MLDS HW1–Language Model Report
TensorJoe

r05922063 陳啓中 r05921032 陳昱維 r05943093 蔣君涵 d04921018 艾弗里

Environment 環境設置

OS: Ubuntu 16.06

CPU: Intel Core i7-3770

3.40GHz, 8 cores, 2 threads each.

GPU: Nvidia TITAN X

12GB on board frame buffer

Python Library:

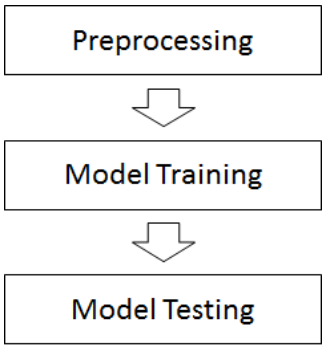
Tensorflow 1.0

Numpy 1.12.1

FastText

Model Description 模型描述

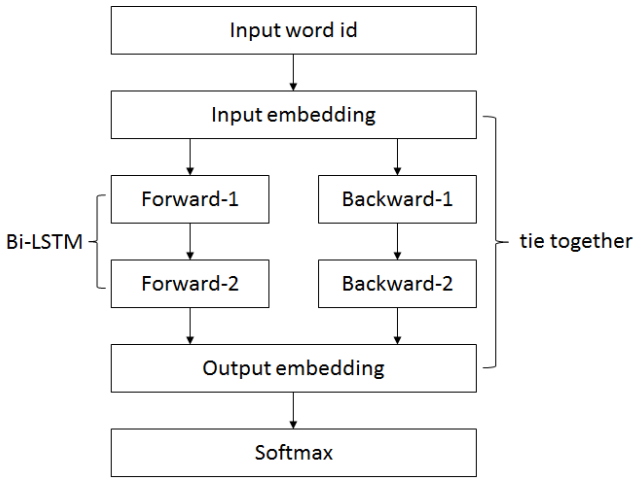
我們架構主要分為三個部分，架構內容以及流程如下圖所示：



在 Preprocess 資料預先處理的部分主要是用人類對於英文已知的知識處理，利用 Word Embedding 的方式，將字詞轉換為向量形式，以方便之後的模型處理運算，在這個專題中我們使用的是 Facebook 公司所提供的 FastText 函式庫的資訊來轉換我們訓練資料中的字詞。

有了數值會的資料後以下是模型訓練的部分架構圖參照下頁的架構：

模型上我們使用 Tensorflow 內部的 LSTM 作為基礎修改，搭出兩層雙向的 RNN，分別順著句子順序及倒序讀取句子中的文字，我們正反方向分別使用 200 及 200 個神經元作為傳遞，最後再通過 Softmax 輸出結果



Performance Improvement 模型調整

在模型的調整上我們使用以下三個手法來增進效能模型訓練效能：

1. Negative Sampling

   在模型訓練上，由於模型訓練的資料量過於龐大，我們用 Negative Sampling 來減少字詞向量之間相似度 (cross-entropy) 的計算量，相較於比較所有可能的字詞選項，我們隨機挑選 4000 個 Negative Sample 來配合正確答案訓練，這能有效減少模型運算時總體時間。我們實驗後發現使用 0.0001 能夠達到足夠的計算速度優化並且相較而言保持了模型判斷的精確程度。

   選擇一個字詞作為 Negative Sample 的機率與該詞出現的頻率有關係，以下為字詞被選為 Negative Sample 的機率

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_j)^{3/4}\right)}$$

2. Subsampling

   在英文句法使用上，經常出現例如“the”, “a”, “or”這種類型的字詞，雖然出現頻率高，然而對於判斷句子或段落意思的幫助卻不大，這類型的字容易導致 LSTM 在學習的時候，會讓那些字被選到的機率比較高，所以我們用 subsampling 來平衡出現頻率高的字被選到的機率，增加真正有代表意義的字詞被選中的機率。

   為了有效去除高機率出現相對兒言較無意義的字詞，我們使用以下機率公式作為判斷是否保留一個字詞，或者是為無意義字詞去除的機率。$z(w_i)$為該字詞在整體資料庫中所在比例。

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1\right) \cdot \frac{0.001}{z(w_i)}$$

3. Optimizer

   我們用 Tensorflow 裡面的 Adam Optimizer 來做優化。利用平均移動量的參數來有效減少演算法的 step size，就可以不需要一直調整模型學習的狀況，並讓訓練過程更穩定。

Experience Setting and Results 實驗設定與結果

以下是我們的最終 model 所使用的參數:
Hidden neurons: 正反方向均為 200-200
Batch size: 20
Negative sampling number: 4000
Adam Learning rate: 初始為 0.001，當 10% epoch 過後 loss 都沒有進步時會降低 50%
Subsampling rate: 0.0001
Pretrained embedding: FastText

訓練時間大概是 12 小時，總共跑了一個 epoch，
得到的 public score 是 0.48077，private score 是 0.50385

Team Division 組員分工

討論： 陳啟中、艾弗里、蔣君涵、陳昱維

資料搜集：陳啟中、艾弗里、蔣君涵

程式： 陳啟中

報告撰寫：陳啟中、艾弗里、蔣君涵、陳昱維

報告彙整：陳昱維