MLDS 2017 Spring HW1 - Language Model

B03901056 孫凡耕 B03901070 羅啓心
B03901032 郭子生 B03901003 許晉嘉

1 Environment

1. 硬體資訊：

OS	CPU	GPU	Memory
`Arch linux 4.10`	`i7 3.4 GHz`	`GTX 1070`	`32 GB`

2. 所使用的 `python` library:

`spacy 1.6.0`	`nltk 3.2.2`

2 Model description

LSTM based RNN Language Model

1. Performance：public score : **47.308** %, private score : **50.962%**
   使用參數：
   word vector : glove.6B.300d
   rnn cell : full LSTM (with peephole)
   optimizer : Adam
   clip gradient norm : 5
   sampled softmax : used

2. 流程：
   處理資料 : 將 Training data 轉成一個個單句或是以 dependency tree 的形式轉成若干單句，再將句子中每個單詞轉爲對應的 word vector
   訓練模型 : 使用變動長度多種的 rnn cell，輸入是除掉最後一個字的單句，輸出是通過 softmax 之後與輸入算 loss，使用不同的 gradient descent 方法來訓練模型。
   測試資料 : 將 Testing data 以與 Training data 相同的方式進行處理，將各個選項填入的單句個別丟入訓練完成的模型中，依據個別可能的機率來決定填空的答案。

3. 可調整之參數：
   word vector : 使用的 word vector 來源及維度（選項：`glove.6B.50d`, `glove.6B.100d`, `glove.6B.200d`, `glove.6B.300d`, `glove.42B.300d`, `glove.840B.300d`）
   RNN layer : RNN 層數
   sampled number : 在 sampled softmax loss 中的 classes 數目
   optimizer : optimizers 的類型（選項：`GradientDescent`, `Adadelta`, `Adagrad`, `Momentum`, `Adam`, `RMSProp`）
   rnn type : RNN Cell 的種類（選項：`Basic`, `basic LSTM`, `full LSTM`, `GRU`）
   learning rate : Learning rate 的大小
   max grad norm : 所允許的 gradient norm 最大值，作爲 clipping 之用
   bidirectional : 是否使用 bidirectional RNN model
   epoch num : 一個 epoch 中總共使用多少個 batch
   train num : 總共使用 Holmes Training Data 中做 training 的檔案數目
   keep prob : Dropout layer 所使用的 keeping probability
   batch size : Training 時每個 batch 中 sentence 的數目
   hidden size : Hidden layer 的維度

4. 使用的外部資源：
   Word vector : GloVe (`https://nlp.stanford.edu/projects/glove/`)
   Dependency Tree Parser : spacy (`https://spacy.io/docs/usage/dependency-parse`)

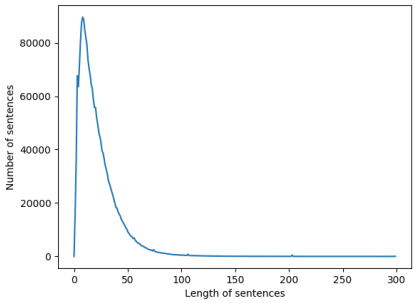

Figure 1: 句子長度分佈

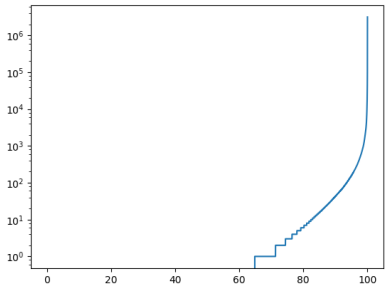

Figure 2: 單字出現次數分佈

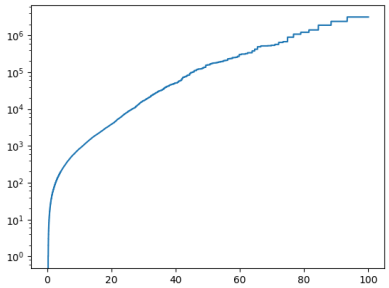

Figure 3: 單字出現次數比例分佈

3 Improvement

1. 僅使用 Training Data 中，長度在特定範圍之內的句子
   特短的句子，大多屬於雜訊或是較無意義的短句（e.g. somebody said,）；特長的句子，也均爲雜訊居多，若納入特長的句子亦會使 training model 維度過高，造成記憶體空間不足。且特短及特長的句子，本身在 Training Data 中所佔的比例也不高（如 Figure 1）。

2. 將出現次數過少的單字從 corpus 中移除
   原本 corpus 的字量過大，會造成記憶體空間不足以及訓練困難，因此，將在 Training Data 中出現次數較少的單字移除（視爲 unknown word），可使訓練學習的過程加速。從上圖 Figure 2 可看出七成左右的單字不在 pretrained corpus 內或是出現次數少於兩次，而從 Figure 3 可看出這些單字又僅佔不到總單字量的一個百分點，因此，將其刪除對於訓練的過程有較大的助益。

3. 將 Training Data 中，開頭及結尾的部分刪去
   由於 Training Data 中幾乎所有文章，開頭及結尾皆相當於目錄或版權資訊等，較不爲一般常用語的句子。有利訓練過程的進行。

4. Dependency Tree
   Dependency tree 可表示句子當中各單字之間的關聯。因此，將資料轉爲 dependency tree 後，可更爲有效的分出每個句子中各個合法的語句，使訓練的資料更爲廣泛且一般。
   舉例來說：對於句子 `An inventory of syntactic functions is taken to be primitive.` 我們可以將其轉爲如圖 Figure 4 的 dependency tree。

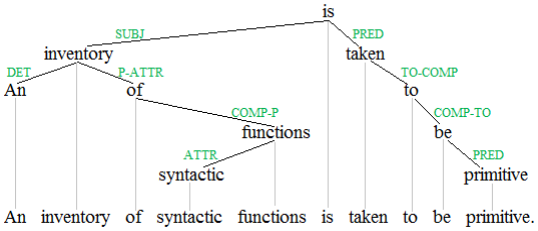

Figure 4: Dependency Tree

5. Word Normalization
   在原先預處理 Training data 時，我們是利用 nltk 套件中的 sent tokenize 以及 word tokenize，但由於 Training data 中，有些單字前後連接一些不重要的字元，形如：`'admit`、`.walk`、`*what`、等單字。而造成原本應是常用的單字，卻被我們判定成 unknown word 的情形。因此，我們將 nltk word tokenize 出來的單字，再進一步判斷，如果他包含兩個以上的英文字母，就將其他所有非英文字母的字元剔除。經過這一個正規化的步驟後，unknown word 的比例降低了 0.5%，並且我們在 public score 上提升了 **7%** ！

4 Experiment

除了嘗試不同的 model 以及各種 improvement，我們也針對以下幾種參數進行實驗，除了試圖尋找最佳的參數，也嘗試驗證關於參數設定的各種傳聞。

1. RNN Cell
   我們共嘗試了四種 RNN Cell，分別為 TensorFlow 1.0 所提供的：BasicRNNCell, BasicLSTMCell, LSTMCell, 以及 GRUCell。其中 LSTMCell 不同於 basic 的版本是有使用 peep-hole。從 Figure 5 可以發現四種 cell 隨著 training epoch 數目的增加，perplexity 下降的幅度並無明顯差異，且基本上在經過一個 epoch 之後 perplexity 便無明顯變化，惟 BasicRNNCell 的振幅略大。儘管如此，在 private data score 的表現上，四種 cell 的分數依序是 0.40769, 0.46731, 0.48077, 0.46923，因此我們仍選擇有 peep-hole 的 LSTMCell 作為 model 中的 RNN cell。

2. Learning Rate
   我們嘗試三種不同的 learning rate，分別為 0.1, 0.01, 0.001。由 Figure 6 可以發現 learning rate 越大，perplexity 下降的速度也越快。然而正如同許多文獻所指出的：太大的 learning rate 可能無法到達 optimal 的點，這個說法亦在我們的實驗中得到驗證。

3. RNN Layer 層數
   在 RNN model 中，我們除了使用單一層的 RNN cell，也有嘗試堆疊至 2 層及 3 層。由 Figure 7 可以發現 RNN cell 的層數越少，training 的速度越快，原因是參數較少的關係。三者在 private data score 上的表現別為 0.47308, 0.48077, 以及 0.49231，可以發現當 RNN cell 的層數為三層的時候，在這個 task 上有較佳的 performance。

4. N-gram Model
   除了以上參數之外，我們也試圖做 model ensemble。其中包含 n-gram、pointwise mutual information (PMI) 等等。在 n-gram 的例子當中，$n = 1 \sim 4$ 於 private data 的分數分別為 0.24615, 0.25962, 0.27885, 0.25769，其中 trigram 的表現最佳。由於我們發現在某些題目中，當空格代入不同的的選項時整句話所計算出的機率相同，因此我們認為或許是 Holmes training corpus 不夠大，使得許多 n-gram 的機率相同，因此沒有辦法有效提升分數。

5. Pointwise Mutual Information (PMI)
   PMI 是我們嘗試使用的另一個 model，他的想法是將一個 sentence 先經過各種方法處理成為各種的 feature sets，這些 feature sets 包含的是一個 sentence 經過 filter 過後剩下的詞彙。得到 feature set 的方法有許多種，例如考慮 stop-word、使用 dependency tree、甚至只取空格前後兩個字等等。將處理完後的 feature sets 中的單字取出，根據 information theory 計算空格候選字與這些單字的關聯性，取出分數最高的即可。這個 model 在 MSR Sentence Completion Challenge 的分數表現高達 61.44%，然而最終我們因為時間因素來不及完成。

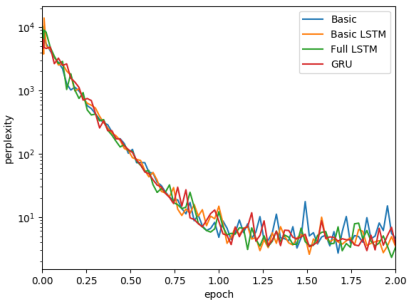

Figure 5: 不同 RNN cell

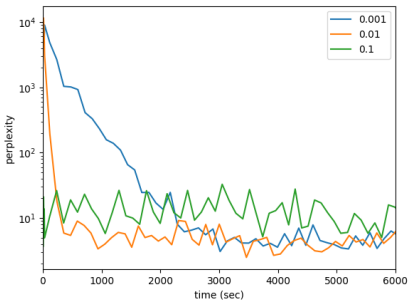

Figure 6: 不同 learning rate

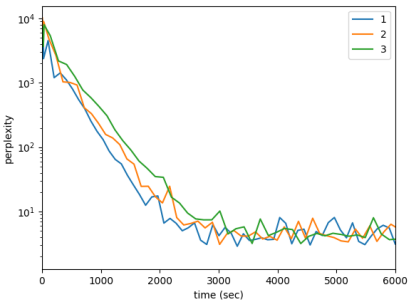

Figure 7: 不同 RNN 層數

5 Team division

孫凡耕	寫 RNN 模型、分配組內工作、教導組員
羅啓心	嘗試寫 Improvement、撰寫報告的模型描述
郭子生	N-gram、跑實驗
許晉嘉	資料處理、PMI、統整撰寫報告、優化 kaggle 分數表現