AICUP-2020

「醫病訊息決策與對話語料分析競賽」

秋季賽：醫病資料去識別化

班(成大):電機所己組 電機所己組 工科系

姓名: 蔡亞倫 廖毅軒 李姵萱

學號: N26091801 N26092027 E94061165

老師：高宏宇

中華民國 109年12月30日

目錄

一、問題與分析方法

1. 問題

根據 Health Insurance Portability and Accountability Act (HIPAA) 規定，在臨床醫療端的文字紀錄中，有關病人隱私資料的內容 (Protected Health Information, PHI) 都要被清除掉或是修改掉。而在門診醫病對話資料中，含有許多求診民眾的隱私內容，如此大量的資料，需要有自動化的方式去辨識出這些隱私內容，方便醫療人員的作業也加速醫療大數據的建立。

本競賽主要目標為從醫生與看診民眾對話中辨識和提取含有隱私資訊的內容，並分類出該內容屬於何種隱私類型。以 F1-Score 評估參賽者在測試語料集上預測結果的正確率。

2. 分析方法

我們從三個部份下去分析。Data processing, Model, Feature。
此處先說明Data processing，Model和feature會在稍後說明。

Data Processing：
有嘗試先對Dataset做一些額外處理

- 角色統一：
  "角色"可以變化各句對話是誰說的，但沒有"角色"可能會混淆前後句的語意資訊，因此我們嘗試讓角色單一化，在問診過程中可能有多位醫師、個管師、病人及病人家屬，所以將醫師方的角色統一改成醫師，病人、病人家屬統一更改為病人，此動作減少混淆機器判斷的機會。
- 文字全形轉半形：Dataset中有些英文字採全形顯示有的是半形，藉由增加文字一致性進而提升辨識率。
- 文字全形轉半形 + 角色統一：結合上述兩種處理。

二、模型介紹

1. NER的model

此處嘗試的模型為BiLSTM-CRF，BiLSTM-CRF模型主體由雙向長短時記憶網路（Bi-LSTM）和條件隨機場（CRF）組成，模型輸入是字元特徵，輸出是每個字元對應的預測標籤。

2. Word Embedding的model

此處嘗試word2vec、fasttext、GloVe，共3種word embedding model。

- word2vec
  - Google 於 2013 年由 Tomas Mikolov 等人所提出。
  - 透過學習大量文本資料，將字詞用數學向量的方式來代表他們的語意。並將字詞嵌入到一個空間後，讓語意相似的單字可以有較近的距離。
  - 在學習相似性和線性規律上有很好的表現。
  - 未充分使用co-occurrence的資訊。
- fastText
  - 由Facebook的AI Research（FAIR）實驗室創建。
  - 於官網提供了294種語言的pretrain model。
  - 考慮character-level的特徵，並可以對沒出現過的字詞(out-of-vocabulary words)作猜測、產生其向量。
- GloVe
  - 即Global Vectors
  - 由史丹佛(Stanford)研究團隊所創造，改進word2vec的不足。
  - 相較於word2vec未充分使用co-occurrence資訊，GloVe多考慮了整體資料的統計資訊。

三、實驗環境與結果

1. Pre-Processing
    - 環境
        - Windows10
        - python 3.7.3
        - Anaconda
        - pytorch 1.7.0
    - 資料集
        - 訓練資料集train_2.txt作為training data。沒有區分訓練或validation set
    - 實驗設計與參數設定
        - 使用baseline中的CRF模型(Sklearn-crfsuite)
        - 將三個修改過的dataset分別做training
        - 使用development_2.txt產生output.tsv上傳至系統做最後評估
    - 實驗結果

        可以發現"角色統一"得到的效果最好

Pre-processing	f1 score(系統)
角色統一	0.452431
文字全形轉半形	0.406249
文字全形轉半形+ 角色統一	0.381581

2. NER的model
    - 環境
        - Windows10
        - python 3.7.3
        - Anaconda
        - pytorch 1.7.0
    - 資料集
        - 訓練資料集train_2.txt作為training data。沒有區分訓練或validation set
    - 實驗設計與參數設定
        - 嘗試過許多種的組合最後發現以下的組合結果最好
        - 句子長度40個字切開

        - Epoch = 200,Embedding = 300,Hidden = 256
    - 實驗結果

        F1-score : 0.61

3. Word Embedding
    - 環境
        - Windows10
        - python 3.7.3
        - Google Colab / Anaonda
    - 資料集
        - 訓練資料集train_2.txt，切分2/3為train data，剩下的1/3當作test data作為初步評估。
        - 若test data評估結果佳，才會使用development_2.txt產生output.tsv上傳至系統做最後評估。
    - 實驗設計與參數設定
        - 皆使用baseline中的CRF模型(Sklearn-crfsuite)
        - 比較各種word vector的表現差異
            - word2vec : 使用套件gensim.models中的Word2Vec()，設定參數min_count=1
            - fasttext : 使用套件gensim.models中的FastText()，設定參數 window =3,min_count = 1,min_n =3,max_n =6,word_ngrams =0,iter=500，size=200
            - GloVe : 使用套件glove中的Glove()與Corpus()，Corpus()中參數window=10，Glove()中no_components=200, learning_rate=0.05,epoch=500
            - 由於環境問題，GloVe相關實驗於Google Colab上進行
        - 比較fasttext維度不同時的表現
            - 使用套件gensim.models中的FastText()，設定參數 window =3,min_count = 1,min_n =3,max_n =6,word_ngrams =0
            - 改變維度(參數size):150、200、250
        - 比較fasttext官方pretrain model與自行訓練model 的表現
            - 使用套件gensim.models中的FastText()，設定參數 window =3,min_count = 1,min_n =3,max_n =6,word_ngrams =0,iter=500，size=250

        - 使用fasttext官方pretrain好的dictionary(200萬詞彙)，直接讀入最常用的20萬個詞彙，size=300(由於RAM不夠，無法動態調整其維度)
        - 加入使用jieba做的POS tagging
    - 加入POS tagging的表現
        - word vector固定使用fasttext官方pretrain好的dictionary
        - 觀察不加POS tagging、使用jieba做POS tagging、使用ckip做 POS tagging的差異
- 實驗結果
    - 各種word vector

word vector	f1 score(test)	f1 score(系統)
word2vec	0.41	x
fasttext	0.525	0.48
GloVe	0.456	0.455

    - fasttext維度不同

fasttext 維度	f1 score(test)	f1 score(系統)
150	0.516	x
200	0.525	0.48
250	0.527	x

    - fasttext官方pretrain model與自行訓練model

model	f1 score(test)	f1 score(系統)
官方pretrain	0.628	0.567
自行訓練	0.58	x

    - 加入POS tagging

Pos tagging	f1 score(test)	f1 score(系統)

不加	0.56	x
jieba	0.628	0.567
ckip	0.631	0.45

- 結論
  使用fasttext官方pretrain好的dictionary(300維)，並且加入使用jieba做的POS tagging效果最佳。

四、Dataset bug report

在實作的過程中在 train_2.txt 有發現一些標註有問題的地方，以下分點說明：

1. 應標註而未標註：

atticle_id	未標到	類型
107	下個月	time
124	媽媽	family
109	阿公	family
109	阿嬤	family
116	阿公	family
116	阿嬤	family
116	爸爸	family
116	媽媽	family
38	405	med_exam
38	10	med_exam
38	19	
28	曾進忠	name
28	四周	time
26	十樓	location
26	一百三	med_exam
26	一百四	med_exam

2. 錯字：

atticle_id	錯誤	更正
117	12崴多	12歲多
117	民衆	民眾
115	民衆	民眾

五、心得

身為第一次接觸NLP的新手，在參加這次的比賽其實挫折不少，從一開始不知從何下手到開始查找資料、尋找適合的特徵與模型都遇到不少困難，甚至也遇過預測結果幾乎完全錯誤的慘況，但看到f1 score有慢慢的成長還是很開心。

一開始採取分頭進行的策略是我們最大的失策，原先以為這樣會更有效率，最後要合併時卻完全無法相容，導致三個人的努力最後只能呈現出一人的結果，這是我們感到最可惜的部分。

我們在聽取其他組別的報告後也從他們的做法中吸收到了不少精華與有趣的想法，並且大概知道怎樣的做法會得到不錯的結果，像是模型或特徵的選擇、資料集如何事先分析與然後找到預測重點等。

各種專案的實作都必須有一定的基礎知識，並且對資料集有足夠的了解，才能夠實作的更有深度。有了這次的比賽經驗與同學們的分享，往後若是再參與類似的比賽或專案相信我們會更有頭緒與效率去執行。在時間允許的情況下，應該先去補足基礎知識、看一些Paper去了解目前學業界最新的想法以及做法後仔細分析資料集及其所適合的演算方式，並結合這次所吸取的經驗，說不定可以因此激盪出更好的實作方式。

六、外部資源集參考文獻

1. 中研院斷詞系統 : https://github.com/ckiplab/ckiptagger/wiki/Chinese-README
2. fasttext官網(pretrain model下載) : https://fasttext.cc/docs/en/crawl-vectors.html
3. ADVANCED: MAKING DYNAMIC DECISIONS AND THE BI-LSTM CRF:

   https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html