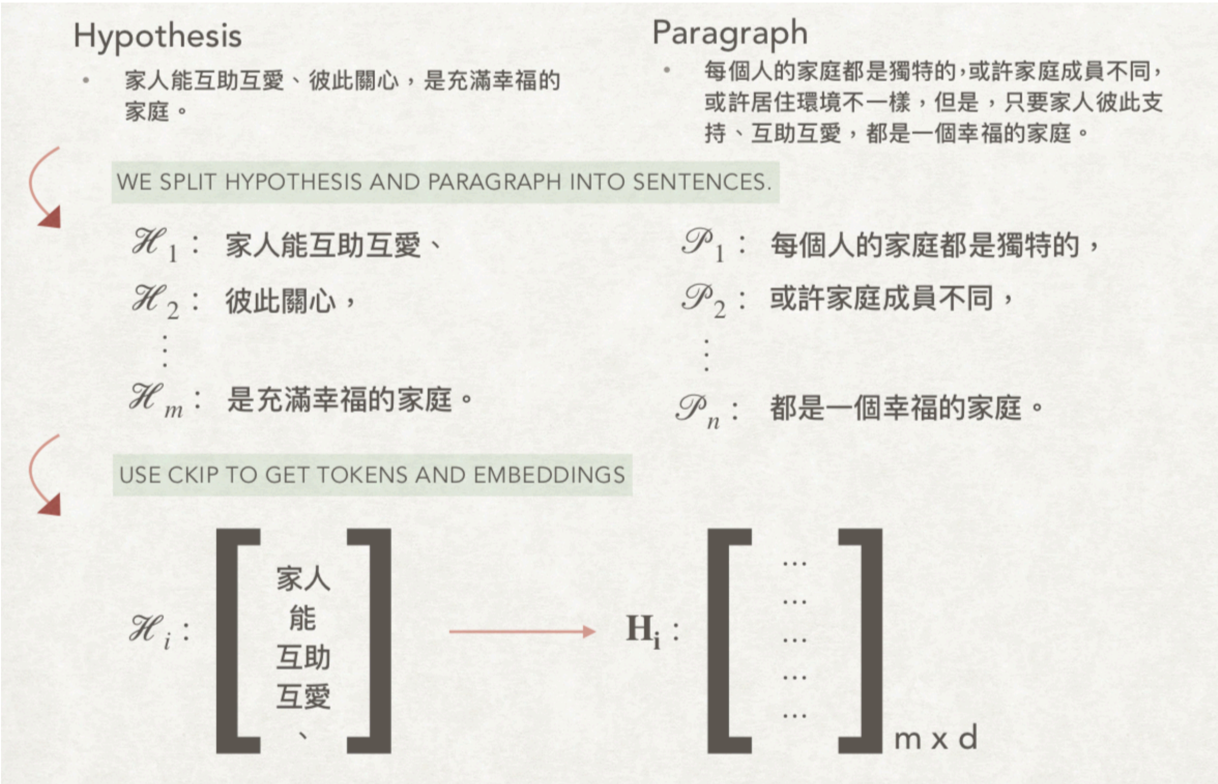
Technical Report
Supporting Evidence Retrieval on SSQA

最近更新時間：2018/09/13 周良冠

Data Formulation (Hypothesis & Paragraph -> matrices)

For each data, there are two kinds of content. One is called “hypothesis”, indicated by $\mathscr{H}$; and the other is called “Paragraph”, indicated by $\mathscr{P}$. Both of them are sets of sentence, $\mathscr{H}_i$ and $\mathscr{P}_j$, where i and j are their orders in the contents. For each sentence $\mathscr{H}_i$ and $\mathscr{P}_j$, we turn them into matrices $\mathbf{H_i}$, $\mathbf{P_j}$ by tokening and word embedding. Therefore, every sentence is represented by a matrix, and each hypothesis and paragraph are both a set of matrix.



$\mathscr{H} = \{\mathscr{H}_1, \mathscr{H}_2, \ldots, \mathscr{H}_m\}$ $\quad$ $\mathscr{P} = \{\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_n\}$

$\mathscr{H}_i$ -> $\mathbf{H_i}$ $\quad$ $\mathscr{P}_j$ -> $\mathbf{P_i}$

（這張圖有錯，$\mathbf{H_i}$ 並不是m x d，而是要看這句話有幾個tokens。）

Model Structure

The structure of the model can be split into three parts: sentence embedding, fusing and function. (名字暫定). We’ve tried different method in each part.

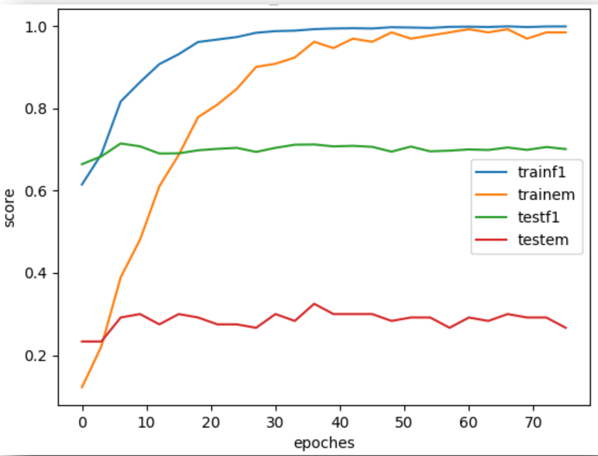
1. **Sentence Embedding**: As mentioned above, a hypothesis (so as to paragraph) is represented by a matrix. This part is aimed to condense the matrix into a vector, which is called sentence embedding. For formulation, we can write $\mathbf{h_i} = \mathbf{sent\_emb}(\mathbf{H_i})$, where lower case indicates vector. We use the last hidden state of BiLSTM to be the embedding and the dimension of the hidden state is d. After this part, a sentence can be represented by an embedding vector $\mathbf{h_i}$ (or $\mathbf{p_j}$), and it should contain semantics meanings of the corresponding sentence.

2. **Fusing**: This part takes in sentence embeddings of both hypothesis and paragraph and generate scores for each sentence of paragraph. That is, each pari of $\mathbf{h_i}$ and $\mathbf{p_j}$ will go through a scoring function $\alpha()$ and generate s_{ij}. $\alpha()$ could be any trainable function. In our case, we simply compute the inner product. Then, we compute $s_j = \beta(s_{ij}),\ where\ i \in [1,m]$. We have tried two different $\beta()$, one is mean function and the other is max function. We will compare the difference in the results.
3. **Function**: This part takes in scores s_j and output the probability of the corresponding $\mathbf{p_j}$, which indicate how possible $\mathbf{p_j}$ is the supporting evidence for $\mathscr{H}$. We use two different functions to transform scores into probability, $Sigmoid()$ and $tanh(abs())$. Having probabilities, we can use negative log likelihood as our loss function.

Result

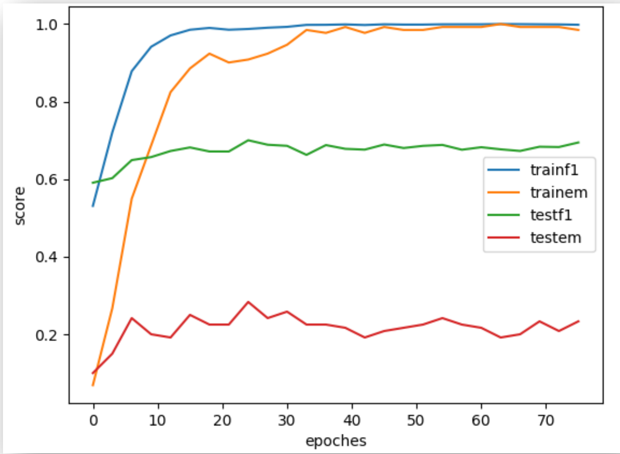
We evaluate results by F1 score and exact match score.

從下面兩張圖可看出，training data的曲線很快速地上升，然而testing data幾乎沒有成長的趨勢，這不是一個好結果。其中可能是因為model設計錯誤，另一個可能原因是data太少導致overfit。



$\mathbf{max}(\alpha(\mathbf{p_j}, \mathbf{h_{1:m}}))$

$a_j = Sigmoid(s_j)$



$\mathbf{mean}(\alpha(\mathbf{p_j}, \mathbf{h_{1:m}}))$

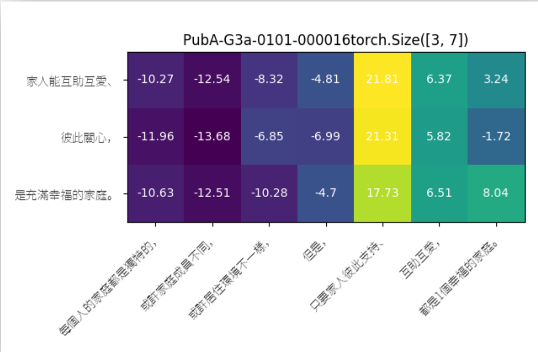
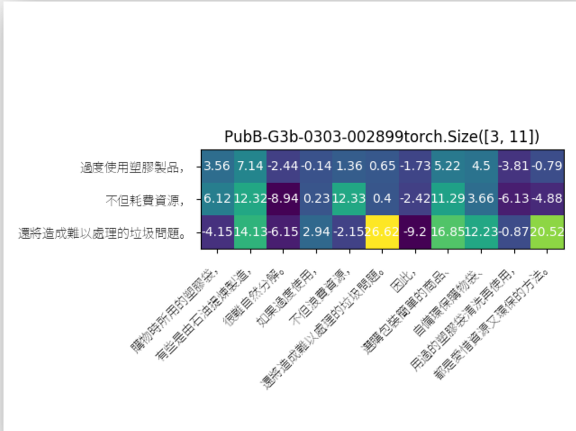
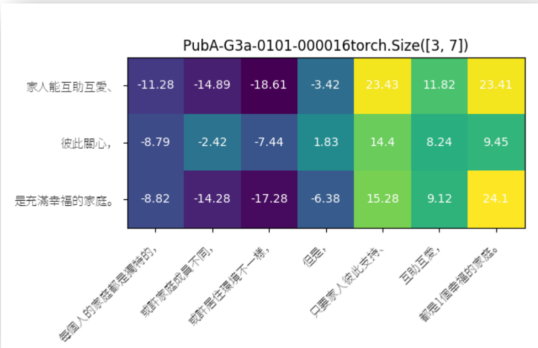
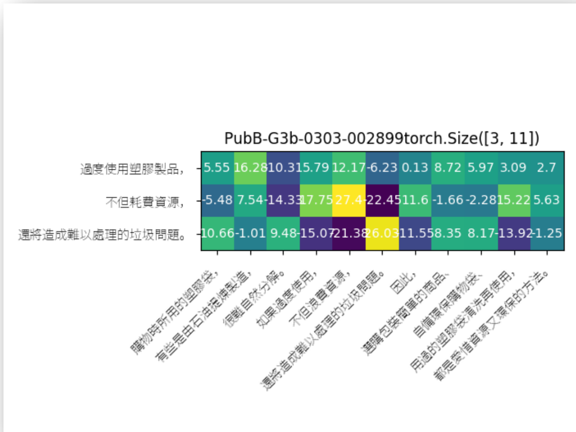
$a_j = Sigmoid(s_j)$

後來做了錯誤分析，去看句子與句子之間的分數是不是真的我們預期的那樣，有相關的句子分數就會高，不相關的句子分數就會很低。
我們分別看training和testing的圖發現，在training的圖中，分數是很離散的，因為model就是照著training data學的，但testing的分數就沒有分的很清楚，很多分數都落在模稜兩可的值，儘管如此，我們依然能夠觀察到，如果兩個句子意思相近的話，他們彼此的分數也會傾向高一點。

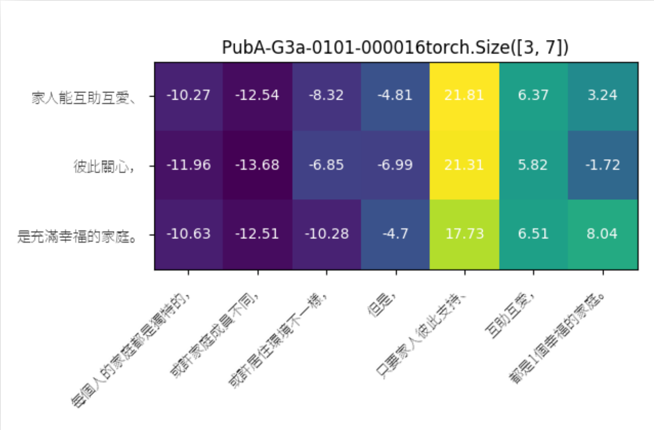
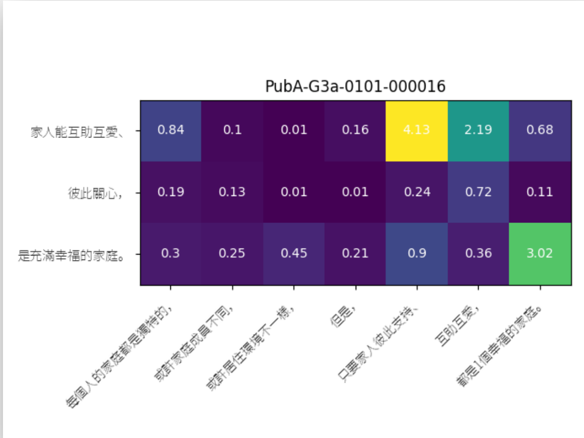
另外，我們觀察到，paragraph中的句子如果是supporting evidence，通常只會和hypothesis裡面的某一句子有關聯。如果$\beta()$使用mean，那麼即使paragraph有個句子和hypothesis其中一個句子高度相關，只要它和其餘句子不相關，它的score就會被其餘句子稀釋掉。因此，我們認

為max會比mean合理。從下圖也能發現，如果使用mean，分數會傾向更均勻；相反的，使用max，分數會較為離散。

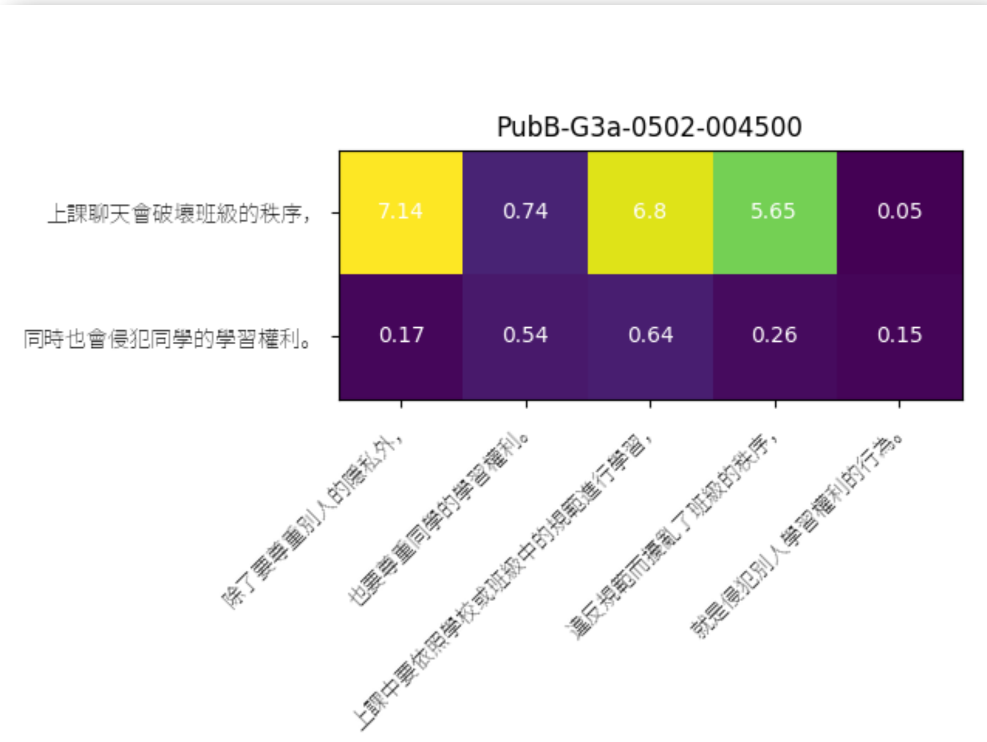
除此之外，我們意識到，計算兩個向量的相關程度時，越不相關分數不應該是越低，而是趨近於0。如果分數是個非常小的負數，應該也能夠算是某方面的相關。因此，我們嘗試在score之後加絕對值，讓分數的最小值為0。但這麼做會遇到一個問題，因為Sigmoid(0)=0.5，如此一來所有預測機率都會大於0.5。為了解決這個問題，將Sigmoid改為tanh。（這實在是非常不可取，當初只是為了跑出結果，先擋著用，老師表示：必須了解機率真正的意義）

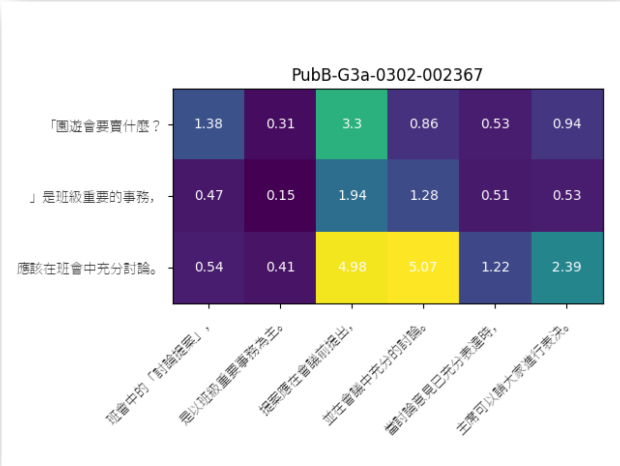
	Training	Testing
Max Sigmoid		
Mean Sigmoid		

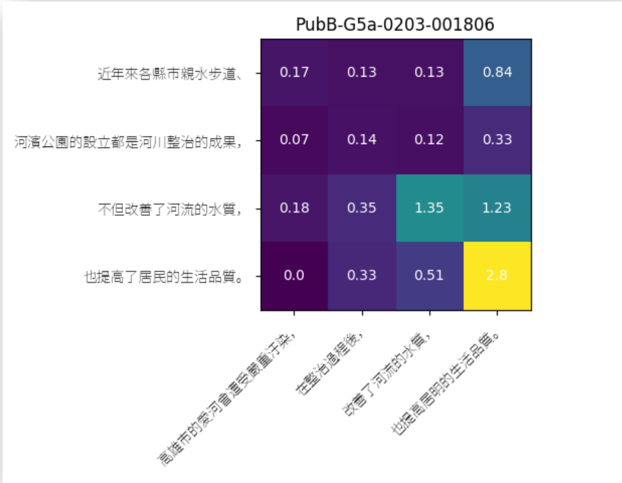
後來實習接近尾聲，也沒多少時間測試了，於是做了max+tanh(abs())的實驗。training set是訓練集與發展集併在一起，而testing set就只有測試集。跟之前不同的是，之前train的時候都只有用到訓練集，想說data多一點，或許會好一點（但依然不多）。實驗結果的分數趨勢圖如下，過程中model卡住了，重新執行後分數才開始往上爬，從圖可以看出，結果依然不佳。不過，新的model產生的分數矩陣圖比起之前更合理。下兩張圖為訓練資料其中一筆的分數，可以看出使用絕對值讓句子與句子之間的分數更合理。

max+Sigmoid	max+tanh+abs
	

在testing data中也可以看得出來。







當然也有不太合理的例子。

一些個人想法與未來發展

1. code是用pytorch寫的，因為我整理資料的方式讓input data不是每一筆都是一樣形狀的tensor，導致我無法善用pytorch平行的特性，在model裡面刻了一些很醜的for迴圈，跑起來超級慢，建議要把code大大翻修。
2. 我覺得hypothesis和paragraph的命名不太好，尤其是paragraph，改成knowledge應該比較好，但因為一直以來都這樣稱呼，後來就沒有改了。
3. 從分數轉換到機率的公式是不太合理的，待修改。
4. 有label的資料真的太少了，或許可以試試unsupervised或semi-supervised。
5. 目前的方法其實是完全依賴sentence embedding的，但是這個model產生sent_emb的機制並不會考慮整個文章，而是每個句子彼此獨立，這並不合理。而且可能有種情況是，多個句子組起來才有意義，單獨存在很難看出其意義。
6. 老師的建議是，直接排列組合出所有可能的candidate，例如一段paragraph有五個句子abcde，那麼candidate就是，a, b, c, d, e, ab, bc, ..., abcde。然後直接去比這些candidate是supporting evidence的可能性。