請清楚標示你選的題目 (1)

Chinese QA

Team name, members and your work division (1)

組員	分工
B04705003 資工三 林子雋	實作及使用R-NET、測試不同word embedding和不同feature的影響、error analysis的介面設計
B04901117 電機三 毛弘仁	製作 baseline model、做 paragraph segment 的 preprocessing、實作 data augmentation、進行 error analysis
B04901018 電機三 游昇融	製作 baseline model、嘗試將與答案重複的字詞從答案中移除、將多個R-NET的預測結果做不同方式的 ensemble

Preprocessing/Feature Engineering (3)

1. Rule-based segmentation

以「。」將 paragraph 切割成數個 segment，並且抽出與 question character 重複最多 character 的 segment + 排序下一個的 segment，當作 model input。可參考底下 baseline model 的說明。原本用整段 paragraph 當 input 所得的 F1 score 為 0.48，使用這個 preprocessing 方法則提升到 0.54。

2. Word embedding

使用gensim在這份資料上面訓練word embedding。

3. 增加統計 feature

Exact match, term frequency 的 feature 接在 word embedding 後面當作一個字的 feature。我們發現兩項 feature 都有使 performance 提升。

4. Ensemble：聯集 >= 機率總和 >> 交集

聯集：將多個 model 預測的結果取聯集，因此答案可能是不連續的。主要效果的提升應該是因為整體的 F1 score 是每題 F1 score 的平均，因此比起完全答錯（回答與正確答案毫無交集，F1 score = 0），如果能夠將延伸到答案，就能夠提升分數，這個方法以我們其中一

個取 F1 score 約 0.503 的 4 個model 來說，可以提升至 0.514，平均每個回答的長度為原本的 2.738 倍。

機率總和：將 model 預測出的 start 和 end 在各個分布的機率相加，得到整體機率最大的 start 和 end pointer，以我們其中一個取 F1 score 約 0.503 的 4 個model 來說，可以提升至約 0.512。

交集：將多個 model 預測的結果取聯集，結果很糟，原因來自多個 model 答案如果差異很大，會得到許多空集合，以我們其中一個取 F1 score 約 0.503 的 4 個model 來說，有 628 題會得到空集合。

5. Data augmentation

如果 answer 含有數字，將每個 digit 改成 0-9 的 random integer；否則將 answer 抽換成另一個長度大於 5 Chinese characters 的名詞。我們發現用這個 preprocessing 方法使 performance 大幅下降，因此決定先不要再胡亂試著調東西，而做扎實的 error analysis。分析結果顯示 jieba 斷詞會把答案「切碎」，而我們 augmented data 長度都不短，這很有可能是 error rate 上升的原因，並非 data augmentation 本身是有錯誤的。若斷詞能力變好，相信還是可以嘗試 data augmentation！

Model Description (At least two different models) (7)

1. Baseline model (rule-based)

paragraph 以「，」或「。」去切割（如圖一），得到多個 paragraph segment。接著看 question characters 在哪個 paragraph segment 出現最多次，就拿那個 paragraph segment 當作最終 answer。

Paragraph：周易的成書時間歷來頗多爭論。傳說遠古的伏羲創八卦、夏禹將其擴充為六十四卦，六十四卦被記載在《連山》一書，《連山》以「艮」為第一卦。到了商朝，六十四卦的次序被重新排列，被記載在《歸藏》一書，以「坤」為第一卦。\n依據司馬遷《史記》的記「囚羑里，蓋益易之八卦為六十四卦。」，後人因此認為《易經》是商朝末年、西周之初的時候確立，是周文王奠定了《易經》以「乾」為第一卦，並為每一卦寫下「卦辭」。周文王之子、周武王之弟周公旦則被認為是「爻辭」的創立者。卦辭和爻辭的內容不單影響周朝的歷史，也影響到「詩經」的文學風格。\n《周易》起源相當早，相傳「文王拘而演周易」，所以坊間認為西周初年由文王所著，因此較春秋時代的哲學著作為早。\n廣義的《易》包括《周易》和《易傳》。由於《周易》文字含義隨時代演變，內容在春秋戰國時便已不易讀懂，因此孔子時代的人撰寫了《十翼》，又稱為《易傳》，以解讀《周易》。現今學者多認為《周易》七卷書中最早的《易傳》是戰國時代的作品。\n《周易》以一套符號系統來描述狀態的變易，表現了中國古典文化的哲學和宇宙觀。它的中心思想，是以陰陽交替的變化描述世間萬物的變化。雖然秦朝時已普遍認為《易經》可用以占卦筮卜，但它的影響遍及中國的哲學、宗教、醫學、天文、算術、文學、音樂、藝術、軍事和武術。自從十七世紀開始，《易經》也被介紹到西方。

Question：卦辭以及什麼的內容影響到「詩經」的文學風格?

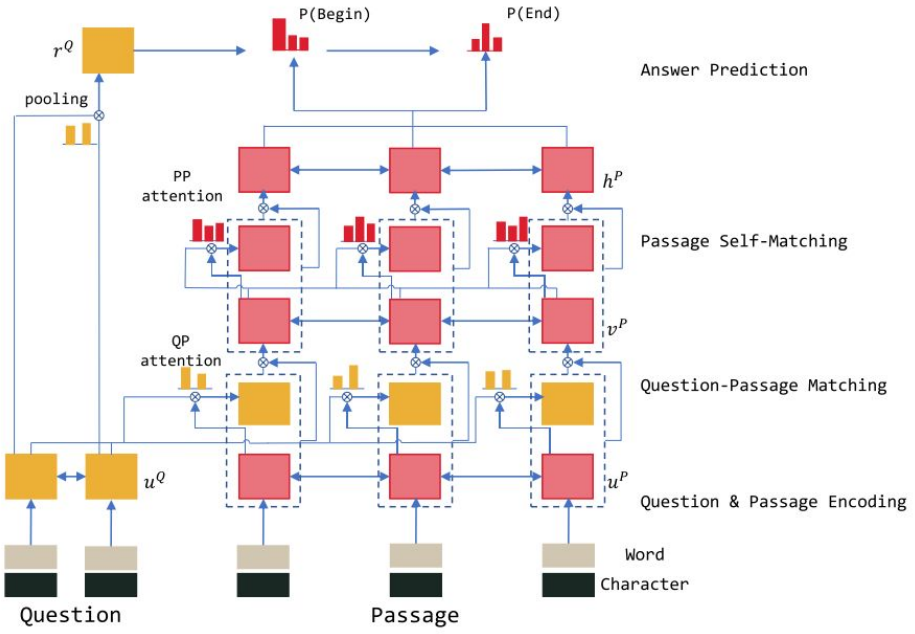
圖一：範例題目及 paragraph segment 選取。
藍字 + 綠字是以「。」去切 paragraph segment；綠字是以「，」切。

2. R-NET model實作

第一步：先將文章段落 (Passage) 和問題 (Question)通過 GRU encode 之後，再做 question-aware 的 attention (也就是將 question 的資訊做摘要之後接在每個 encode 過的 passage hidden vector 後面)。

第二步：讓 Passage 對自己做 attention，目的是讓神經網路再看一次 Passage 使得神經網路能夠更加理解和整合 Passage 資訊。

第三步：把 Passage 和 Question 的資訊放入 Answer RNN 的神經網路中去預測 Passage 的 start 和 end。



圖二：R-NET 總攬。

Experiments and Discussion (8)

1.Baseline model (rule-based)

1.1. 使用的 rule

經過觀察，我們發現 paragraph 中的答案，旁邊所包圍的 context 往往和 question 的 characters 有不少重疊，因此我們才決定以此建立 baseline model。我們發現，以「。」切 paragraph segments 可以得到比用「，」切還來得高的 recall，但相對的， precision 也可能比較低。儘管如此，F1 score 還是比較高（表一），而且輕易超過 simple baseline。

表一：Baseline model 的 F1 Scores

Model	F1 Score
Simple baseline	0.107
「，」segmentation	0.123
「。」segmentation	0.147
Strong baseline	0.167
「。」segmentation + remove question words from segment	0.231

因為「。」切出來的 paragraph segment 可能具有較低 precision，我們想了一個增加 precision 的簡單方法：將 jieba 斷詞過後所得的 question words，從 paragraph segment 裡移除。這樣的簡單方法，使我們到達 0.231 的 F1 score，過 strong baseline。

受到這樣的啟發，我們後面的 R-Net model，決定以 paragraph segment （底下簡稱「segment」）當 input 去訓練，這樣可以讓 model 更專注在重要的句子上。

1.2. Baseline model 方法來做 preprocessing

因為 training data 當中，有 0.164 比例的 single-sentence segment 沒有包含到正確答案，因此我們將 segment 改成 2-sentence（後面多加一個 segment），將比例降低為 0.127。不含 answer 的 2-sentence segment 我們先移除，留其他「完好」的 2-sentence segment 當作 training data。這樣的 preprocessing，可以使 R-NET 獲得 performance gain（圖三）。

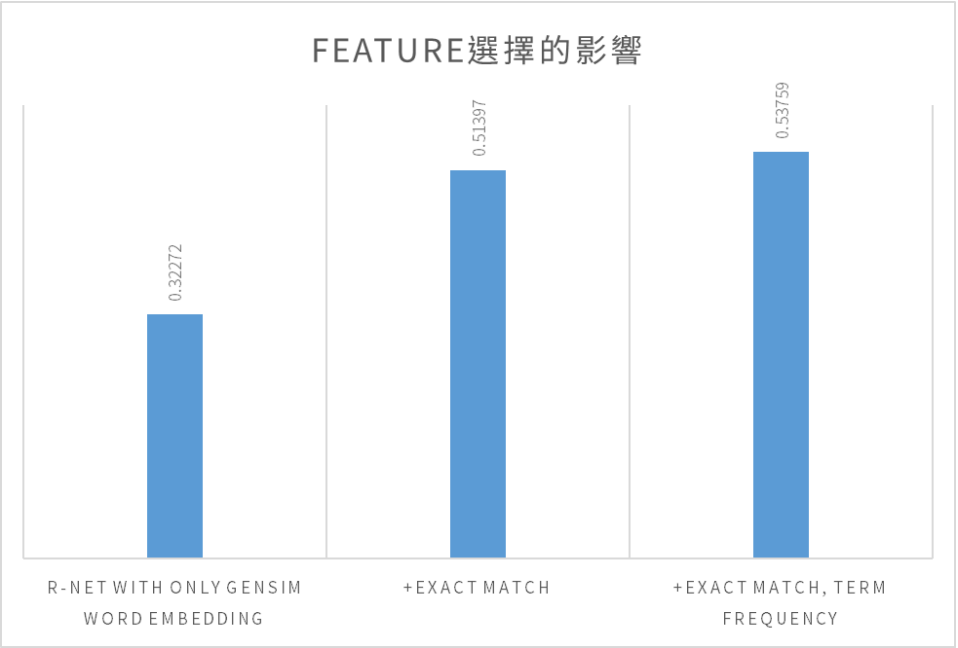
2. R-NET model

一開始，我們直接拿 R-NET 直接訓練在原本的 Passage 上，並直接在原本的 Passage 和 Question 上訓練，大約訓練一天半之後預測，結果上傳 F1 Score 約 0.13。

後來我們在思考，會不會是因為太過冗長的句子使得神經網路不好對答案聚焦。因此我們結合了 Rule-based 的方法，將 R-NET 直接訓練在擷取出來的 2-sentence segment 身上，F1 Score 為 0.33，顯示出神經網路在短的句子的確能夠抓取到資訊。

我們繼續發想到，其實可以直接把 Rule-Based 的資訊直接 encode 起來並接在 word embedding 後面，於是我就將 Passage 當中每個字的 embedding 後面，都接上「這個字是否出現在Question (Exact Match Feature) 」的一個binary feature，也將Question的每個字做同樣的事情，F1 Score 上升到 0.51。

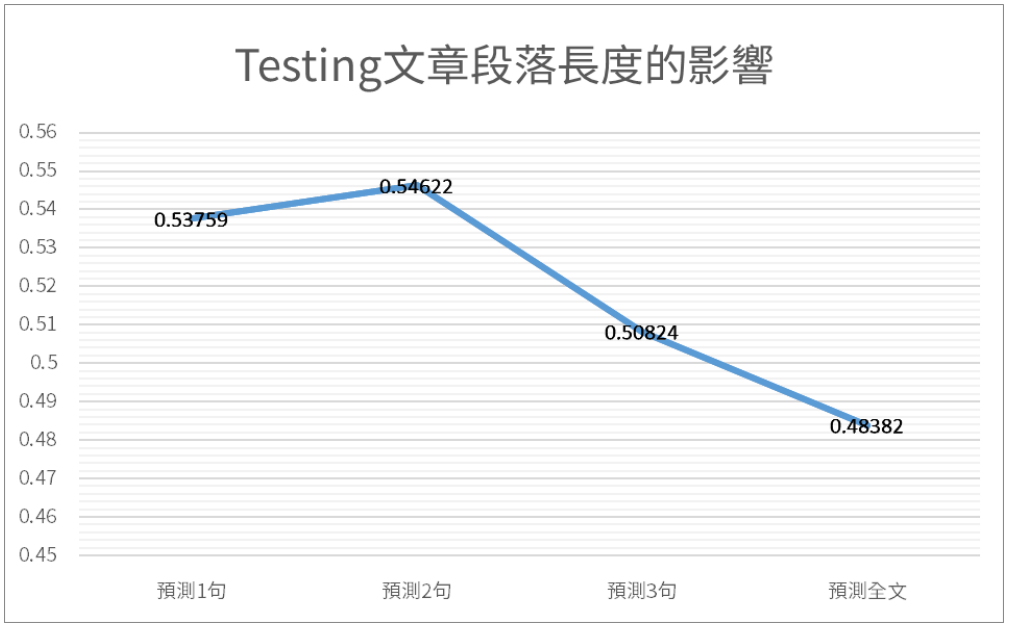
最後，我們認為一般人在閱讀文章的時候，通常字數越少的字(代表比較重要的字)，比較容易成為答案，因此為了將這個想法放入模型當中，我們在word embedding後面除了接上Exact Match之外也接上「文字頻率 (Term Frequency) 」，結果使得 F1 Score 上升到0.53。



圖三：R-NET訓練在兩句的文章段落，並在一句的 Testing Data 上預測

說明:

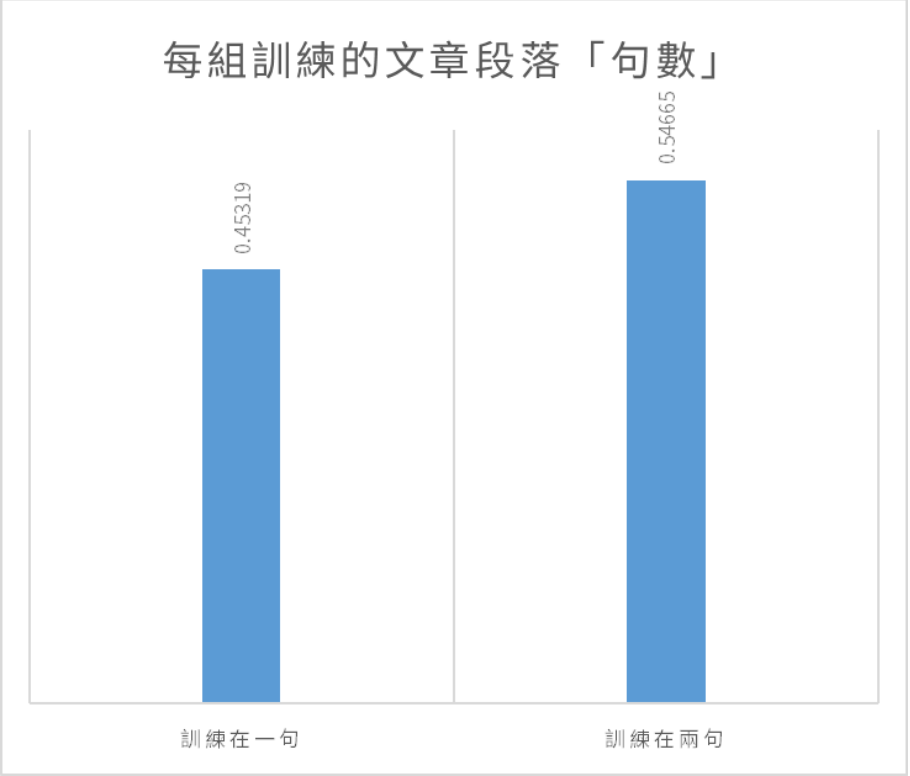
這個圖說明了 feature 對最終結果的影響非常大，尤其可以看到加上 Exact Match 之後，效果大增，我們認為是因為一個字若是出現在 Question 當中，代表這個字跟問題的相關度極大，因此使得 F1 Score 大大提升。另外，再多加上 Term Frequency 之後，可以再多增加一些效果，我們認為是因為重要的字的 Term Frequency 比較低。



圖四：訓練在兩句的Training data上的R-NET，預測不同長度的Testing data

說明:

我們發現同樣訓練在同樣文章段落為兩句的R-NET，在預測不同長度的文章段落時，聚焦答案的F1 Score是在預測兩句時最好，我們認為是因為文章段落如果太長，會造成神經網路無法聚焦在正確答案，而文章段落太短又會讓語意的表達不夠清楚以至於不好預測到正確的答案。



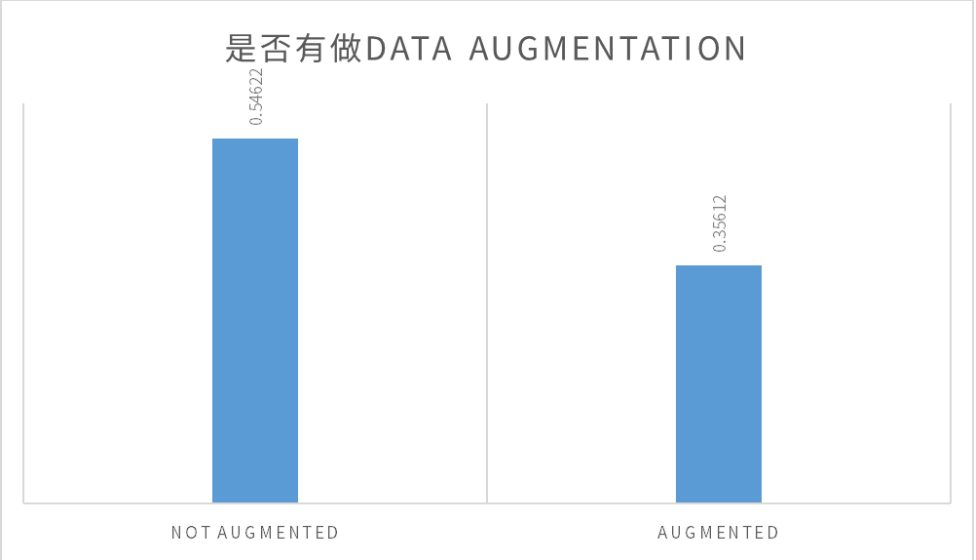
圖五：R-NET 訓練在文章段落只有一句和只有兩句的比較

說明:

可以發現，在使用同樣的參數、模型時，訓練在文章段落為兩句上的效果會比單單只有訓練在一句上的效果還要好，推測是因為訓練（證據？）

3. Data augmentation

我們發現大部分的 answer 都是名詞，或含有阿拉伯數字，因此設計簡單的 data augmentation 方法：如果 answer 含有數字，將每個 digit 改成 0-9 的 random integer；否則將 answer 抽換成另一個名詞。由於我們沒有找到 categorized Chinese words 的 knowledge base，因此下載了政府的 open data，手動去 parse 「本國專利技術名詞」，將這些當作可以替換的 answer。當每個「專利技術名詞」都使用一次過後，我們成功將 training data 擴大成 10 倍。但我們發現效果不好（如圖六），因此決定進行 error analysis，不要繼續瞎加 feature 或亂加 data。



圖六：R-NET 訓練在沒有 Augmented 的資料與 有 Augmented 的資料比較

說明：

我們發現，將R-NET模型訓練在沒有 Augmented 的資料上效果會比較好，我們認為是因為 Augmented 的資料對於模型來說，Out Of Vocabulary和語意被置換的情形下會使得模型的語意理解被打亂（證據？跑一些換字實驗看機率）

4. Error analysis

以下的圖片當中，紅色文字為 OOV，藍字為我們的解讀。句子已經過 jieba 斷詞。

4.1. 斷詞錯誤

我們的 input 和 output 都是 jieba 斷詞的 word，但經過錯誤分析，發現人名、專有名詞容易被 jieba 斷開，使 model 在選取答案的時候不會完整選到。或許可以使用 named entity processing（將 named entity 抽出，在 train 和 test time 的時候換成某個 vector，之後再將 output named entity 文字換成原本的）。

參考問題：在項立志以及何人的率領下.....
參考解答：「在 | 項 | 立志 | 、 | 董 | 在華 | 率領 | 下 | 」 → 「董在華」

「在 | 項 | 立志 | 、 | 董 | 在華 | 率領 | 下」→ 「董」
「在 | 項 | 立志 | 、 | 李 | 老師 | 率領 | 下」→ 「李」：sidenote--「老師」這個字似乎會被忽略
「在 | 項 | 立志 | 、 | 老師 | 李 | 宏毅 | 率領 | 下」→ 「李」
「在 | 項 | 立志 | 、 | 李 | 老師 | 宏毅 | 率領 | 下」→ 「李」
「在 | 項 | 立志 | 、 | 本田 | 彌 | 兵衛 | 率領 | 下」→ 「本田彌」
「在 | 項 | 立志 | 、 | 毛澤東 | 率領 | 下」→ 「毛澤東」：當 named entity 被 word vector 完整呈現的時候，model 答題正確

圖七：斷詞不當影響 model performance

4.2. OOV 的影響

我們發現，OOV 似乎會切斷 model output，有點像是閱讀一段文字時，看到不懂的單字後「傻住」。可能的解決方式有使用更大的 corpus （我們這次只有使用 train、test set）去訓練 word vector，或是融合 character embedding。

參考問題：「自哪個時期.....」
參考解答：「唐 | 宋 | 以來」 → 「唐宋」

「我 | 偷偷 | 哈哈 | 以來」 → 「我偷偷哈哈」
「我 | 認真 | 哈哈 | 以來」 → 「我認真哈哈」
「我 | 偷偷 | 認真 | 哈哈 | 以來」 → 「我偷偷認真哈哈」
「我 | 開始 | 哈哈 | 以來」 → 「哈哈」
「我 | 開始 | 用力 | 哈哈 | 以來」 → 「用力哈哈」
「我 | 開始 | 玩 | 遊戲 | 以來」 → 「玩遊戲」
「我 | 開始 | 大便 | 以來」 → 「我」 ：此 OOV 似乎會切斷句子
「我 | 開始 | 大便 | 哈哈 | 以來」 → 「哈哈以來」
「我 | 開始 | 大便 | 用力 | 哈哈 | 以來」 → 「用力哈哈」
「阿拉 | 巴 | 古奇 | 蒙 | 時代 | 以來」 → 「蒙時代」 ：別的 OOV 似乎也會切斷句子

圖八：OOV 對 output 的影響

4.3. 無法判斷 negation

當我們在文字敘述中加入否定詞時，發現 model 無法正確考慮到否定現象，output 錯誤答案。我們想到的一種解決方式是用 dependency parsing，將被否定的詞後面加上一個 bit 的 binary feature，使 model 明確接收這個資訊。

參考問題：在項立志以及何人的率領下.....
參考解答：「在 | 項 | 立志 | 、 | 董 | 在華 | 率領 | 下 | 」 → 「董在華」

Question：1952 | 年時 | 在 | 何人 | 的 | 率領 | 下對 | 黃河 | 河源 | 進行 | 了 | 數月 | 的 | 勘查 | ? |
Segment：1952 | 年 | ， | 不 | 是 | 在 | 毛澤東 | ， | 是董 | 在華 | 的 | 率領 | 下 | ， | 對 | 黃河 | 河源 | 進行 | 了 | 數月 | 勘查 | 。 |
Model output：毛澤東

圖九：「不」被 jieba 斷開，model 沒有學會考慮到它代表的意思