



深度學習TensorFlow實務

循環神經網路

Lab5

-TA-

李偉弘

廖宜健

林佑昌

蔡明諺

彭冠偉

1. 快速回顧RNN

快速回顧 RNN

- 在深入 Sequence to Sequence 的細節之前，先和各位介紹一位老朋友——RNN (Recurrent neural network)
- 傳統上，我們假設神經網路的每個輸入是相互獨立的，意即對於輸入 I^i 、 I^j 而言， I^i 取什麼值，並不會影響 I^j 取什麼值，因為 I^i 與 I^j 沒有關係。但這個假設有個很大的缺點，就是在處理序列 (Sequence) 時不太管用，因為序列內的元素長幼有序、先後有別，這種順序性導致了輸入間彼此相依

快速回顧 RNN

- 股票走勢就是種典型的數值序列：

9 : 00	10 : 00	11:00	12:00
123	128	132	136

- 如果有人問：「不知道下午一點是會漲還是會跌」，我們多半會回答：「當然會漲，因為九點在漲，十點在漲，十一點也在漲」
- 當我們想預測十二點的股市指數時，不是選擇隨手丟枚骰子，而是選擇參考以前的股市指數，就說明了十二點當下的指數與十二點前的股票指數其實是有相依性的

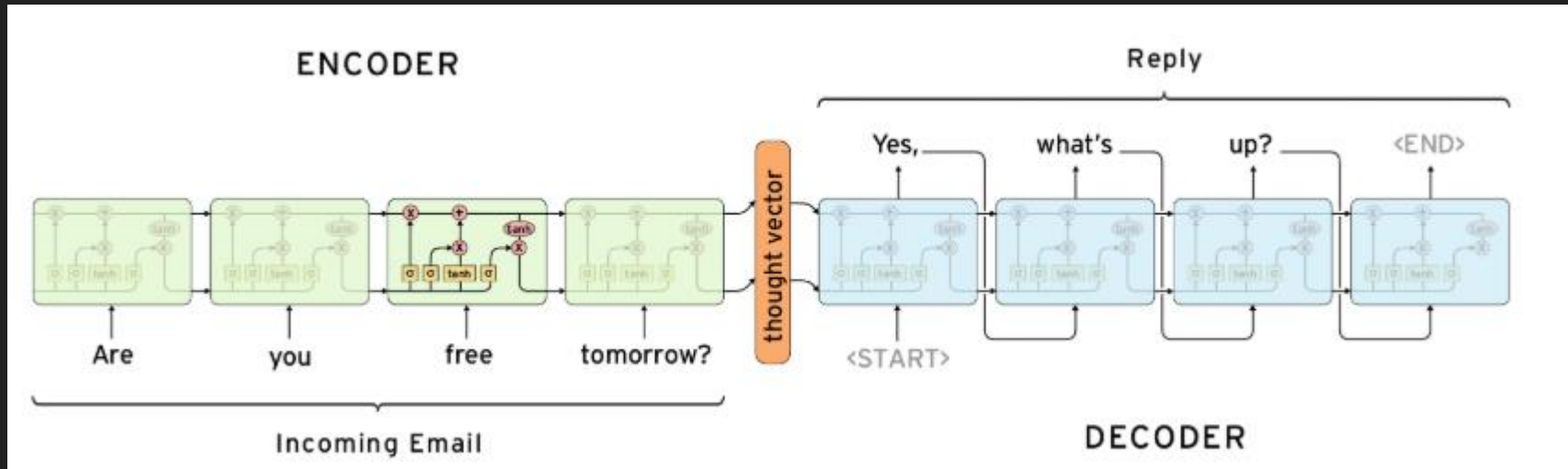
快速回顧 RNN

- 再以文字序列舉個例子，比如說同樣都是用到了「不」、「歡」、「喜」這三個字，但「喜歡不？」是一個男孩切切於心的期盼，而「不喜歡。」則是女孩流水無情的漠然。
- 這兩組序列有相同的構成，卻因順序，而讓彼此的結局殊如雲泥。

2. Sequence to Sequence

Sequence to Sequence

- Sequence to Sequence 是由 Encoder 與 Decoder 兩個 RNN 構成，它的運作原理其實與人類的思維很相似，當我們看到一段話時，會先將這句話理解吸收，再根據我們理解的內容說出回覆
- Encoder 就是負責將輸入序列消化、吸收成一個向量，我們通常把這個向量稱為 context vector；Decoder 則是根據 context vector 來生成文字



Sequence to Sequence

- 不過我們只有一個輸入，要怎麼生成出超過 1 個輸出呢？只要把目前的輸出當成之後的輸入就可以了

```
1 while True:
2     output = decoder(output)
3     outputs.append(output)
```

- 由於我們總是將前個輸出當成後個輸入，這個 while 不會結束，所以我們得設置一個終止信號 EOS (End Of Sentence)，告知 Decoder 到此為止就好：

```
1 while output != 'EOS':
2     output = decoder(output)
3     outputs.append(output)
```


Sequence to Sequence

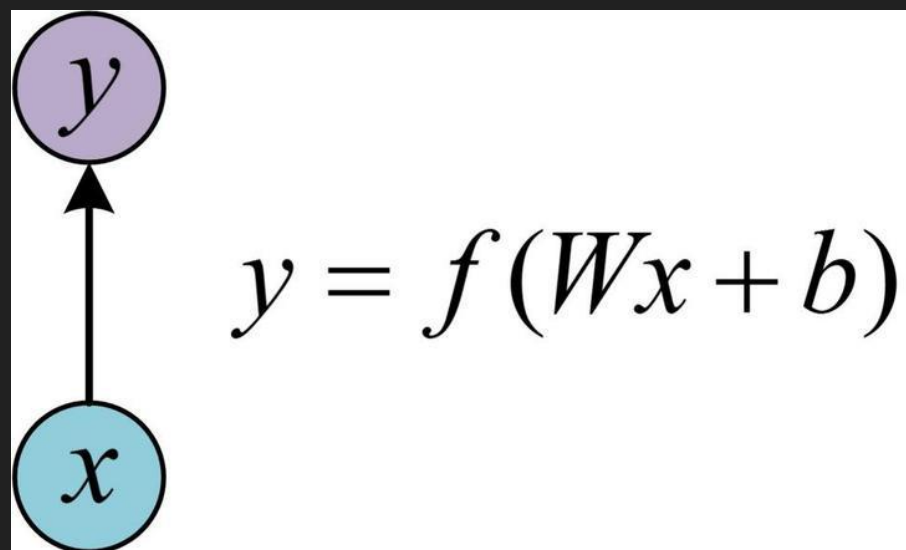
- 對於輸入序列 x_1, \dots, x_T ，與輸出序列 y_1, \dots, y_T 而言，透過 Encoder 我們能將輸入序列轉換成 context vector v ，我們希望能在 Decode 階段最大化條件機率
- $P(y_T) = \prod_{t=1}^T P(y_t | y_1, \dots, y_{t-1})$
- 對時間點 T 而言，模型知道已經聽到了什麼(v , context vector)，以及之前說了些什麼 y_1, \dots, y_{t-1} ，以這兩件事為基準，來評估現在該說什麼 y_T

Sequence to Sequence

- 總歸而言，Sequence to Sequence 的精華在串接了兩個 RNN，第一個 RNN 負責將長度為 M 的序列給壓成 1 個向量，第二個 RNN 根據這 1 個向量產生出 N 個輸出，這 $M \rightarrow 1$ 與 $1 \rightarrow N$ 相輔相成下就構建出了 $M \rightarrow N$ 的模型，能夠處理任何不定長的輸入與輸出序列，好比：
 - 輸入一句英文，輸出一句法文，就寫好了一個翻譯系統
 - 輸入一個問題，輸出一句回覆，就架好一個聊天機器人
 - 輸入一篇文章，輸出一份總結，就構成一個摘要系統
 - 輸入幾個關鍵字，輸出一首短詩，就成就了一名詩人

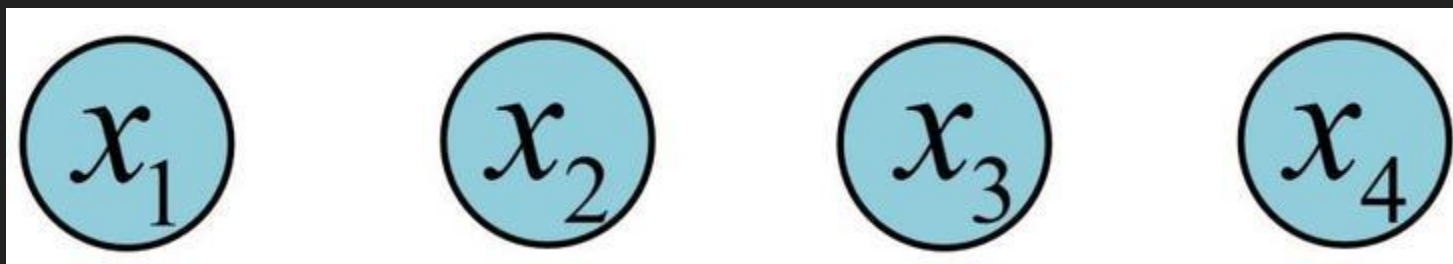
單層網路結構

- 在學習RNN之前，首先要了解一下最基本的單層網路
- 輸入是 x ，經過變換 $Wx + b$ 和激活函數 f 得到輸出 y



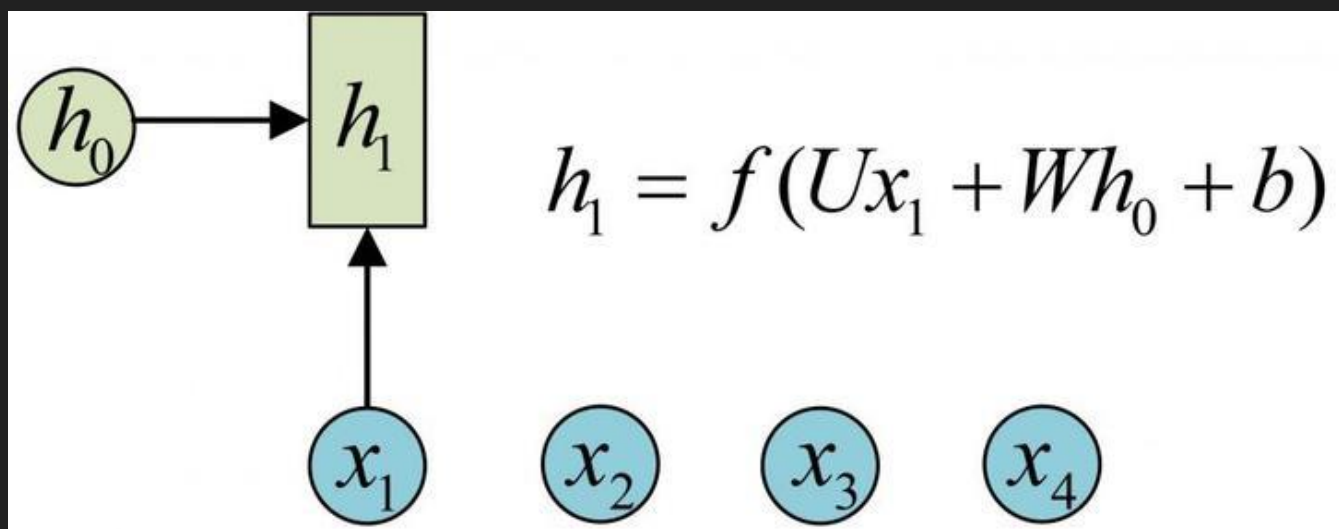
經典的RNN結構 (N vs N)

- 自然語言處理問題。X1 可以看做是第一個單詞，x2 可以看做是第二個單詞，以此類推
- 語音處理。此時，x1、x2、x3.....是每幀的聲音信號
- 時間序列問題。例如每天的股票價格等等



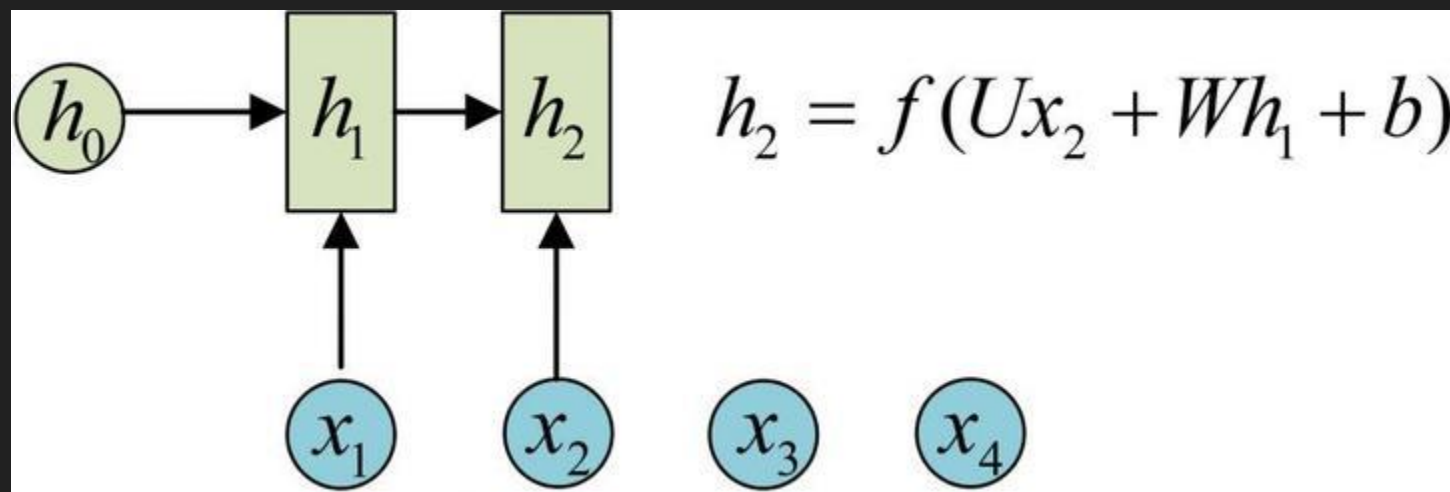
經典的RNN結構 (N vs N)

- 為了建模序列問題，RNN 引入了隱狀態 h (hidden state) 的概念， h 可以對序列形的數據提取特徵，接著再轉換為輸出
- 一個箭頭就表示對該向量做一次變換。如上圖中 h_0 和 x_1 分別有一個箭頭連接，就表示對 h_0 和 x_1 各做了一次變換



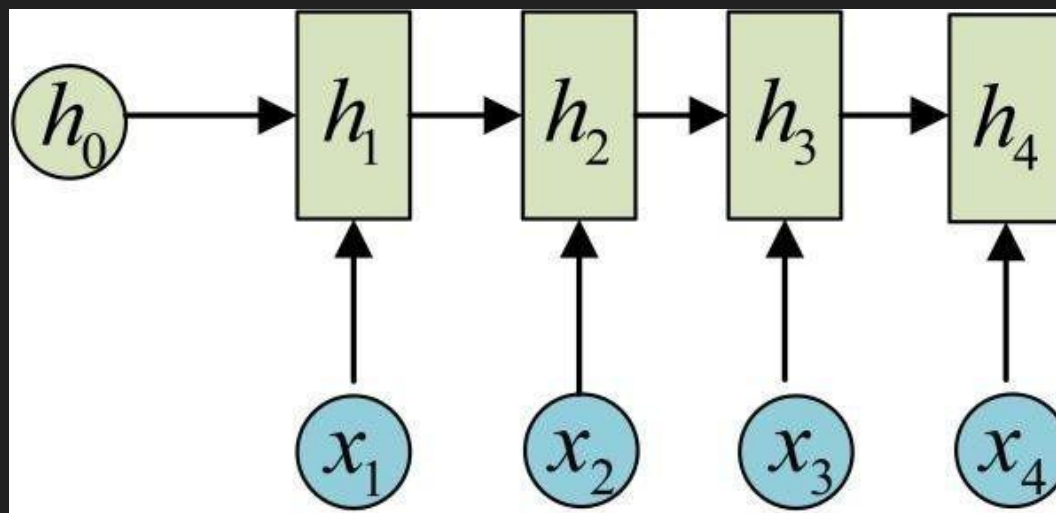
經典的RNN結構 (N vs N)

- h_2 的計算和 h_1 類似在計算時，每一步使用的參數 U 、 W 、 b 都是一樣的，也就是說每個步驟的參數都是共享的，這是 RNN 的重要特點，一定要牢記



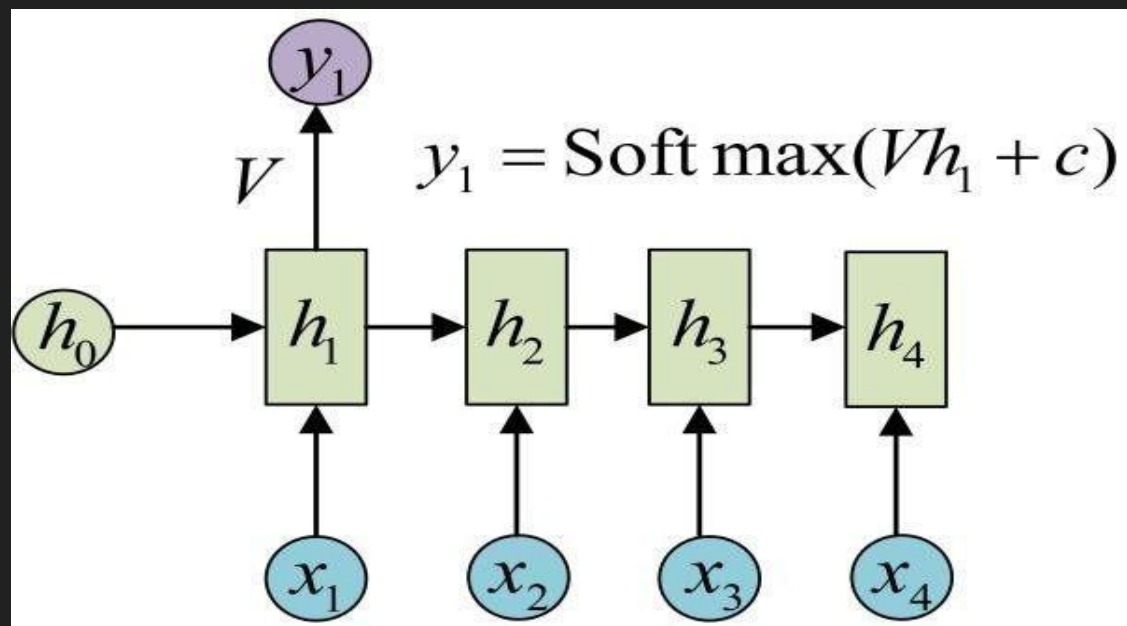
經典的RNN結構 (N vs N)

- 依次計算剩下來的 (使用相同的參數 U 、 W 、 b)
- 這裡為了方便起見，只畫出序列長度為4的情況，實際上，這個計算過程可以無限地持續下去



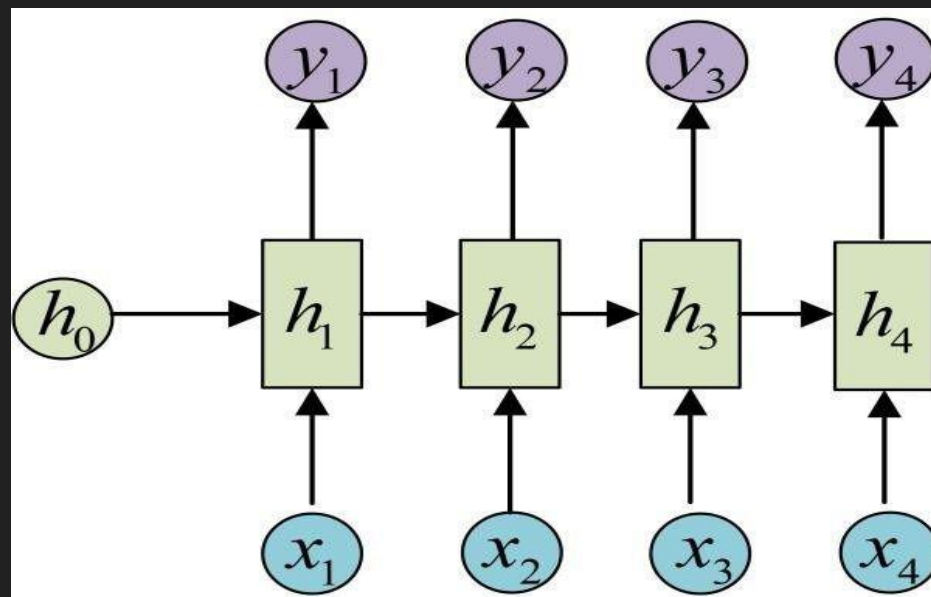
經典的RNN結構 (N vs N)

- 我們目前的 RNN 還沒有輸出，得到輸出值的方法就是直接通過 h 進行計算，這裡的這個箭頭就表示對 h_1 進行一次變換，得到輸出 y_1



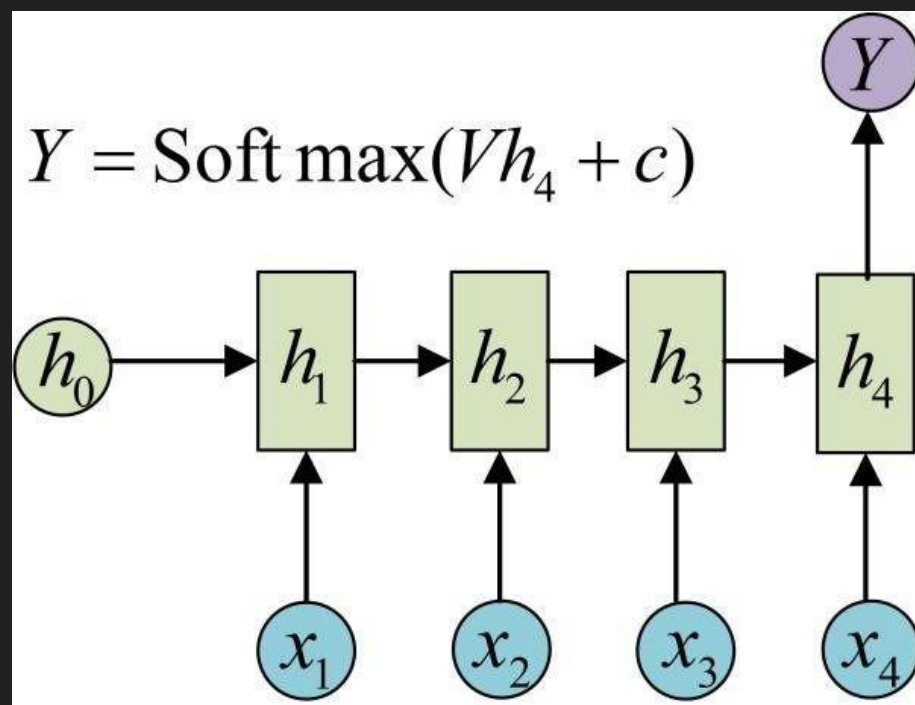
經典的RNN結構 (N vs N)

- 這就是最經典的RNN結構，它的輸入是 $x_1, x_2 \dots x_n$ ，輸出為 $y_1, y_2 \dots y_n$ ，也就是說，輸入和輸出序列必須要是等長的
- 由於這個限制的存在，經典RNN的適用範圍比較小



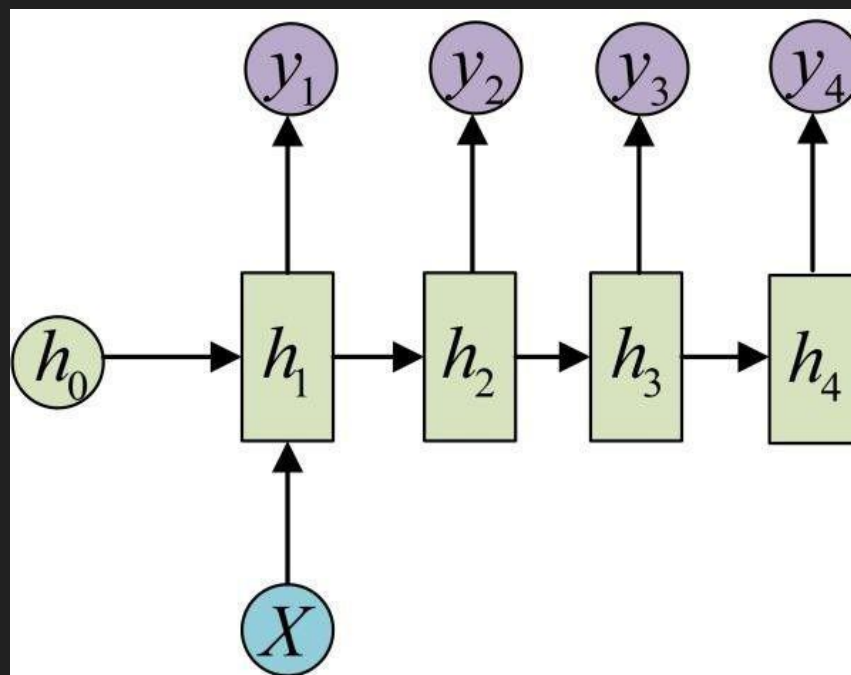
經典的RNN結構 (N vs 1)

- 我們要處理的問題輸入是一個序列，輸出是一個單獨的值而不是序列，只要在最後一個h上進行輸出變換就可以了
- 這種結構通常用來處理序列分類問題



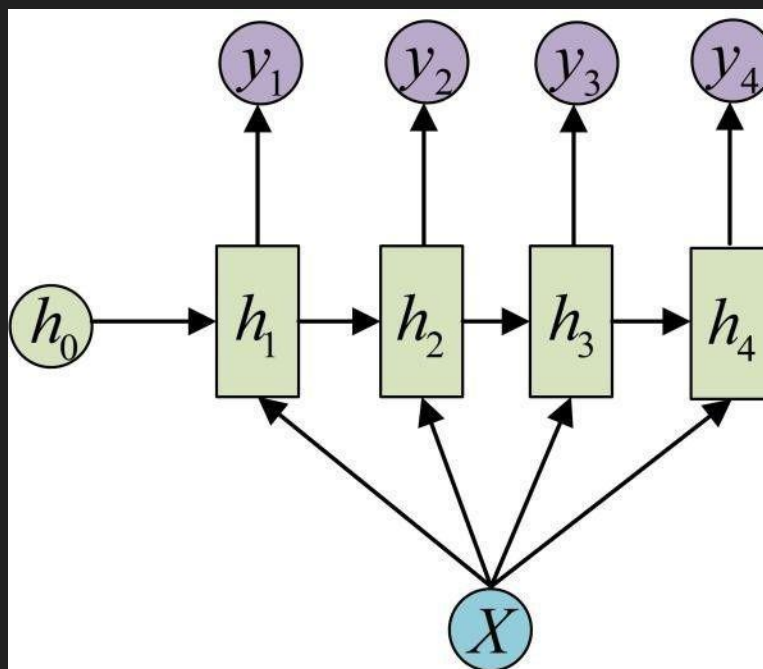
經典的RNN結構 (1 vs N)

- 輸入不是序列而輸出為序列的情況
- 可以只在序列開始進行輸入計算



經典的RNN結構 (1 vs N)

- 一種結構是把輸入訊息 X 作為每個階段的輸入
- 這種 1 VS N 的結構可以處理的問題：從圖像生成文字，此時輸入的 X 就是圖像的特徵，而輸出的 y 序列就是一段句子

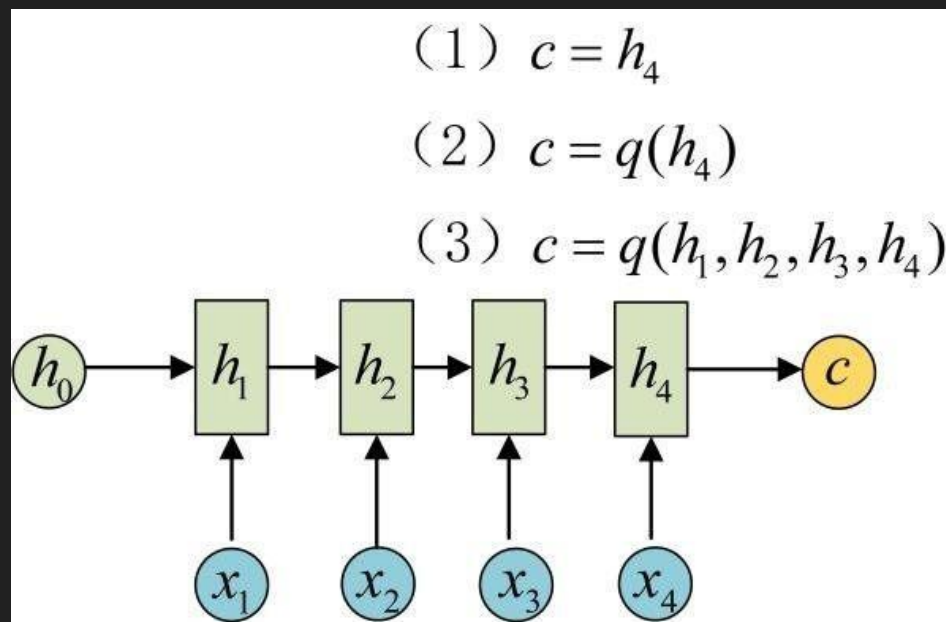


RNN變體 (N vs M)

- RNN最重要的一個變種：N vs M，這種結構又叫Encoder-Decoder模型，也可以稱之為Seq2Seq模型
- 原始的 N vs N RNN要求序列等長，然而我們遇到的大部分問題序列都是不等長的，如機器翻譯中，源語言和目標語言的句子往往並沒有相同的長度
- 這種 Encoder-Decoder 結構不限制輸入和輸出的序列長度，因此應用的範圍非常廣泛，例如：機器翻譯、語音識別、閱讀理解

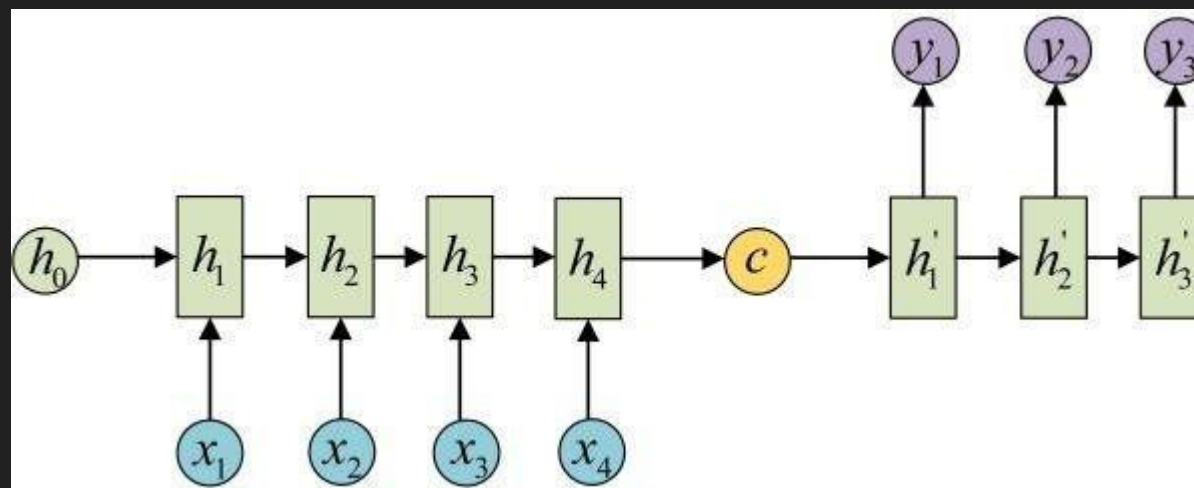
RNN變體 (N vs M)

- Encoder-Decoder 結構先將輸入數據編碼成一個上下文向量 c
- 得到 c 有多種方式，可以把 Encoder 的最後一個隱狀態賦值給 c ；還可以對最後的隱狀態做一個變換得到 c ；也可以對所有的隱狀態做變換



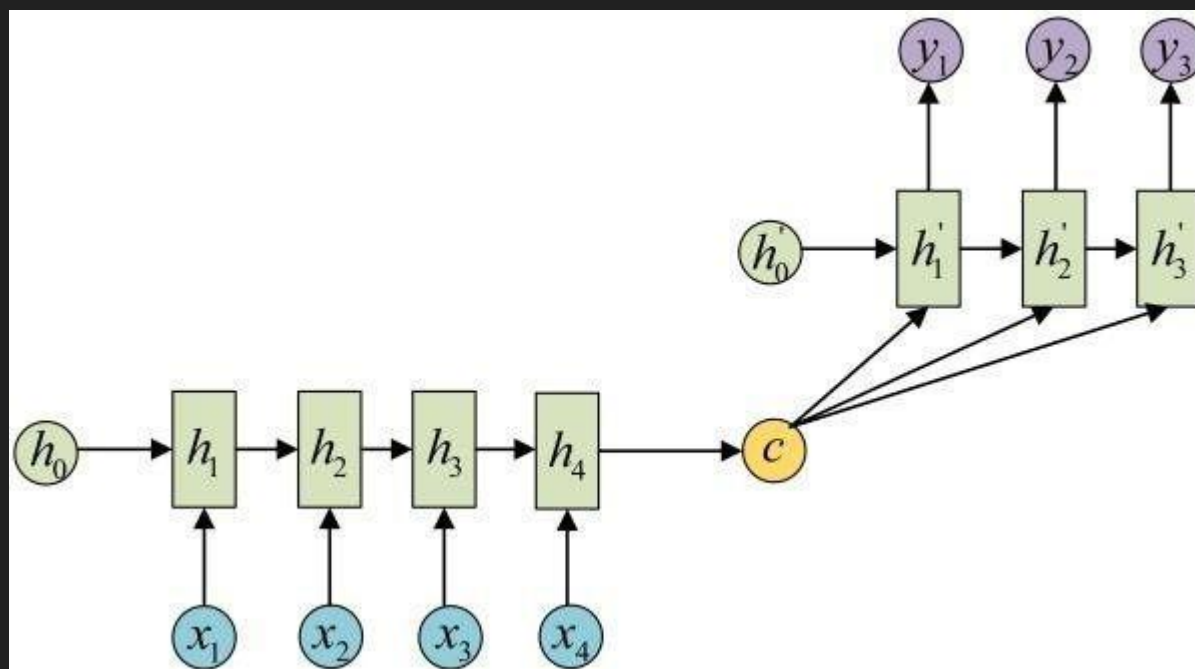
RNN變體 (N vs M)

- 拿到 c 之後，就用另一個 RNN 網路對其進行解碼，這部分 RNN 網路被稱為 Decoder
- 將 c 當做之前的初始狀態 h_0 輸入到 Decoder 中



RNN變體 (N vs M)

- 另一種做法是將 c 當做每一步的輸入

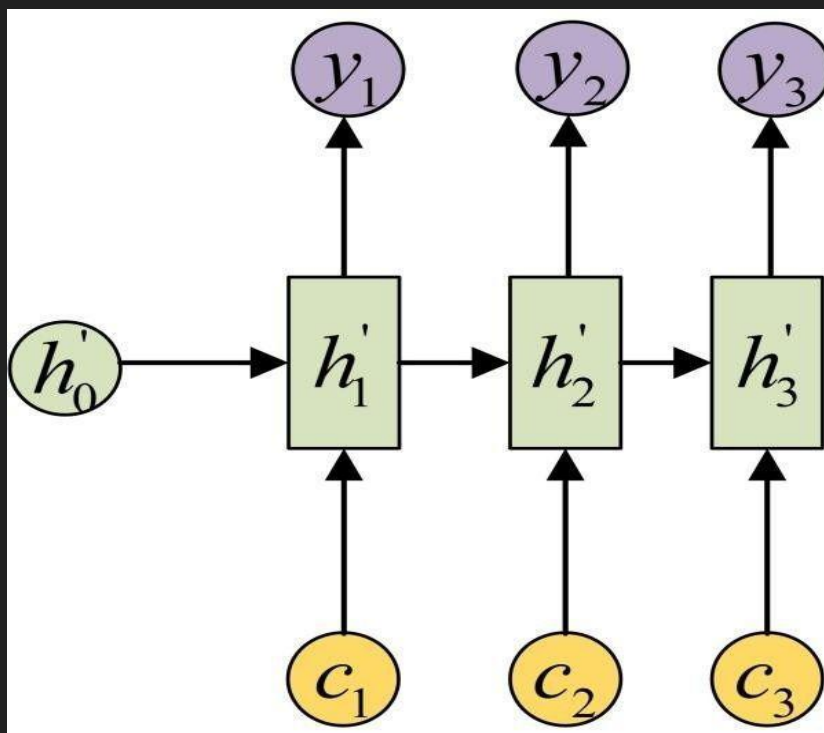


Attention機制

- 在 Encoder-Decoder 結構中，Encoder 把所有的輸入序列都編碼成一個統一的語義特徵 c 再解碼， c 中必須包含原始序列中的所有訊息，它的長度就成了限制模型性能的瓶頸
- 例如機器翻譯問題，當要翻譯的句子較長時，一個 c 可能存不下那麼多訊息，就會造成翻譯精度的下降。

Attention機制

- Attention 機制通過在每個時間輸入不同的 c 來解決這個問題
- 每一個 c 會自動去選取與當前所要輸出的 y 最合適的上下文訊息



Attention機制

- 「我」、「愛」、「台」、「灣」在翻譯成英語時，第一個上下文 c_1 應該和「我」這個字最相關， c_2 應該和「愛」最相關，最後的 c_3 和「台」、「灣」最相關

我		愛		台		灣	
h_1	$*$	a_{11}	$+$	h_2	$*$	a_{12}	$+$
h_3	$*$	a_{13}	$+$	h_4	$*$	a_{14}	$= c_1 \longrightarrow I$
h_1	$*$	a_{21}	$+$	h_2	$*$	a_{22}	$+$
h_3	$*$	a_{23}	$+$	h_4	$*$	a_{24}	$= c_2 \longrightarrow LOVE$
h_1	$*$	a_{31}	$+$	h_2	$*$	a_{32}	$+$
h_3	$*$	a_{33}	$+$	h_4	$*$	a_{34}	$= c_3 \longrightarrow Taiwan$

2. 使用TensorFlowd完成實驗

使用 TensorFlow 完成實驗

- TensorFlow 實現的開源聊天機器人項目 DeepQA
- 從數據集上和一些重要代碼上進行了說明和闡述，最後針對於測試的情況
- <https://github.com/Conchylicultor/DeepQA>

Conchylicultor / DeepQA

Watch 196 Star 2,270 Fork 1,013

Code Issues 82 Pull requests 4 Projects 0 Wiki Insights

My tensorflow implementation of "A neural conversational model", a Deep learning based chatbot

chatbot deep-learning tensorflow seq2seq

182 commits 1 branch 0 releases 12 contributors Apache-2.0

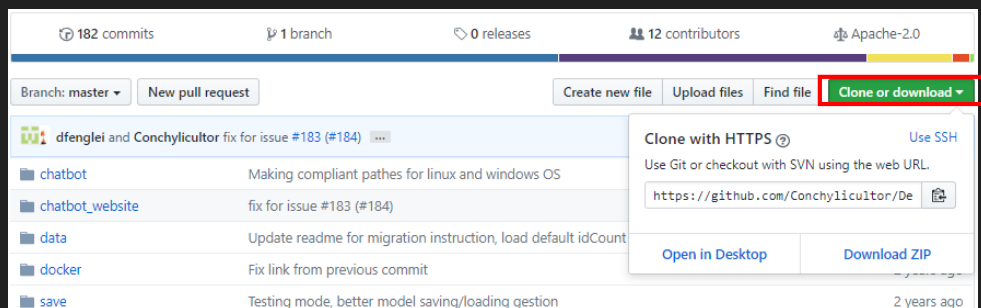
Tree: efcfbf3d4b New pull request Create new file Upload files Find file Clone or download

dfenglei and Conchylicultor fix for issue #183 (#184) Latest commit efcfbf3 on 8 Apr 2018

chatbot	Making compliant paths for linux and windows OS	a year ago
chatbot_website	fix for issue #183 (#184)	9 months ago
data	Update readme for migration instruction, load default idCount for bac...	2 years ago
docker	Fix link from previous commit	2 years ago
save	Testing mode, better model saving/loading gestion	2 years ago

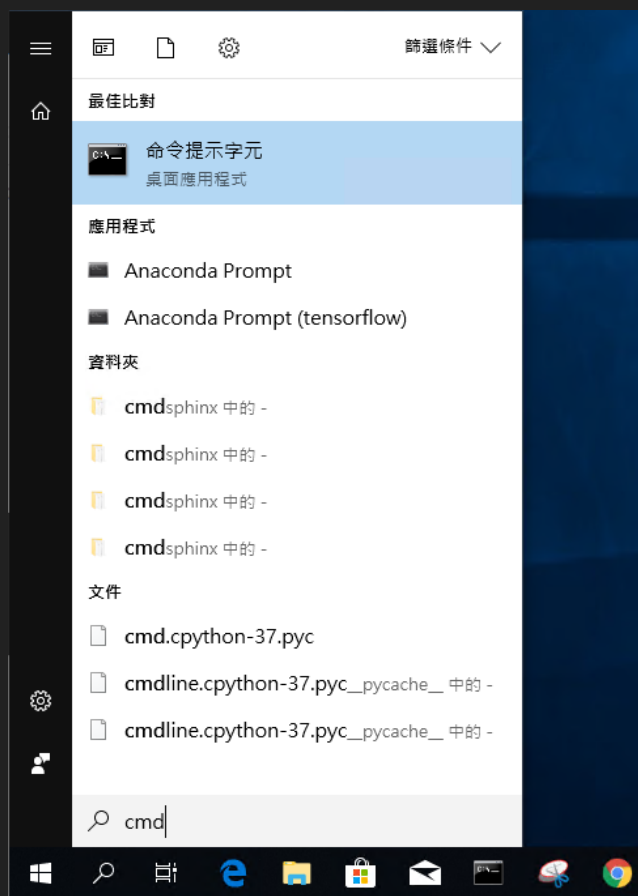
使用 TensorFlow 完成實驗

按下網頁右上角的 Clone or download 並選擇 Download ZIP 並解壓縮至 C:\ 目錄之下



使用 TensorFlow 完成實驗

- 在搜尋輸入 cmd，開啟命令提示字元

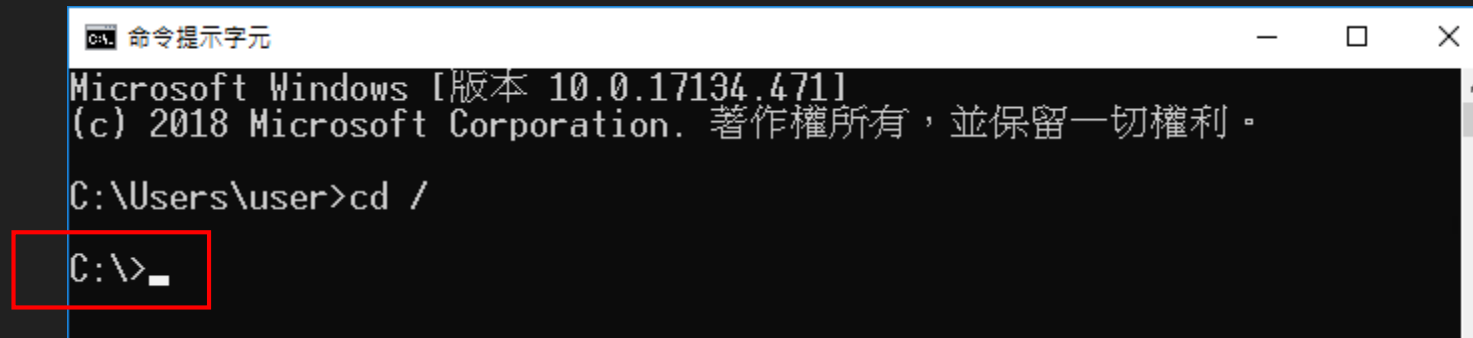


使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

```
> cd /
```

- 工作路徑由 C:\Users\user 變成 C:\



A screenshot of a Windows Command Prompt window. The title bar reads "命令提示字元". The window content shows the following text: "Microsoft Windows [版本 10.0.17134.471] (c) 2018 Microsoft Corporation. 著作權所有，並保留一切權利。 C:\Users\user>cd / C:\>". The final prompt "C:\>" is enclosed in a red rectangular box.

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

> cd DeepQA-master

- 工作路徑由 C:\變成 C:\DeepQA-master




```
命令提示字元
Microsoft Windows [版本 10.0.17134.523]
(c) 2018 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user>cd \
C:\>cd DeepQA-master
C:\DeepQA-master>
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

> **Activate tensorflow** # 之前已經建立該虛擬環境



```
命令提示字元
Microsoft Windows [版本 10.0.17134.523]
(c) 2018 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\user>cd \

C:\>cd DeepQA-master

C:\DeepQA-master>activate tensorflow

(tensorflow) C:\DeepQA-master>
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

> `pip install -r requirements.txt`

```
(tensorflow) C:\DeepQA-master>pip install -r requirements.txt
Collecting tensorflow-gpu>=1.0 (from -r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/ed/1f/b9cf0932b7720be8f31f898926353494ed9585d8534aecbb5fe0f5b52c89/tensorflow_gpu-1.12.0-cp35-cp35m-win_amd64.whl
Requirement already satisfied: numpy in c:\users\user\anaconda3\envs\tensorflow\lib\site-packages (from -r requirements.txt (line 2)) (1.14.3)
Requirement already satisfied: nltk in c:\users\user\anaconda3\envs\tensorflow\lib\site-packages (from -r requirements.txt (line 3)) (3.3)
```



```
Installing collected packages: tensorflow-gpu, tqdm
Successfully installed tensorflow-gpu-1.12.0 tqdm-4.29.1
You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

> `python -m nltk.downloader punkt`

```
(tensorflow) C:\DeepQA-master>python -m nltk.downloader punkt
C:\Users\user\Anaconda3\envs\tensorflow\lib\runpy.py:125: RuntimeWarning: 'nltk.downloader'
' found in sys.modules after import of package 'nltk', but prior to execution of 'nltk.dow
nloader'; this may result in unpredictable behaviour
  warn(RuntimeWarning(msg))
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令進行訓練

> `python main.py --numEpochs=5`

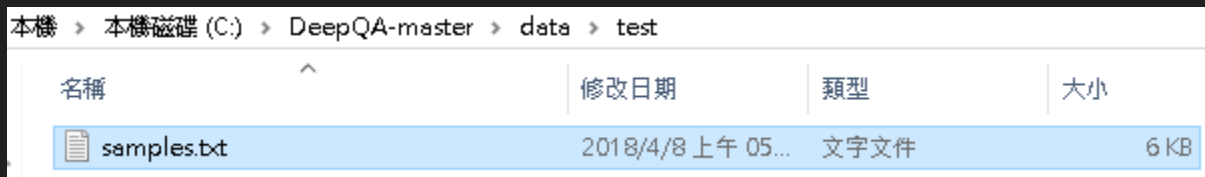
```
(tensorflow) C:\DeepQA-master>python main.py --numEpochs=5
C:\Users\user\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from 'float' to 'np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Welcome to DeepQA v0.1 !
```



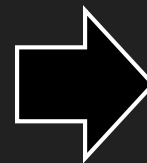
```
██████████| 624/624 [05:09<00:00, 2.15it/s]
Epoch finished in 0:05:09.638149
Checkpoint reached: saving model (don't stop the run)...
Model saved.
The End! Thanks for using this program
(tensorflow) C:\DeepQA-master>
```

使用 TensorFlow 完成實驗

- 查看 C:\DeepQA-master\data\test\samples.txt 的測試問題



本機 > 本機磁碟 (C:) > DeepQA-master > data > test			
名稱	修改日期	類型	大小
samples.txt	2018/4/8 上午 05:00	文字文件	6 KB



```
1 Hi
2 Hi!
3 Are you conscious?
4 How are you?
5 What is your name ?
6 Are you alive ?
7 Luke, I am your father!
8 You shall not pass!
9 I'm going to kill you!
10 Are you ready ?
11 When are you ready ?
12 What color is the sky?
13 How old are you ?
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令進行測試

> `python main.py --test`

```
(tensorflow) C:\DeepQA-master>python main.py --test
C:\Users\user\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from 'float' to 'np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Welcome to DeepQA v0.1 !

TensorFlow detected: v1.12.0
```



```
██████████| 328/328 [00:10<00:00, 30.60it/s]
Prediction finished, 5/328 sentences ignored (too long)
All predictions done
The End! Thanks for using this program
```

使用 TensorFlow 完成實驗

- 查看 C:\DeepQA-master\save\model\model_predictions.txt
- model_predictions.txt 為測試結果

本機 > 本機磁碟 (C:) > DeepQA-master > save > model

名稱	修改日期	類型	大小
checkpoint	2019/1/14 下午 0...	檔案	1 KB
events.out.tfevents.1547464706.DESK...	2019/1/14 下午 0...	DESKTOP-N7T7S...	3,667 KB
model.ckpt	2019/1/14 下午 0...	CKPT 檔案	1 KB
model.ckpt.data-00000-of-00001	2019/1/14 下午 0...	DATA-00000-OF...	261,948 KB
model.ckpt.index	2019/1/14 下午 0...	INDEX 檔案	2 KB
model.ckpt.meta	2019/1/14 下午 0...	META 檔案	3,562 KB
model_predictions.txt	2019/1/14 下午 0...	文字文件	14 KB
params.ini	2019/1/14 下午 0...	組態設定	1 KB



```
Q: Hi
A: I'm not a little.

Q: Hi!
A: I'm not a little.

Q: Are you conscious?
A: I'm not.

Q: How are you?
A: I'm not.

Q: What is your name ?
A: I'm not.

Q: Are you alive ?
A: I'm not a little.
```


使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令進行聊天機器人互動

> `python main.py --test interactive`

```
(tensorflow) C:\DeepQA-master>python main.py --test interactive
C:\Users\user\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from 'float' to 'np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Welcome to DeepQA v0.1 !

TensorFlow detected: v1.12.0
```



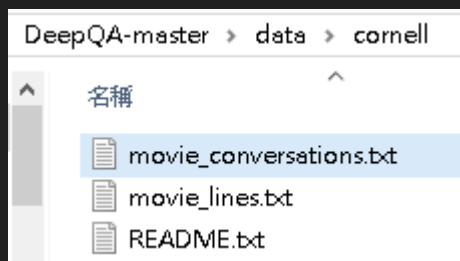
```
Welcome to the interactive mode, here you can ask to Deep Q&A the sentence you want. Don't
have high expectation. Type 'exit' or just press ENTER to quit the program. Have fun.
Q: Hi
A: I'm not a little.
Q:
```

DeepQA

- DeepQA 是 Tensorflow 實現的開源的 seq2seq 模型聊天機器人
- 出自 Google 的一篇關於對話模型的論文 A Neural Conversational Model
- 訓練的語料庫包含電影台詞的對話 Cornell 對話庫、Scotus 對話庫、及 Ubuntu 的對話
- 數據都能在項目的 data 裡找到，目前只能針對某一個對話數據庫進行訓練，還沒有支持混合對話的訓練

Cornell數據集

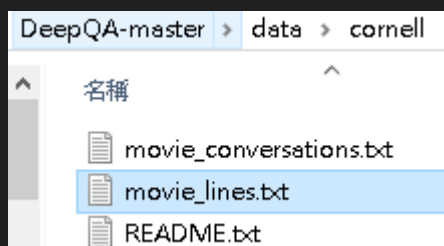
- DeepQA 默認的是 Cornell 對話數據，一共兩個文件：人物對話信息 `movie_conversations.txt` 和具體對話內容 `movie_lines.txt`
- `movie_conversations.txt` 裡每一行的第一個數據代表對話人物 1 的 ID，第二個數據代表對話人物 2 的 ID，第三個數據代表電影 ID，後面是對話的 ID，`+++$$$$` 為分隔符



```
movie_conversations.txt x
1  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L194', 'L195', 'L196', 'L197']
2  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L198', 'L199']
3  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L200', 'L201', 'L202', 'L203']
4  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L204', 'L205', 'L206']
5  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L207', 'L208']
6  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L271', 'L272', 'L273', 'L274', 'L275']
7  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L276', 'L277']
8  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L280', 'L281']
9  u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L363', 'L364']
10 u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L365', 'L366']
11 u0 +++$$$$ u2 +++$$$$ m0 +++$$$$ ['L367', 'L368']
```

Cornell數據集

- movie_lines.txt 每一行的第一個數據代表對話的 ID，第二個數據表示說話的人物 ID，第三個數據電影 ID，第四個是此人物的名字，最後是這句話的具體內容，+++\$\$\$\$ 為分隔符



```
1 L1045 +++$$$$ u0 +++$$$$ m0 +++$$$$ BIANCA +++$$$$ They do not!
2 L1044 +++$$$$ u2 +++$$$$ m0 +++$$$$ CAMERON +++$$$$ They do to!
3 L985 +++$$$$ u0 +++$$$$ m0 +++$$$$ BIANCA +++$$$$ I hope so.
4 L984 +++$$$$ u2 +++$$$$ m0 +++$$$$ CAMERON +++$$$$ She okay?
5 L925 +++$$$$ u0 +++$$$$ m0 +++$$$$ BIANCA +++$$$$ Let's go.
6 L924 +++$$$$ u2 +++$$$$ m0 +++$$$$ CAMERON +++$$$$ Wow
7 L872 +++$$$$ u0 +++$$$$ m0 +++$$$$ BIANCA +++$$$$ Okay -- you're gonna need to learn how to lie.
8 L871 +++$$$$ u2 +++$$$$ m0 +++$$$$ CAMERON +++$$$$ No
```

Model.py 程式碼解析

```
# Creation of the rnn cell
def create_rnn_cell():
    encoDecoCell = tf.contrib.rnn.BasicLSTMCell( # Or GRUCell, LSTMCell(args.hiddenSize)
        self.args.hiddenSize,
    )
    if not self.args.test: # TODO: Should use a placeholder instead
        encoDecoCell = tf.contrib.rnn.DropoutWrapper(
            encoDecoCell,
            input_keep_prob=1.0,
            output_keep_prob=self.args.dropout
        )
    return encoDecoCell
encoDecoCell = tf.contrib.rnn.MultiRNNCell(
    [create_rnn_cell() for _ in range(self.args.numLayers)],
)
```

建立單個LSTM單元

給lstm單元添加dropout

最後用參數
numLayers決定多
少層的RNN

Model.py 程式碼解析

- 定義網絡的輸入值，根據標準的 seq2seq 模型，一共四個：
- Encoder 的輸入：人物 1 說的一句話 A
- Decoder 的輸入：人物 2 回覆的對話 B
- Decoder 的 target 輸入：decoder 的輸入在時序上的結果，比如說完這個詞後的下個詞的結果
- Decoder 的 weight 輸入：實際句子的長度，因為不是所有的句子的長度都一樣，在實際輸入的過程中，各個句子的長度都會被用統一的標示符來填充（padding）至最大長度，weight 用來標記實際詞彙的位置，代表這個位置將會有梯度值回傳

```
with tf.name_scope('placeholder_encoder'):
    self.encoderInputs = [tf.placeholder(tf.int32, [None, ])]
    for _ in range(self.args.maxLengthEnco) # Batch size * sequence length * input dim

with tf.name_scope('placeholder_decoder'):
    self.decoderInputs = [tf.placeholder(tf.int32, [None, ], name='inputs')]
    for _ in range(self.args.maxLengthDeco) # Same sentence length for input and output (Right ?)
    self.decoderTargets = [tf.placeholder(tf.int32, [None, ], name='targets')]
    for _ in range(self.args.maxLengthDeco)
    self.decoderWeights = [tf.placeholder(tf.float32, [None, ], name='weights')]
    for _ in range(self.args.maxLengthDeco)
```

Model.py 程式碼解析

- Tensorflow 把常用的 seq2seq 模型都封裝好了，比如 embedding_rnn_seq2seq，這是 seq2seq 一個最簡單的模型

```
# Define the network
# Here we use an embedding model, it takes integer as input and convert them into word vector for
# better word representation
decoderOutputs, states = tf.contrib.legacy_seq2seq.embedding_rnn_seq2seq(
    self.encoderInputs, # List<[batch=?, inputDim=1]>, list of size args.maxLength
    self.decoderInputs, # For training, we force the correct output (feed_previous=False)
    encoDecoCell,
    self.textData.getVocabularySize(),
    self.textData.getVocabularySize(), # Both encoder and decoder have the same number of class
    embedding_size=self.args.embeddingSize, # Dimension of each word
    output_projection=outputProjection.getWeights() if outputProjection else None,
    feed_previous=bool(self.args.test) # When we test (self.args.test), we use previous output as next input (feed_previous)
```

Model.py 程式碼解析

- RNN 輸出一個句子的過程，其實是對句子裡的每一個詞來做整個詞彙表的 softmax 分類，取機率最大的詞作為當前位置的輸出詞，但是若詞彙表很大，計算量會很大，那麼通常的解決方法是在詞彙表裡做一個下採樣
- 把隱藏層的輸出映射到整個詞彙表，這種映射需要參數 w 和 b

```
def sampledSoftmax(labels, inputs):
    labels = tf.reshape(labels, [-1, 1]) # Add one dimension (nb of true classes, here 1)

    # We need to compute the sampled_softmax_loss using 32bit floats to
    # avoid numerical instabilities.
    localWt = tf.cast(outputProjection.W_t, tf.float32)
    localB = tf.cast(outputProjection.b, tf.float32)
    localInputs = tf.cast(inputs, tf.float32)

    return tf.cast(
        tf.nn.sampled_softmax_loss(
            localWt, # Should have shape [num_classes, dim]
            localB,
            labels,
            localInputs,
            self.args.softmaxSamples, # The number of classes to randomly sample per batch
            self.textData.getVocabularySize(), # The number of classes
            self.dtype)
```


Model.py 程式碼解析

```
# For training only
else:
    # Finally, we define the loss function
    self.lossFct = tf.contrib.legacy_seq2seq.sequence_loss(
        decoderOutputs,
        self.decoderTargets,
        self.decoderWeights,
        self.textData.getVocabularySize(),
        softmax_loss_function= sampledSoftmax
    )
    tf.summary.scalar('loss', self.lossFct) # Keep track of the cost

    # Initialize the optimizer
    opt = tf.train.AdamOptimizer(
        learning_rate=self.args.learningRate,
        beta1=0.9,
        beta2=0.999,
        epsilon=1e-08
    )
    self.optOp = opt.minimize(self.lossFct)
```

對整個詞彙表的做
softmax loss

更新方法採用默認參
數的Adam

-END-