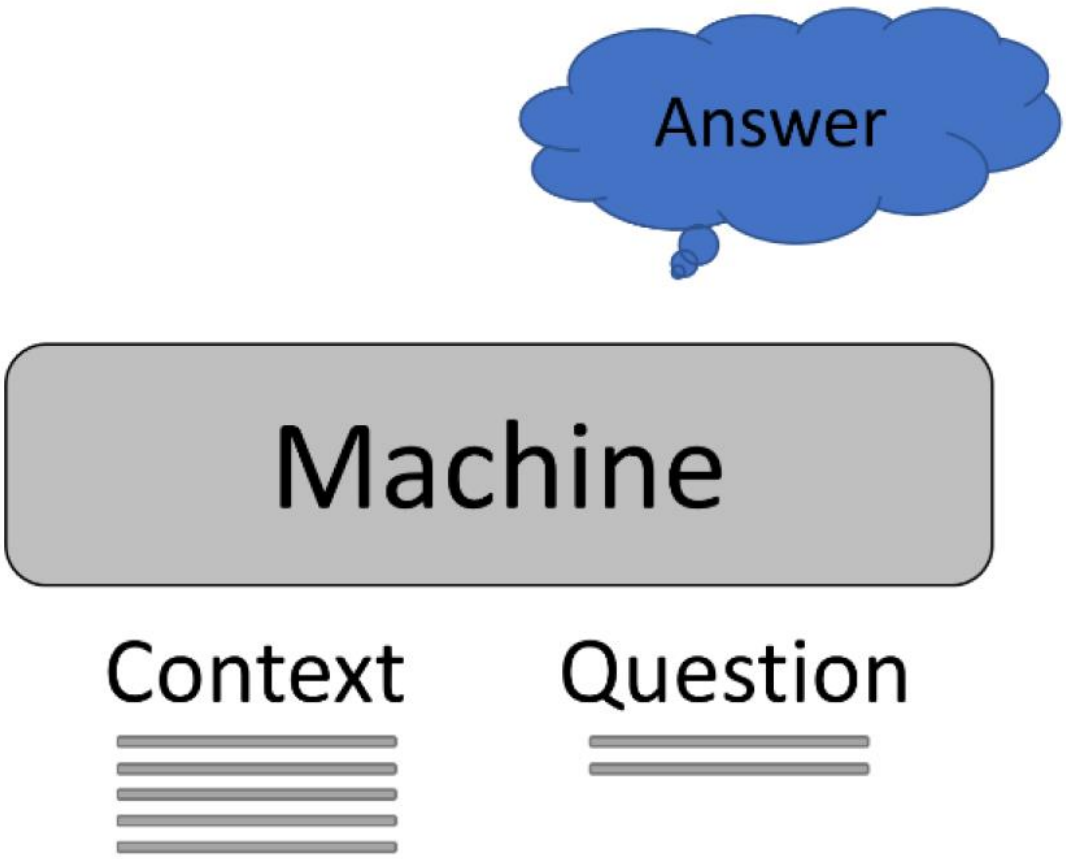


MACHINE LEARNING FINAL

CHINESE QUESTION & ANSWER

TEAM: NTU_b02902131_旁門左道

B02902131 曾大祐: model implementation, model fine tune and experiment analysis

R05943148 蕭家堯: function implementation and model study

B03902125 林映廷: paper survey and model study

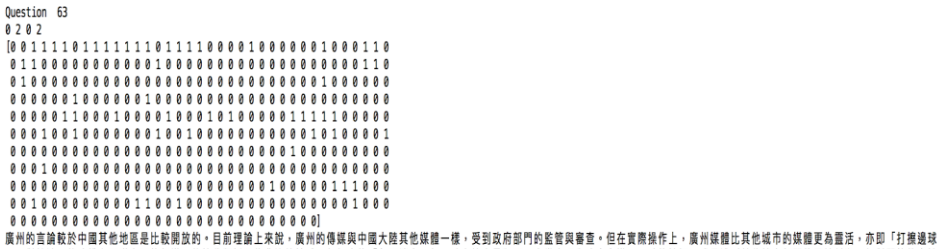
B03901115 吳宇: preprocessing and experiment analysis

1. Preprocessing and Feature Engineering

在機器學習的領域中，資料的前處理與特徵的抽取與模型的表現息息相關。在這個章節中，我們會介紹我們的前處理與特徵抽取方法，其中主要可以分為兩個部分：CinQ 與 Word Embedding。其中 CinQ 是 Character in Question 的簡稱，是我們這次模型的核心精神，而後期加入 Word Embedding 像是一個輔助資料，協助 Model 對文本可以有更全面的理解。

(1) CinQ (Character in Question)

對於 context 當中的每一個中文字，我們都會和問題做比較，如果這個字在問題句中也有出現，我們就標記它為 1，反之如果沒有出現就標 0 。舉例來說，如果 context 是「小明今天想吃滷肉飯」，問題句是「小明今天想吃什麼」，那麼 context 就會被 encode 成「111111000」。用這樣的前處理方法得到的 feature 來實作 sliding window 的時候，找到 context 中 1 最多的一段句子，正確答案就有很大的機率出現在這段句子前後的連續的 0 當中。例如下圖的例子中，feature 中的第一句裡面有很多 1，所以預測這一串 1 的前面兩個 0 的位置就是被挖空的答案。



(2) Multi-Layer CinQ

CinQ 的方法並沒有用到任何 embedding 的技巧，完全沒有考慮每個字的語義上的異同，僅靠 feature 中包含的 context 與問題句的重複率信息，就能答對一部分的問題，得到超過 simple baseline 的分數。但是只考慮單個中文字也帶來了諸多局限性。首先，要預測的答案應該是一個完整的詞，而 CinQ 的方法可能會造成預測的答案位置切在一個詞的中間(即預測成「州市」而非「廣州市」。) 其次，某些 context 中的字雖然在問題中高頻出現，但是跟問題句的語義並沒有關聯。例如「的」會在 context 的各處出現，這些 feature 中的 1 不僅不能幫助訓練 model，反而會加入雜訊造成干擾。

為了解決 uni-gram 的局限性，我們在 feature 中加入了 bi-gram 和 tri-gram。實作的方法類似一維 CinQ 用 0 和 1 表示是否在這個位置出現重複的字。因為 bi-gram 和 tri-gram 的 segment 長度大於 1，所以在比較完 context 和 question 之後要在每一個 1 後面加 padding 使各維度的位置都有對齊。

最後我們一共加到 5-gram，從實驗結果看來每增加一維 n-gram 都會對答題準確度有一些幫助。我們分析原因可能是用 bi-gram 和 tri-gram 還是不能排除很多長度大於 1 的中文 stop words。而且因為同一段 context 通常是在講同一個話題，有很多字彙反復出現，例如在描述廣州相關話題的時候，可能會多次出現「廣州」，但是假如問題是問「廣州的歷史有多長」，雖然「廣州」也屬於在問句中出現過的詞，但是因為它對應到 context 中各個位置，對於預測答案的位置就不會提供太大的幫助。換言之，如果加入對長度為 5 的片語的考慮，我們可能會發現 context 中只有一處出現了連續的「廣州的歷史」這 5 個字，從而幫我們找到正確的答案的位置。

(3) Weighted CinQ

如上面提到的，有很多不重要的 stop words 在 context 和問題句中都常有出現。為了盡可能降低這些字的干擾，我們對 context 中每一個有在問題句中出現的字或者詞都計算一個 weight 值，降低經常出現的比較不重要的字的 weight。

實作中計算 weight 的方法是用原本 context 去掉我們要計算的這個字剩下的長度直接作為 weight。例如，「的」在 context 中經常出現，而將 context 中所有的「的」都減掉之後剩下的長度就會變得很短；而「廣」在 context 中出現頻率不多，減掉所有「廣」之後剩下的 context 長度會大於去掉「的」剩下的長度。用這樣的方法，少出現的字的 weight 就被提高了。

(4) Punctuation Layer

在 CinQ 基礎上，我們又將標點符號也加入考慮之中。雖然在很多其他訓練 model 辨識語義的案例中，大部分的標點符號不能提供多少信息量反而會對 model 理解其他字彙的意思造成干擾，因而常會被建議和其他 stop words 一起從字典裡面刪除。但是在 QA 的情況中，標點符號有重要的斷句的功能。被句號隔開的兩段文字通常是在講兩個不同的主題和內容，所以答案的位置通常不會是跨過句號和逗號的兩邊，也比較少和 CinQ 中連續 1 的片段分屬被句點隔開的兩個不同的句子。此外，我們在觀察 training data 的時候發現，一些問題要問某個書名或者是特殊用語的時候，答案會包在引號中間。因此我們猜測，加入區分各種標點符號的信息，可以幫助 model 提高答題的準確度。

我們實作的方法是，在原本的 CinQ 的基礎上，再增加幾維表示標點符號的維度。例如，我們會用 one-hot encode 的方法表示句號在 context 中出現的位置，用 0 和 1 來表示這個位置上的 character 是否是一個句號。然後再將它和一維 CinQ 疊在一起變成兩維的 feature 一起作為網路的 input。以此類推，對於每一類標點符號，我們都用 one-hot 作為一維 feature 疊加上去。最後實驗的結果顯示，加入標點符號之後，model 答題的正確率有了很大的提升。

(5) Simple Word Embedding

CinQ 雖然用單純比較重複率和相似度的方法就可能成功超越 baseline, 但是還有很多題目不可能用 CinQ 答對。以下面這個問題為例：

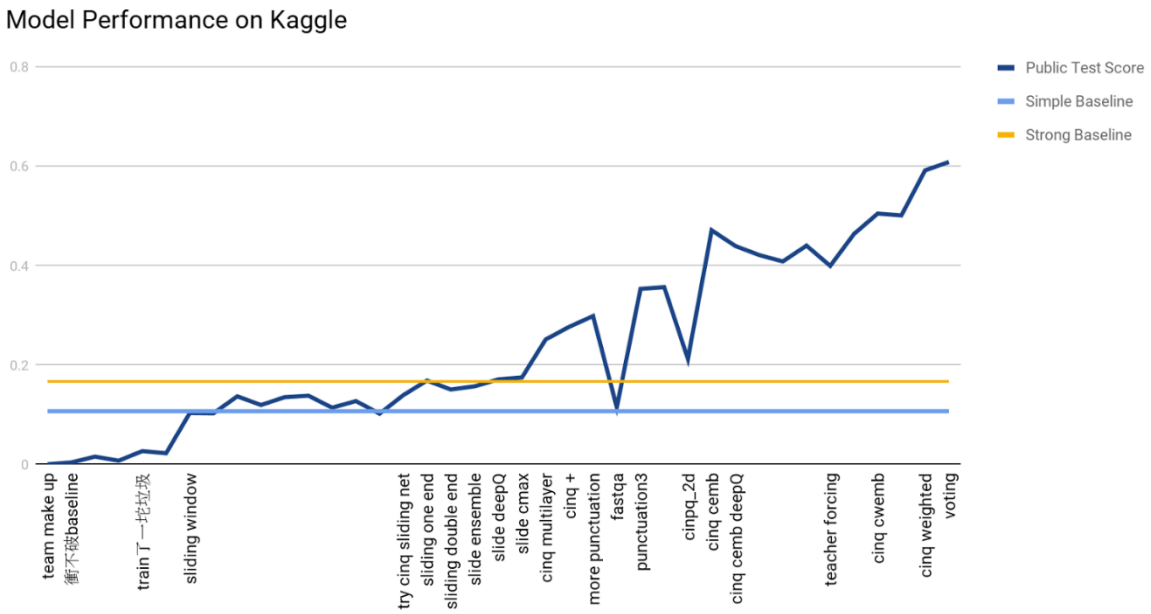
C: “...按城市轄區人口計算，也僅次於中國的重慶市，是全世界人口第二多的城市行政區。...”
Q:全世界人口第一多的城市行政區為?
A:重慶市

問題問的是人口第一多的城市，但是 context 中只出現了「第二多」。如果用比對重複字的方法，我們沒有辦法找到正確的答案位置。要從「第二多」推測出答案，model 必須要理解「第二多」，「第一多」，和「僅次於」這三個詞之間的語義關聯。

為了答對這一類需要語義知識的題目，我們加入了 character embedding 和 word embedding，進一步提高了 model 的表現。

2. Model Description and Discussion

在前一個章節中，我們討論了資料的前處理與有效特徵的抽取，在這個章節中我們會介紹我們在這次的期末專題中，如何從中觀察出更好的做法並加以改良以及這些模型的細節描述，如何對應不同的前處理與特徵。這個章節的排列方式如下圖的時間軸，我們依序介紹數個重要且關鍵的模型，並對他們的原理與結果做快速的討論。



(1) sliding window with parameters

--length : 固定的長度
--shift：找出某個 window 後回傳平移後的位子
--enlarge：找出某個 window 後，（平移後）伸縮某個量

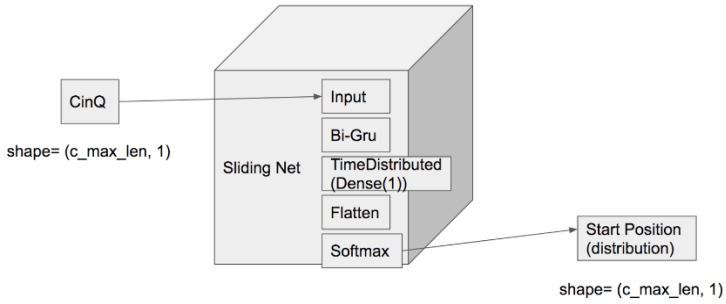
此方法啟發於 TA 的 hint ，在手把手之前我們並不明白 sliding window 的確切意思，所以我們就揣摩其意。我們的做法是將 context 中每個字是否出現在 question 裡頭當作一個

feature，將 sliding window 中每個切出來的 segment 計算出現在 question 裡面的字數最多的 segment ，將該段平移、伸縮後當作答案輸出。

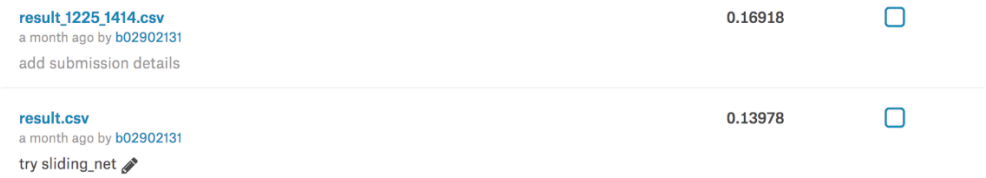
調整參數後得到上圖表現。過 simple baseline。

(2) Simple Word Embedding

這是我們自己亂取名的 model ，從這邊開始進入 neural network 的方法。我們的想法是，既然 sliding window 的人工 rule-based 方法可以過 simple baseline ，而調整那些參數也不過是重複搜尋最佳解的行為，那何不讓 neural network 來替我們做這件事。於是我們就將 cinq (character in question) 作為 feature ，餵進 model ，output 一個 start position ，加上一個 fixed length 為 end position ，作為輸出答案。如下圖所示：
c_max_len: context max length



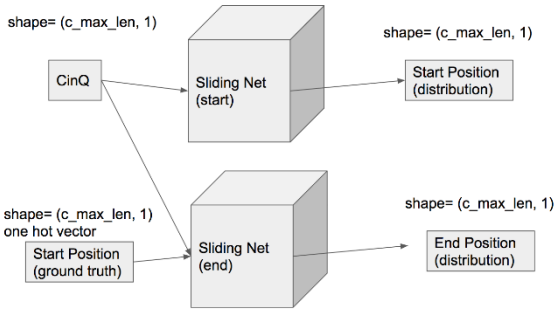
Input 過一層 bidirectional gru 得到 (cmaxlen, hidden_size) 的 sequence output，將每個 time step 分別過 cell=1 之 Dense layer 即可得到 (cmaxlen, 1) 的 distribution 作為 start position 的機率分佈，輸出前先 flatten 調整形狀， softmax 把 output 的範圍縮限到 0~1



表現直接到 0.13978 勝過原本手調 segment，將 model 調參用心 train 一下之後就到達 0.16918 過了 strong baseline。

(3) Sliding net +

CinQ 基本上跟 sliding net 運作方式差不多，但慢慢做了一些修正如下：
one end (start pos) -> two end (start and end position)
one model -> two model
擷取 context length 400 -> 不擷取，直接用全長 c_max_len = 1160



model 分開成兩個來 train，第一個跟原本相同，用於 predict start position，第二個 model 在 train 的時候多餵入 ground truth 的 start position ，此 feature 為 context 長度的 one hot vector ，只有 start index 為 1 其他都為 0，用於 predict end position distribution。在 infer 的時候，先以第一個 model 預測出 start position ，將這個預測出來的值以 argmax 取出 start index ，產生一個 predict 出來的 one hot vector 為給第二 model 來 predict end position distribution，一樣再以 argmax 取出 end index。

result_slide_ensemble_cmax.csv 24 days ago by b02902131 add submission details	0.17522	☐
result_1227_slide_ensemble_grape.csv 24 days ago by b02902131 add submission details	0.17149	☐
result_slide_ensemble.csv 24 days ago by b02902131 add submission details	0.15742	☐
result_1226_double_end.csv 25 days ago by b02902131 add submission details	0.15109	☐

Kaggle 分則是每有驚奇，不斷讓我們感嘆「這樣也可以？」，雖有起伏但仍有突破。

(4) Sliding net + Multi-level

爾後，我們發現 character 在中文裡雖然比起英文多了一點文意，但仍不足以表達完整意思，於是我們加上 bigram, trigram 到 5-gram 的 CinQ feature，以 context 長度乘 5 維的 input 輸入，內部運作照樣，輸出亦同。

Example:
Context: 小明吃滷肉飯，小明飽了，之後他洗碗。
Question: 小明吃飽之後他做什麼？

	小	明	吃	滷	肉	飯	·	小	明	飽	了	·	之	後	他	洗	碗	。
1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0
2	1	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

result_multilayer_32hidden.csv 24 days ago by b02902131 add submission details	0.25174	☐

效果顯著，我們欣喜若狂。此時我們超過許多還在 simple 到 strong 爬行的組別。但與此同時，我們的 model 仍未看到任何問句與文本，只看到若干 0 與 1 ，5 維的 input ，這 model 對於別人用 GPU 我們用 deepQ 甚至 queue 到只能用本機燒 CPU 的我們這組真的是非常實用。

(5) Sliding net + Punctuation

再之後我們繼續在這條歧途上前行，我們檢視 output 發現有時回答會包含標點符號，而我們思考，逗號能作為句子的分段，句號能隱含文章主題的轉變，若我們加入句讀作為 feature 或許能提升表現。因此，我們將 逗號、句號 作為 feature 加入。

	小	明	吃	滷	肉	飯	，	小	明	飽	了	，	之	後	他	洗	碗	。
3	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
，	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
。	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

result_1228_punctuation.csv
23 days ago by b02902131
add submission details
0.27741

(6) Sliding Net + Multi-layer Punctuation

我們食髓知味越走越偏，進入邪門歪道，走火入魔，把各種標點符號都列進去，最後總共加了 9 種標點符號。而結果不意外的，這九層的變相 rule-based 也為我們帶來更顯著的分數提升，雖然說工作我們就輸了，但是 Kaggle 分數卻又贏了一些。

result_1229_punctuation3.csv
22 days ago by b02902131
add submission details
0.35360

(7) CinQ + Char Embedding

最後我們還是逐步走回正軌，首先加入了 Character Embedding，我們的方式是以 training data 的所有 character （大約兩千多）做 size 8 的 embedding ，沒有 pre-trained，直接跟 model 一起 train。

Question 自己 attention 之後與 Context 乘起來，跟 Context 還有原本的 CinQ 全部 concatenate 起來，Time Distributed Dense 、Flatten、Softmax、輸出。

result_0105_cinq_cemb.csv
16 days ago by b02902131
add submission details
0.47160

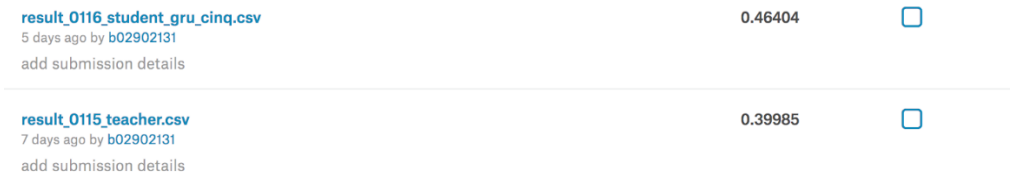
(8) Teacher Forcing

從這邊我們回歸到只有一個 model 。在某次下課請教教授時，我們得到了 Teacher Forcing 這個關鍵字，原來其實我們原本把 model 分成兩個的方法，就等同於 Teacher Forcing 。於是我把 code 改寫回只有一個 model ，在 training 的時候下參數

isTeacherForcing = True ，此時 model 會將 ground truth 的 start position 用於 train end position，但在 infer 的時候則取消 teacher forcing ，model 會改以 predict 出來的 start position 作為 predict end position 的 input。

小小需注意的地方是如此一來 model 在 train 跟 predict 的時候 layers 跟參數數量不一致，因此 load weights 的時候要 by name=true。這樣 keras 只會 load 名字相同的 layers 跟 weights。

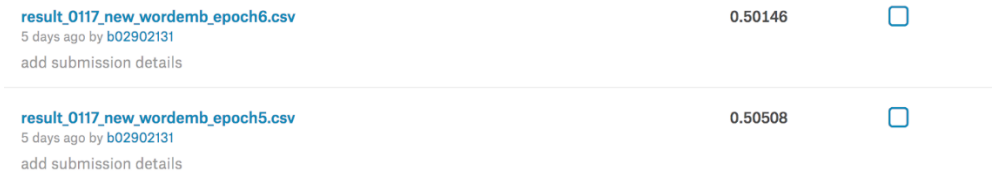
還有個小地方是教授當時教我先用 ground truth start position train 到一定水準後再拿到 teacher forcing train 第二階段。但實際實驗結果這樣還是會爛掉，training 整個過程都用 teacher forcing 才比較好。



但實際上效果好像都略輸 model 分開。好像無論如何設定 loss weights ，都不比兩個 model 分開專門 predict start position 跟 end position 來得準。但基於這樣的 model 比較簡潔，我們就暫時沒改回去了。實作上我們 loss wieght 設定 0.9: 0.1 出。

(9) Word Embedding

在之後我們加上 word embedding ，使用的套件是 gensim ，以及 wiki corpus ，取 mincount=300， size=100。 我們試過 trainable=True 也試過 False ，效果差不多。也想試試 size 調更大，但我的 mbpr 2014 with retina 撐不下去，索性放棄。



(10) CinQ weighted

由於 Character in Question 有時候會出現一些多餘的、詞頻高的字，例如 “的” 、“有” 之類的，或是同一個名詞出現在不同段落，因此我們將原本的 CinQ vector 再加上 weighted ，weighted 是將有出現的字用 1 除掉在 context 裡面出現的次數，例如廣州如果在文章裡出現三次，則除 3，標上 0.333。

(11) Voting

從我們觀察出，有些 model 可能在某些題目答對，某些答錯，而另外一個 model 可能相反，這是因為 training set 是有 randomize 過的。 為了使這些正確的答案能被盡可能保留助，我們又採取了複數個 model 投票的方式。其中投票的方式我們又試了兩種，如下：

Voting net：將多個 model output 出來的 distribution 直接接在一起，餵進另一個專門用來投票的 model，讓 model 來決定到底該 output 什麼範圍，output 一樣是 start position 跟 end position

Union vote：這個嚴格說來其實不算投票，就只是把多個 model 的 output segment 取 union，犧牲 precision 來換取 recall，這個方式的 segment 其實不連續，我們發現這樣做其實有點犯規不符合題目要求，但在 Kaggle 上是 work 的，可能是這題目在 Kaggle 上的漏洞。

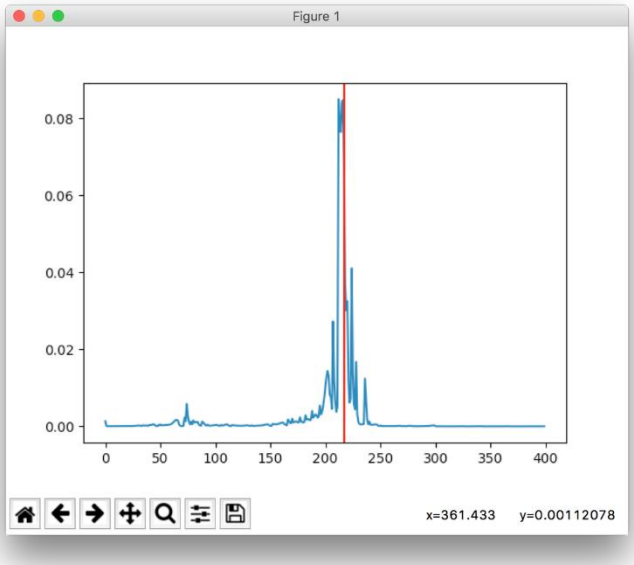
在 Kaggle 上的測試的結果分數略微上升。

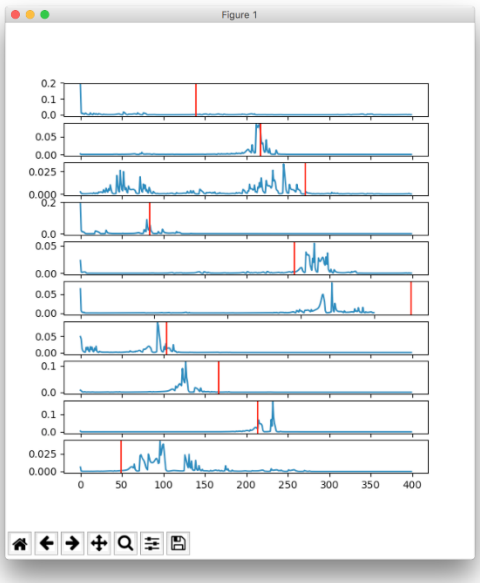
3. Experiments and Discussion

由於在上個章節中，我們有對 Model 的實作技巧做了原因與成果探討，這個章節我們主要會著重在 Model 的行為上。由於上傳 Kaggle 次數有限且可貴，我們希望能夠再上傳之前就可以預估這個模型可能的表現，同時也希望建立一個穩定而可靠的檢討方式。這個章節主要分為三個部分：觀察模型的輸出機率分布、觀察模型輸出的文字答案以及失敗的模型介紹。

(1) Output Distribution

示意圖如下，藍色為預測開始位置的 distribution 紅色為 ground truth。以下圖左側的例子來看，Model 有相當高的信心可以預測出接近 Start bit 的位置，且周圍的機率分布呈現高斯分布，這可能就會是一個成功的模型。從右邊的圖中可以看的出來，我們的 Model 是有從中抓到不少關鍵資訊的





(2) Case Discussion

在這個小節我們列出三個 Case 在不同模型下輸出的結果，比較不同模型之間可能的行為，並做簡單的討論。

Case 1 (ID: ef24f39f-aa85-4515-94ce-aee4403affc9)

Context:黃河，在中國古代稱作河水，簡稱河，是中國的第二長河，僅次於長江，
Question: 中國的第一長河為?
Solution: 長江

Sliding window with parameters:僅次於長江，也是世界第六長河流。中國的兩
Simple word embedding: 黃河，在中
Sliding net + : 黃河
Sliding net + Multi-level : 黃河
Sliding net + Punctuation : 黃河
Sliding Net + Multi-layer Punctuation : 黃河
CinQ + Char Embedding : 兩條母親河
Teacher Forcing : 黃河
Word Embedding : 第二長河
CinQ weighted : 第二長河
Voting : 黃河第二長河

Discussion: 在這個 task 中，我們所有演變的 model 都無法預測正確，推測其原因可能為雖有捕捉到「第一」和「第二」的詞義，但相近詞義的「僅次於」卻沒捕捉到。儘管經歷過如此多 model 的演變，但仍有些力有未逮的地方。

Case 2 (ID: 141af4ce-2aec-43dc-854b-448478d15a78)

Context: …明洪武十五年，宗泐和尚奉使西藏歸朝，經過河源地區時，曾對其進行考察，並賦詩一首《望河源》，在詩序中他指出河源出自巴顏喀拉山的東北，而且是黃河和長江上源的分水嶺，這在河源認識上成為了一次突破。
Question: 宗泐和尚所賦的哪一首詩使得在河源認識上成為了一次突破?
Solution: 《望河源》

Sliding window with parameters: 黃河和長江上源的分水嶺，這在河源認識上成
Simple word embedding: 水嶺，這在
Sliding net + :分水嶺，這
Sliding net + Multi-level：序中他指出
Sliding net + Punctuation：而且是黃河
Sliding Net + Multi-layer Punctuation：《望河源》
CinQ + Char Embedding :《望河源》
Teacher Forcing : 黃河
Word Embedding : 黃河
CinQ weighted :《望河源》
Voting : 明洪武十五年《望河源》

Discussion: 在這個 task，Sliding Net + Multi-layer Punctuation 和 CinQ + Char Embedding 這兩個 model 在 accuracy 上就有很明顯地提升，比之前的 model 更好聚焦，而更容易找到答案。

Case 3 (ID: 141af4ce-2aec-43dc-854b-448478d15a78)

Context : …紅橡樹的木材因為多孔道而不能防水，應用價值少於白橡樹，但紅橡樹木材的紋路生動，在家具製造方面卻很有用處。…
Question : 由於紅橡樹的木材具有什麼特性而應用價值少於白橡樹?
Solution : 多孔道而不能防水

Sliding window with parameters: 道而不能防水，應用價值少於白橡樹，但紅橡
Simple word embedding: 是很典型的
Sliding net + : ，但
Sliding net + Multi-level：因為多孔道而
Sliding net + Punctuation：為多孔道而
Sliding Net + Multi-layer Punctuation：為多孔道而
CinQ + Char Embedding : 多孔道
Teacher Forcing : 紋路生動
Word Embedding : 紋路生動
CinQ weighted : 不能防水
Voting : 多孔道不能防水紋路生動

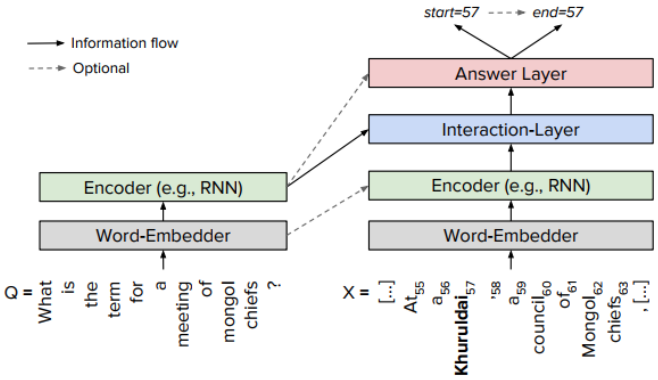
Discussion: 在這個 task 中，與其他 model 相比，voting 起到互相結合的效果，因為其他 model 大部分只答對一半，但 voting 則是兩半都可以找到，並將答案組合在一起。

(3) Failure Models

Fast-QA：我們這群迷途的小羊，亦曾想過回頭，轉往 squad 的 model 嘗試，雖然想當初 simple baseline 的 deadline 我們就曾為了一個 train 不起來的 r-net 弄到失魂落魄，但我們仍勇於面對，又再找了一個號稱很快的 fastqa 來 train 看看。

沒錯 Fast-QA 真的很快，也讓我們很快又再次面對挫敗。再度想起人類第一次 train 出一坨垃圾的恐懼。所以我們放下了 fastq-QA 再度重回 CinQ model 的懷抱。

Making Neural QA as Simple as Possible but not Simpler
Dirk Weissenborn, Georg Wiese, Laura Seiffe
https://arxiv.org/pdf/1703.04816.pdf



result.csv
23 days ago by Janus Shiau
fastqa
0.11532

CinPQ-2D：由於答案有時出現在問題裡出現過的字的前面，有時候在後面，因此我們推測只使用 CinQ 不足以表達前後順序的關係，而嘗試了 CinPQ 2d array ，使用 context 長度乘上 Question 長度的 array 來存取 context 中的字是否出現在 question 且存在相對應的位子，示意圖如下圖。但結果並沒有特別好，但是花費的時間劇增，因此放棄此方法。

question\context	小	明	今	天	吃	了	泡	麵
小	1	0	0	0	0	0	0	0
明	0	1	0	0	0	0	0	0
今	0	0	1	0	0	0	0	0
天	0	0	0	1	0	0	0	0
吃	0	0	0	0	1	0	0	0
了	0	0	0	0	0	1	0	0
什	0	0	0	0	0	0	0	0
麼	0	0	0	0	0	0	0	0

result_cinpq2d.csv
17 days ago by b02902131
add submission details
0.21246