 1/1
256/256 [==============================] - 1s 2ms/step - loss: 2.8868
<BOS>妳廣告公司早點下班來百貨公司幫金城巡視<EOS>_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD

<BOS>我我我我我我_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD_PAD
Epoch 1/1
```

經過網上查找資料以及詢問有相關經驗的學長後,發現將輸出的資料的前一個字的預測結果和 state 作為下一個字的輸入，可以解決連續輸出相同的字的問題。

```
Input sentence: 關馬西在船上祈禱未來會一帆風順

Output sentence 雅信也一樣表心冀望多年來努力認真苦讀

Decoded sentence: ['你', '一定', '要', '好好', '照顧', '自己', '\t', '我', '也', '會', '好好', '照顧', '自己', '\n']
-
Input sentence: 祈禱未來會一帆風順雅信也一樣表心冀望

Output sentence 多年來努力認真苦讀可以回家鄉服務

Decoded sentence: ['讓', '你', '的', '機會', '\t', '你', '一定', '要', '好好', '照顧', '自己', '\n']
-
Input sentence: 雅信也一樣表心冀望多年來努力認真苦讀

Output sentence 可以回家鄉服務其實雅信不知道

Decoded sentence: ['讓', '他', '知道', '\t', '你', '的', '心情', '一定', '會', '很', '難過', '\n']
-
Input sentence: 多年來努力認真苦讀可以回家鄉服務

Output sentence 其實雅信不知道在台灣

Decoded sentence: ['你', '放心', '\t', '我', '一定', '會', '好好', '照顧', '自己', '\n']
-
Input sentence: 可以回家鄉服務其實雅信不知道

Output sentence 在台灣早就有很多人在等待他了

Decoded sentence: ['你', '的', '事', '\t', '你', '要', '知道', '\n']
```

綜上所述，雖然我們在訓練 seq2seq 的 model 的時候遇到了許多問題也成功的解決了部分的問題，但是實驗結果依然不盡如人意。在 training data 的 predict 上面雖然可以根據不同的 input 產生不同的 output，但是產生的 output 大部分與正確答案不符。在 testing data 上的表現就更差了，都回覆相同的結果。最後附上實驗結果。在 training set 的 loss 最低為 1.6089，acc 最高為 0.7083。在 kaggle 的 public score 最高為 0.39946。

```
Input sentence: 你才剛生完小孩體力都還沒復原

Decoded sentence: ['不要', '這樣', '\t', '我', '知道', '\n']
Input sentence: 你躺在床上好好休息晚點護士會抱小孩過來

Decoded sentence: ['不要', '這樣', '\t', '我', '知道', '\n']
Input sentence: 像誰都一樣只要是我們的孩子不管她長什麼樣都沒關係

Decoded sentence: ['不要', '這樣', '\t', '我', '知道', '\n']
Input sentence: 你放心有清標他在這裡顧著

Decoded sentence: ['不要', '這樣', '\t', '我', '知道', '\n']
Input sentence: 當媽媽了突然就覺得自己變老了

Decoded sentence: ['不要', '這樣', '\t', '我', '知道', '\n']
Input sentence: 她吃過飯了沒她還沒吃她還正在餵奶

loss: 1.6089 - acc: 0.7083 - val_loss: 1.7919 - val_acc: 0.6936
```

4.2. Model 2

在 Model 的結構上，我們測試過在 lstm 上是否使用 Bidirectional，結果顯示使用 Bidirectional 會約略提升 0.01 左右的分數。

另外我們也試過，將 dot layer 改為 concatenate layer，看看效果如何，實驗的結果是 dot layer 相較 concatenate layer 有 0.02 左右較佳表現。

在 Word2Vec 上，若是使用 CBOW(sg=0)，需要去除 stopword 與標點才會獲得較佳得結果，反之，使用 skip-gram(sg=1)，去除與否的表現差異並不大，整體來說，skip-gram 的表現會比 CBOW 來得更好。

在訓練模型的過程中，我們發現影響結果的主要因素來自於 data 的處理，首先我們發現，擷取句子的長度，會影響結果的好壞，下表為我們的實驗結果，max sentence length 為一個句子最多取幾個詞

Max sentence length	66	27	10
Score	0.381	0.421	0.417

一開始，我們沒有對句子做裁減，結果為 0.38，而當我們觀察 data 後發現，test 的句子往往比 train 還要長，因此我們試著只取句子的最後十個詞彙當作向量，反而分數上升了 0.3 左右，我們又漸漸增加 sentence 的維度，當 sentence length=27 時，我們獲得 0.42 的成績，透過實驗，我們認為擷取與 training data 差不多的長度句子，會使得結果較好。

接著，我們測試 training data 的截取，一開始是一行當作一句話，但觀察 data 後發現，test data 有一部分的題目是兩句對話，因此我們也試著將兩句合併當作 training data

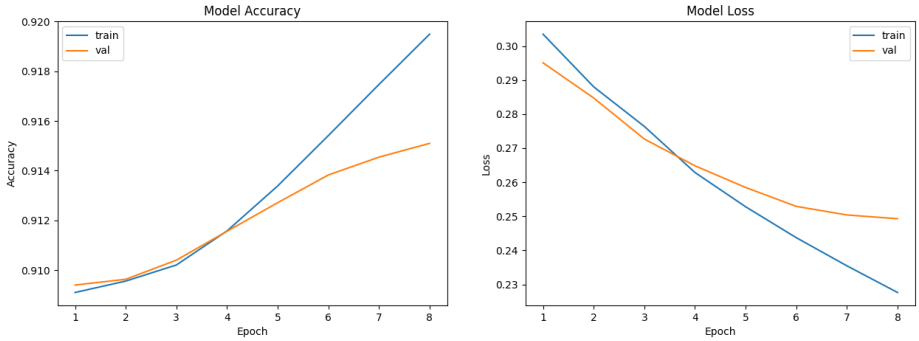
例如：

在劇本檔中，句子為 S1,S2,S3..Sn，除了取(S1,S2),(S2,S3)...之外，還取了(S1+S2,S3),(S2+S3,S4)...作為 training data，結果如下表，可以看到有加入這樣的 training data 效果會來得較佳。

Training data 取樣	1 對 1	1 對 1+2 對 1
Score	0.421	0.434

最後，我們嘗試增加更多的 training data 筆數，來看結果是否會比較好，我們試著增加更多的 False Data，有利於模型的訓練，但也發現當 True 與 False 超過一定比例，結果反而開始下降，最終，我們選擇 True:False 資料比例為 1:8，實驗結果如下：

T/F ratio	1:1	1:4	1:8	1:15
Score	0.434	0.441	0.488	0.471



4.3. Model 3

基於 Model 2:相似度-LSTM with Attention 架構，採用的正確對話集與錯誤對話集的比數為 1:4，我們嘗試針對以下幾個地方去進行調整，第一個部分為時間序列模型的選擇，分別嘗試了 LSTM / BiLSTM / BiGRU，實驗結果如下：

Model	LSTM	BiLSTM	BiGRU
Score	0.410	0.417	0.415

原本期望藉由將 attention 計算得到的權值應用到答案特徵，用以加強對問題較為重要的詞的權值，降低對問題關聯性較小詞的權重，但與原先的相似度 model 比較（以相同資料比數），所得出的 Score 並未有提升，反而較差，故最終仍選擇 Model 2 進行改良，比較的實驗結果如下 (True:False 資料比例皆為 1:4)：

Model	相似度 model	相似度 model with Attention
Score	0.441	0.417

4.4. 結果

以下總結不同 Model 在 Kaggle 上 Score 的影響：

Model	Score
seq2seq	0.399
相似度 model(BiLSTM)	0.488
相似度 model with Attention	0.417

綜合上述所言，所以最終我們選擇 model 2 作為我們在 github 上所執行的模型。