ADL Final Project Report

Machine Comprehension (MC)

B01901073 李佳軒
R04942056 余朗祺
R05942066 張瀞婷
Codalab 帳號：r04942056

本次的 task 是 machine comprehension, 目的是讓機器可以理解文章內容，並回答對應的問題。使用的資料為 Squad 和 TOEFL 的聽力問答。本報告架構如下：先介紹我們使用的 model 架構，再略述我們實作的 code 架構，接著是實驗細節和實驗結果，再來是結果分析，最後是結論。

Model Architecture

Question Answering is a rather mature challenge in Natural Language Processing. In this project, it is a single choice problem. The task demands the machine to hold the ability to understand article and choose from four choices after reading the question. The state-of-the-art is memory network (including its multiple variants).

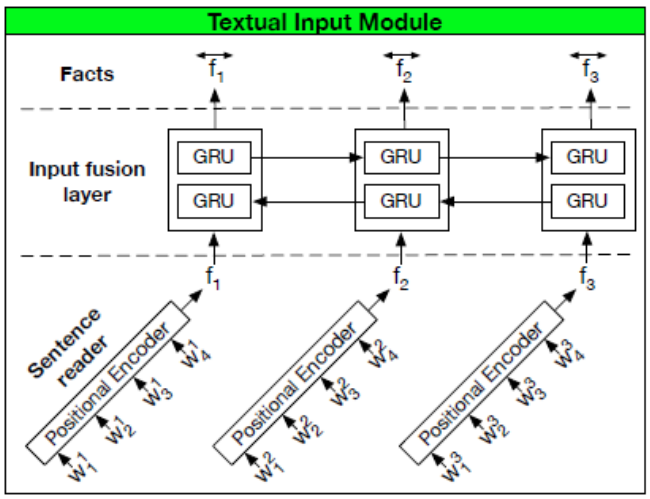
Memory Network features (1) attention (2) long-term-like memory.

(1) Questions trigger an iterative attention process which allows the model to condition its attention on the inputs and the result of previous iterations. (2) And the result computed by the hierarchical recurrent sequence model is like a long term memory.

In this project, we build the model from Dynamic Memory Networks for Visual and Textual Question Answering. There are four modules in this model.

Input Module:

The first component is a sentence reader, responsible only for encoding the words into a sentence embedding. The second component is the input fusion layer, bi-directional GRU for sentences embedding because it allows information from both past and future sentences to be used.



Question Module:

This module computes a vector representation of the question, which is the final hidden state of a GRU over the words in the question.

The Episodic Memory Module:

The high level goal of this module is to produce an embedding that contains the meaning of the whole article and question. It receives the fact vectors(the output of the input module) and q vectors (the output of the question module) .

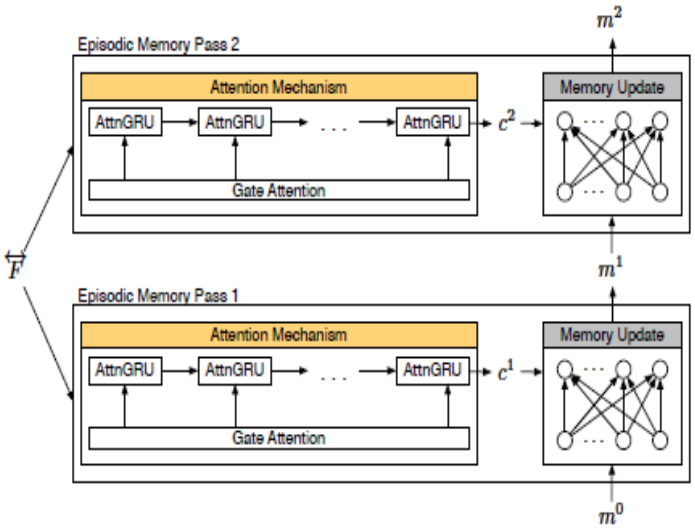


Figure 4. The episodic memory module of the DMN+ when using two passes. The $\overleftrightarrow{F}$ is the output of the input module.

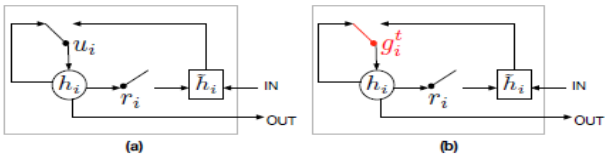


Figure 5. (a) The traditional GRU model, and (b) the proposed attention-based GRU model

The way the model implements the attention mechanism is called *Attention based GRU.* The original update gate of GRU is replaced by the computed g. so the hidden state updating formula becomes

$$h_i = g_i^t \circ \tilde{h}_i + (1 - g_i^t) \circ h_{i-1}$$

$$z_i^t = [\overleftrightarrow{f_i} \circ q; \overleftrightarrow{f_i} \circ m^{t-1}; |\overleftrightarrow{f_i} - q|; |\overleftrightarrow{f_i} - m^{t-1}|]$$

$$Z_i^t = W^{(2)} \tanh\left(W^{(1)} z_i^t + b^{(1)}\right) + b^{(2)}$$

$$g_i^t = \frac{\exp(Z_i^t)}{\sum_{k=1}^{M_i} \exp(Z_k^t)}$$

The g value is computed by a series of math computation. But the spirit here is still attention mechanism and quite intuitional. We look at previous memory and fact vectors to decide the focus of our reading. The final hidden state of the attention based GRU will become the contextual vector C. I think this is the most innovational part of this paper. Because the old memory network usually generate contextual vector C by weighted sun g over fact vectors F. This improvement allows the attention mechanism to be sensitive to both the position and ordering of the input facts.

Episode Memory Updates

$$m^t = ReLU\left(W^t[m^{t-1}; c^t; q] + b\right)$$

The final hidden state of attention GRU is the new information. In each pass we take a look at the previous episodic memory and the new information and decide the new episodic memory. This procedure can be operated several times (we generate different episodes of memory). These times are called “hops” or “passes”. The hop mechanism is inspired by the

intuition that we might recursively query information from long story to solve a problem. The example below explains the idea perfectly.

Question: Where was Mary before the Bedroom?
Answer: Cinema.

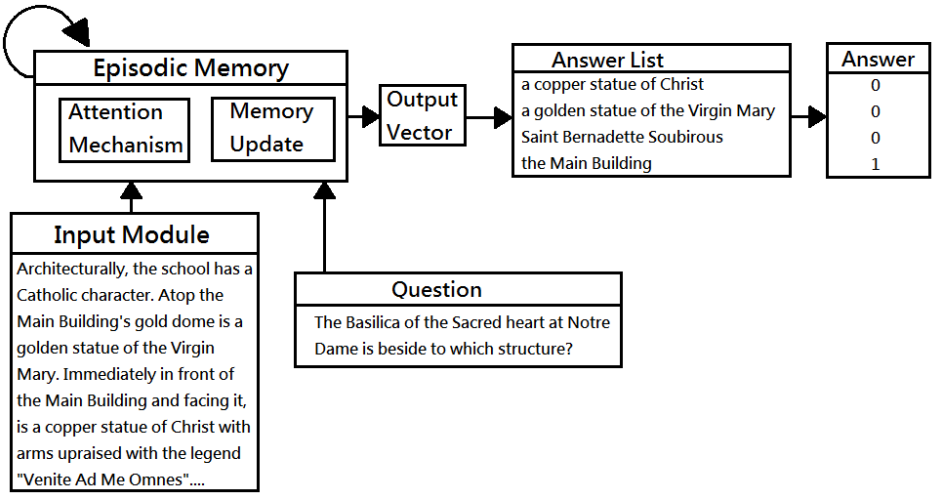
Facts	Episode 1	Episode 2	Episode 3
Yesterday Julie traveled to the school.			
Yesterday Marie went to the cinema.		■	
This morning Julie traveled to the kitchen.			
Bill went back to the cinema yesterday.			
Mary went to the bedroom this morning.	■		
Julie went back to the bedroom this afternoon.			
[done reading]			■

Table 5. An example of what the DMN focuses on during each episode on a real query in the bAbI task. Darker colors mean that the attention weight is higher.

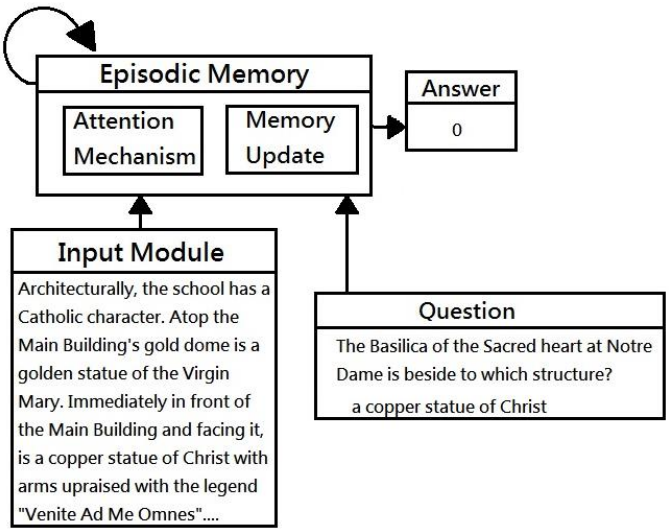
While there is paper suggesting that using different weights for each pass through the episodic memory may be advantageous. But since the task TOEFL training data size is too small so we choose the "tied weights version", that is we use the same parameter in every episode.

Answer Module (CHOICE VERSION)

This is where the original DMN+ different from our implementation. Our implementation works as follows: First, we encode the four choice sentences and then compute cosine similarity with the episode memory module output respectively. The highest score will be our predicted answer.



YES-OR-NO VERSION



This version is different from the choice version in three places:
First, each choice sentence is concatenated with the question sentence as a new question sentence.
Second, in the answer module we put the output of the episode memory module into one layer multiple perceptron. The output is a scalar between 0 to 1(since the activation function is sigmoid). This scalar can be viewed as a confidence score of the model predicting the choice to be true.
And we predict the choice to be the answer if the output scalar is >0.5. The two classes are [0,1] corresponds to [no, yes].
Third, the loss function is thus binary cross entropy.
The above is what we do in the training phase. When predicting answers, we use batch_size 4 and compare their confidence score. Argmax to get the resulting choice.

Code Description

Most functions are quite straight-forward if you read carefully through the model architecture introduction. So only explains special functions in specific files below :

toefl_input.py, squad_input.py

These two files are only different in the input files.

def init_babi(fname):

The parser directly load data from data json file and store it as list of dictionary.

task = {"C": "", "Q": "", "A": "", "S": ""}

def load_glove(dim):

We use glove vector to initialize the words in the data.

def get_lens(inputs, split_sentences=False):

def get_sentence_lens(inputs):

And since every articles are different in sentence numbers and each sentence is different in length. We need to pad the sentences to align the longest sentence/article. Then they can be feed to GRU.

squad_plus.py toefl_plus.py

These two files are only different in the input files and hyperparameters

class Config(object):

"""Holds model hyperparams and data information."""

def _position_encoding(sentence_size, embedding_size):

"""Position encoding described in section 4.1 in "End to End Memory Networks" (http://arxiv.org/pdf/1503.08895v5.pdf)"""

This function encodes the position of words within the sentence. The order of the words now affects our model.

def add_loss_op(self, output):

Since it's a four choice problem, we can think it as a classification problem. And to prevent overfitting, we use l2 regularization loss. add l2 regularization for all variables except biases

def add_training_op(self, loss):

optionally cap and noise gradients to regularize

Adds gradient noise as described in http://arxiv.org/abs/1511.06807. The input Tensor `t` should be a gradient. The output will be `t` + gaussian noise. 0.001 was said to be a good fixed value for memory networks.

Cap gradient simply bound the gradient value.

實驗數據與分析

由於 Codalab 上傳次數的限制，為了分析不同參數下 model 的表現，也為了增進 model performance，我們訓練 model 時使用 10-fold cross validation，也就是將 training data 分為 10 等份，每次取其中一分作為 validation set，最後將 10 個 validation accuracy 平均，作為該 model 的 performance measure。下列各項表現都是這種方式算的。

我們 model 固定的參數與其值如下：batch size 為 32, embedding size 和 hidden state vector size 為 100, 總共跑 256 個 epoch, gradient 上限為 10，最多一次 input 100 個句子，使用預先訓練好的 100 維 GloVe (glove.6B.100d.txt)作為 word embedding。這些數據之所以不改動，有些是因為我們參考用 code 本身的限制，有些是因為改動後影響結果不大，為簡化分析，故固定之。

可以調整的參數包含：initial learning rate, number of hops, dropout rate，是否將 dropout 套用至各別 GRU 上。以下表格列出 TOEFL 或 SQUAD 各種不同參數 model 的表現：

TOEFL

	Initial learning rate	#hops	Dropout	Drop GRU or not	Validation accuracy
(a)	0.01	2	0.1	False	0.303
(b)	0.01	2	0.3	False	0.427
(c)	0.01	2	0.45	False	0.459
(d)	0.01	2	0.5	False	**0.474**
(e)	0.01	2	0.6	False	0.396
(f)	0.1	2	0.5	False	0.455
(g)	0.01	1	0.5	False	0.388
(h)	0.01	3	0.5	False	0.443
(i)	0.01	2	0.5	True	0.46
(j)	0.01	2	0.3	True	0.414

比較(a)~(e), dropout 太低的結果(a)會比太高(e)離理想值(d)差很多。這大概是因為 TOEFL 的 training data 量相較 SQUAD 太少了，很容易產生 overfitting 問題，這也讓 dropout 在這個 task 顯得很重要。然而，如果進一步 dropout 各個 GRU，比較(d)(i)和(b)(j), 可發現 drop GRU 的話都會降低 accuracy, 可能是因為這樣會造成整體 dropout 比例太高，資訊損失太多(像是 GRU 的 memory 部分)，故我們選擇不 dropout GRU. 另外，我們也針對 memory network 中的 hop 數做調整，(g)(d)(h)分別是 1~3 hop 數的結果，可發現 2 個 hop 數有最好的表現。最後，我們有嘗試比較不同初始 learning rate 的結果，但因時間關係只有另外試 0.1，結果略遜於理想。

另外，我們尚有嘗試更動一些原本固定的參數，因為時間關係並沒有做系統性的完整實驗，只能大致分析之。數據如下。

hop	dropout	lr	l2	batch size	Embed size	hidden_size	train_data	drop_gru	run 1	run2	run3	
2	0.55	0.01	0.001	32	300	300	670	FALSE	Best_acc 0.3125	Best_acc 0.28125	Best_acc 0.34375	
2	0.5	0.01	0.001	32	200	50	670	TRUE	Best_acc: 0.31			
1	0.3	0.001	0.1	10	50	50	670	FALSE	Overfit			
1	0.5	0.01	0.005	32	300	75	670	TRUE	0.28			

參照第三項，若 L2-norm 太大、初始 learning rate 太小，似乎會產生嚴重 overfitting. 比較一項和二項，hidden size 變小與 dropout GRU 也會讓結果變略差。大致看來，若讓 hidden size/embedding size 變大，表現應該也會變好，但與上面的數據相比，size 為 200 或 300 似乎結果會變差(原本固定值都是 100)，推測這應該是因為 TOEFL 資料量不夠，若 vector size 太大，資訊會變太分散，以致於學習效果變差。如果資料量足夠，那麼較大的 vector size 應該也會有較好結果，雖然學習時間也會因此變得較長。

SQUAD

	Initial learning rate	#hops	Dropout	Drop GRU or not	Validation accuracy
(a)	0.01	2	0.1	False	0.35
(b)	0.01	2	0.3	False	0.474
(c)	0.01	2	0.5	False	**0.521**
(d)	0.01	1	0.5	False	0.33
(e)	0.01	3	0.5	False	0.425
(f)	0.01	2	0.5	True	0.519

因為 SQUAD 資料量較多，training 花費時間較久，故我們直接參考 TOEFL 的結果選擇比較需要嘗試的參數組以節省時間。基本上，dropout 表現(a~c)和 hop 數表現(d~e)與 TOEFL 部分相去不遠，值得注意的是 dropout GRU 部分，(f)的表現與最佳表現(c)其實是差不多的，可見 dropout rate 高一些對 SQUAD 而言是可接受的。因為時間關係我們沒有嘗試更高的 dropout rate, 但相信應該是可以得到更好的結果的。

結論

我們改寫了 Dynamic memory network 來處理本次 machine comprehension 問題，提出兩種可能架構：(1)是非題，(2)選擇題，其中選擇題架構的結果比較好，在 SQUAD 和 TOEFL 問題 validation set 上皆得到約五成的正確率。可惜因為時間關係，我們在 codalab 上只有是非題架構的結果，TOEFL 正確率約三成(達標)，SQUAD 正確率只有 25%。不過，最後在報告截止前我們還是得到了滿意的結果，這次的 project 確實讓我們受益良多。

參考文獻

- Caiming Xiong, Stephen Merity, Richard Socher. Dynamic Memory Networks for Visual and Textual Question Answering. arXiv preprint arXiv:1603.01417
- Wei Fang, Jui-Yang Hsu, Hung-yi Lee, Lin-Shan Lee. Hierarchical Attention Model for Improved Machine Comprehension of Spoken Content. arXiv preprint arXiv:1608.07775
- Dynamic Memory Networks in TensorFlow (https://github.com/barronalex/Dynamic-Memory-Networks-in-TensorFlow)