ML Final - TV conversation

Introduction & Motivation (1%)

我們這次選擇的題目是 tv conversation，主要是給定一段中文電視劇的對話後，從六個選項中選出最適合的回應。選擇這個題目主要是因為雖然 machine learning 運用在自然語言處理上已行之有年，但由於分詞難度或語言普及程度等等的關係，中文的文本處理相較於英文來說，相對資源較少，難度也更高。在認為這題目充滿挑戰性的情況下，我們最終決定選擇它作為 final project 的題目。

Data Preprocessing / Feature Engineering (2%)

Simple Text Replacement

用正則表達式函式庫 `re` 將以下文字做處理。

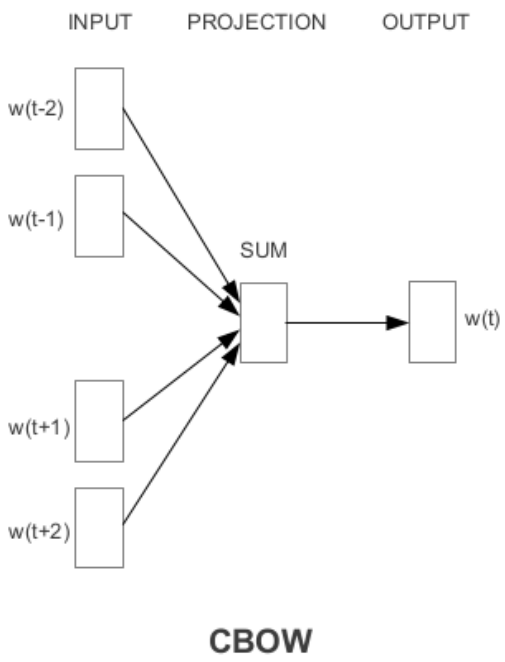
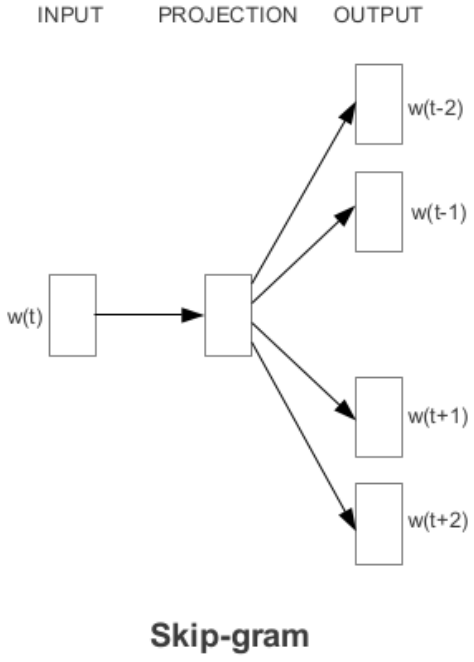
```
pat_remove = re.compile('["\-()]')
pat_punc = re.compile('[，、？\.+]')
```

其中`pat_remove`會被移除，`pat_punc`會被轉成單格空白。

Jieba

將一句話拆成一個一個詞，並往後以詞作為一個有意義的單位。若沒有分詞而只以單個字來當作 embedding 的單位，可能會造成內含的訊息壓縮度過高。

Pretrain CBOW & skip-gram

CBOW	skip-gram
	

我們分別使用 skip-gram 和 CBOW 對 training data 做 word embedding。 並在接下來訓練模型的時候，會將文字利用上述兩種 pretrain-model 轉換成長度較為固定的 vector。

" special case

我們觀察testing data時，發現有 " 的句子通常都是歌詞的一開始，而在對話中較少有一句正常的話接歌曲。因此我們會將這些選項刪除，使干擾選項減少，進而增加正確率。

Model Description (4%)

mLSTM Model

pretrain

word embedding 用 word2vec(skipgram) pretrain。

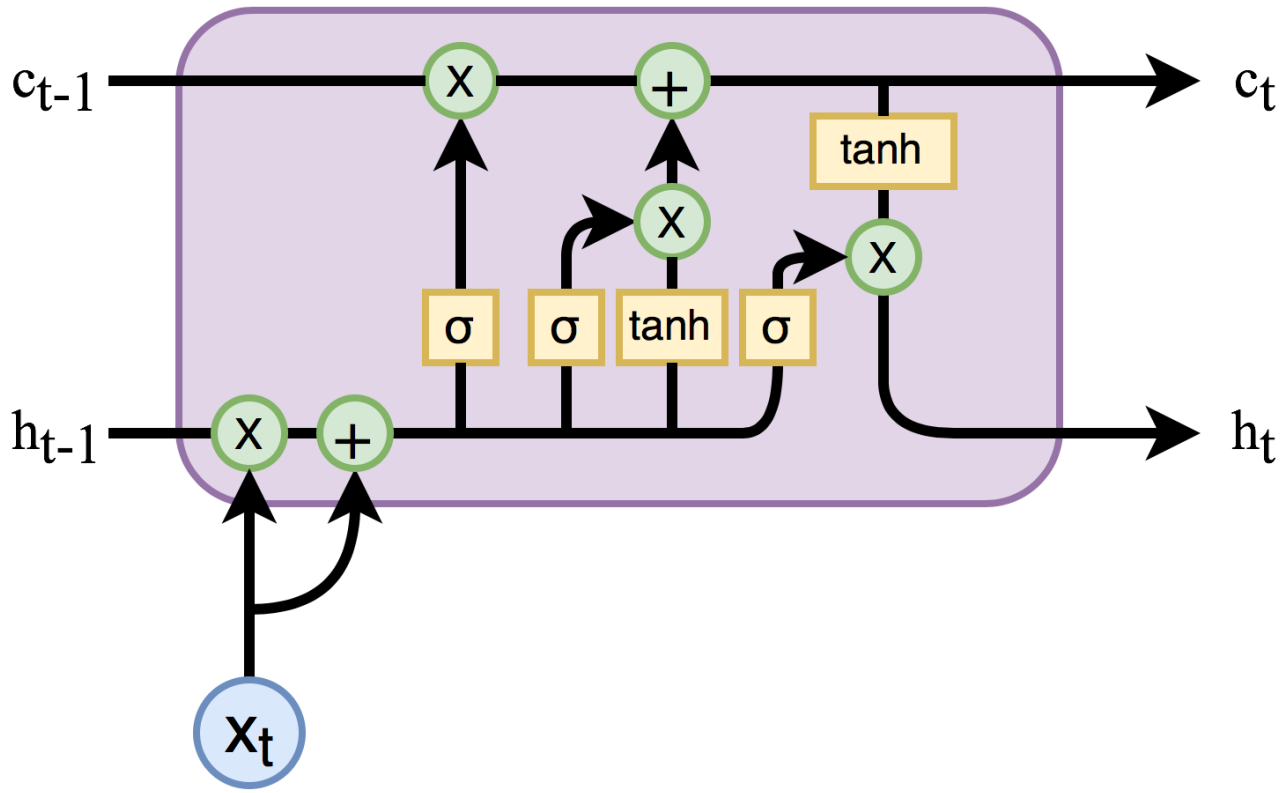
training

從每個句子的下三句裡面選出一句當成正確答案，另外 sample 五句當成錯誤答案。把題目和選項 encode 之後做內積，然後 6 個內積值取 softmax，loss 用 cross entropy。 optimizer 用 Adam，learning rate 從 5e-4 線性下降到 0，訓練 300 個 epoch。 （訓練時內積換成 `element-wise product -> dropout(0.5) -> sum`）

encoder

Layer	Params
word embedding	256d
dropout	p=0.5
mLSTM	512d 取最後的 c 作為 output (h, c) 的初始值設為可訓練參數 所有 W 都加 weight normalization

mLSTM cell



$$m_t = (W_{hm}h_{t-1} + b_{hm})(W_{im}x_t + b_{im})\ i_t = \sigma(W_{ii}x_t + b_{ii} + W_{mi}m_t + b_{mi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{mf}m_t + b_{mf})\ g_t = \tanh(W_{ig}x_t + b_{ig} + W_{mg}m_t + b_{mg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{mo}m_t + b_{mo})\ c_t = f_t c_{t-1} + i_t g_t\ h_t = o_t \tanh(c_t)$$

BOW Model

pretrain

word embedding 用 word2vec(CBOW) pretrain。

training

把每個句子當成問題句，該問題句的下一句當成正確答案，另外 sample 下一句以外的一個句子當成錯誤答案。把問題句和答案句經過 bag of words得出的向量(shape=(embed_size,))經過 DNN 得到的結果做內積之後 sigmoid，loss 用 binary cross entropy。 optimizer 用 Adamax。

- $lr = 0.002$

- $\beta_1 = 0.9$
- $\beta_2 = 0.999$
- $decay = 1e-5$

訓練 15 個 epoch。

（訓練時內積換成 `sum -> Dot -> Dense(1,sigmoid)`）

encoder

Layer	Params
word embedding	64d
Dense	1024
Dropout	0.2
LeakyReLU	0.1
BatchNormalization	<keras default>
Dense	4096
Dropout	0.3
LeakyReLU	0.1
BatchNormalization	<keras default>

double encoder Model

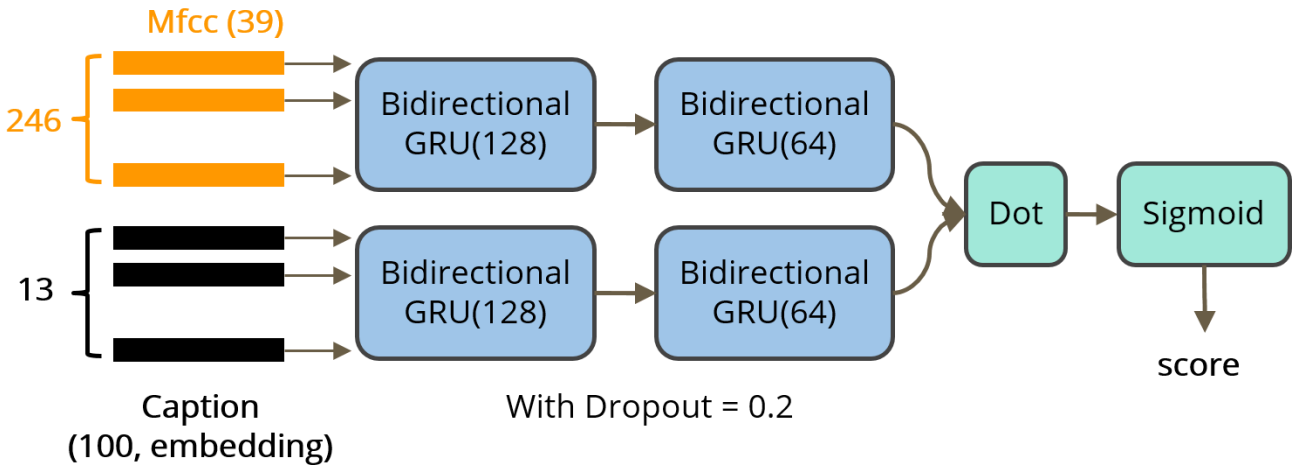
pretrain

word embedding 用 word2vec(skipgram) pretrain。

training

把每個句子當成問題句，該問題句的下一句當成正確答案，另外 sample 下一句以外的一個句子當成錯誤答案。把問題句和答案句經過 word embedding 得出的矩陣 (shape=(sentence_length, embed_size)) 分別經過兩個不同的 encoder 後得到的結果做內積之後 sigmoid，loss 用 binary cross entropy。

此方法是由 https://github.com/thtang/ML2017FALL/tree/master/final 所啟發，差別僅在於我們的 input (橘色和黑色部份) 皆為 (sentence_length, embed_size) 的矩陣，而 mfcc 處為問題句的 word embedding sequence，caption 處為回答句的 word embedding sequence。



optimizer 用 Adam。

- $lr = 0.001$
- $\beta_1 = 0.9$
- $\beta_2 = 0.999$
- $decay = 0.0$

訓練 20 個 epoch。

encoder

Layer	Params
word embedding	300d
Bidirectional GRU	128
Dropout	0.2
Bidirectional GRU	64
Dropout	0.2

Experiment and Discussion (6%)

以下的正確率，皆以 public/private 形式表示。

有無刪去選項中的引號的比較

- 有刪除：0.50395 / 0.49683
- 沒刪除：0.51264 / 0.50671

若刪除選項中的引號，可以刪除那些像是歌詞的選項，進而增加正確率，實驗模型為上述的 double encoder model。

mLSTM 與 LSTM 的比較

- LSTM: 0.54071 / 0.52885
- mLSTM: 0.55652 / 0.54545

使用由原本 LSTM 修改而來的 mLSTM ，發現跟 LSTM 相差並沒有很多。

word embedding 維度的比較

- 100d: 0.38418 / 0.37667
- 150d: 0.42055 / 0.43201
- 200d: 0.42569 / 0.42252
- 250d: 0.48063 / 0.48221
- 300d: 0.51264 / 0.50671

在做 word embedding 時，維度跟結果會有很明顯的相關，因此我們也做了跟 embedding 維度有關的實驗。實驗用的 model 為上述的 double encoder model。由結果可以發現維度越大越好，至於維度再繼續增加的情形因為訓練時間會過長的緣故，就沒有繼續嘗試。

訓練集正反樣本比例的比較

- 正確佔1/2，錯誤佔1/2: 0.54387 / 0.53833
- 正確佔1/6，錯誤佔5/6: 0.55652 / 0.54545

在預測時，因為要從六個選項中選一個正確的選項，因此我們讓訓練時的正反樣本比例和測試時相同，發現結果有變好。實驗模型為上述的 mLSTM model。

訓練時的問題句由多句接合與否的比較

- 無接合: 0.49604 / 0.48893
- 有接合: 0.51264 / 0.50671

在預測時，因為問題句幾乎都是由多句接合的，因此我們在訓練時的問題句由三句接合，判斷第四句是否正確作為訓練的樣本，發現結果有變好。實驗模型為上述的 double encoder model。

截斷長句方法的比較

- 最大長度為15，保留前面: 0.50750 / 0.50237
- 最大長度為15，保留後面: 0.50750 / 0.50355
- 最大長度為20，保留前面: 0.50592 / 0.49920
- 最大長度為20，保留後面: 0.50671 / 0.50237
- 最大長度為25，保留前面: 0.50988 / 0.50434
- 最大長度為25，保留後面: 0.51185 / 0.50513

在訓練時，我們因為有時句子會太長，為了加快訓練速度而截斷一些句子，只保留句子的前半部或後半部，發現都較沒截斷稍微差一點點。而保留句子的後段結果看起來比保留前段好，推測是因為會被截斷的句子通常都是在問題句有多句接合的情況，而此時較為重要的訊息應該是接近接點(即問題句和答案句的連接處)的地方，因此保留後段較佳。但有些差距不太明顯，可能是實驗誤差。實驗模型為上述的 double encoder model。

BOW 和 encoder 的比較

- BOW: 0.50750 / 0.50079

- double encoder: 0.51264 / 0.50671

將 BOW 改成 encoder 之後效果較好，因為 BOW 不考慮詞序變換的問題。但是相對於 HW5，這次的差距並沒有那麼明顯，推測是因為 HW5 是判斷語意正面或負面，但這次是判斷接起來的語意是否通順。 因此在遇到「Q:今天天氣如何? A1:今天天氣熱，但是個好日子。 A2:今天是個好日子，但天氣熱。」這種問題時，詞序變換造成的語意分歧對結果的影響較不明顯。

關於 **ensemble** 的比較

- 只拿數個 mLSTM: 0.53438 / 0.55849
- mLSTM + BOW: 0.55889 / 0.56482
- mLSTM + BOW + double encoder: 0.55968 / 0.57312

從實驗結果來看， ensemble 在此次任務中，仍是非常實用的方法。

Conclusion (1%)

經由以上的實驗結果，以下是我們認為較佳的訓練方法:

- embedding 的維度越大越好(我們採用的維度是300)。
- 不截斷長句，而是將短句 padding 至所有句子中最長的長度。
- 正反樣本比例數應為 1:5。
- 訓練時應將多句相連的問題句接合在一起。
- 刪除選項中的引號。
- 使用多樣化的模型進行訓練，並將結果 ensemble。

Reference (1%)

Multiplicative LSTM for sequence modelling, https://arxiv.org/pdf/1609.07959.pdf

An efficient framework for learning sentence representations https://arxiv.org/pdf/1803.02893.pdf

Lecun uniform initializers http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

Adamax/Adam Optimizers https://arxiv.org/pdf/1412.6980v8.pdf

Software Framework for Topic Modelling with Large Corpora https://radimrehurek.com/gensim/lrec2010_final.pdf

Dropout: A Simple Way to Prevent Neural Networks from Overfitting http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Empirical Evaluation of Rectified Activations in Convolution Network https://arxiv.org/pdf/1505.00853.pdf

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift https://arxiv.org/pdf/1502.03167.pdf

Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks https://arxiv.org/pdf/1602.07868.pdf