國立臺灣大學電機資訊學院電機工程學系

106 上 機器學習 期末專案

Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Machine Learning 2017 FALL

邁向中文問答之路

Chinese QA

NTU_r05546016_宏駒團隊

林冠廷 Kuan-Ting, Lin

洪唯凱 Wei-Kai, Hung

洪紹綺 Shao-Chi, Hung

張少豪 Shao-Hao, Chang

指導教授：李宏毅 老師

Advisor: Hung-Yi Lee, Ph.D.

中華民國 107 年 1 月

January 2018

Section 1 Team Work Division

林冠廷 R05546016
組長

大學就讀政大應用數學系的冠廷，擅長數學建模求解，我們信任數學系四年來精實的邏輯訓練，因此推舉他當組長。並將較複雜的多維度詞向量類神經網路模型交給他建置。

洪唯凱 R04546031
爬蟲 Pro

擅長搜尋資料與統整資訊的唯凱，這次為中文問答期末專案，下遍各式各樣 Google 關鍵字，網羅所有可能適用的模型。並負責建置字典，將 Input Data 逐字轉換為其對應的 key，後續嘗試與 Attention Model 接軌。

張少豪 R05546038
抵霸葛 Pro

喜歡 Debug 的少豪負責延續助教提供的 Sample Code 完整的加入所有更符合需求的資料前處理。包含段落、問句筆數的對齊以及資料陣列長度補零，同時負責處理多維度 Word2Vec 詞向量，後續嘗試與模型接軌。

洪紹綺 R05546014
暴力解 Pro

喜歡從不同角度看問題的紹綺，決定嘗試以深度學習以外的方法求解。多方嘗試後，發現以問題做為索引回去段落抓答案便可達到 Strong Baseline 的標準。後續負責斷詞建字典的工程，以及架構可供訓練的類神經網路模型。

Section 2 Preprocessing and Feature Engineering

2.1. Jieba 斷詞

Jieba 是目前表現最好的 Python 中文分詞套件，因此使用其作為 RNN model input 資料前處理的斷詞工具。因為 jieba 預設主要是簡體字，所以加入'dict.txt.big.txt' 增加字典內容讓繁體字的分詞更好，同時選擇適合文本分析的精確模式將文句做較精確的斷詞。另根據實作 word2vec 的結果，保留標點符號會有較好的 performance。

2.2. Gensim - Word2Vec

本次 word2vec 的套件選擇使用 gensim。參數設定方面，sg=1 表示使用了 skip-gram 而非 cbow，這是因為 bag of words 會忽略掉字詞的前後順序，在文意判斷上會有不顯著的情況；window=5 表示對一個詞會往左右看五個字，由於答案的長度多半不會太長，因此這邊維持預設值；min_count=1 是為了防止出現頻率較少的詞被忽略，因為答案的詞不一定會出現很多次，在資料筆數沒有相當大的情況下，仍然設定為 1。

2.3. Answer - One hot encoding

為了方便 model 的 output 能讀取訓練資料集，將每個答案字詞的位置做 one hot encoding，亦即，在斷詞後將答案字詞相對於整段文章詞數的 start 及 end 位置轉換成 1*最大文章長度的矩陣，這樣也方便 predict 完後轉回我們要的答案位置。

2.4. 資料除錯化

由於經過計算後發現有問題字數過長，因此透過統計後找出 outlier，發現有部分題目及答案格式錯誤，在前處理部分預先去除。

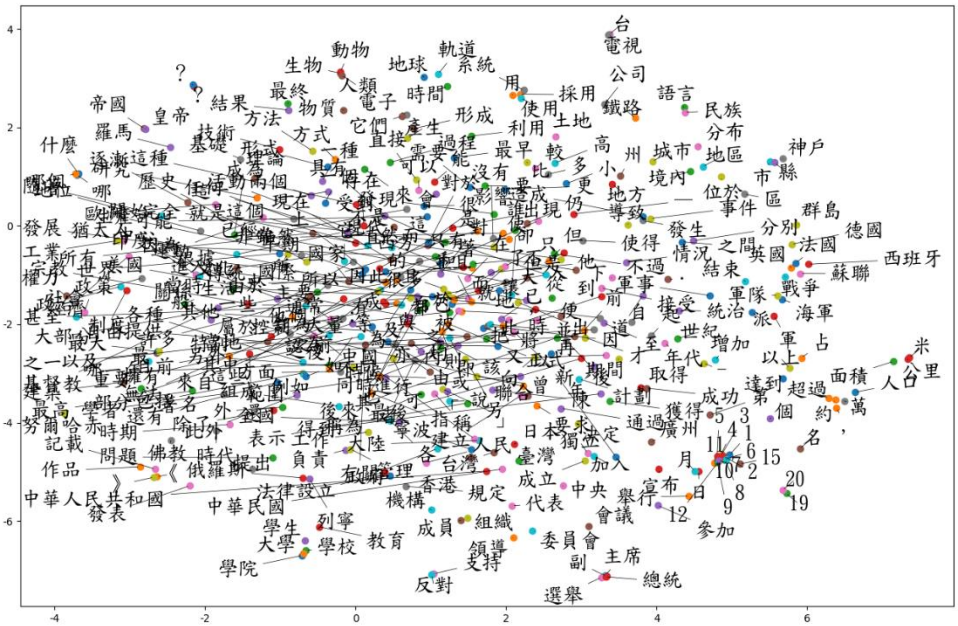


Figure 1. embedding 視覺化 (min_count=1000)

Section 3 Model Description

3.1. 本文特徵提取演算法

雖然類神經網路可以解決很多不同類型的問題，但由於無法拆解其學習過程的緣故而為人詬病，因此除了建構類神經模型外我們也嘗試了以一般演算法的概念來求解此問題。首先，我們先嘗試直接輸出整篇文章作為解答，得到 f1 score 為 0.023，觀察 simple baseline 為 0.107，相差不遠。

秉持「答案就在問題附近」這個原則，使用 sklearn 套件裡的 TfidVectorizer 對文章進行資料前處理，提取關鍵字後以問題作為索引，回去段落內抓取句子，再輸出整句作為解答，獲得 f1 score 高達 0.20584。

有趣的是，之前在作業包含標點符號的訓練模型，可以協助機器學習到不同的情緒，但在嘗試不同模型的過程中，我們發現對於此種作法，包含標點符號的關鍵字提取方式會獲得較差的表現。

由於對本問題做深度學習需要用到 attention 或是透過 sliding window 去降低文章長度等處理，相較之下特徵提取的作法是我們最快取得成效的方法，根據問題的特性尋找針對性的演算法效果可見一斑。

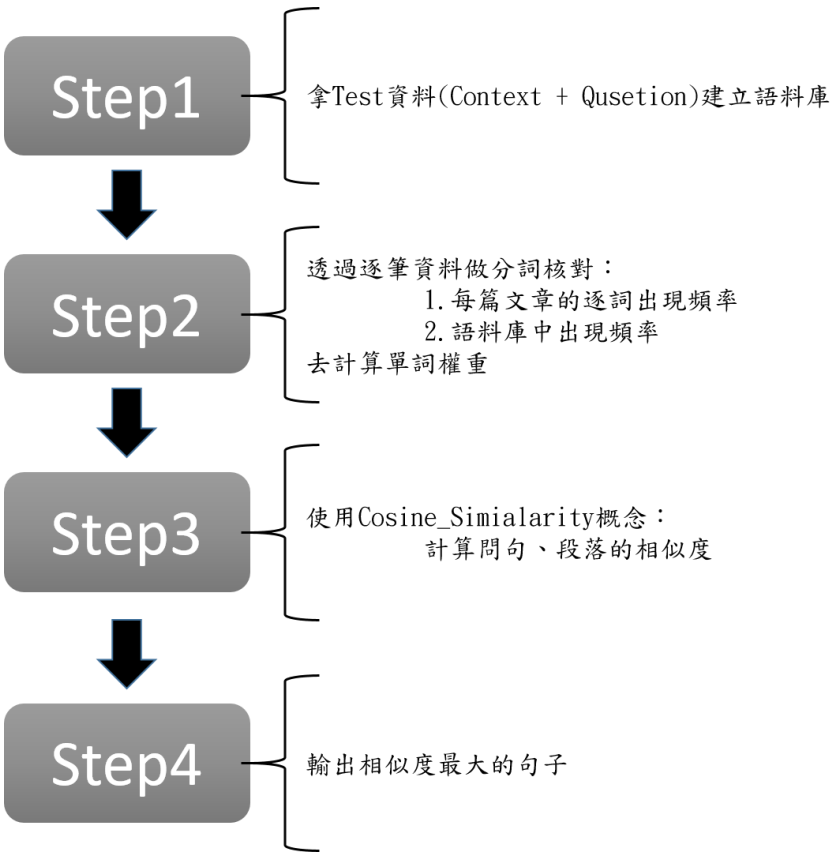


Figure 2. 本文特徵提取流程

3.2. RNN & LSTM

本次 QA 問題我們同時嘗試以 recurrent neural networks 求解，整體概念是先建置兩個 LSTM 的 model 分別對文章內容以及問題做訓練，再透過相乘的 merge 結合兩個 model 的 output 繼續訓練問題的答案。

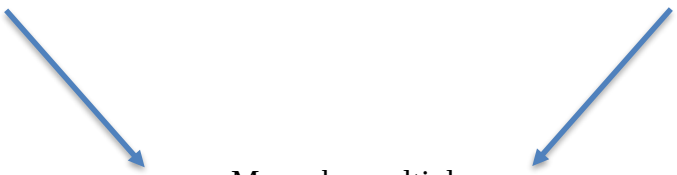
經過第二章節的前處理後，我們可以將 padding 過的文章以及問題向量分別丟入模型中，這邊由於一篇文章對應多個問題，因此我們也將文章複製到與問題相同的數量方便對應。至於 output 的部分則採用前面提到的 one hot encoding 方便轉換。model 前部分的 dropout 的值設定為 0.5 避免 overfitting，後半部則嘗試為 0.2。optimizer 的部分則選擇過去較為常用的 adam。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 256)               365568
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
=================================================================
Total params: 497,152
Trainable params: 497,152
Non-trainable params: 0
_________________________________________________________________
```

LSTM model of context

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 256)               365568
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               65792
=================================================================
Total params: 497,152
Trainable params: 497,152
Non-trainable params: 0
_________________________________________________________________
```

LSTM model of question



Merge by multiply

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
merge_1 (Merge)              (None, 256)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 645)               330885
=================================================================
Total params: 1,456,773
Trainable params: 1,456,773
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. RNN Model

Section 4 Experiments and Discussion

4.1. 本文特徵提取模型討論

資料前處理使用 sklearn 套件的本文特徵提取，先以 fit 函式建置包含段落與問題的語料庫，再以 TfidVectorizer 分析單詞。

TfidVectorizer 合併了 CountVectorizer 與 TfidTransformer 的功能計算單詞權重，以免被過於常見的語助詞、冠詞如的、和、了.....等單詞誤導模型權重。CountVectorizer 方法只會統計詞彙出現次數，例如「了」這個字出現很多次，但對訓練模型而言並非十分具有實際意義，因此使用 TfidVectorizer 方法。

TF-IDF 統計方法主要用以評估字詞對於段落或檔案的重要程度，其主要概念為：字詞的重要性與檔案中出現的次數呈正比增加，但隨著在語料庫中出現的頻率呈反比下降。此方法建立在 CountVectorizer 基礎上結合整個語料庫，考慮單詞權重，而非單純考慮單詞出現次數。

最初提取本文特徵時，以 word n-gram 的方式分詞，f1 score 僅達 0.03，改以 char-n-gram 方式分詞，f1 score 達 0.14，通過 simple baseline。觀察答案與問句後發現，以此方式輸出句子，極為容易將問句提到的單詞一起輸出，因此嘗試將答案去掉問句出現過的單詞，f1 score 達 0.2415，一舉突破 strong baseline。

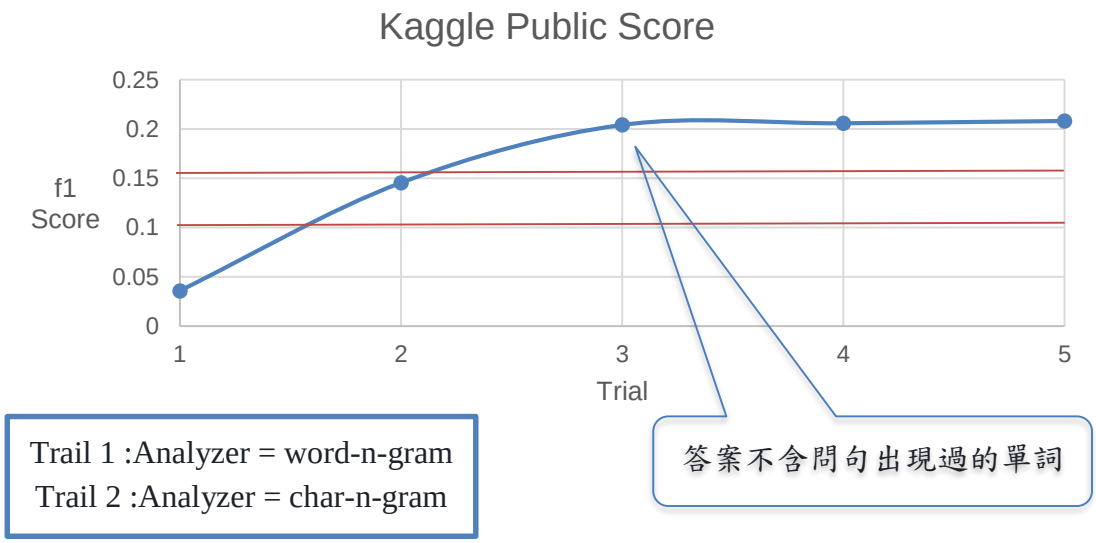


Figure 3. Kaggle f1 score

4.2. Recurrent Neural Network 模型 討論

在資料前處理的部份曾嘗試「切字」以及「切詞」兩種方式。

最初使用「切字」的目的是想要先快速測試架構的模型是否可以執行，並認為機器可以透過複雜的類神經網路，學習到開頭字和結尾字，但效果並不如預期。經過組員互相討論爭辯，同時上網查找不同資料後，意識到如果採用「切字」的方式訓練模型，機器只會學習到所有碎字之間的關聯，不會了解字與詞、詞與詞的關係，舉「黃河」兩字為例，在這種作法中會被切為「黃」與「河」，但電腦不會了解到「黃河」代表的意義。這也正是 Embedding 層存在的理由，但我們一開始並沒有在模型中架設 Embedding 層。

於是最後決定使用「切詞」的方式訓練模型，但是這時候會出現一個問題，例如：答案「香港國際機場」，但答案抓的 index，會因為 jieba 切詞，而被切成「香港」和「國際機場」，所以答案可能只會是兩種其中一種，雖然可以解決不用找出開始和結尾的麻煩，同時可以有效建置 Word2Vec，但是最後出來的結果也不盡人意。

對於如何架構 Attention 模型及其背後理論，我們還是沒有很熟悉，要如何教機器去學習哪一個字詞是比較重要的，並給予權重上的分配，還有後續如何將訓練完的權重丟回模型訓練。

4.3. 專案 Chinese QA 問題討論

在看過很多網站的介紹後發現很多類似模型，都是在訓練預測英文的問答，甚至是圖片問答，但是英文字母只有 26 個字，分詞也較為容易，中文的字詞卻有好多種組合，訓練也嘗試了不同的斷詞方式建置詞向量庫，可以看出精確模式與搜尋模式斷詞效果較好，但是套用至類神經網路模型，執行的表現都沒有很好。

Jieba 斷詞結果	斷詞方式
['廣州 / 的 / 快速 / 公交 / 運輸系統 / 每 / 多久 / 就 / 會 / 有 / 一輛 / 巴士 / ？']	精確模式 可以準確分出文章中的名詞。
['廣州 / 的 / 快速 / 公交 / 交運 / 運輸 / 運輸系統 / 系統 / 每 / 多久 / 就 / 會 / 有 / 一輛 / 巴士 / / ']	全模式 為了保留完整字串，多出「交運」此類無義字詞。
['廣州 / 的 / 快速 / 公交 / 運輸 / 系統 / 運輸系統 / 每 / 多久 / 就 / 會 / 有 / 一輛 / 巴士 / ？']	搜尋模式 可以準確分出文章中的名詞，並額外斷出「運輸」、「系統」以及「運輸系統」。

對於拆解機器如何學習的過程，本組尚有大量理論需要加強補充，除了 Attention 的概念以外，還有網路常見的 Pointer Netwoek 架構，另外 R-Net 的實作，都是未來可以繼續研究並嘗試實作的方向。

References

自然语言处理中的 Attention Model：是什么及为什么
http://blog.csdn.net/malefactor/article/details/50550211

Attention-based Recurrent Neural Networks for Question Answering
https://web.stanford.edu/class/cs224n/reports/2761224.pdf

Pointer Networks
https://papers.nips.cc/paper/5866-pointer-networks.pdf

Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models
https://explosion.ai/blog/deep-learning-formula-nlp

浅谈 Attention-based Model【原理篇】
http://blog.csdn.net/wuzqChom/article/details/75792501

The Stanford Question Answering Dataset
https://rajpurkar.github.io/SQuAD-explorer/