HW3_NER_CRF

F74084012 竇賢祐

一、 簡介

利用 word vector 加上自定義的 feature 等做訓練，找出不同的 tag 來計算 f1score

二、 過程

1. 以上課所介紹的 BiLSTM 做嘗試
2. 增加自定義的 feature
3. 更改 word vector 檔案
4. 綜合 2.3 點再次測試

嘗試以上三種方法來找出不同的 f1 score

一開始先嘗試修改 CRF 架構，讓 BiLSTM 能夠融入其中，過程中皆使用 Colab 做測試，找過網路上多種範例 code 來修改，但不確定問題出於何處，每次到了要 model fit 的時候總是跑一半就斷線，也未 output 出任何問題，猜可能是記憶體太滿，所以有嘗試過把一些參數調低，但仍然到 fit 的階段就斷線，因此第一個方法沒有成功實作。

由於以上方法實作一直失敗，因此往其他方向做測試，先嘗試增加 feature，除了以 pretrained 中 vector 的數值以外，自己加入三種不同的 feature 來看結果是否有變得更好，但因結果有時並不會達到作業目標的 0.45，所以又再想了其他的方式來測試。

更改 word vector 來測試看看不同的 vector 是否能夠造成不同的效果，原先助教給的 pretrained data 長度為 512，而後來改用的 word vector 長度為 300。

三、 實驗結果

第一個方法因 code 不確定何處出問題因此尚未成功，故接下來以另外兩種方法來說明實驗結果。

第二個方法增加自定義的 feature，在 f1score 表現上確實有些許提升，比起原先只有 vector 的情況可以提升約 0.1~0.2，就結論而言可以推斷說這個 feature 是有助於 CRF 在判斷字詞的，但猜測因為 feature 原先的 vector 就有 512 個，因此即使再加上 1~3 個 feature，實際影響的效果可能也沒那麼顯著，所以提升的範圍不大。

第三個方法修改 word vector，原先助教給的 pretrained data 裡面的內容都是將字轉換為長度為 512 的 vector，然而效果並不是很好，因此嘗試更改 word vector 後，發現 f1score 可以有大幅的提升，因此可以看出在 CRF 的框架下 pretrained data 占了很大的比重，若 pretrained data 訓練得不錯，就可以大幅提升辨識的程度。

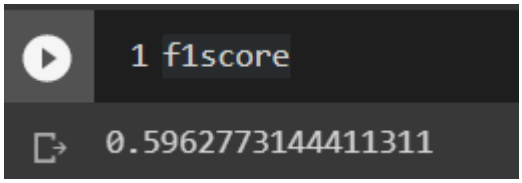
最後一種方法增加自定義 feature + 更改 word vector 檔案，此種方法感覺上應該要有更佳的成效，然而在實作的結果發現和第三種方法相比 f1score 反而稍稍降低了 0.001~0.005 不等，但和第二種方法相比仍然是有大幅的提升，

Feature 代號 O:僅 word vector A: 是否為數字 B:是否為標點符號 C:詞的長度增加的 feature

	O	O + A	O + B	O + C	O+A+B+C
can.cbow.cwe_p.tar_g.512d.0	0.360877	0.352952	0.419648	0.403024	0.374521
cc.zh.300.vec	0.602380	0.601201	0.605160	0.6058434	0.596277

不管是增加哪種 feature，或是綜合起來都能或多或少提升 score，雖然在 cc.zh.300.vec 這個 vector 下，綜合三種 feature 反而降低了 score，但仍能使結果達到 0.59

cc.zh.300.vec + 三種 feature 的 f1score 截圖:



四、 心得

NER 在文章判讀、AI 對話上是一個很重要的部分，之前也有修過課程嘗試做一個聊天機器人，那時也是利用 NER 來幫助判別相關的詞語，並設計相對應的回應，因此 NER 在 AI 對話上佔有一個很重要的地位，藉此可以幫忙把看似複雜的文句，一一拆解，找出一個相對制式化的格式，如此才能方便電腦做判讀。

從實作中知道 NER 存在的困難點，詞語放在不同的句子中可能代表的意義不同，字也會因為前後接的字不同而有不同的意義，這也就導致辨識出來的 tag 不同，雖然兩個詞語中可能共同含有某個字，但實質上所代表的 tag 可能是不同的，這些都大大的提升了 NER 的困難性。