Delta Chinese QA 邁向中文問答之路

隊名 : NTU_r05922037_AI 人工智障

組員 :r05922037 資工所碩二 邱德旺

r05922042 資工所碩二 吳肇中

r05944018 網媒所碩二 賴至得

一、分工(1) :

1.邱德旺:

(1) Try & Implemant “conversations in TV shows” model
(2) Write report

2.吳肇中:

(1) Try & Implemant “Listen and Translate” model
(2) Write report

3.賴至得:

(1) Try & Implement “Delta Chinese QA 邁向中文問答之路” model
(2) Write report

二、Preprocessing/Feature Engineering(3):

1.中文分詞:

使用 jieba[7]進行中文分詞，在這次 project 中，我們分別使用 jieba 預設的字典和官網中附有的支持繁體中文的字典進行分詞，然後分別訓練出不同的模型。

2. Part-of-Speech (POS) tagging:

我們嘗試給予模型更多資訊，企圖提升預測的結果。其中，最成功的方式為加入每個詞彙的詞性資訊。我們使用 jieba 的詞性標注功能找出每個詞彙的詞性。加入詞性的方式為在每個字的 word embedding 之後額外再加入一個代表詞性的 POS embedding。

3. Pre-trained embedding:

(1) Word embedding:

使用分詞後的所有中文句子訓練 word embedding。我們選用 Gensim[8]中的 word2vec 作為訓練模型，並調整訓練參數以得到多樣的 word embedding。

(2) Character embedding:

由於我們大部分的 word embedding 沒有包含所有的詞彙，需要藉由 character embedding 處理 out-of-vocabulary (OOV)的狀況，因此，我們將所有中文字分開後訓練中文字的 character embedding。我們仍是選用 Gensim 的 word2vec 作為訓練模型。

(3) Part-of-Speech (POS) embedding:

將所有中文句子使用 POS tagging 後，利用結果訓練 POS embedding，用意是找出良好的 representation 來代表每個詞性彼此的關係，我們仍是使用 Gensim 的 word2vec 作為訓練模型。

4. 設定 word-level answer

因為提供的 training data 中的 answer start 表示文章中代表 answer 起始的字的位置。不過由於我們的模型是以詞做為單位進行預測，因此需要進行轉換，具體而言，我們將 answer 第一個字所在的詞的位置做為 answer start，answer 的最後一個字所在的詞的位置做為 answer end。

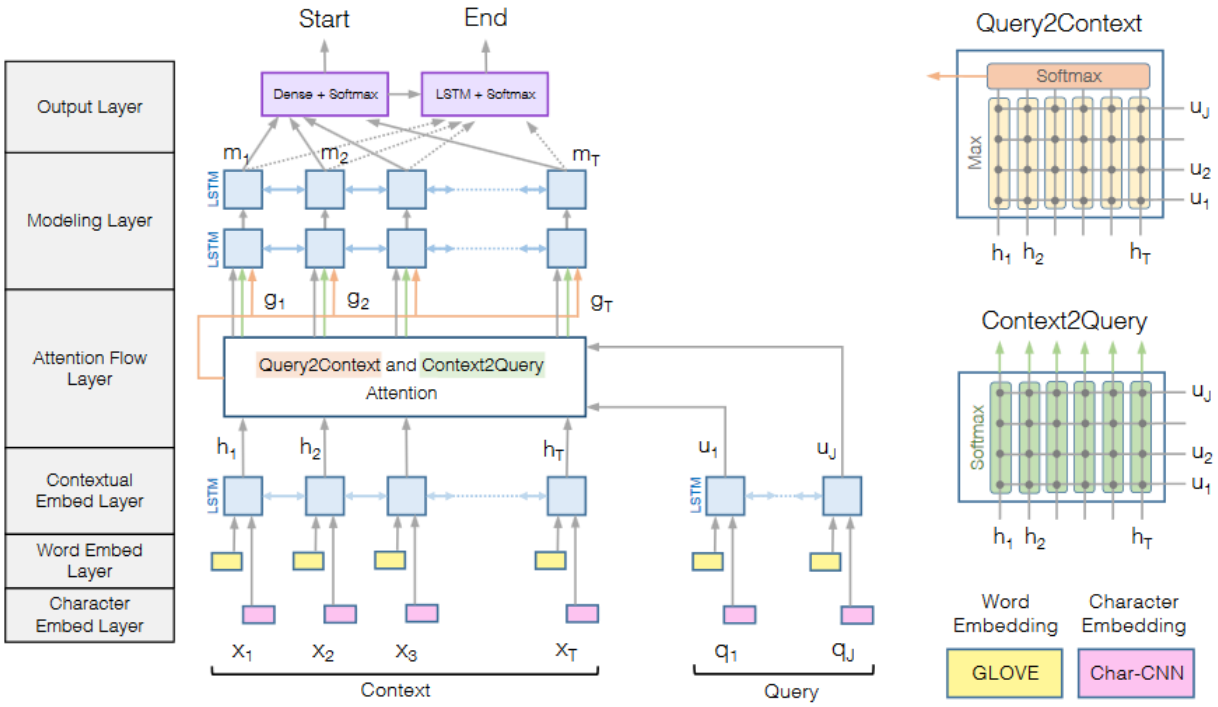
5. 篩選 training data

由於此次 project 中文章和問題的字數多，為了節省記憶體的使用，我們首先設定不同的文章長度和問題長度上限，並且將超過上限的字捨棄。但是上述作法可能會導致模型學習在不完整的資料上，因此，我們後來採取篩選 training data 的方式，將超過文章長度上限或問題長度上限的 training data 去除。經過統計僅有 3 筆 testing data 的文章長度超過 400，11 筆 testing data 的問題長度超過 40，移除影響應該不大，所以對於有使用篩選 training data 的模型，我們都將文章長度上限設為 400，問題長度上限設為 40。

三、Model Description (At least two different models) (7):

1. Bidirectional Attention Flow(BIDAF)[1]:

(1) 模型概觀圖:



(2) BIDAF 各層介紹:

(a) Character Embedding Layer:

利用 character-levelCNN[2]來將 input 的文字轉換成一個 vector

(b) Word Embedding Layer:

直接將 input 的文字 map 到 pre-train 的 word embedding，得到對應的 vector(這裡使用的是 GloVe pre-train embedding)

(c) Contextual Embedding Layer:

這裡用一個 bidirectional LSTM 來將前兩層得到的 embedding 作為 input，並將輸出的兩個方向的 LSTM concate 起來作為 output

(d) Attention Flow Layer:

這層使用雙向的 attention mechanism:用 query attend context(Query-to-context Attention Component) 與用 context attend query(Context-to-query Attention Component)

最後將 context vector 還有兩個方向的 attention vector 通過 Beta function 的轉換，如下:

$$\mathbf{G}_{:t} = \beta(\mathbf{H}_{:t}, \tilde{\mathbf{U}}_{:t}, \tilde{\mathbf{H}}_{:t}) \in \mathbb{R}^{d_{\mathbf{G}}}$$

這裡的 Beta function 可以是任何的 function，而這個 model 用的是 simple concatenation 來作為 Beta function

(e) Modeling Layer:

將上一層得到的 G vector 作為 input，放到一個兩層的 bidirectional LSTM 裡得到 output

(f) Output Layer:

由於 QA task 是要輸出答案位於 context 裡的開頭與結尾位置，所以這個 model 分別用不同的結構來輸出開頭與結尾位置:

(i) 開頭位置:

開頭位置產生的方式是用以下公式，產生位置的機率分布:

$$\mathbf{p}^1 = \text{softmax}(\mathbf{w}_{(\mathbf{p}^1)}^\top [\mathbf{G}; \mathbf{M}])$$

(ii) 結尾位置:

結尾位置則是先將上一層的輸出 M 通過一個 bidirectional LSTM 得到 M2 在放到以下公式，產生位置的機率分布:

$$\mathbf{p}^2 = \text{softmax}(\mathbf{w}_{(\mathbf{p}^2)}^\top [\mathbf{G}; \mathbf{M}^2])$$

(g) 小細節:

(i) 前三層(Character Embedding Layer, Word Embedding Layer, 以及 Contextual Embedding Layer)是對 context 跟 query 都會做的

(ii) 前兩層(Character Embedding Layer, Word Embedding Layer)的 output 在輸入 Contextual Embedding Layer 之前，有通過一個兩層的 Highway Network[3]，再將這個 Highway Network 的輸出作為 Contextual Embedding Layer 的輸入

(iii) 在實作 Word Embedding Layer 和 Character Embedding Layer 時，我們採用 pre-trained embedding 並且在訓練 BIDAF 時就固定 embedding 的參數，原因是我們認為 pre-trained embedding 可以表現比較好而且可以節省訓練 BIDAF 的時間。

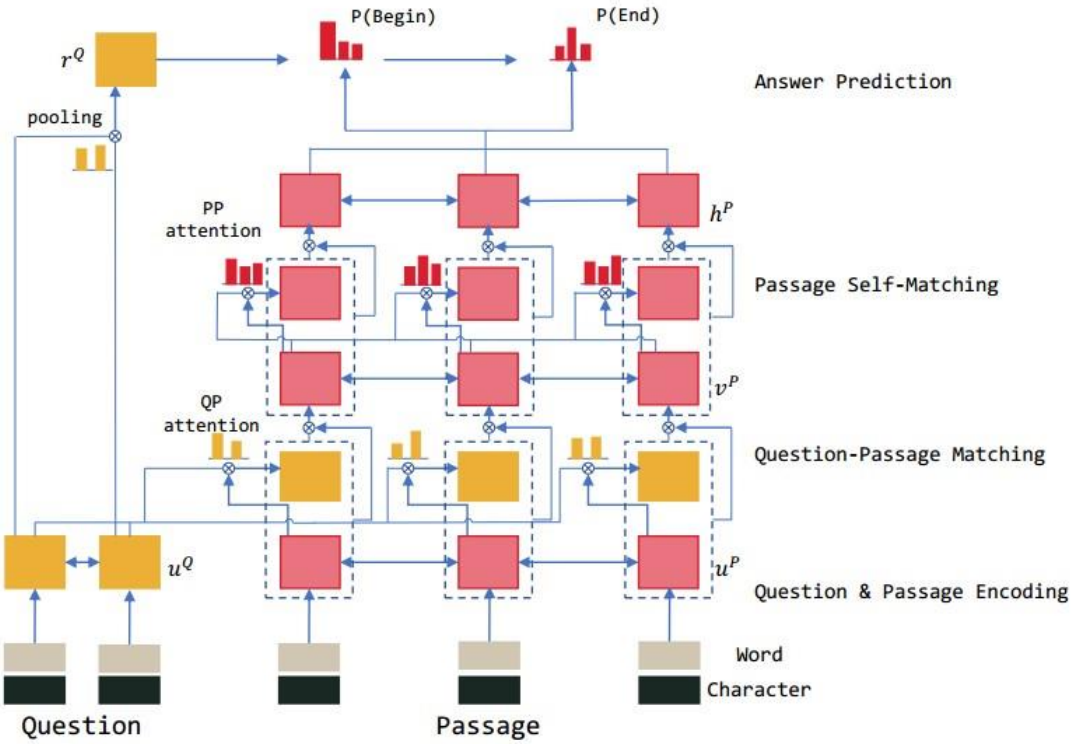
(3) Loss function 定義:

model 所定義的 loss function 是以開頭位置以及結尾位置的機率分布來做定義，定義如下:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N} \log(\mathbf{p}_{y_i^1}^1) + \log(\mathbf{p}_{y_i^2}^2)$$

2. R-Net [4]:

(1) 模型概圖:



(2) 介紹:

(a) Q and P encoding:

這是將 question Q, 跟對硬的 Passage encode 成 representation 的過程，是分別使用 character 跟 word level 的 Q 和 P 作為輸入 通過一層 bidrectional GRU 得到一個能夠與 context 相關的 word representation.

(b)Q and P Matching:

然後將 P 的 encoding, 通過一個 Gated attention-based RNN, 與 Q 做 attention 得到 與 Q 相關的 word representation。這個 RNN 的詳細作法是在 attention RNN 的輸出上在學一個門閥值，與 RNN 輸入做運算，當作第一步的 matching v^p. 這個 gated attention RNN 的原理:

$$v_t^P = \mathrm{RNN}(v_{t-1}^P, [u_t^P, c_t]) \quad \text{.....(1)}$$

此 RNN[5] 可以額外輸入值 attention, 此值是將 RNN 的輸入[u^p, c_t] 透過一個 gated function 學到一個能夠表達 passage 中各部位 attention 的值 gt, 在透過 gt 得到一個可以放入(1) 式中的[u^p,c_t]作為 attention RNN input.

$$g_t = \text{sigmoid}(W_g[u_t^P, c_t])$$
$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t] \quad \text{.....(2)}$$

(c)Self-Matching:

接著的部分其實是針對這個 representation v^P 做再改善，來捕捉 long term dependency. 實際做法是再使用一層 gated attention RNN

$$h_t^P = \text{BiRNN}(h_{t-1}^P, [v_t^P, c_t]) \quad ,$$

不過此時的 c_t 是指使用 Passage v^p 對自身做 attention pooling 得到的 attention vector, 所以在得到 h^p 之前的 [v^p,c_t] 也是經過 此 c_t 與 v^p 通過 gated function 得到的 representation, 來當作 RNN 的 addtional input.

(d) prediction , 即通過 pointer network 來預測開頭跟結束位置

(3) 總結：可以將 R-net 理解為 同時使用 word, char embedding 作為輸入，並使用兩層 gated attention RNN 來做 文章的 representation,這個 representation 實際是由 從問題對到文章的理解 (在 Q 跟 P 之間的 matching)、文章自身對文章的理解(利用前一層對 QP 的理解，再對文章本身做 attention) 得到。

四、Experiments and Discussion (8):

1. Introduction:

我們將介紹我們的 Experiments and Discussion。我們將 Experiments 分成 Embedding, BIDAF, R-net，分別介紹不同模型的設定以及我們有興趣的實驗和比較。在 Discussion 中，我們談論我們在這個 final project 中的發現、想法和心得，包括成功和失敗的嘗試，那些 feature 和 model 是有用的，以及我們推測那些 feature 和 model 為什麼會有用。

2. Embedding

在這次的 project 中，我們使用了在 preprocessing 所提到的 embedding。具體的參數如表 1。在表 1 中，type 表示 embedding 為 word embedding, character embedding, 或是 POS embedding， size

表示 embedding size, dictionary 表示輸入的資料經由 jieba 的何種字典分詞過，model 表示訓練 embedding 的模型。

Name	Type	Size	Dictionary	Model	Window	Min. count
w1	word	256	Default	Skip-gram	9	5
w2	word	128	Default	Skip-gram	21	3
w3	word	128	dict.txt.big.txt	Skip-gram	21	3
c1	char	256		Skip-gram	15	1
c2	char	128		Skip-gram	31	1
p1	POS	32	dict.txt.big.txt	Skip-gram	21	1

表 1: All Embedding

3. Bidirectional Attention Flow(BIDAF):

(1) 實驗概述:

我們參考論文使用 keras 實作 BIDAF model，並且我們使用不同的參數訓練不同的 BIDAF 模型，參數包括文章長度，問題長度，詞彙長度等等。此外，由於課堂中介紹的 question answering model 的 attention 僅有 question-to-context attention，BIDAF 同時具有 Query2Context 和 Context2Query attention，為了確認 Context2Query attention 是否真的有用，我們額外實作一個 simple BIDAF model，去除 Context2Query attention。最後，透過實驗發現，加入 POS embedding 能有效增強 BIDAF 的預測能力。

(2) 有無 Context2Query 對結果的影響

我們實作 simple BIDAF，去除 Context2Query attention 以及其他與之相關的部分，與 BIDAF 進行比較，結果如表 2 所示。表 2 中，word F1 score 表示使用模型 word-level 的 prediction 與 word-level answer 進行 F1 score 的計算。由表 2 可以看出，BIDAF 在 training 和 testing data 上皆比 simple BIDAF 表現更好，表示 Context2Query attention 確實提供模型更多的資訊，增進模型的能力。另外，值得一提的是，simple BIDAF 為我們在 Kaggle 上第一個超過 simple baseline 的模型。

Model	Word F1 score	Training F1 score	Testing F1 score
Simple BIDAF	0.36422	0.35899	0.14752
BIDAF	0.44844	0.43738	0.19679

表 2: Simple BIDAF 與 BIDAF 比較

(3) 有無 POS embedding 對結果的影響

由於人類在做閱讀測驗時，時常需要將文句切斷並分析段落中詞彙的詞性以利閱讀，或許詞性也能夠幫助機器學習 question answering。基於這個猜想，我們將每個字的 POS embedding 和 word embedding 連接起來作為新的 word embedding 並作為 BIDAF 的 embedding layer，我們稱這個模型為 POS BIDAF。POS BIDAF 和 BIDAF 的比較如表 3。由表 3 可以看出，加入 POS embedding 大幅增加 BIDAF 的表現，顯示出詞性在這個 task 上是個很強力的 feature，幫助模型對於文章和問題的意義有更明確的理解。在這次 project 中，POS BIDAF 為我們在 Kaggle 上 public score 最高的模型。

Model	Word F1 score	Training F1 score	Testing F1 score
BIDAF	0.44844	0.43738	0.19679
POS BIDAF	0.85816	0.85302	0.57636

表 3: BIDAF 與 POS BIDAF 比較

(4) 結果總覽

表 4 記錄所有 BIDAF 的模型實驗結果。表 4 中，embed 表示模型所使用的 embedding，p 表示文章長度，q 表示問題長度，w 表示詞彙長度，h 表示 contextual embedding layer 的 hidden size, m 表示 model layer 的 hidden size。Word F1 表示 word-level prediction 和 answer 的 F1 score

ID	Model	Embed	p	q	w	h	m	Word F1	Train F1	Test F1
1	Simple BIDAF	w1, c1	250	15	3	128	64	0.36090	0.35096	0.16175
2	Simple BIDAF	w1, c1	400	40	3	128	64	0.36422	0.35899	0.14752
3	BIDAF	w1, c1	400	30	3	128	64	0.68338	0.66481	0.24936
4	BIDAF	w1, c1	400	40	3	128	64	0.44844	0.43738	0.19679
5	**POS BIDAF**	**w3,c2,p1**	**400**	**40**	**5**	**64**	**64**	**0.85816**	**0.83188**	**0.57636**

表 4: All BIDAF results

4. R-net:

(1) 實驗概述:

我們使用 Github 上公開的 R-net implementation[6]，並且使用不同的參數訓練不同的 R-net 模型，參數主要有文章長度，問題長度，詞彙長度，embedding size, question-passing matching 和 self-passing matching 的 attention size，question-passing matching 和 answer prediction 的層數。除此之外，我們嘗試課堂中提到的限制長度的做法，設一個長度上限，並觀察結果的變化。最後，我們也嘗試使用不同的 pre-trained word embedding，進一步提升 R-net 的結果。

(2) 有無預測長度上限對結果的影響:

在課堂中有提過設定預測長度上限，而我們認為或許會有幫助，原因在於，直接輸出 R-net 的預測可以發現許多答案長度都非常長，但是，經過統計，有 13810 筆 (95%) training data 的答案長度小於 5 個詞，表示 testing data 中超過 5 個詞的答案應該不多。因此，我們設定預測長度上限為 5，比較結果如表 5 所示，由表 5 可以看出設定長度上限後些微提升模型的表現，和我們的預想類似。

Model	Word F1 score	Training F1 score	Testing F1 score
R-net	0.33714	0.33504	0.27477
R-net (pred<5)	0.33987	0.33776	0.27577

表 5: 長度上限對 R-net 的影響

(3) word embedding 對結果的影響:

在 project 一開始並不知道 jieba 預設是基於簡體中文分詞，後來在作業中知道 jieba 可以使用預設的字典以更好的分詞繁體中文。因此我們想要比較有無繁體字典對於模型的表現是否有影響，因此我們使用不同的字典進行分詞，訓練 word embedding，然後作為 R-net 的 input 訓練出不同的模型，比較結果如表 6 所示。由表 6 可以發現，使用支持繁體中文的字典分詞確實更加良好的掌握了文章和問題的語意，進而增加預測的 F1 score。

Model	Embed	Word F1	Training F1	Testing F1
R-net (pred<5)	w2, c2	0.29247	0.29017	0.21575
R-net (pred<5)	w3, c2	0.38198	0.38337	0.28300

表 6: word embedding 對 R-net 的影響

(4) 結果總覽

表 7 記錄所有 BIDAF 的模型實驗結果。表 7 中，embed 表示模型所使用的 embedding，p 表示文章長度，q 表示問題長度，w 表示詞彙長

度，a 表示 question-passing matching 和 passage self-matching 的 hidden size, n 表示 question-passing matching 和 answer prediction 的 LSTM 的層數。Word F1 表示 word-level prediction 和 answer 的 F1 score

ID	Model	Embed	p	Q	w	a	n	Word F1	Train F1	Test F1
1	R-net	w1, c1	400	40	10	64	3	0.33714	0.33504	0.27477
2	R-net (pred<5)	w1, c1	400	40	10	64	3	0.33987	0.33776	0.27577
3	R-net	w1, c1	400	40	5	32	5	0.30664	0.30459	0.23621
4	R-net (pred<5)	w1, c1	400	40	5	32	5	0.30990	0.30813	0.24036
5	R-net (pred<5)	w2, c2	400	40	10	64	3	0.29247	0.29017	0.21575
6	**R-net (pred<5)**	**w3, c2**	**400**	**40**	**10**	**64**	**3**	**0.38198**	**0.38337**	**0.28300**

表 7: All R-net results

5. Discussion

我們羅列了數種成功的模型及其變化，不過不乏有其他不成功的嘗試有待改進。例如，使用 POS embedding 在 R-net 上無法訓練出有用的模型，設定預測長度上限在 POS BIDAF 上 F1 score 不增反減，顯示出這些 feature, model 和 rule 之間的關係還有待釐清。我們想要特別討論 POS tagging 對這個 project 的影響，我們認為詞性是判斷 answer 的重要指標，可以想見在 QA 中，答案通常都會是名詞，因此詞性就很好的指出了答案的開始和結束的位置。不過，在我們的實驗中，POS embedding 僅對 BIDAF 有幫助，卻無法對 R-net 有用。我們推測原因有可能是我們 R-net 的參數沒有調好，另外，也有可能和 BIDAF 以及 R-net 的 attention 機制有關。BIDAF 的 attention 機制是 question 和 context 中每個字做 pairwise 的 similarity，POS embedding 可以很明確的表示在這種 similarity matrix 中，然而在 R-net 當中，question-passage matching 是藉由一個 LSTM 動態的賦予 attention，在每個 time step，LSTM 都考慮當前的 passage 的字,LSTM 的 hidden state 和所有 question 的字，缺乏對於整體文章的掌握，詞性可能就無法明確的表示在這種 attention 之中，或許也是 R-net 使用 POS embedding 結果不如預期的原因之一。

五、Reference:

[1] Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[2] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[3] Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." *arXiv preprint arXiv:1505.00387* (2015).

[4] R-Net: Machine reading comprehension with self-mathcing networks, ACL (2017)

[5] Machine Comprehension Using Match-LSTM and Answer Pointer, Arxiv (2016)

[6] https://github.com/minsangkim142/R-net

[7] https://github.com/fxsjy/jieba

[8] https://github.com/RaRe-Technologies/gensim