智慧化企業整合

Project 3

英文文法校正

110034545 張芳綺

指導教授：邱銘傳 教授

目錄

一、 摘要

傳統上當英文文法有錯誤時通常需要依靠人工檢查及除錯，本研究使用 Seq2Seq 架構與 LSTM 模型訓練出一個能自動偵錯並校正文法的模型，input 有錯誤文法的英文句子後能夠 output 出修正文法後正確的句子。最後呈現出校正後的結果並與原本錯誤的句子進行比對。

二、 研究目的與動機

1. 目的與動機

在進行英文寫作時常常會犯一些文法上的小錯誤，若無細心檢查或對文法不熟悉的話可能會繼續使用錯誤的語句而不自知，因此我們希望訓練一個能夠自動偵錯並進行文法除錯的模型，輸入錯誤的英文句子後能夠輸出校正過後文法正確的英文句子，以節省人工檢查及除錯的時間。

2. 5W1H

Who	進行英文寫作者或文法校正人員
What	進行英文文法校正除錯
Where	要進行文法除錯的任何地點
When	當文法有錯誤時
Why	自動偵測錯誤並校正，節省人工檢查時間
How	Seq2Seq、LSTM

三、 文獻回顧

1. Seq2Seq(Sequence to Sequence)

Seq2Seq 被提出於 2014 年，最早由兩篇文章獨立地闡述了它的主要思想，分別是 Google Brain 團隊的《Sequence to Sequence Learning with Neural Networks》和 Yoshua Bengio 團隊的《Learning Phrase Representation using RNN Encoder-Decoder for Statistical Machine Translation》。Seq2Seq 解決問題的主要思路是通過深度神

經網絡模型（常用的是 LSTM）將一個作為輸入的序列映射為一個作為輸出的序列，這一過程由編碼（Encoder）輸入與解碼（Decoder）輸出兩個環節組成，前者負責把序列編碼成一個固定長度的向量，這個向量作為輸入傳給後者，輸出可變長度的向量。

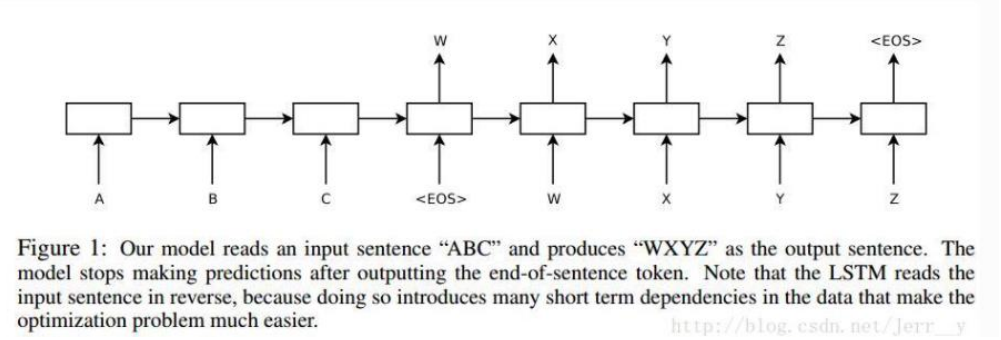
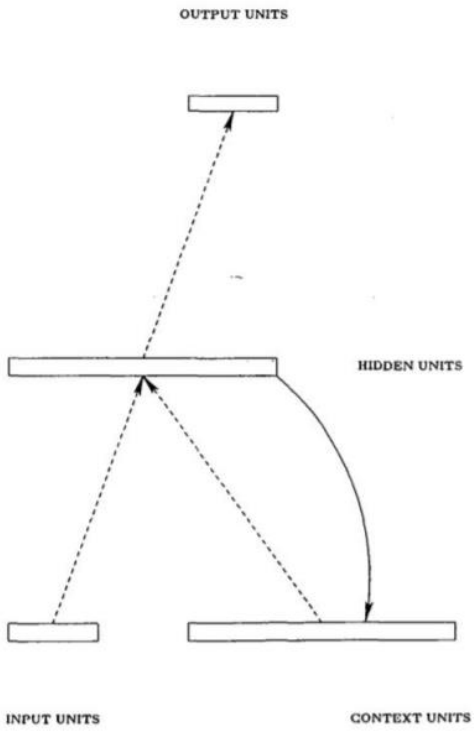


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Seq2Seq 示意圖

2. 遞歸神經網路(Recurrent Neural Network, RNN)

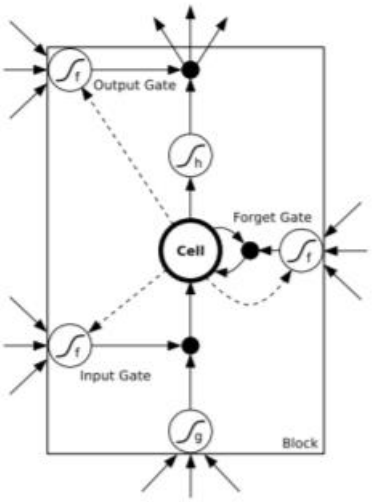
遞歸神經網路不同於一般的神經網路，其能夠處理具有序列關係的數據，Elman 在 1990 年提出了簡易的 RNN 架構，如下圖所示。RNN 常被用於自然語言處理(Natural Language Processing, NLP)，但在 Hochreiter(1991) and Bengio,et al.(1994)這兩篇文獻當中有提到 RNN 模型隨著序列的增長會有無法學習到序列較前的問題。



RNN 基本架構

3. 長短期記憶網路(Long Short-Term Memory, LSTM)

長短期記憶網路是一種特殊的 RNN 模型，其最早由 Hochreiter, Sepp, and Jürgen Schmidhuber 在 1997 年時提出，解決了 RNN 在處理長序列數據時，會有梯度消失或梯度爆炸的問題。其在神經單元中加入了「閥(gate)」的概念，用來控制有多少的數據需要被遺忘或是更新，分別為輸入閥(input gate)、輸出閥(output gate)以及遺忘閥(forget gate)，透過這樣的方式，可以避免掉這樣的問題。



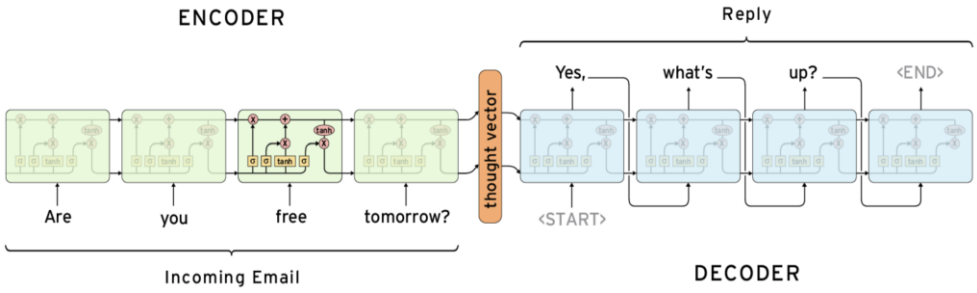
LSTM 基本架構

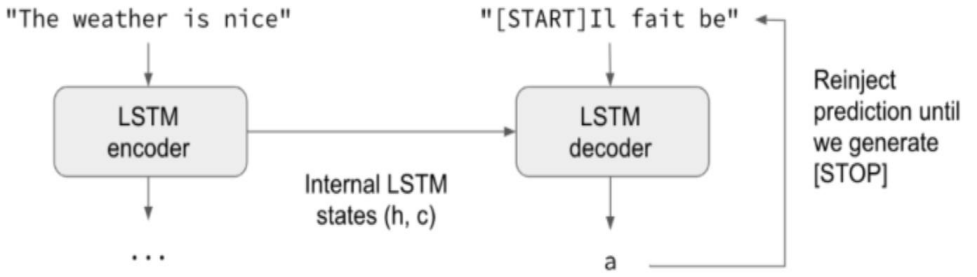
四、 方法

1. Seq2Seq 架構

Seq2seq 模型為 Sequence to Sequence 的縮寫，也被稱作 encoder-decoder framework，encoder 把輸入的文字轉換成機器理解的 context vector，而 decoder 把 context vector 轉換成我們能理解的文字。而訓練好的模型面對意義相近的輸入文字所產生的 context vector 在相同的向量空間上距離會是相近的。在這個架構下，我們可以選擇適合的 encoder 和 decoder model，套用在任何端對端的學習任務（End-to-End Learning）上。選擇合適的 encoder-decoder 可以

讓機器合理地理解人類的輸入資料，從而省去人工特徵選擇和文本資料的意涵難以抽取的麻煩。而 Sequential model 擅長的就是處理具有序列特徵的資料，像是文字或語音。常見的 Sequential model 基本組成為 RNN、LSTM、GRU 等。本研究所使用的架構串連兩個 LSTM 隱藏層，一個隱藏層擔任 encoder 的角色，以 LSTM 處理，但不管輸出，只保留記憶狀態(State)，讓另一個隱藏層使用，另一個隱藏層額外再考慮前文，兩者綜合起來，預測下一個翻譯的字，這個機制即為 decoder。

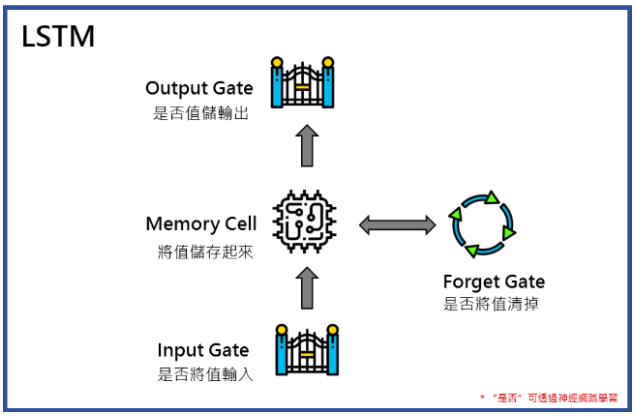




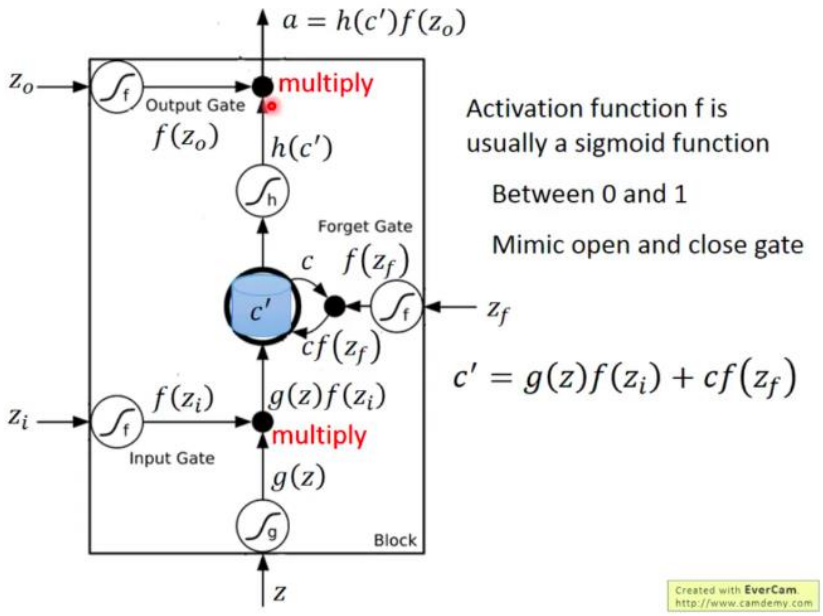
2. LSTM

LSTM(Long short-term memory)，主要改善了以前 RNN 的一些問題(Ex: Memory 的設計問題)，而 LSTM 由四個 unit 組成: Input Gate、Output Gate、Memory Cell 以及 Forget Gate。

- Input Gate: 當資料輸入時，input gate 可以控制是否將這次的值輸入，並運算數值
- Memory Cell: 將運算出的數值記憶起來，以利下個 cell 運用
- Output Gate: 控制是否將這次計算出來的值 output，若無此次輸出則為 0
- Forget Gate: 控制是否將 Memory 清掉(format)



一個 cell 的運作流程：當資料 input 進一個 LSTM cells，數學上可表示為 g(z)，第一個先遇到的就是 input gate，input gate 會使用 f(Zi) (Activation function f)，來表示 input gate 開啟的機率。接下來遇到第二關會是 Memory cell，首先，先紀錄當下 input 值加上前一次 Memory cell 裡的值並乘上 forget gate 的機率，看是否要遺忘前一次紀錄 。最後一關就是 output gate，output gate 會確認是否把值放出，也是以機率的方式來使用。



五、 個案研究

A. 資料集描述

資料集類似文本的形式，兩句為一組意思一樣的句子，前句為錯誤的文法，後句為正確的文法。資料集共有 153182 組句子，其中屬於錯誤

文法的句子中共有 41936 個相異的單字，而屬於正確文法的句子中共有 41713 個相異的單字。

```
In my opinion the TV channels should show the violent movies and TV shows in night , not in the afternoon .     In my
opinion the TV channels should show the violent movies and TV shows at night , not in the afternoon .
There I began to study Family Therapy , discipline that I enjoy too much , because I 've seen I can help people about their
troubles ..     There I began to study Family Therapy , discipline that I enjoy too much , because I 've seen I can help
people with their troubles ..
It seems  me , that in this song there are romantic notes  .    It seems to me , that in this song there are romantic notes
.
So , I want to say that I admire your right disposition and sacrifice in solve the problems related a your phobia , as for
example going out the company . So , I want to say that I admire your right disposition and sacrifice to solve the problems
related to your phobia , as for example going out the company .
I  graduated  Chung university .       I  graduated from Chung university .
I like listening  music and travelling on the weekends .       I like listening to music and travelling on the weekends .
Let me tell you why you should apply  this job .       Let me tell you why you should apply for this job .
In Munchen I went been in Beer Square . In Munchen I went to in Beer Square .
Next I go to the gym and I practice exercises for a little while , because I need to go for the college .       Next I go
to the gym and I practice exercises for a little while , because I need to go to the college .
I am now at Beijing , it 's amazing here ; new modern buildings with a lot of Chinese history . I am now in Beijing , it 's
amazing here ; new modern buildings with a lot of Chinese history .
Here in Brazil it is very common we have problems with family business .       Here in Brazil it is very common to have
problems in family business .
I go for every game and cheer for the home team .       I go to every game and cheer for the home team .
I love painting romantic landscapes , reading books , listening  all types of music and playing video games .   I love
painting romantic landscapes , reading books , listening to all types of music and playing video games .
Shirt  $ 30.00  Shirt for $ 30.00
```

B. 實作流程步驟

1. 資料預處理：將文字轉換成成數字的形式

將文本資料讀取進來後，把錯誤文法的句子(input) 和正確文法的句子(target)分別用 list 存取，接著再把正確文法和錯誤文法句子中出現過的所有單字用 dictionary 分開存取，方便之後將句子中的單字轉換成數字。target 句子部分要加上<Start>、<End>的標籤，我使用 ‘\t’ 、 ‘\n’ 來當作 Start 和 End。

```
with open(data_path, 'r', encoding='utf-8') as f:
    lines = f.read().split('\n')
for line in lines[: min(num_samples, len(lines) - 1)]:
    input_text, target_text = line.split('\t')
    # We use "tab" as the "start sequence" character
    # for the targets, and "\n" as "end sequence" character.
    target_text = '\t' + target_text + '\n'
    input_texts.append(input_text)
    target_texts.append(target_text)
    # for char in str(input_text).split():
    for char in mysplit(input_text):

        if char not in input_characters:
            input_characters.add(char)
    # for char in str(target_text).split():
    for char in mysplit(target_text):

        if char not in target_characters:
            target_characters.add(char)
```

```
input_token_index = dict(
    [(char, i) for i, char in enumerate(input_characters)])
target_token_index = dict(
    [(char, i) for i, char in enumerate(target_characters)])
```

接著用 np.zeros 的方式將每個句子做 padding，保證每個句子長度一樣。

```
encoder_input_data = np.zeros(
    (len(input_texts), max_encoder_seq_length),
    dtype='int32')
decoder_input_data = np.zeros(
    (len(input_texts), max_decoder_seq_length),
    dtype='int32')
decoder_target_data = np.zeros(
    (len(input_texts), max_decoder_seq_length,1),
    dtype='int32')
```

接著將句子根據 dictionary 來轉換成數字的形式，其中 0 代表 <pad> 沒有意義。

```
for i, (input_text, target_text) in enumerate(zip(input_texts, target_texts)):
    input_text=mysplit(input_text)
    target_text=mysplit(target_text)
    for t, char in enumerate(input_text):
        encoder_input_data[i, t] = input_token_index[char]
    for t, char in enumerate(target_text):
        # decoder_target_data is ahead of decoder_input_data by one timestep
        index=target_token_index[char]
        decoder_input_data[i, t] = index
        if t > 0:
            # decoder_target_data will be ahead by one timestep
            # and will not include the start character.
            decoder_target_data[i, t - 1,0] = index
# Define an input sequence and process it.
```

Dirty data 的清理：因為原本的資料集中屬於正確文法的句子裡也有一些錯誤，所以手動進行糾正，把 discuss、explain、mention、describe 後的 about 刪除。

```
def  data_clean(input_data):
    clean=[]
    for  data  in  (input_data):
        clean_data=re.sub(r'discuss  about',r'discuss',data)
        clean_data=re.sub(r'explain  about',r'explain',clean_data)
        clean_data=re.sub(r'mention  about',r'mention',clean_data)
        clean_data=re.sub(r'describe  about',r'describe',clean_data)
        clean.append(clean_data)
    return  clean
```

2. **Word2vector：用 pretrained 好的 embedding layer 轉換成向量**

利用pretrained好的word2vec模型來實作embedding_layer，只取用有出現在文本內的字。接著把剛剛已轉換成數字的句子單字丟入embedding_layer轉換成向量：

```
EMBEDDING_DIM=300
## input
embedding_matrix = np.zeros((num_encoder_tokens, EMBEDDING_DIM))
for word, i in input_token_index.items():#word_index.items():
    if i >= MAX_NB_WORDS:
        continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # input data中的詞不再word2vec裡向量取0，有的話取word2vec的向量
        embedding_matrix[i] = embedding_vector

## target
embedding_matrix1 = np.zeros((num_decoder_tokens, EMBEDDING_DIM))
for word, i in target_token_index.items():#word_index.items():
    if i >= MAX_NB_WORDS:
        continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # target data中的詞不再word2vec裡向量取0，有的話取word2vec的向量
        embedding_matrix1[i] = embedding_vector
# 將訓練好的词向量加載如embedding layer
#  trainable = False，代表词向量不更新
embedding_layer = Embedding(num_encoder_tokens,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix],
                            trainable=False)
embedding_layer1 = Embedding(num_decoder_tokens,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix1],
                            trainable=False)
```

```
encoder_inputs =  Input(shape=(None, ), dtype='int32',)
encoder_embedding = embedding_layer(encoder_inputs)
```

3. **Encode：將 embedded data 丟入 encoder(LSTM1)後再 output 一個 context vector**

用一個 LSTM 將 embedded data 當作 input 丟入 encoder 中 encode 成一個 context vector (encoder states)並傳入 decoder。

```
encoder = (LSTM(latent_dim, return_state=True))
encoder_outputs, state_h, state_c = encoder(encoder_embedding)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

4. **Decode：將 context vector 丟入 decoder(LSTM2) decode 出 target sentence**

將 encoder 產生的 context vector 當作 decoder hidden layer 的 initial state。decoder 部分在訓練時會將上一個 cell decode 時的 hidden layer 傳入當前 cell 並搭配 target input 來 decode 出字。在 predict 時則是會將上一個 cell 的 output 當作這次的 target input 來產生 output。

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs =  Input(shape=(None, ), dtype='int32',)
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_embedding = embedding_layer1(decoder_inputs)
decoder_lstm = (LSTM(latent_dim, return_sequences=True, return_state=True))
decoder_outputs, _, _ = decoder_lstm(decoder_embedding,
                                     initial_state=encoder_states)
decoder_dense = (Dense(num_decoder_tokens, activation='softmax'))

# decoder_dense = TimeDistributed(Dense(num_words, activation='softmax'))
decoder_outputs = decoder_dense(decoder_outputs)
```

C. 模型架構

此模型共有兩層 Input layer，兩層 Embedding layer，兩層 LSTM，一層 Dense

Encoder：input_1、embedding_1、lstm_1

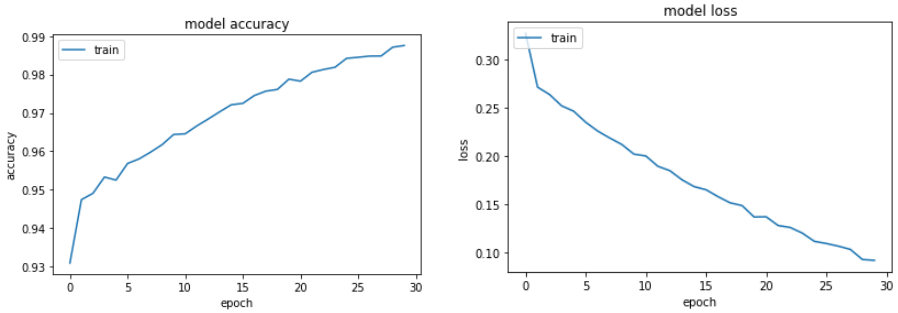
Decoder：input_2、embedding_2、lstm_2、dense_1

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, None)]	0	[]
input_2 (InputLayer)	[(None, None)]	0	[]
embedding_1 (Embedding)	(None, None, 300)	12580800	['input_1[0][0]']
embedding_2 (Embedding)	(None, None, 300)	12513900	['input_2[0][0]']
lstm_1 (LSTM)	[(None, 300), (None, 300), (None, 300)]	721200	['embedding_1[0][0]']
lstm_2 (LSTM)	[(None, None, 300), (None, 300), (None, 300)]	721200	['embedding_2[0][0]', 'lstm_1[0][1]', 'lstm_1[0][2]']
dense_1 (Dense)	(None, None, 41713)	12555613	['lstm_2[0][0]']

Total params: 39,092,713
Trainable params: 13,998,013
Non-trainable params: 25,094,700

D. 模型訓練成果



模型在短句上的表現較佳，大部分都能準確的校正文法上的錯誤，但偶爾會改變原本的語意：

```
Input sentence: I really like  wear jeans and t - shirts .
Decoded sentence:  I really like to wear jeans and T - shirts .

Input sentence: Many people like  play golf .
Decoded sentence:  Many people like to play volleyball .

Input sentence: I 'm working on my office .
Decoded sentence:  I ' m working in my office .

Input sentence: Also you should n't yell in the street .
Decoded sentence:  Also you should n ' t yell at the street .

Input sentence: I also like  play a guitar .
Decoded sentence:  I also like to play a guitar .

Input sentence: Let me tell you why you should apply  this job .
Decoded sentence:  Let me tell you why you should apply for this job .

Input sentence: I am sending out invitations on 30 of my friends .
Decoded sentence:  I am sending out invitations to 30 of my friends .

Input sentence: Welcome  my home .
Decoded sentence:  Welcome to my home .

Input sentence: Sometimes I talk  phone and send emails .
Decoded sentence:  Sometimes I talk on the phone and send emails .

Input sentence: I am  good shape .
Decoded sentence:  I am in good shape .

Input sentence: I worked with him  5 years  in Metis company .
Decoded sentence:  I worked with him for 5 years at a company .

Input sentence: It will be in Rio de Janeiro and I will arrive there  about 6 pm .
Decoded sentence:  It will be on July , , I will go there and and I will go there .
```

模型在長句上的表現較差，可能完全改變語意且文法仍是錯誤的：

```
Input sentence: I think our future is positive and colourful ; every day we have new chances  working on the quality of our thoughts and words .
Decoded sentence:  I think the future is very good and and have a better work in the market and the quality of the people will be more careful with you .

Input sentence: Before the age of laptops we used note books to take notes of everything , but now we can use the laptops also on the meetings .
Decoded sentence:  Before the online catalog they have to spend time on the internet , I will send it to the people to the people in the world and s more .

Input sentence: In my opinion the TV channels should show the violent movies and TV shows in night , not in the afternoon .
Decoded sentence:  In my opinion the TV channels should show the violence and sex and The movie is to go to the movies .
```

六、 結論與改善方向

這次我共跑了 30 個 epoch，花了快兩個禮拜，因為時間較來不及，如果可以跑 50 個 epoch，訓練的結果應該會更好。另外，從成果可以看出目前的模型在短句的表現成果較佳，長句的處理沒辦法準確的更改錯誤，因

為只用 LSTM 的 encoder 和 decoder 在處理長句時沒辦法很好的學到正確的句子，整個長句 encode 成一個 context vector 會損失很多 input 的資訊，可以改善的方法是加入 attention 機制，讓每個文字都產生一個 context vector，而不是將全部的 input 都壓縮成一個 context vector，這樣子 decoder 在 decode 句子時可以利用不同的 context vector 更好的 decode 出 target word，這樣可以幫助模型專注在應該專注的地方。另外，可以考慮把 LSTM 換成 Transformer 來提升效果, 或是參考 google 的 bert 模型架構。

參考資料

https://gau820827.medium.com/%E6%95%99%E9%9B%BB%E8%85%A6%E5%AF%AB%E4%BD%9C-ai%E7%90%83%E8%A9%95-seq2seq%E6%A8%A1%E5%9E%8B%E6%87%89%E7%94%A8%E7%AD%86%E8%A8%98-pytorch-python3-31e853573dd0

https://ithelp.ithome.com.tw/articles/10194403?fbclid=IwAR1PegT2qp7KBMaIusY6YANhEsmbmHFS9PFGl0CBkHI7k38-JRBbKnAPOFc

https://ithelp.ithome.com.tw/articles/10223055

https://dataxujing.github.io/seq2seqlearn/chapter1/