# Delta Chinese QA

## NTU_r05943110_想題目好難...先寫作業好了

**R05943110 蕭　堯**

**R05943167 譚雋飛**

**B02202070 林智捷**

**B03505042 張翼麟**

# Preprocess

**Word Embedding :** 使用 Facebook 提供在 Wiki 訓練的中文 fastText，300 維

**OpenCC :** 使用 OpenCC 將 Facebook 的 tokens 由簡體轉為繁體

**Jeiba :** 使用 Jeiba 切字

**OOV :** 把 OOV 表示成 0 向量

**Tokens 長度:** 因為有很多 tokens 是我們不需要的(ex:日文、較長的英文)，這些 tokens 出現在 task 裡的機率不高。所以我取長度小於等於 4 的 tokens，這樣大多數的 tokens 都會是中文，也包含到了 4 字的成語。最後總共有 185848 個 tokens，是原先的 6 成，有效降低了計算量。

**數字 Tokens:** 我們發現雖然已經有中文數字的 tokens，但是卻沒有阿拉伯數字的 tokens，這會讓年份相關的預測大打折扣，於是我們自己多補了數字 0~2200 的 tokens，以便涵蓋大多數的西元年份。這幾個 tokens 對應到同一個 vector，我們自定義 vector 為每個維度都為 0.5 的 300 維向量，這樣的設定就讓我們在年份的預測已經相當準確了。

**Context 長度:** 我們發現 training data 的長度普遍比 testing data 長，以 jeiba 切字後一個詞作為單位長度的話，training 前十長的 context 長度都在 550 以上，testing 第一名卻只有 431，這樣的差異也許會影響訓練的結果，於是我們只選取 context 長度在 500 以下的文章 training，這樣只會捨棄掉 1%的 training data，而且也可以預測到所有的 testing data。

**Question 長度:** Training data 的 question 長度也有普遍篇長的情形，以 jeiba 切字後一個詞作為單位長度的話，training 前十長的 question 長度都在 50 以上，testing 第一名卻只有 28，於是我們只選取 question 長度在 35 以下的問題 training，這樣只會再捨棄掉 0.7% 的 training data，而且也可以預測到所有的 testing data。

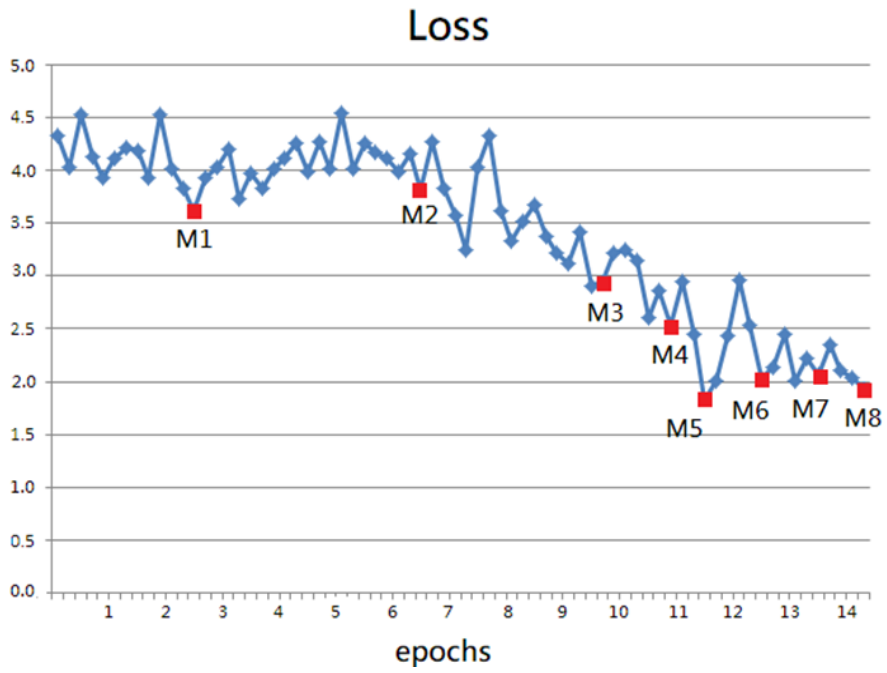
**Padding:** 所有長度未滿 500 的 contexts 都在後面補 0 到長度為 500；所有長度未滿 35 的 questions 都在後面補 0 到長度為 35。

**標記:** 因為最後 predict 的是 answer 再 context 裡的 start 與 stop，而非利用 jeiba 切字後 context 裡的 start 與 stop，所以我們先用一個標記記住 jeiba 切字的所有斷點，以利之後能夠準確的預測到 answer 在 context 的位置。

**Validation Set:** 從 training data 中隨機取 20%的 contexts 作為 validation set

# Model

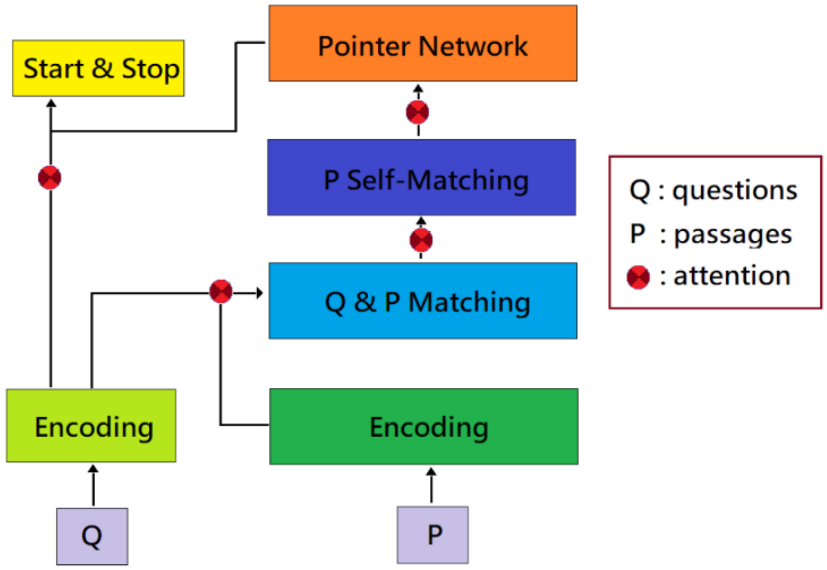
下圖為我們的 model 在不同訓練過程下的 Loss，Loss 的定義為標準答案與預測答案之間的 Cross Entropy。在過程中我們取了其中 8 個 model(M1~M8)，可以看出隨著 epochs 上升，Loss 雖然有劇烈的抖動，但整體上是有降低的，Kaggle 的表現也有越來越好的。也因為如此，我們也希望可以看看從 M1 到 M8 的過程中，model 到底學到了甚麼，詳細分析的結果會在後面提到。




Kaggle

M1: 0.084
M2: 0.110
M3: 0.205
M4: 0.267
M5: 0.392
M6: 0.428
M7: 0.435
M8: 0.457

我們參考了 Microsoft 開發的 R-NET，示意圖如下。model 主要分成 5 個 blocks 分別為 Embedding、Encoding、Q&P Matching、P Self-Matching 和 Pointer Network。比較特別的是，R-NET 使用了 2 層 Interaction Layers 分別為 Q&P Matching 和 P Self-Matching。下面會針對們這幾個 block 作介紹。



**Embedding:** 原本的 R-NET 有 word embedding 和 char embedding 兩部分，不過對於中文的 task，char embedding 似乎比較麻煩，加上 Github 上有提到 char embedding 的表現並沒有好很多，因此我們的 model 只保留了 word embedding 的部分。

**Encoding:** 將 Q 和 P 的 word embedding 用 bi-directional RNN 去 encoding，以便了解字詞之間的前後的交互關係。為了降低計算量，採用的是 GRU 而非 LSTM。

**Q&P Matching:** 主要是對文章中與題目相關的地方做強調，原理類似於 Match-LSTM，也就是把文章中每個詞對問題作 attention，再與問題一同放進 RNN 去 encoding，最後得到每個詞的 query-aware 的表示方式。另外，R-NET 還多使用了 attention-based 的 gate，其與 LSTM 和 GRU 的 gates 不同之處在於 attention-based gate 是基於問題的 attention-pooling vector 和文章的內容決定閥門開關的程度。

**P Self-Matching:** 有的時候，問題的答案常常會跟有用的線索距離很遠，所以讓 model 理解文章前後關係是必要的。方法與 Q&P Matching 類似，只是 Self-Matching 是對文章和文章本身進行比對。

**Pointer Network:** 使用了 Pointer Network 決定答案的 start 與 stop，比較不同的是，將問題的 attention-pooling vector 做為 Pointer Network 的初始狀態，與 Self-Matching 的 output 共同輸入 Network 得出問題 start 的概率，然後再利用 start 的概率分布與 Self-Matching 的 output 再輸入進 Network 一次，更新後得到問題 stop 的概率。最後，start 和 stop 的概率會與 target answer 去做 Cross Entropy 計算出 Loss，以最小化 negative log probabilities 的方式去更新 model 的參數。

## ■ Ans. Sentence M+

我們發現這個 task 有一個很重要的特點，這有助於改善我們的 model，對於後面的圖表分析也有很大的幫助，以 training data 其中一題為例。

**Q:** 因斯布魯克機場負責處理哪座山周圍地區季節性國際航班?
**Ans.:** 阿爾卑斯山
**M+:** 因淤布魯克機場位於城市的西部，處理阿爾卑斯山周圍地區前往歐洲航班及季節性國際航班。

Ans.一定在 context 的其中一小段，如果 context 以句號分開可以拆成 3~5 小段，觀察後會發現有其中一段會與 question 有較高的重複字，我們稱 context 的這一小段為 M+，通常，Ans.都會落在 M+裡面。統計上，training data 有約 9 成的 Ans.會落在 M+裡。 此題的 M+和 Q 重疊的字以藍色標註，Ans.以紅色標註。

## ■ Questions of「以及」

當我們在把問題分類的時候發現，問題的難度是有分級的，有些問題可能不太容易回答，要以較複雜的 model 來預測；但有些問題卻相對簡單，以 rule-based 的方法判斷反而有較好的表現。最後我們決定以 rule-based 的方法來 predict 問題中有「以及」字眼的答案，舉一個問題為例。

**Q:** 地殼可依其結構分為大陸地殼以及?
**Ans:** 海洋地殼
**M+:** 地殼的質量只占全地球 0.2%，按結構分為大陸地殼和海洋地殼兩種。

在 M+中，我們針對特定連接詞「與」、「、」、「和」、「及」、「或」和「以及」(藍色標註)的下一串詞(紅色標註)作為 predict 的答案。

## ■ Ensemble

根據上面的原則，我們可以做一個簡易卻有效的 Ensemble。

**Method:** 取 M8、M7、M6 做 ensemble，如果 M8 對某題的 predict 有落在 M+內，則選用 M8 的 predict 做為此題答案；如果沒有，再繼續判斷 M7 predict 有沒有落在 M+內，如果有則以 M7 的 predict 做為此題答案；如果沒有再繼續判斷 M6 有沒有落在 M+，有的話就採用 M6 的答案，都沒有的話再採用 M8 的 predict 做為此題的答案。

**Result:**

| | M6 | M7 | M8 | M6+M7+M8 | M6+M7+M8+以及 |
|---|---|---|---|---|---|
| **Kaggle** | **0.428** | **0.435** | **0.457** | **0.462** | **0.470** |

之後我們的 model 還有再繼續 train，多了 M9 和 M10 兩個 model，在 Kaggle 上分數分別為 0.461 和 0.465。我們把 M8+M9+M10+「以及」做 ensemble，最後得出了我們的 Kaggle Best 0.491。到 M10 以後 model 的 Loss 和 Kaggle 表現都沒有變更好，也就是說 model 在 M10 後差不多就 overfitting 了，大約發生在 16 個 epochs 的時候。

## ■ Other methods

在發現 M+這個重要的指標後，我們就試著把 testing data 的 context 中 M+以外的內容都刪掉，拿 train 好的 M8 直接從 testing data 的 M+中找答案，理論上準確率會提高，不過上傳 Kaggle 後表現是變差的。我們認為可能是因為文長不匹配的關係，因為 M8 在 train 的時候所讀的是全文，長度較長；而 test 的時候是用 M+，長度較短。要改善這個問題其實很簡單，就是在 train 的時候也只拿 M+去訓練。不過我們的運算資源有限，加上也沒時間重 train 了，只好作罷……。

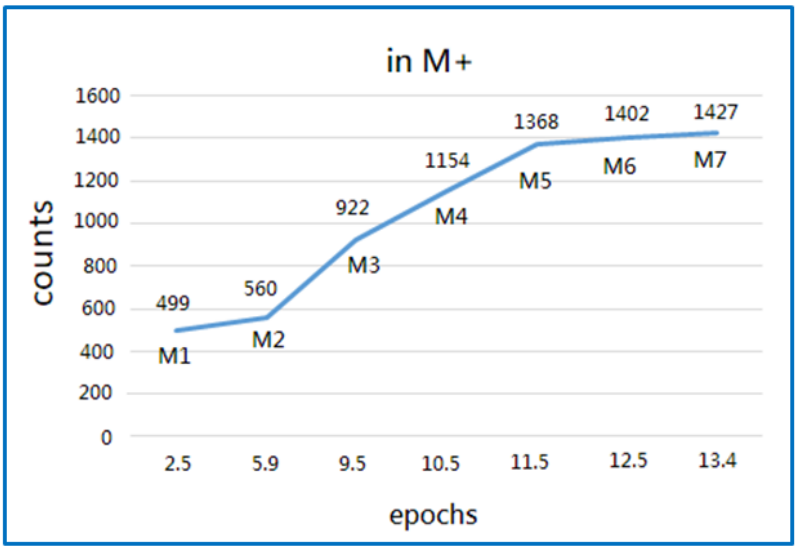
# Experiments and Discussion

**Motivation:** 我們認為 M+是個很重要的指標，比較好的 model 預測的結果應該大多在 M+內，所以我們選擇了在訓練過程中不同階段的 8 個 model (M1~M8)，想分析這 8 個 model 在 M+內的數量差異。

**Discussion:** 從下圖中的趨勢看出來 training 的 epochs 上升是有助於找到答案所在的段落，可以看到我們最後一個 model 預測的結果有 1427 個問題在 M+裡面，這大概佔了全部問題的 8 成。前面提到答案在 M+中的約 9 成，這也間接說明了我們 model 中 Q&P Matching 的 interaction layer 是有學到東西的。

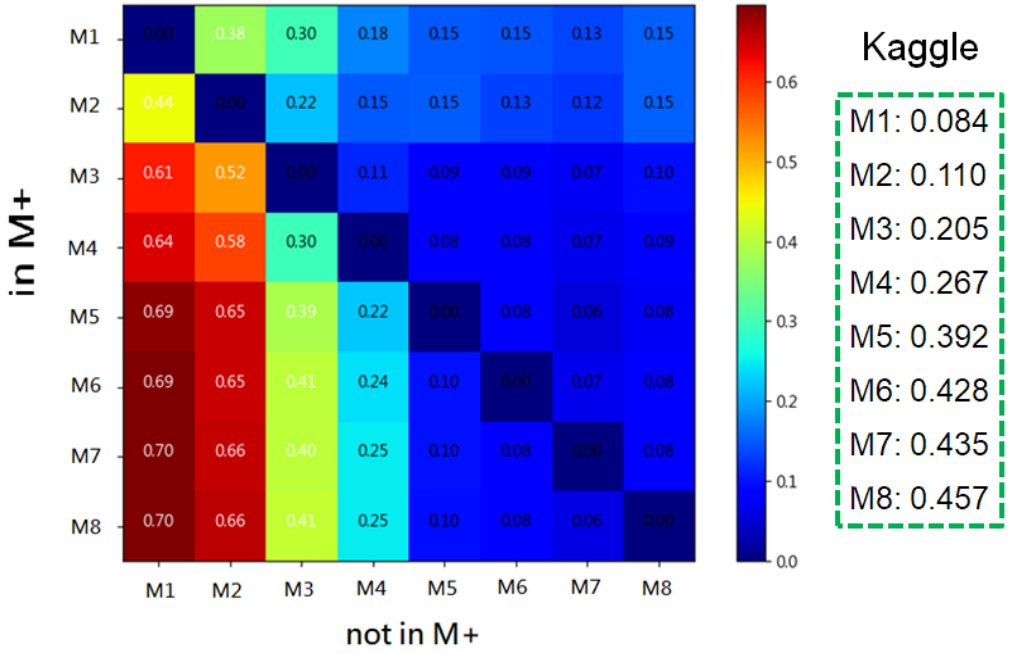
Kaggle

M1: 0.084
M2: 0.110
M3: 0.205
M4: 0.267
M5: 0.392
M6: 0.428
M7: 0.435




**Motivation:** 因為用 M+的 ensemble 是有讓表現變好的，所以我們想要探討一下 M1~M8 這 8 個 model 之間 M+分布之間的交互關係，以便能夠更了解 ensemble 能夠變好的原因。

**Discussion:** 下圖中第 i row 第 j column 指的是所有問題中，第 i 個 model 在 M+裡，但是第 j 個 model 不在的比例，可以觀察到其實 6 7 8 個 model 彼此之間差異並不大，我們也肉眼看過這幾個 model 有差異的題目，沒有發現甚麼比較重要的關聯性，所以用後面幾個 model ensemble 的效果就沒有很顯著。另外我們實驗過拿前幾個 model 互相 ensemble 的效果會不錯，例如前三個 model ensemble 大約可以得到 0.25 左右的成績，而前三個 model 最好的也只有 0.205，這個部分可以從圖左上角看出來，推測在前期的幾個 model 學到的東西是不一樣的，因為右上角的部分 pivot 兩側的值都不像右下角那麼小，也就是說他們可能 focus 在不同的問題上面，所以用前期來 ensemble 是會有不錯的效果的。




**Motivation:** 我們想要進一步了解model到底比較擅長或不擅長回答什麼問題，以及 M1~M8 在不同問題上預測的差異為何，所以我們隨機選取了 20 篇文章約 80 個問題進行分析。

| | 特定字 | 問法 |
|---|---|---|
| 變異度 | **Fig.1** | **Fig.2** |
| 準確度 | **Fig.3** | **Fig.4** |

**變異度:** 我們發現，對於有些問題，M1~M8 預測的結果不會有太大的改變，有些則相反。為了方便分析，我們定義變異度為$8^2/\sum x_i^2$ (對於一個問題，若 8 個 model 有 2 類不同的預測，則 i=1, 2，若第 1 類被 5 個 model 預測，即 x1=5，第 2 類被 3 個 model 預測，即 x2=3)。因此變異度最大的情況是 8 個 model 都預測不同的答案，即變異度為 $8^2/(1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2)=8$。而變異度最小的情況是 8 個 model 都預測相同的答案，變異度為 $8^2/8^2=1$。

**準確度:** 即 M8 這個 model 預測的正確率，問題的準答案由人工方式判定。

**特定字:** 對於某些問題，文章中與題目裡的特定字詞重疊的部分剛好答案就會在附近。舉例如下，Q:神奈川曾於江戶時代末期時由於哪個條約而被定為開港地? context: 江戶時代末期，神奈川在美日修好通商條約中被定為開港地，此題關鍵字為「條約」。

**問法:** 我們將問題粗略分為 8 類，分別為「哪"一"個」、「哪"一"種」、「哪一"單位"」、「有/是甚麼」、「何處」、「何時」、「何人/誰」、「為?」

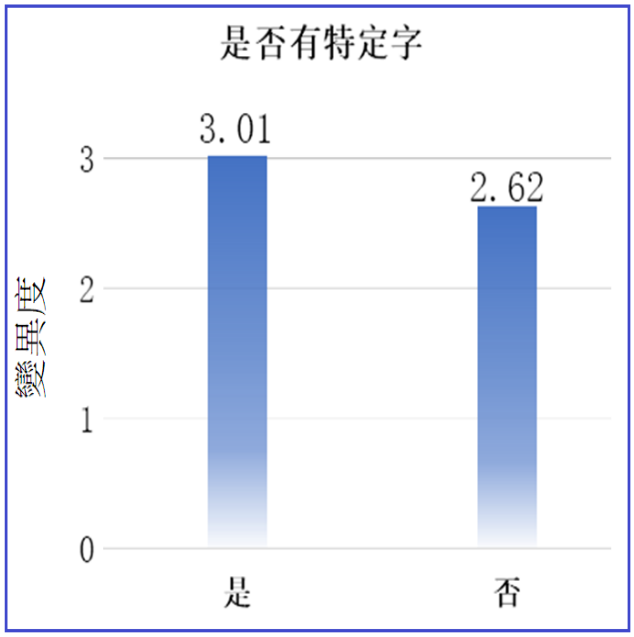



**Fig.1**

**Discussion:** 若沒有特定字，則代表該問句使用相似的文字取代文本中的敘述，亦或以通稱問特定答案，例如：哪種地形？而此時文本中能回答的選項則有：丘陵、平原、台地...等。而上圖顯示，若有特定字則變異度較高，但差異並不是很明顯。我們觀察數十筆資料後發現，8 個 model 可能會預測的答案多在特定字附近，因此 model 多樣的預測在特定字附近的字詞，所以變異度較高；而若無特定字，則 model 可能是依其他方式預測，導致變異度較低。

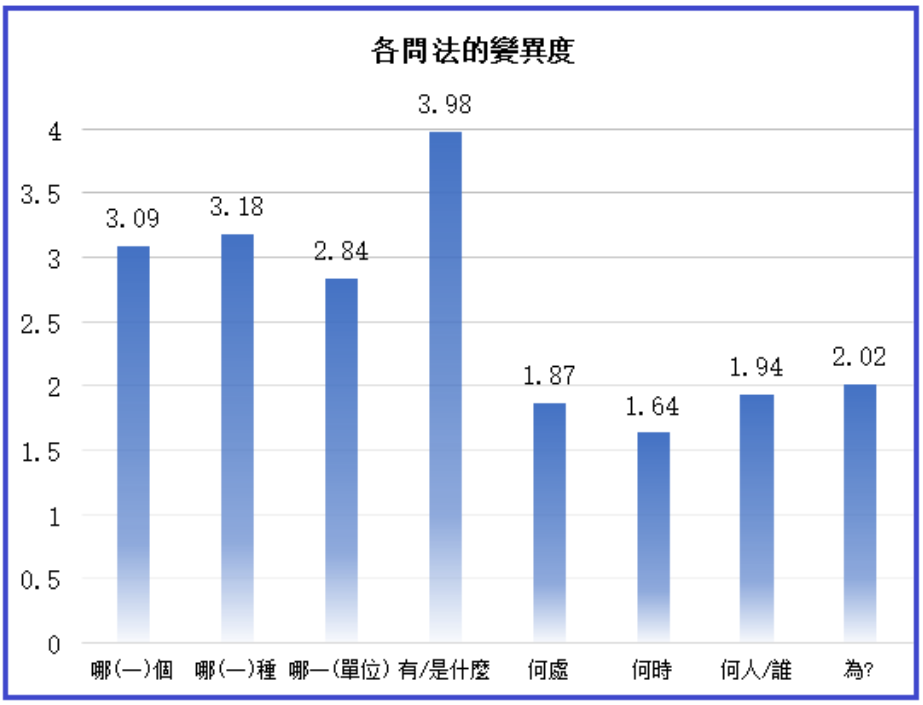
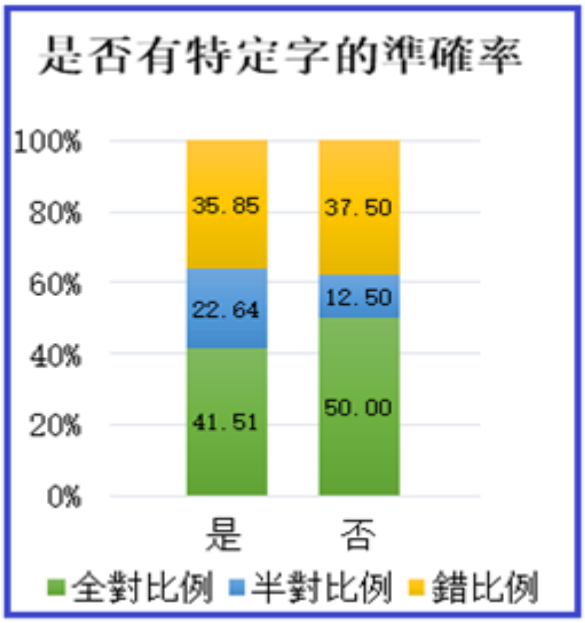



**Fig.2**

**Discussion:** 這近 80 個問題，以「哪"一"個」和「哪"一"種」佔的比例最高，而其他問法的比例都相對較少。我們可以發現，8 個 model 對於以"什麼"當疑問詞的問法的變異度較高，或許此種問題對 model 的難度較高。而佔最大比例的兩種問法，其變異度較接近我們在 Fig.1 所記錄的平均變異度。另外，我們的 model 對「何人」、「何時」、「何處」問法的變異度明顯較低，也就是此種問法，8 個 model 得到的答案較單一，且準確率也較高。




**Discussion:** 如左圖所示，M8 預測有特定字的問題，全對的準確率略低，然而把半對也涵蓋進去的話，與沒有特定字的問題的準確率相差並不大，兩者比例皆靠近 65%。我們可由先前在 Fig.1 和此圖推論，model 在預測有特定字的問題時，可能得出的答案較多樣，且多在特定字附近，所以預測的答案常有與正確解重疊的字詞，也因此導致半對比例較高。

**Fig.3**

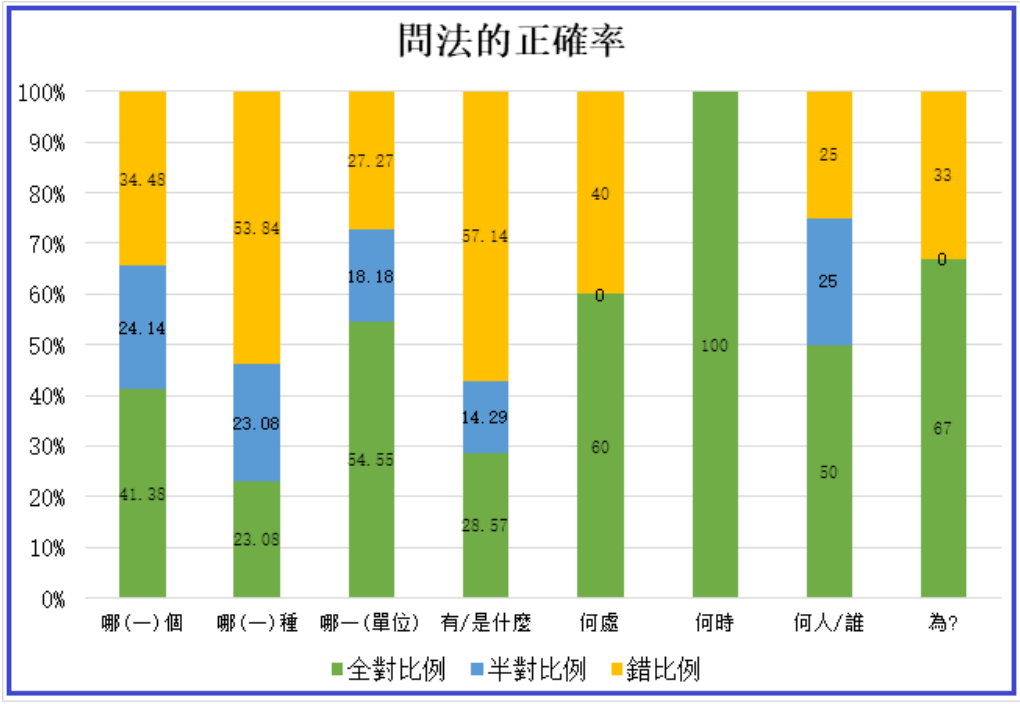



**Fig.4**

**Discussion:** 明顯可看出，M8 對後四種問法，有較高的準確率，且與前四種問法相差不少。而這八種問法當中，準確率最差的是「有/是什麼」，且由 Fig.2 所得的變異度和此圖推論，model 在預測此種問法的問題時，變異度最大又在此處，所得準確最差，因此 model 極不擅長預測此種問法的問題。而 model 在預測「何處」、「何時」和「何人」時變異度最低，且此處正確率也較高，因此我們的 model 皆較擅長此類問題，在文本中找尋特定地名、時間或人名。而在所有問題所佔比例最大的兩類問題：「哪"一"個」和「哪"一"種」，我們簡單計算該全對和半對所得的比例為 0.440，相當接近我們使用 M8 在 Kaggle 所得的分數 0.457。

## 相關參數

Batch: 50
Training / Validation: 0.8 / 0.2
Optimizer: Adam
Passage/Question length: 500/35
Embedding Dim: 300
Token numbers: 185848

## 參考論文

R-NET: Machine Reading Comprehension with Self-matching Networks.

## 組員分工

| **R05943110 蕭　堯** | **寫 code、整理 report** |
|---|---|
| R05943167 譚雋飛 | 提供運算資源 |
| B02202070 林智捷 | 實驗分析、繪製圖表 |
| B03505042 張翼麟 | 實驗分析、繪製圖表 |