深度學習TensorFlow實務

循環神經網路

Lab4

-TA-
李偉弘
廖宜健
林佑昌
蔡明諺
彭冠偉

1. 循環神經網路介紹

循環神經網路

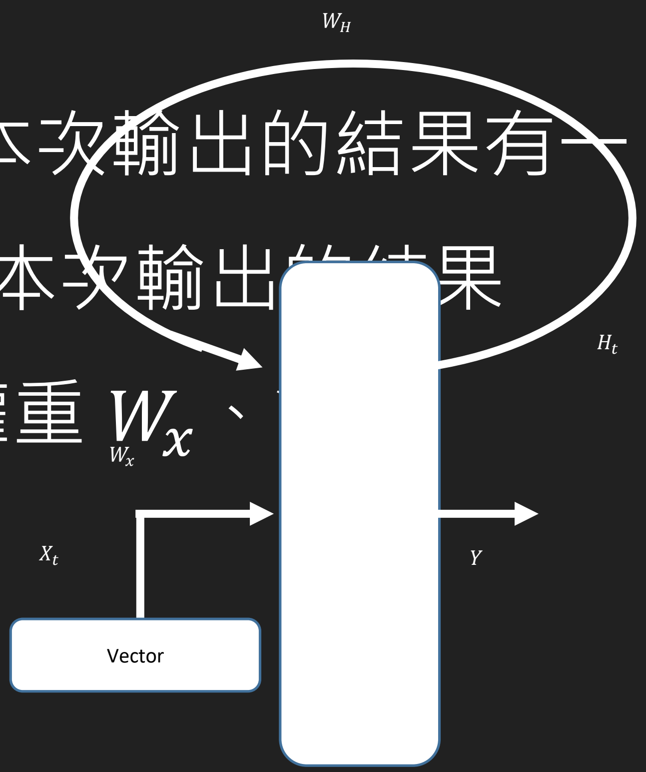
- 全名: Recurrent Neural Networks, RNN
- 與前饋神經網路、卷機神經網路最大的不同為 - 記憶暫存功能
- 在自然語音處理 (natural language processing, NLP) 應用廣泛

隱馬可夫模型

- 馬可夫鏈的核心是:
  - 給定當前知識、資訊的情況下，觀察對象過去的歷史狀態對於將來的預測來說是無關的
  - 一個系統變化時，它下一個狀態 (第n+1狀態) 如何的機率只需觀察和統計當前狀態 (第n個狀態) 即可以正確得出
  - 隱馬可夫鏈是一個雙重的隨機過程，不僅狀態轉移之間是個隨機事件，狀態和輸出之間也是一個隨機過程

RNN結構

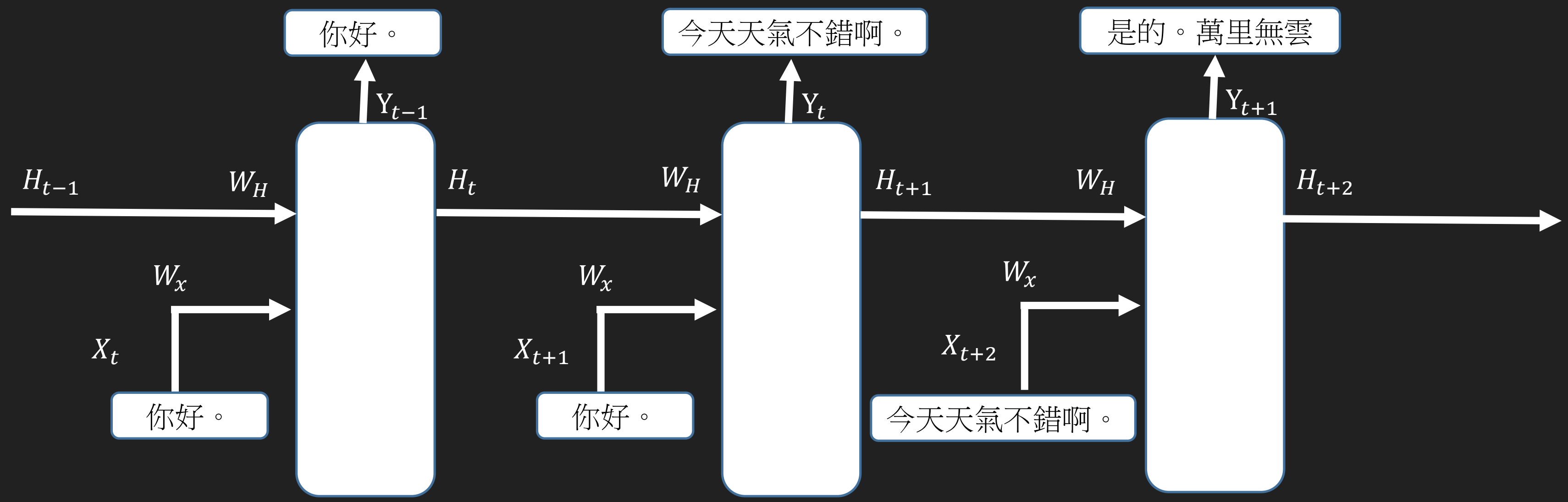
- 兩個特定係數 W_x、W_H，W_x 與 X_t 向量做乘積，作為輸入
- 向量Y由前一次輸出的 H_{t-1} (暫存起來)和 W_H 相乘產生的向量和 W_x 與 X_t 相乘產生的向量做 Softmax 得到
- 前一次輸入的向量 X_{T-1} 所產生的結果對於本次輸出的結果有一定的影響，甚至 X_{T-2}、X_{T-3} 都有可能影響本次輸出的結果
- 中間具體量化的邏輯關係需透過訓練得到權重 W_x、



RNN訓練過程

- W_x、W_H矩陣被初始化後，在 Y 側必有輸出，也就會有殘差產生 E_i
- 放入第一句和第二句後產生 E_1，加上第二句和第三句後產生 E_2，加到倒數第二句和最後一句後產生 E_{n-1}，因此可簡寫成

$$LOSS = \sum_{i=1}^{n-1} E_i$$



RNN艱難的誤差傳遞

- W_X、W_H 是我們最終要學習的內容
  - $LOSS = \alpha E_X + \beta E_H$
- E_X、E_H 表示由 W_X、W_H 引起的誤差，α、β 表示由樣本產生的係數
  - $\mathrm{H_T} = W_H f(H_{t-1}) + W_X X_t$
  - $Y_T = SOFTMAX(f(\mathrm{H_T}))$
- 如果只有 X_t、Y_t，那殘差就成了
  - $\mathrm{H}_1^o = W_H f(\,) + W_X X_1$
  - $E_1 = \frac{1}{2}(SOFTMAX\big(f(\mathrm{H}_1^o)\big) - Y_1)^2 \Rightarrow E_1 = \frac{1}{2}(W_S\big(f(\mathrm{H}_1^o)\big) - Y_1)^2$
- 其中 W_S 是指 Softmax 中的 W_S 矩陣

RNN艱難的誤差傳遞

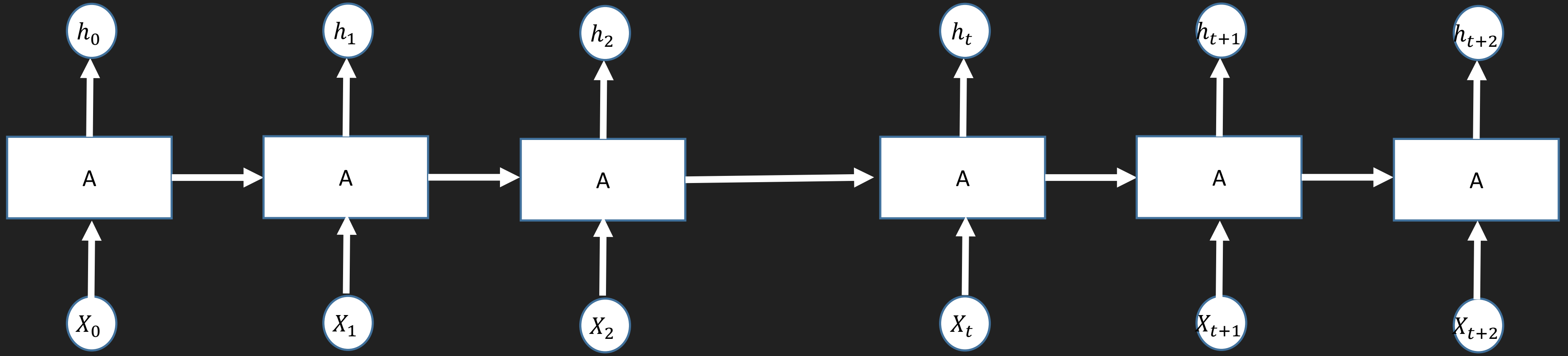
- 根據 E_1 這個殘差分別來求出 W_{X}、W_H 的偏導數來得到梯度大小
  - W_{X} 的偏導數：$\frac{\partial \mathrm{E}_1}{\partial \mathrm{W}_\mathrm{X}} \Rightarrow \frac{\partial W_s f(\mathrm{H}_1^o)}{\partial \mathrm{W}_\mathrm{X}} \Rightarrow W_s \frac{\partial f(\mathrm{H}_1^o)}{\partial \mathrm{W}_\mathrm{X}} \Rightarrow W_s \frac{\partial f(\mathrm{H}_1^o)}{\partial (\mathrm{H}_1^o)} \frac{\partial (\mathrm{W}_\mathrm{H} f(\,) + \mathrm{W}_X X_1)}{\partial \mathrm{W}_\mathrm{X}} \Rightarrow W_s X_1 \frac{\partial f(\mathrm{H}_1^o)}{\partial (\mathrm{H}_1^o)}$
- 當僅有 1 個樣本對輸入，殘差在 W_X 的斜率僅僅和 X_1 向量有關，若有 2 個樣本對，就與 X_1、X_2 有關
  - W_H 的偏導數：$\frac{\partial \mathrm{E}_1}{\partial \mathrm{W}_\mathrm{H}} \Rightarrow \frac{\partial W_s f(\mathrm{H}_1^o)}{\partial \mathrm{W}_\mathrm{H}} \Rightarrow W_s \frac{\partial f(\mathrm{H}_1^o)}{\partial \mathrm{W}_\mathrm{H}} \Rightarrow W_s \frac{\partial f(\mathrm{H}_1^o)}{\partial (\mathrm{H}_1^o)} \frac{\partial (\mathrm{W}_\mathrm{H} f(\,) + \mathrm{W}_X X_1)}{\partial \mathrm{W}_\mathrm{H}} \Rightarrow W_s \frac{\partial f(\mathrm{H}_1^o)}{\partial (\mathrm{H}_1^o)} \frac{\partial \mathrm{W}_\mathrm{H} f(\,)}{\partial \mathrm{W}_\mathrm{H}}$
- 最後一個$\frac{\partial \mathrm{W}_\mathrm{H} f(\,)}{\partial \mathrm{W}_\mathrm{H}}$簡化後成了 $f(\,)$，第一次代入一對輸入和輸出值時，這部分值為初始化給的 H_0 值，寫成 $f(\,)$ 問題不大

RNN艱難的誤差傳遞

- 如果是第 3 對輸入的值 $\frac{\partial \mathrm{W_H} f(\mathrm{H_2^0})}{\partial \mathrm{W_H}}$，$\mathrm{H_2} = W_H f(H_1) + W_X X_1$
- 這仍是一個 W_H 函數，要繼續求導求出 $\frac{\partial \mathrm{H_2}}{\partial \mathrm{W_H}}$
- 如果有 1000 對，就需要求這一系列的導數並連乘起來
  - $\frac{\partial \mathrm{H_{1000}}}{\partial \mathrm{W_H}} \frac{\partial \mathrm{H_{999}}}{\partial \mathrm{W_H}} \frac{\partial \mathrm{H_{998}}}{\partial \mathrm{W_H}} \cdots\cdots \frac{\partial \mathrm{H_2}}{\partial \mathrm{W_H}} \frac{\partial \mathrm{H_1}}{\partial \mathrm{W_H}}$
- 將會加大運算的時間複雜度，引發梯度消失、梯度爆炸
- 後人對 RNN 改造發展 LSTM 演算法代替遞歸累積時間演算法

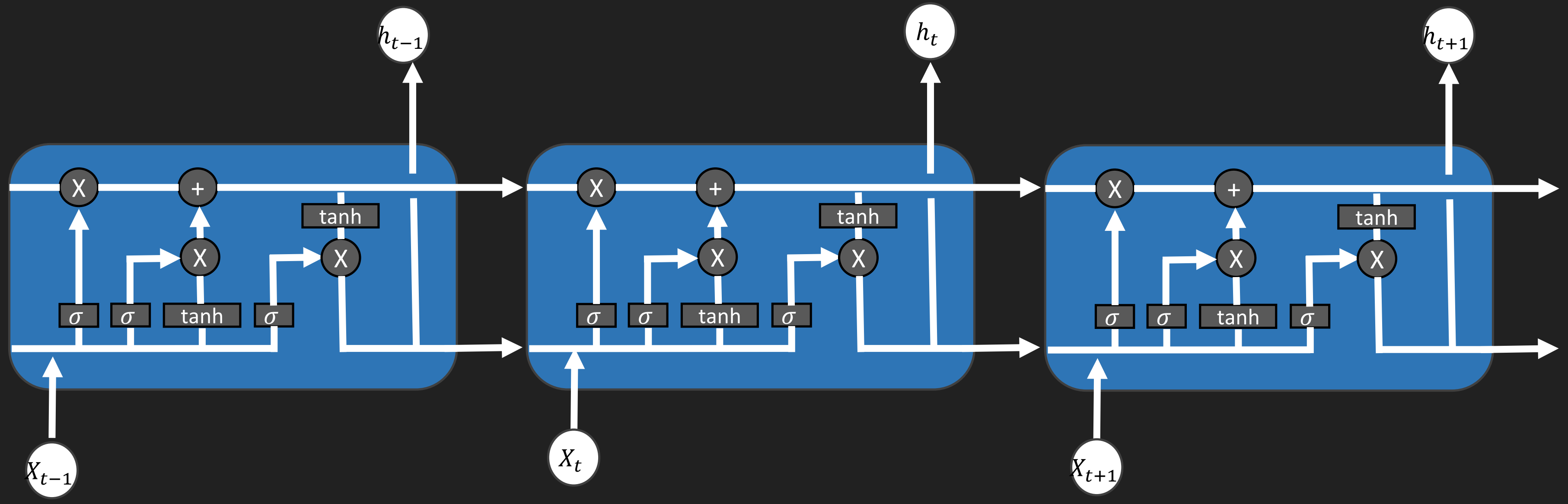
LSTM演算法

- 全名: long short-term memory，簡稱為：LSTM
- 規避了RNN的梯度爆炸和梯度消失問題，學習速度更快
- 多了遺忘閘 (forget gate) 機制
- 對於一個輸入序列 X_i，某一個 X 值會影響一個在時間上或者空間上較遠的 H_j 輸出



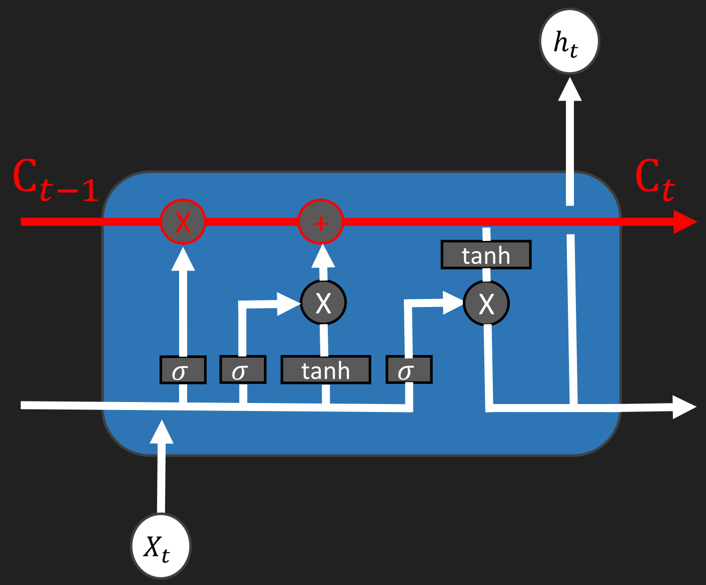
LSTM演算法

- LSTM 輸入層為 X_t，輸出層為 h_t，中間部分為一個個的 LSTM 單元
- LSTM 單元一個一個首尾相接，前一層輸出會作為後一層輸入



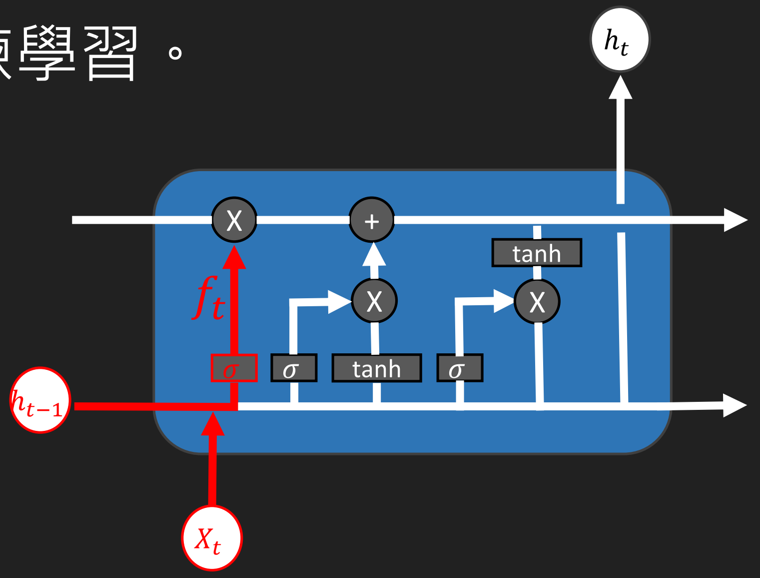
LSTM演算法

- 首先從左到右進行一個向量傳輸，左側的 C_{t-1} 進入單元後，先被一個乘法器乘以一個係數後，再線性疊加一個數值從右側輸出



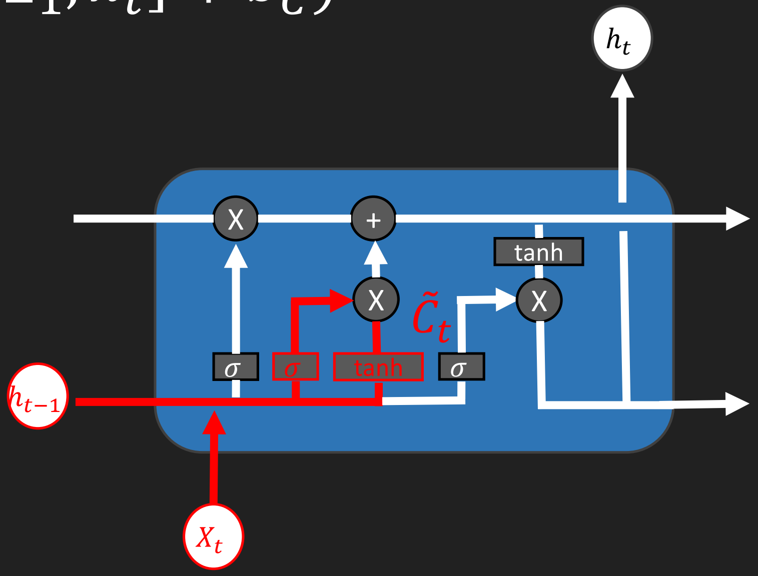
LSTM演算法

- 左側 h_{t-1} 和下方輸入 x_t 經過連接操作，接著透過一個線性單元$\sigma(sigmoid)$函數之後產生一個 0 ~ 1 之間的數字作為係數輸出
- $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- 這就是一個遺忘閘，所謂遺忘就是指相乘的過程，如果 $Sigmoid$ 函數輸出是1，那就是完全記住；如果是 0，那就是完全忘記，中間的 W_f、b_f 作為待定係數需進行訓練學習。



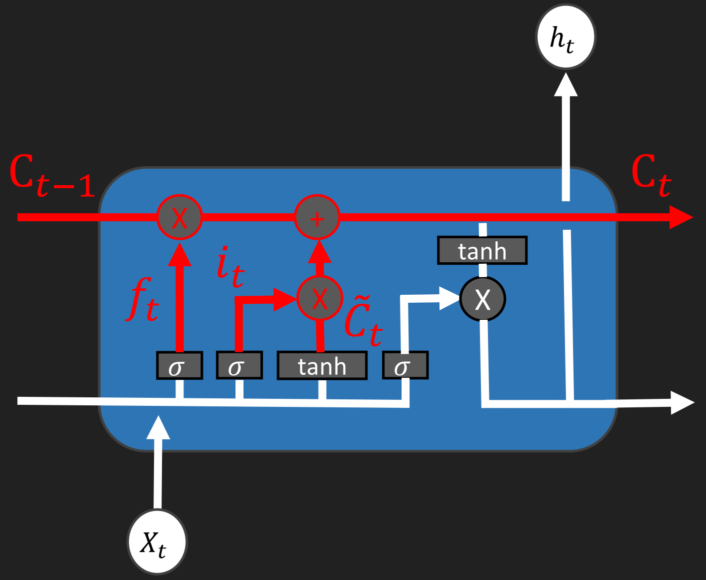
LSTM演算法

- 這裡有兩個神經網路層一個是 σ，表達式為：
  - $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- 一個是 tanh，表達式為：
  - $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$



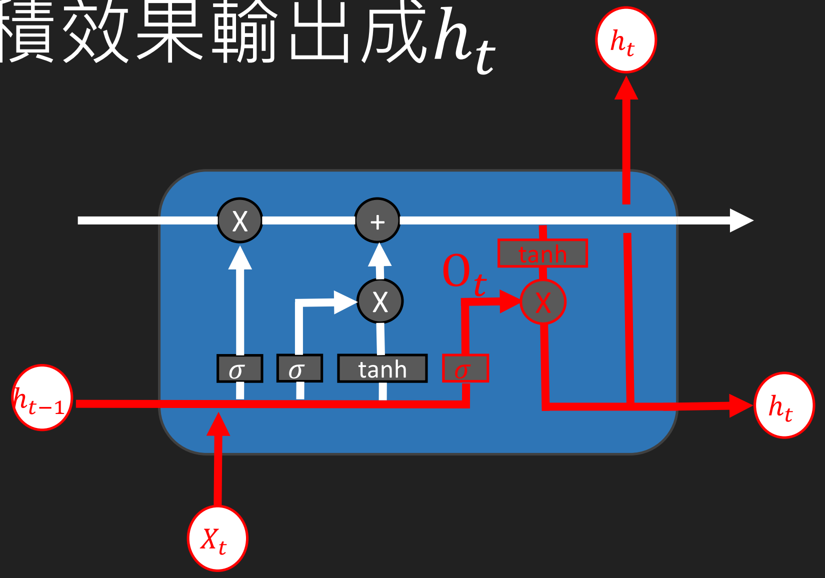
LSTM演算法

- 前一次傳遞過來的 C_{t-1} 向量會和 $\tilde{C}_t$ 線性疊加
  - $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- 那輸出的 C_t 究竟有多少採納本次輸入的訊息？有多少採納上一次遺留下來的訊息呢？



LSTM演算法

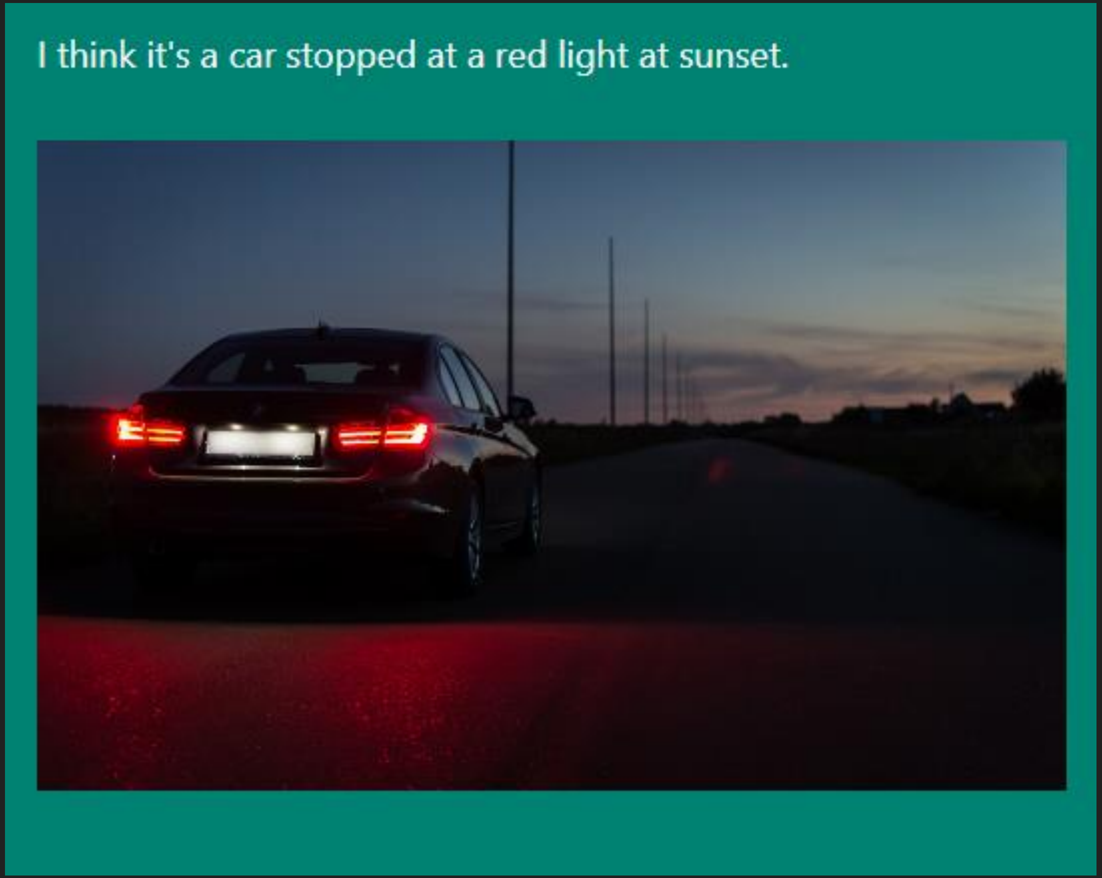
- 最後輸出有兩個h_t，一個輸出到同層下一個單元，一個輸出到下一層單元上
  - $O_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_0)$
  - $h_t = O_t * tanh(C_t)$
- 輸出的C_t向量又經過一個O_t遺忘閘的乘積效果輸出成h_t



2. 應用情境

微軟的識圖機器人CaptionBot

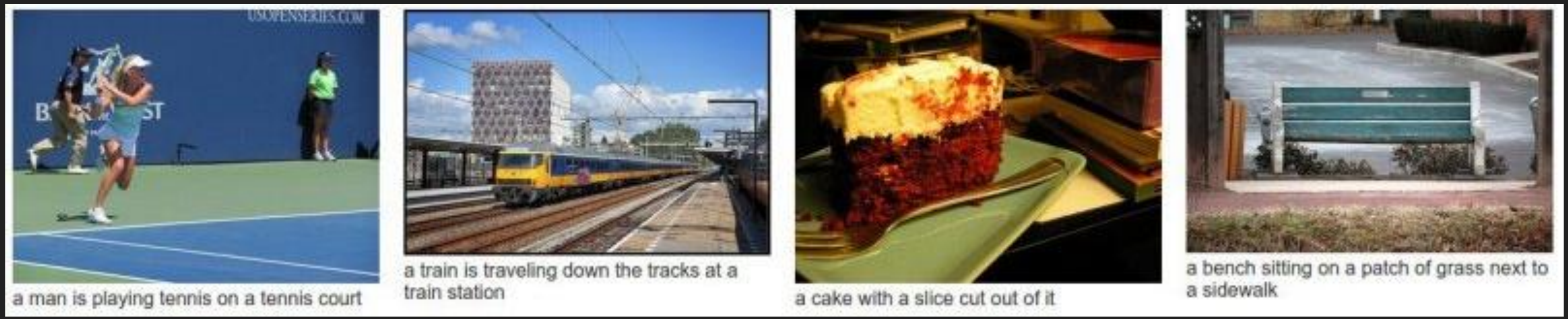
- 是一種單一向量輸入，多向量輸出的情境(描述一張圖上的訊息)



https://www.captionbot.ai/

圖像字幕模型開源碼

- Github 上的開源專案 CNN + RNN 模型
- CNN 用來提取特徵，RNN 用特徵的向量和描述向量來訓練
- 這種模型能夠標示下面這些圖片中有什麼物體(人物)，以及他們的狀態或者動作



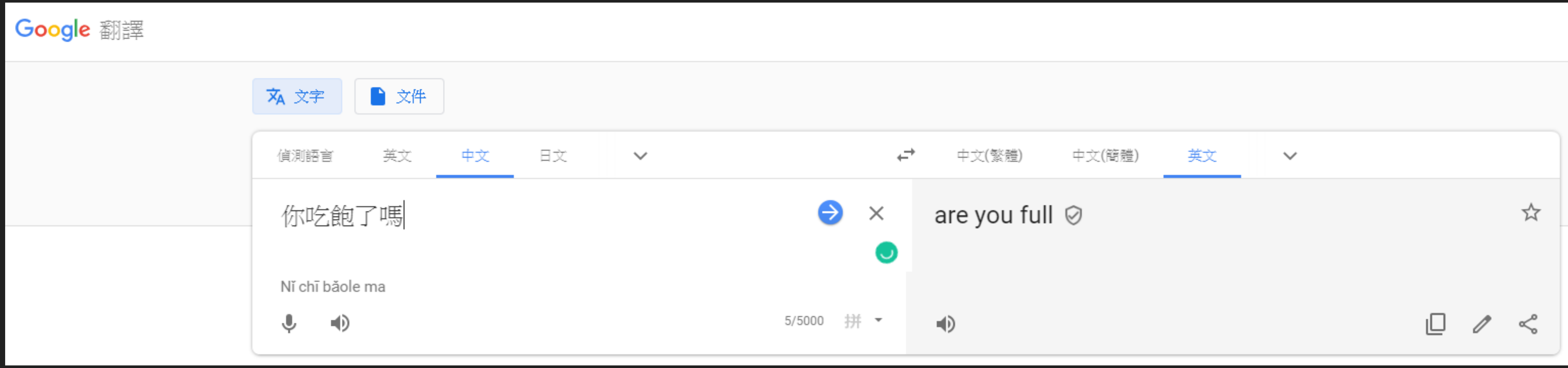
https://github.com/karpathy/neuraltalk2

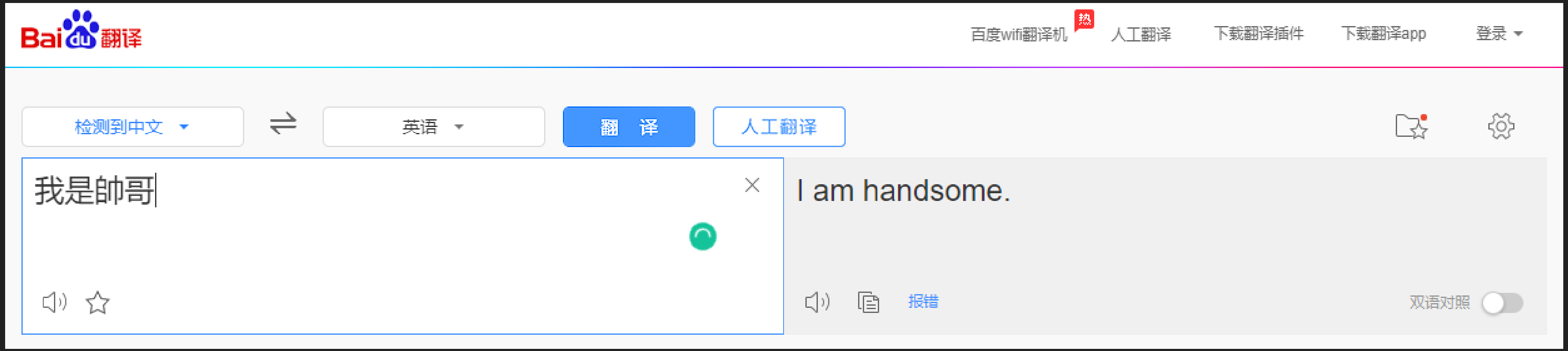
識別影片主題分類

- 專案 C3D: Generic Features for Video Analysis 可以實作主題的識別
- 目前多對撥放的體育競技內容進行分類識別

自動翻譯、聊天機器人

- 在客服、問訊系統等情境下應用，減少人工投入
- Google 翻譯、百度翻譯





描述影片段訊息開源碼

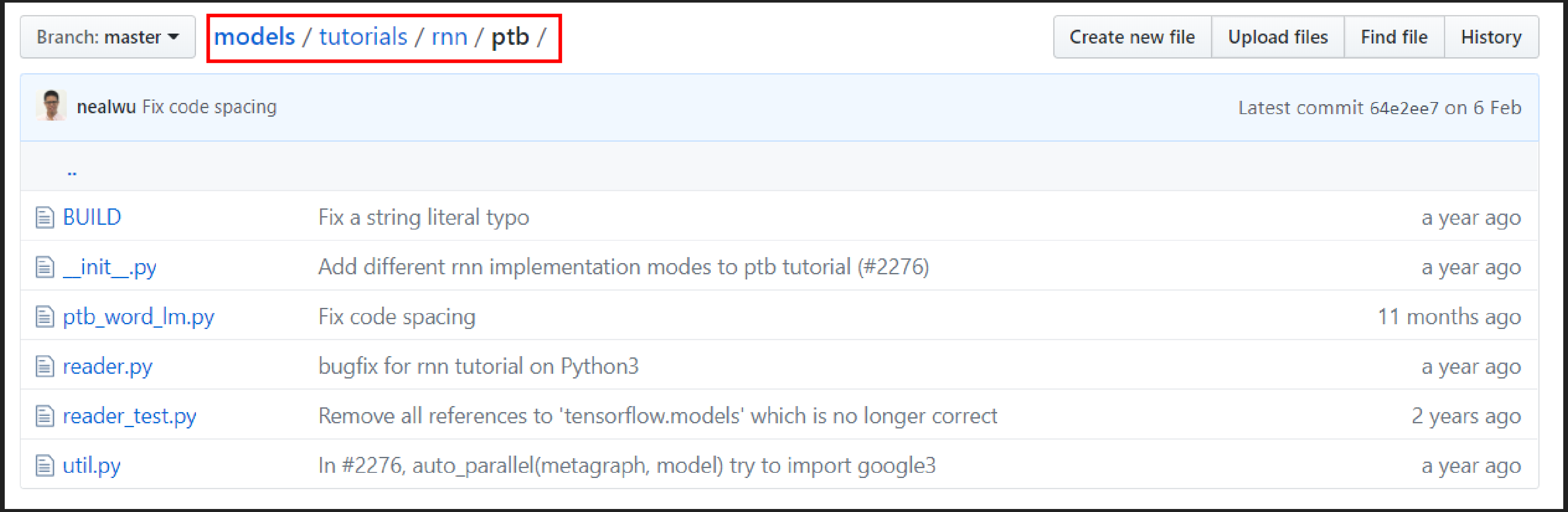
- Github 上的開源專案 RNN 模型
- Demo 影片中，網路能正確識別途中有一群人站在綠色的草坪

https://github.com/samim23/NeuralTalkAnimator

2. 使用TensorFlowd完成實驗

使用 TensorFlow 完成實驗

- 採用循環神經網路 (Recurrent Neural Networks) 完成自動文字生成的小工程
- 使用 TensorFlow 官方提供最為經典的 RNN 入門案例
- https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb



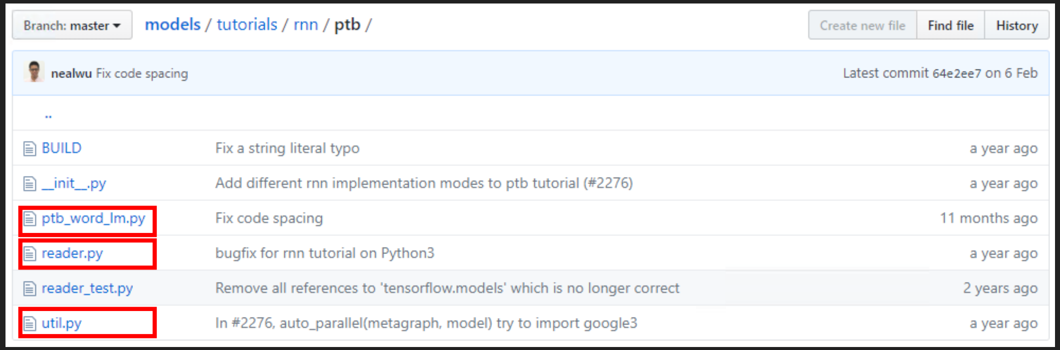
Branch: master ▾ | models / tutorials / rnn / ptb / | Create new file | Upload files | Find file | History

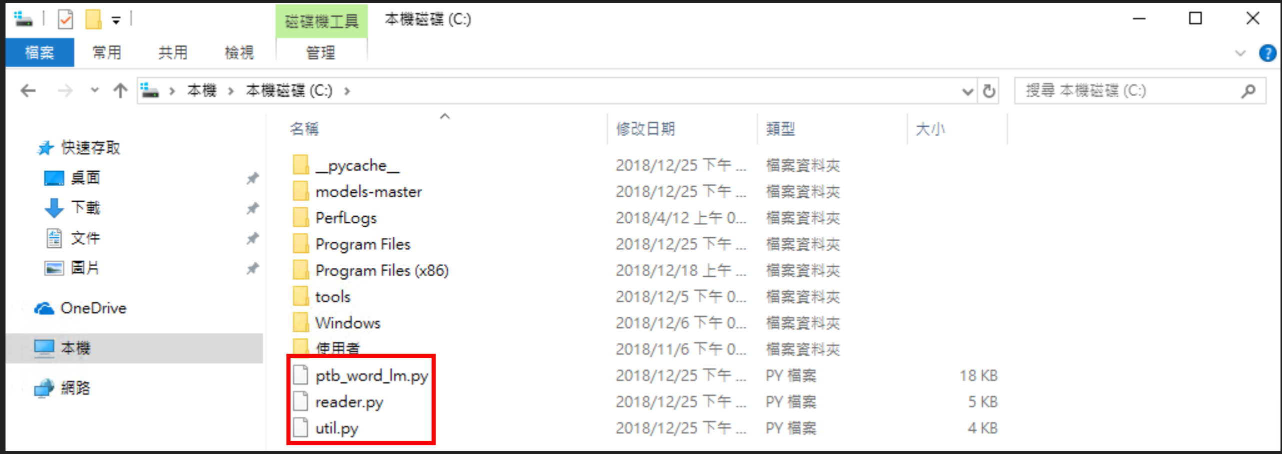
nealwu Fix code spacing — Latest commit 64e2ee7 on 6 Feb

..		
BUILD	Fix a string literal typo	a year ago
__init__.py	Add different rnn implementation modes to ptb tutorial (#2276)	a year ago
ptb_word_lm.py	Fix code spacing	11 months ago
reader.py	bugfix for rnn tutorial on Python3	a year ago
reader_test.py	Remove all references to 'tensorflow.models' which is no longer correct	2 years ago
util.py	In #2276, auto_parallel(metagraph, model) try to import google3	a year ago

使用 TensorFlow 完成實驗

- 下載檔案 ptb_word_lm.py、 reader.py、util.py至 C:\ 目錄之下



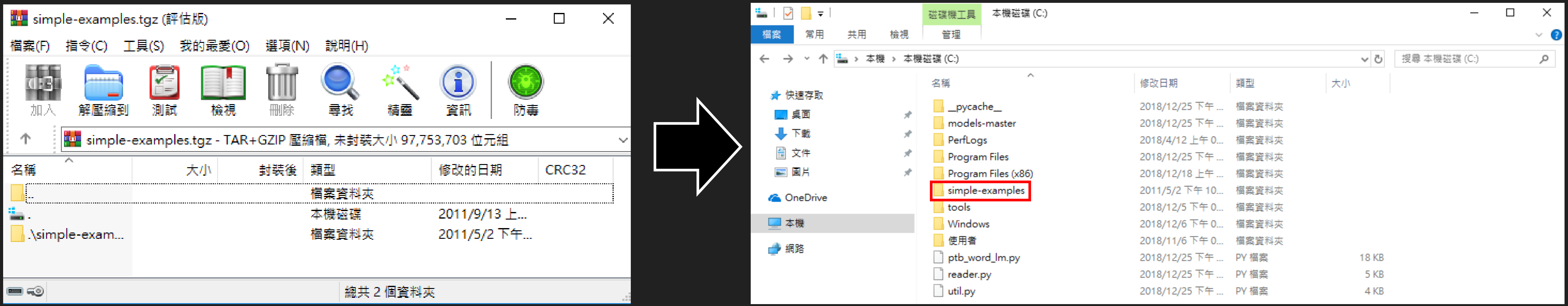


使用 TensorFlow 完成實驗

- 下載 simple-examples
  - http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz

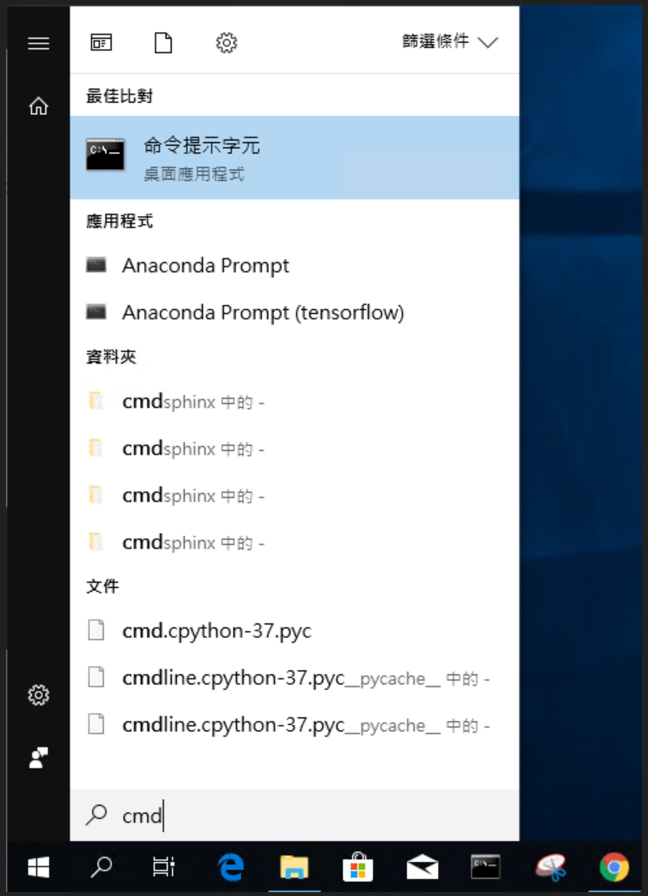
使用 TensorFlow 完成實驗

- 將 simple-examples.tgz 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:\

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
```

使用 TensorFlow 完成實驗

- 在 cmd 視窗，輸入以下指令

```
> python ptb_word_lm.py --data_path=simple-examples/data/
```

命令提示字元 - python ptb_word_lm.py --data_path=simple-examples/data/

```
(tensorflow) c:\>python ptb_word_lm.py --data_path=simple-examples/data/
2018-12-25 22:47:44.138531: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that th
is TensorFlow binary was not compiled to use: AVX
2018-12-25 22:47:44.565338: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1405] Found device 0 with properties:
name: GeForce GTX 1050 major: 6 minor: 1 memoryClockRate(GHz): 1.455
pciBusID: 0000:01:00.0
totalMemory: 2.00GiB freeMemory: 1.61GiB
2018-12-25 22:47:44.575268: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding visible gpu devices: 0
2018-12-25 22:47:45.072588: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device interconnect StreamExecutor
with strength 1 edge matrix:
2018-12-25 22:47:45.080068: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971]      0
2018-12-25 22:47:45.085682: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0:   N
2018-12-25 22:47:45.089311: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created TensorFlow device (/device
:GPU:0 with 1356 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050, pci bus id: 0000:01:00.0, compute capab
ility: 6.1)
WARNING:tensorflow:From ptb_word_lm.py:504: Supervisor.__init__ (from tensorflow.python.training.supervisor) is depreca
ted and will be removed in a future version.
```

使用 TensorFlow 完成實驗

- 出現這個畫面~成功
- Epoch = 13
- 困惑度 (Perplexity)

命令提示字元

```
0.903 perplexity: 41.805 speed: 15583 wps
Epoch: 11 Train Perplexity: 40.901
Epoch: 11 Valid Perplexity: 120.685
Epoch: 12 Learning rate: 0.004
0.004 perplexity: 59.853 speed: 15186 wps
0.104 perplexity: 45.295 speed: 15606 wps
0.204 perplexity: 49.544 speed: 15589 wps
0.304 perplexity: 47.502 speed: 15608 wps
0.404 perplexity: 46.690 speed: 15671 wps
0.504 perplexity: 45.908 speed: 15650 wps
0.604 perplexity: 44.384 speed: 15582 wps
0.703 perplexity: 43.727 speed: 15603 wps
0.803 perplexity: 42.981 speed: 15660 wps
0.903 perplexity: 41.523 speed: 15695 wps
Epoch: 12 Train Perplexity: 40.625
Epoch: 12 Valid Perplexity: 120.372
Epoch: 13 Learning rate: 0.002
0.004 perplexity: 59.583 speed: 15785 wps
0.104 perplexity: 45.086 speed: 15938 wps
0.204 perplexity: 49.330 speed: 15926 wps
0.304 perplexity: 47.309 speed: 15943 wps
0.404 perplexity: 46.509 speed: 15833 wps
0.504 perplexity: 45.736 speed: 15883 wps
0.604 perplexity: 44.221 speed: 15748 wps
0.703 perplexity: 43.566 speed: 15725 wps
0.803 perplexity: 42.824 speed: 15746 wps
0.903 perplexity: 41.371 speed: 15755 wps
Epoch: 13 Train Perplexity: 40.476
Epoch: 13 Valid Perplexity: 120.182
Test Perplexity: 113.936

(tensorflow) c:\>_
```

Ptb_world_lm.py 程式碼解析

```python
def _get_lstm_cell(self, config, is_training):
  if config.rnn_mode == BASIC:
    return tf.contrib.rnn.BasicLSTMCell(
        config.hidden_size, forget_bias=0.0, state_is_tuple=True,
        reuse=not is_training)
  if config.rnn_mode == BLOCK:
    return tf.contrib.rnn.LSTMBlockCell(
        config.hidden_size, forget_bias=0.0)
  raise ValueError("rnn_mode %s not supported" % config.rnn_mode)

def _build_rnn_graph_lstm(self, inputs, config, is_training):
  """Build the inference graph using canonical LSTM cells."""
  # Slightly better results can be obtained with forget gate biases
  # initialized to 1 but the hyperparameters of the model would need to be
  # different than reported in the paper.
  def make_cell():
    cell = self._get_lstm_cell(config, is_training)
    if is_training and config.keep_prob < 1:
      cell = tf.contrib.rnn.DropoutWrapper(
          cell, output_keep_prob=config.keep_prob)
    return cell

  cell = tf.contrib.rnn.MultiRNNCell(
      [make_cell() for _ in range(config.num_layers)], state_is_tuple=True)

  self._initial_state = cell.zero_state(config.batch_size, data_type())
  state = self._initial_state
  # Simplified version of tf.nn.static_rnn().
  # This builds an unrolled LSTM for tutorial purposes only.
  # In general, use tf.nn.static_rnn() or tf.nn.static_state_saving_rnn().
```

建立LSTM單元

給lstm單元添加dropout

建立多層lstm

Ptb_world_lm.py 程式碼解析

```python
with tf.device("/cpu:0"):
  embedding = tf.get_variable(
      "embedding", [vocab_size, size], dtype=data_type())
  inputs = tf.nn.embedding_lookup(embedding, input_.input_data)

if is_training and config.keep_prob < 1:
  inputs = tf.nn.dropout(inputs, config.keep_prob)

output, state = self._build_rnn_graph(inputs, config, is_training)

softmax_w = tf.get_variable(
    "softmax_w", [size, vocab_size], dtype=data_type())
softmax_b = tf.get_variable("softmax_b", [vocab_size], dtype=data_type())
logits = tf.nn.xw_plus_b(output, softmax_w, softmax_b)
 # Reshape logits to be a 3-D tensor for sequence loss
logits = tf.reshape(logits, [self.batch_size, self.num_steps, vocab_size])

# Use the contrib sequence loss and average over the batches
loss = tf.contrib.seq2seq.sequence_loss(
    logits,
    input_.targets,
    tf.ones([self.batch_size, self.num_steps], dtype=data_type()),
    average_across_timesteps=False,
    average_across_batch=True)

# Update the cost
self._cost = tf.reduce_sum(loss)
self._final_state = state
```

每個單詞使用一個唯一向量表示，word_embedding

給 輸 出 層 添 加dropout

過一層全連接

運算loss

Ptb_world_lm.py 程式碼解析

```
outputs = []
with tf.variable_scope("RNN"):
  for time_step in range(self.num_steps):
    if time_step > 0: tf.get_variable_scope().reuse_variables()
    (cell_output, state) = cell(inputs[:, time_step, :], state)
    outputs.append(cell_output)
output = tf.reshape(tf.concat(outputs, 1), [-1, config.hidden_size])
```

開始訓練循環

```
self._lr = tf.Variable(0.0, trainable=False)
tvars = tf.trainable_variables()
grads, _ = tf.clip_by_global_norm(tf.gradients(self._cost, tvars),
                                  config.max_grad_norm)
optimizer = tf.train.GradientDescentOptimizer(self._lr)
self._train_op = optimizer.apply_gradients(
    zip(grads, tvars),
    global_step=tf.train.get_or_create_global_step())

self._new_lr = tf.placeholder(
    tf.float32, shape=[], name="new_learning_rate")
self._lr_update = tf.assign(self._lr, self._new_lr)
```

使用梯度下降最佳化演算法運算梯度，更新降低learning rate，擷取一下梯度值

Ptb_world_lm.py 程式碼解析

```python
sv = tf.train.Supervisor(logdir=FLAGS.save_path)
config_proto = tf.ConfigProto(allow_soft_placement=soft_placement)
with sv.managed_session(config=config_proto) as session:
  for i in range(config.max_max_epoch):
    lr_decay = config.lr_decay ** max(i + 1 - config.max_epoch, 0.0)
    m.assign_lr(session, config.learning_rate * lr_decay)

    print("Epoch: %d Learning rate: %.3f" % (i + 1, session.run(m.lr)))
    train_perplexity = run_epoch(session, m, eval_op=m.train_op,
                                 verbose=True)
    print("Epoch: %d Train Perplexity: %.3f" % (i + 1, train_perplexity))
    valid_perplexity = run_epoch(session, mvalid)
    print("Epoch: %d Valid Perplexity: %.3f" % (i + 1, valid_perplexity))

  test_perplexity = run_epoch(session, mtest)
  print("Test Perplexity: %.3f" % test_perplexity)

  if FLAGS.save_path:
    print("Saving model to %s." % FLAGS.save_path)
    sv.saver.save(session, FLAGS.save_path, global_step=sv.global_step)
```

降低更新learning rate

分別使用訓練資料
和驗證資料執行一
個epoch

使用測試資料直接一個epoch

儲存模型

Ptb_world_lm.py 程式碼解析

```
def main(_):
  if not FLAGS.data_path:
    raise ValueError("Must set --data_path to PTB data directory")
  gpus = [
      x.name for x in device_lib.list_local_devices() if x.device_type == "GPU"
  ]
  if FLAGS.num_gpus > len(gpus):
    raise ValueError(
        "Your machine has only %d gpus "
        "which is less than the requested --num_gpus=%d."
        % (len(gpus), FLAGS.num_gpus))

  raw_data = reader.ptb_raw_data(FLAGS.data_path)
  train_data, valid_data, test_data, _ = raw_data

  config = get_config()
  eval_config = get_config()
  eval_config.batch_size = 1
  eval_config.num_steps = 1
```

使用TensorFlow後首先調用main函數

準備訓練、驗證和測試資料集

獲取訓練參數和驗證相關參數

Ptb_world_lm.py 程式碼解析

```python
class SmallConfig(object):
  """Small config."""
  init_scale = 0.1
  learning_rate = 1.0
  max_grad_norm = 5
  num_layers = 2
  num_steps = 20
  hidden_size = 200
  max_epoch = 4
  max_max_epoch = 13
  keep_prob = 1.0
  lr_decay = 0.5
  batch_size = 20
  vocab_size = 10000
  rnn_mode = BLOCK
```

```python
class MediumConfig(object):
  """Medium config."""
  init_scale = 0.05
  learning_rate = 1.0
  max_grad_norm = 5
  num_layers = 2
  num_steps = 35
  hidden_size = 650
  max_epoch = 6
  max_max_epoch = 39
  keep_prob = 0.5
  lr_decay = 0.8
  batch_size = 20
  vocab_size = 10000
  rnn_mode = BLOCK
```

```python
class LargeConfig(object):
  """Large config."""
  init_scale = 0.04
  learning_rate = 1.0
  max_grad_norm = 10
  num_layers = 2
  num_steps = 35
  hidden_size = 1500
  max_epoch = 14
  max_max_epoch = 55
  keep_prob = 0.35
  lr_decay = 1 / 1.15
  batch_size = 20
  vocab_size = 10000
  rnn_mode = BLOCK
```

```python
class TestConfig(object):
  """Tiny config, for testing."""
  init_scale = 0.1
  learning_rate = 1.0
  max_grad_norm = 1
  num_layers = 1
  num_steps = 2
  hidden_size = 2
  max_epoch = 1
  max_max_epoch = 1
  keep_prob = 1.0
  lr_decay = 0.5
  batch_size = 20
  vocab_size = 10000
  rnn_mode = BLOCK
```

四組不同的訓練參數設定

reader.py 程式碼解析

```python
def _read_words(filename):
    with tf.gfile.GFile(filename, "r") as f:
        if Py3:
            return f.read().replace("\n", "<eos>").split()
        else:
            return f.read().decode("utf-8").replace("\n", "<eos>").split()
```

讀檔案裡的資料，把 “\n” 換行符換成” <eos>” ，返回所有出現的單詞列表

```python
def _build_vocab(filename):
    data = _read_words(filename)
    counter = collections.Counter(data)
    count_pairs = sorted(counter.items(), key=lambda x: (-x[1], x[0]))
    words, _ = list(zip(*count_pairs))
    word_to_id = dict(zip(words, range(len(words))))
    return word_to_id
```

把每個單詞對應一個唯一id

```python
def _file_to_word_ids(filename, word_to_id):
    data = _read_words(filename)
    return [word_to_id[word] for word in data if word in word_to_id]
```

把檔案裡的單詞變成對應的id

```python
def ptb_raw_data(data_path=None):
    train_path = os.path.join(data_path, "ptb.train.txt")
    valid_path = os.path.join(data_path, "ptb.valid.txt")
    test_path = os.path.join(data_path, "ptb.test.txt")

    word_to_id = _build_vocab(train_path)
    train_data = _file_to_word_ids(train_path, word_to_id)
    valid_data = _file_to_word_ids(valid_path, word_to_id)
    test_data = _file_to_word_ids(test_path, word_to_id)
    vocabulary = len(word_to_id)
    return train_data, valid_data, test_data, vocabulary
```

返回訓練、驗證、測試資料集(單詞變成對應id之後的結果)和檔案裡出現的詞彙總數

reader.py 程式碼解析

```python
def ptb_producer(raw_data, batch_size, num_steps, name=None):
  with tf.name_scope(name, "PTBProducer", [raw_data, batch_size, num_steps]):
    raw_data = tf.convert_to_tensor(raw_data, name="raw_data", dtype=tf.int32)

    data_len = tf.size(raw_data)
    batch_len = data_len // batch_size
    data = tf.reshape(raw_data[0 : batch_size * batch_len],
                      [batch_size, batch_len])

    epoch_size = (batch_len - 1) // num_steps
    assertion = tf.assert_positive(
        epoch_size,
        message="epoch_size == 0, decrease batch_size or num_steps")
    with tf.control_dependencies([assertion]):
      epoch_size = tf.identity(epoch_size, name="epoch_size")

    i = tf.train.range_input_producer(epoch_size, shuffle=False).dequeue()
    x = tf.strided_slice(data, [0, i * num_steps],
                         [batch_size, (i + 1) * num_steps])
    x.set_shape([batch_size, num_steps])
    y = tf.strided_slice(data, [0, i * num_steps + 1],
                         [batch_size, (i + 1) * num_steps + 1])
    y.set_shape([batch_size, num_steps])
    return x, y
```

把資料和其對應的標籤
分為若干個batch返回

-END-