Machine Learning (Fall, 2017)

Chinese QA Final Report

隊名

NTU_b04901066_Need4GPU

隊員-分工

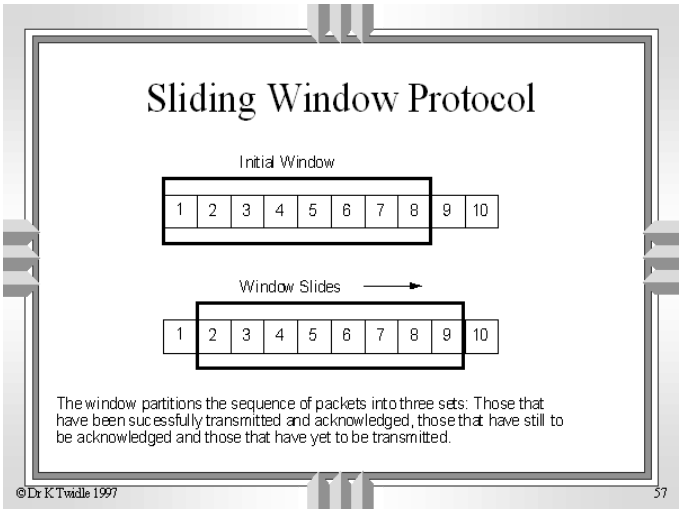
- B04901001 方致偉 - build R-NET model, run train & test
- B04901066 洪國嘵 - 隊長, build sliding window model
- B04901142 陳政曄 - report
- B04901146 黃禹傑 - report

ATTEMPT 1

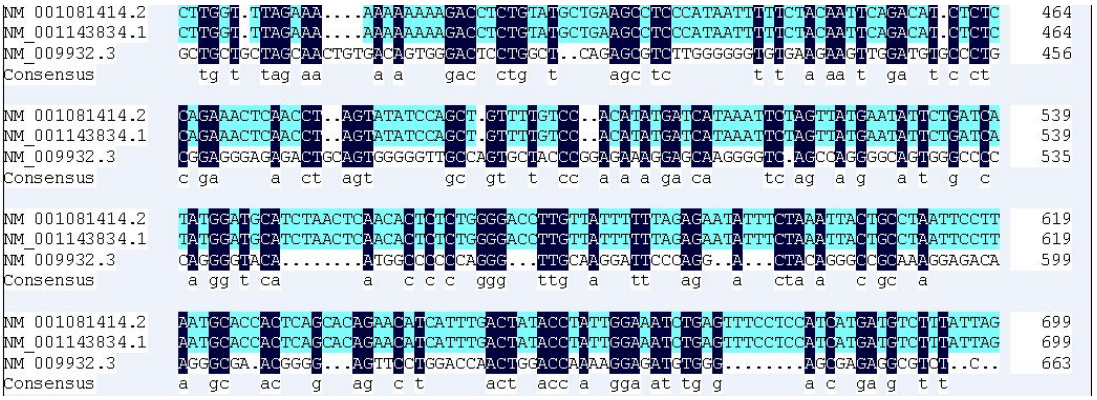
這個題目十分困難，model 要回答的並不是單純的對錯、分類或時序相依性輸入輸出問題，而是必須在給定非常長的題目和文章的情況下，指出正確答案的位置，就算是人類來作答，F1 score 也大約只有 0.91 分。

細思解題辦法，決定先模仿人類找出答案的方法：找出問題關鍵字，然後在文章中找到關鍵字出現的地方，再開始在附近找答案。因為正確答案通常十分鄰近這些關鍵字，因此這是十分有效且高準確率的方式。然而機器並不會自己找關鍵字，也無法直接理解文意來指出正確答案，此時就需要用到一些演算法。

因為學過電腦網路導論，知道 Transmission Control Protocol 的實作方式及其特色，因此我們嘗試使用 sliding window 來框出答案。實作方式十分簡單，一個長度為 n 的 sliding window 在 paragraph 中移動，每次會和問題比較出現的文字，計算文字相同數量當作 score，取 score 最高的 n 個字為答案，此方式實作之 kaggle public 分數約為 0.10。



圖片來源：
http://www.sourcecodefiles.com/wp-content/uploads/2014/11/Sliding-window-protocol.gif



圖片來源：http://liucheng.name/329/

而後，又想到生物學基因序列比對的方式，因此，將一對一的比較，改為使用 python 標準函式庫 difflib 中的 SequenceMatcher，他會盡量找到兩序列的最長連續匹配子序列，而 ratio() 這個 function 會回傳匹配後續列的匹配率，我們以此做為新的score，其餘部分維持和上一段文中的實作方式相同，透過調整 n，可以得到 0.11 ~ 0.136 的 kaggle public 分數。

最後，和系上同學討論時，發現助教的評分程式並不是只看序列的起始和結束位置，而是會去比對每個字來算分數，因此大家便想到將題目出現過的字刪除，有很高的機率就將不是答案的字刪掉，如此便又可提升正確率，以此方式實作之 kaggle public 分數可以來到約 0.18 ~ 0.195。

以上為我們嘗試以 rule base model 來解此問題的過程和結果。

ATTEMPT 2

Preprocessing / Feature Engineering

一開始提供的資料是沒有 validation set 的，因此我們需要從中切出一些作為 validation set。這份 dataset 的形式為，一個閱讀段落搭配數個不同的問題，因此可能的切法會有兩種，一種是將部分的閱讀段落及及其所有子問題作為 validation set，另一種則是每個段落取部分問題作為 validation set。如果想讓 model 在 train 的時候盡可能接觸到越多不同的文章，可以採取第二種作法，但就會產生 training set 與 validation set 有重複內容的疑慮，在 experiments and discussion 會實驗不同做法的差異。

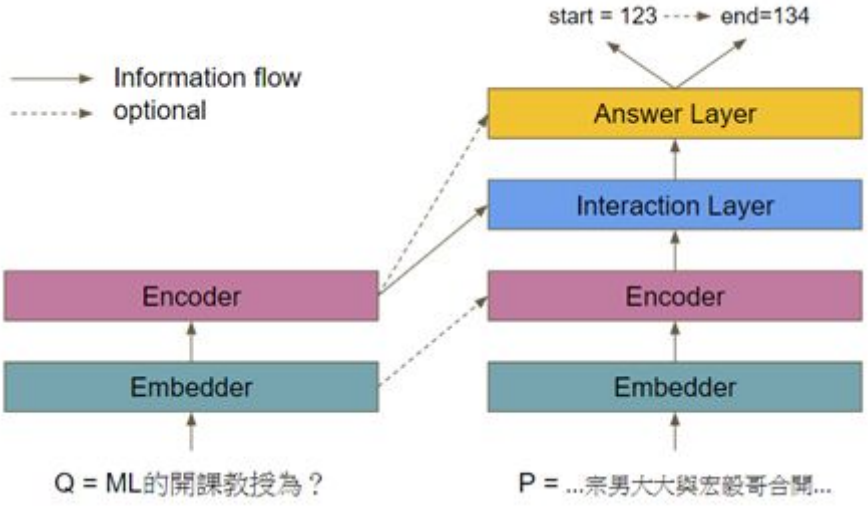
在資料的預處理上，我們使用 jieba 作為分詞用的 library，並且找到的一個 fork 出來的版本（ https://github.com/ldkrsi/jieba-zh_TW ），它將裡面的辭庫及 HMM 替換成了繁體中文版本，根據 fork 版本作者的實驗，其分詞之後與中研院中文斷詞系統的 edit distance，較以原版 jieba 來的好很多，因此我們採用這個修正過的 jieba 版本。

Edit distance	第一段 (92)	第二段 (136)	第三段 (75)	第四段 (52)	第五段 (63)
jieba zh_TW	9	20	12	12	9
jieba轉簡體後斷詞	44	43	31	23	21
jieba直接斷繁體字	53	74	43	34	34

（表格取自 https://github.com/ldkrsi/jieba-zh_TW ）

而在 word embedding 方面，我們使用了 GloVe 演算法做 training，其使用方法與 gensim 不太一樣，gensim 是作為 Python module import 進來跑，GloVe 是需要 compile 它的 C++ source code 之後，到 demo.sh 裡面修改 corpus 位置、參數等等，然後直接執行，就會 train 出一個 model file。

Model Description



我們實做的模型架構與投影片的基本架構類似，主要來自微軟的自然語言運算團隊的 R-Net 在 GitHub 上的非官方版本，實作所用的 framework 為 PyTorch。整個架構可以分成四大部分：embedding layer, encoding layer, attention layer, pointer layer。以下分四部分介紹：

1. embedding layer

首先將文章與問題的字句做embedding，我們使用jieba完成斷詞並用 GloVe 轉成向量（事先 train 好），並將 train 好的 weights 放入torch.nn.Embedding存取。在 paper 中 proposed 的 model，其分別將單字及字元做了 embedding，不過我們在這邊只取單字做 embedding，因為取個別字元做 embedding 過於耗時，而paper中也有提到去掉 character-level embedding 對 F1 score 只有 0.9% 的

影響。

2. encoding layer

Sentence encoding 我們用 bidirectional LSTM 的 RNN 來實作，目的是對文章與問題做前處理，將其中的詞語與文章、問題的關係建立出來。這個階段文章與問題是分開處理的，所以將文章與問題分開傳入兩個biderectional RNN 的模型，並將 output 整理成可以當成下一階段 attention layer 的格式。

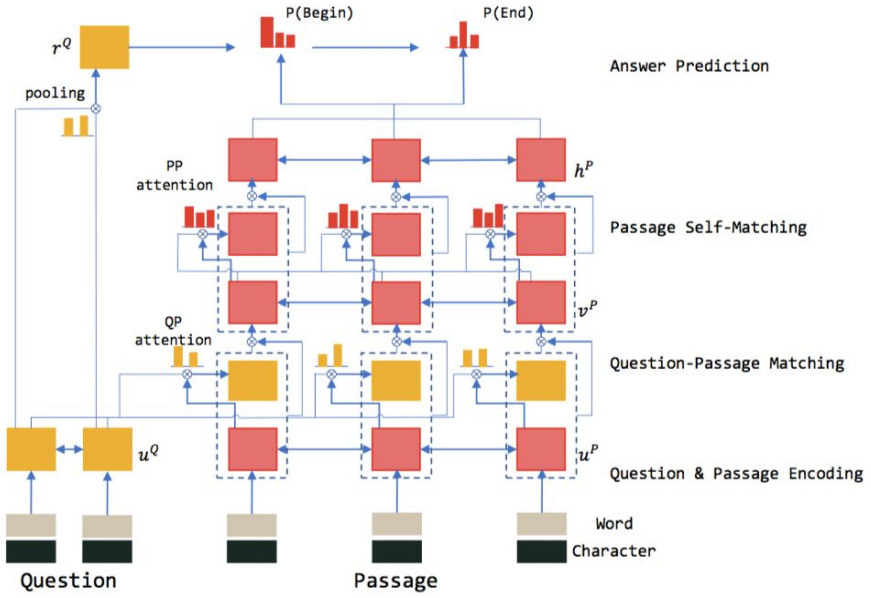
3. attention layer

這部份分成 pair-macthing 和 self-matching 兩部分。兩種 matching 都是利用 NN 去訓練出 match function 以產生 attention weights，最後輸進 GRU 的 RNN cell 產生經過 attention 的 encoded outputs。Pair-matching 是用問題當作 inputs，找出對應的文章中字句的 attention weight，以標出文章中可以回答道問題的部分；self-matching 則是用 pair-matching 的 output 當作 input 找出文章對應的 attention weight，這部分是作者認為 pair-matching 的結果涵蓋的文章資訊量太少，因此再對文章本身做一次 attention。

4. pointer layer

這部分用兩種模型來實作：sequence model, boundary model。Sequence model 會產生可能是答案的序列，boundry model 則標出文章中可能是答案的字句範圍。利用上一部份產生的 attention weight，透過 attention pooling 的方式，將 pair-matching 的結果與問題相比較，找出整篇文章中最可能是答案的範圍，並將開始與結束點標示出來。

我們可以引用一張圖來大略的表示這個 model

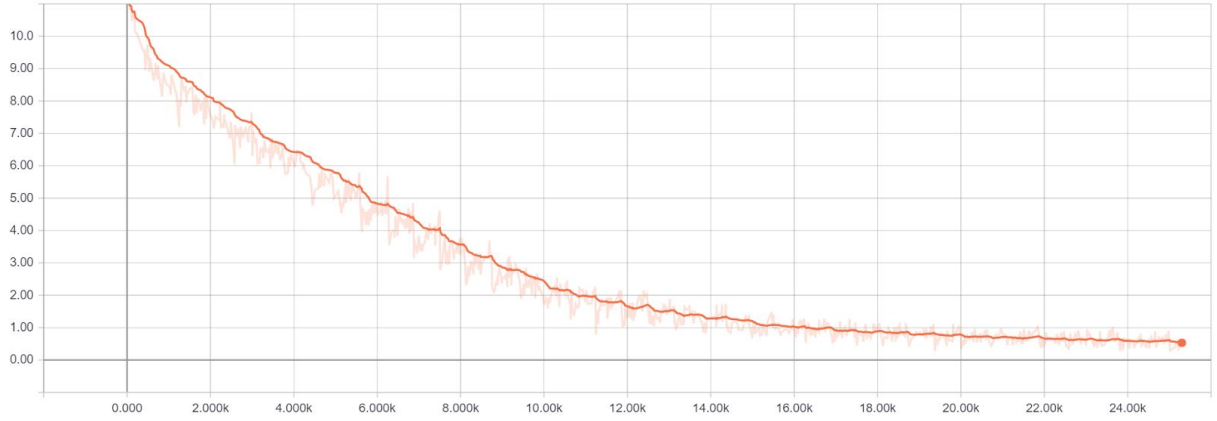


(Reference: Natural Language Computing Group, Microsoft Research Asia. R-NET: Machine Reading Comprehension with Self-matching Networks.)

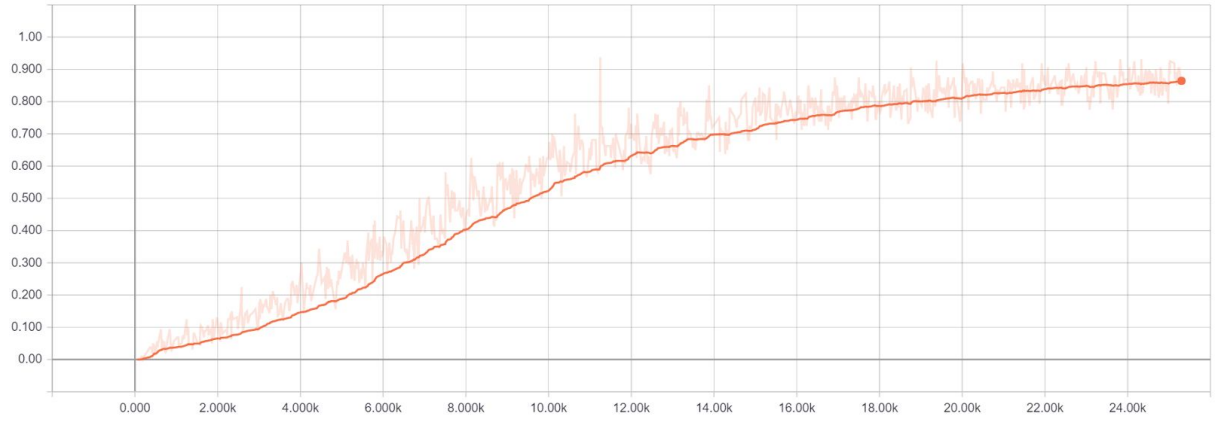
圖中，input 的 Passage 與 Question 即對應到此處的文章以及問題。如前文所述，character-level embedding 基於效率問題並無在此實做。在第一層的地方，對應到上述的第二點 (encodeing layer)，是將文章與問題分別丟入兩個 biderectional recurrent network。接著在圖中的第二層和第三層，對應到上述第三點 (attention layer)，在第二層中，將第一層的輸出整合丟入 gated attention-based recurrent network，得到輸出後，接著再將其丟進第三層，進行圖中所謂的 passage self-matching。最後再經過 pooling layer 得到結果。

Experiments and Discussion

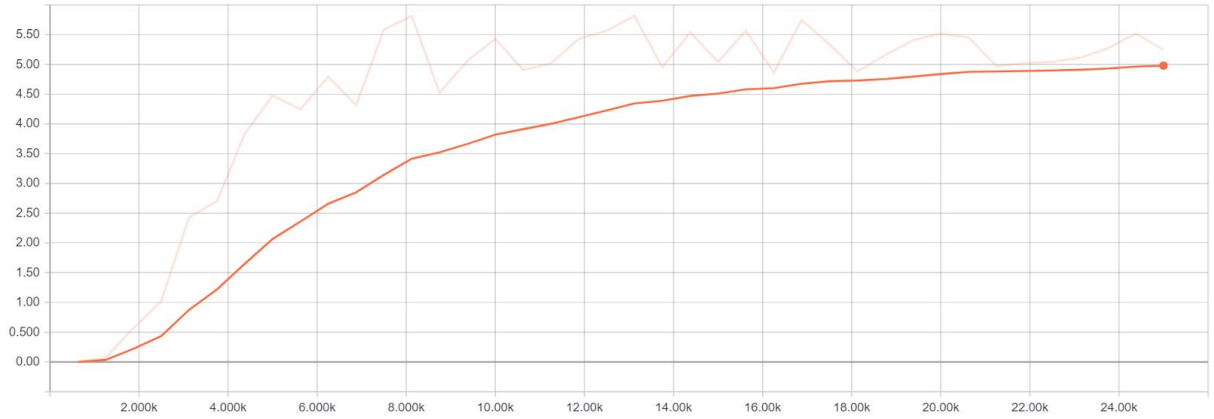
我們的 training 使用 cross entropy 作為 loss function，optimizer 使用與 paper proposed 相同的 Adadelta (rho=0.95, eps=1e-6)。以下為在 training / validation set 上面畫出的 learning curve。



Training set, Loss



Training set, Exact Match score (0-1.0)



Validation set, Exact Match score (0-100)

從以上曲線可以看出，我們的 model overfitting 狀況嚴重，推測原因是這次的 dataset 較小（SQuAD 有 100000 個問題 v.s. Delta 有 15000 個問題）。

以下表格比較單純用 sliding window、R-net，以及兩者合併（R-net 有時會有 begin<end 的輸出，導致答案為空白，因此用 sliding window 補進答案）的表現：

Model	Performance (Kaggle F1 score)
Sliding window (window size = 30)	0.19450
R-net	0.22858
Combined (window size = 30)	0.24083

Reference

An unofficial implementation of R-net in PyTorch.
(https://github.com/matthew-z/R-net)

Natural Language Computing Group, Microsoft Research Asia. R_NET: Machine Reading Comprehension with Self-matching Networks.
(https://www.microsoft.com/en-us/research/publication/mrc/)

Slides in class.

Machine Comprehension Using Match-LSTM and Answer Pointer
(https://openreview.net/forum?id=B1-q5Pqxl)

Gated Self-Matching Networks for Reading Comprehension and Question Answering
(http://www.aclweb.org/anthology/P17-1018)