NLP project 2

第四組 **B03902015** 簡瑋德 **B03902021** 郭冠宏 **B02902005** 張志宏

Division of Work

- Preprocessing - B02902005
- Model Design - B03902015
- Eexperiment - B03902021

Run

```
github url : https://github.com/Cabalelrsky2000/NLP
python model_poolgru_other.py(get outputfile : "output_poolgru_other.txt")
```

Method

Task Introduction

這次的專案是希望給定句子中的特定兩個詞，分析兩者之間的關係，兩個詞之間的關係分為九類加上其他類，我們決定使用句子的依賴樹、詞向量和詞性當作模型的輸入，並運用機器學習架構的模型來預測句中兩詞的關係，我們將問題著重於十八類加上其他類的分類問題，並使用"遞歸神經網路"配上傳統機器學習和深度學習來完成這次的專案。

Featre Extraction

Building Dependency Tree

因為在這次的project中牽涉到字詞與字詞之間的關係，所以我們選擇使用dependency parse tree中的資訊作為進行training的features。

我們使用stanford parser 3.9.1，對每個句子建立帶有POS標記的dependency tree，再從中抽取features作為分類使用。

其中，我們使用了wordsAndTags及typedDependencies兩種outputFormat：先透過前者完成語句的tokenization及POS tagging，接著從後者得出的所有dependency關係建立dependency tree。

另外，由於目前stanford parser這個工具所做的dependency parse提供的結果是dependency graph可能含有環(cycle)，未必是dependency tree，必須加上在執行時加上

```
-outputFormatOptions "treeDependencies"
```

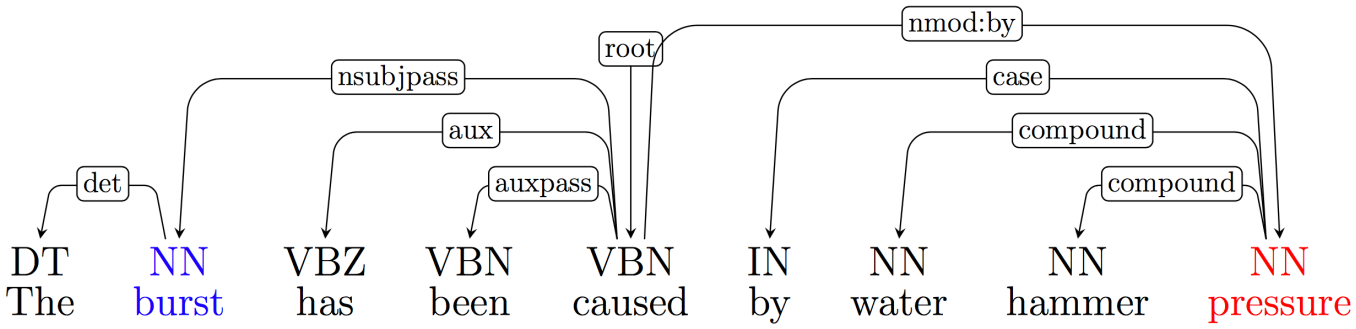
的額外參數以確保dependency graph為最有可能的樹狀結構。

以training data中的其中一個句子"The <e1>burst</e1> has been caused by water hammer <e2>pressure</e2>."為例，以下分別為wordsAndTags與typedDependencies的結果：

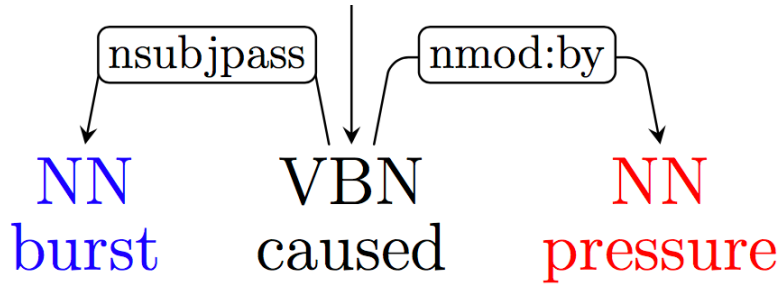
```
The/DT burst/NN has/VBZ been/VBN caused/VBN
    by/IN water/NN hammer/NN pressure/NN ./.

det(burst-2, The-1)
nsubjpass(caused-5, burst-2)
aux(caused-5, has-3)
auxpass(caused-5, been-4)
root(ROOT-0, caused-5)
case(pressure-9, by-6)
compound(pressure-9, water-7)
compound(pressure-9, hammer-8)
nmod:by(caused-5, pressure-9)
```

經過前述的處理我們就可以得到dependency parse tree：



得到dependency tree之後，我們擷取兩個name entity節點之間的最短路徑，以節點(token+POS)-關係(dependency)-節點-...-節點的格式，作為training data。從前面提過的例子，burst(entity1)到pressure(entity2)的最短路徑即為：

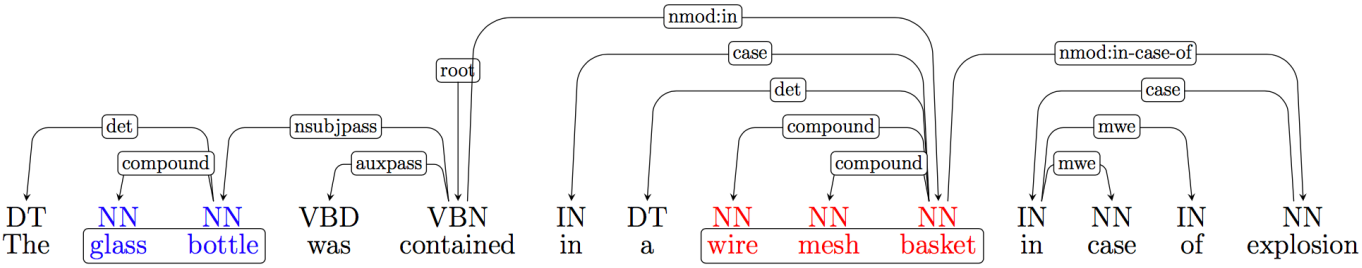


其中我們對路徑上的dependency之方向性做出標記，即此路徑是從樹狀結構中的子節點走向父節點還是相反。並且我們參考OO paper中的nagative sampling的方式同時使用了雙向的路徑作為training data，所以將entity1到entity2以及entity2到entity1兩個方向的路徑都分別抽出。

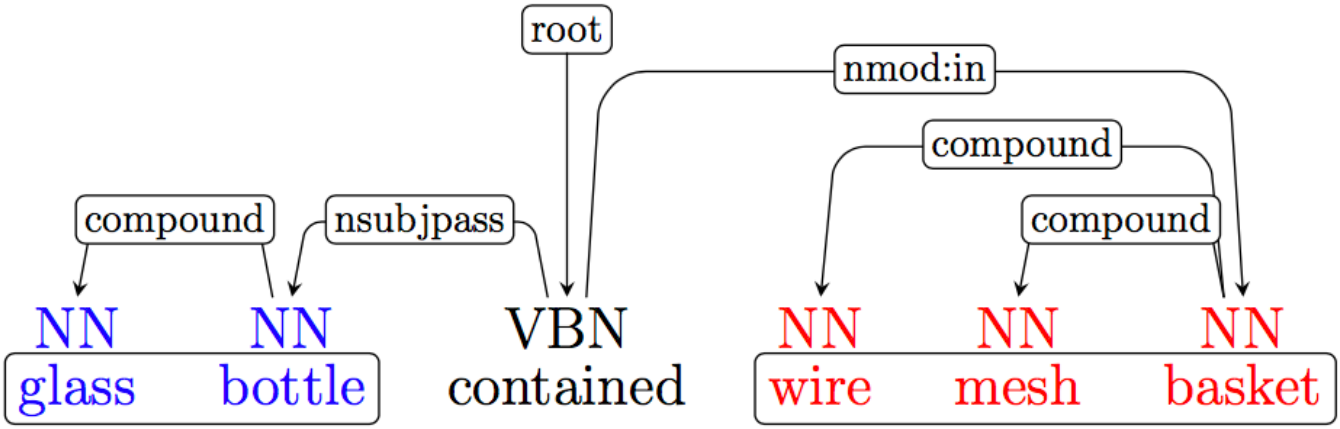
$$\text{burst(NN)} \xrightarrow{\text{nsubjpass(-)}} \text{caused(VBN)} \xrightarrow{\text{nmod:by(+)}} \text{pressure(NN)}$$

$$\text{pressure(NN)} \xrightarrow{\text{nmod:by(-)}} \text{caused(VBN)} \xrightarrow{\text{nsubjpass(+)}} \text{burst(NN)}$$

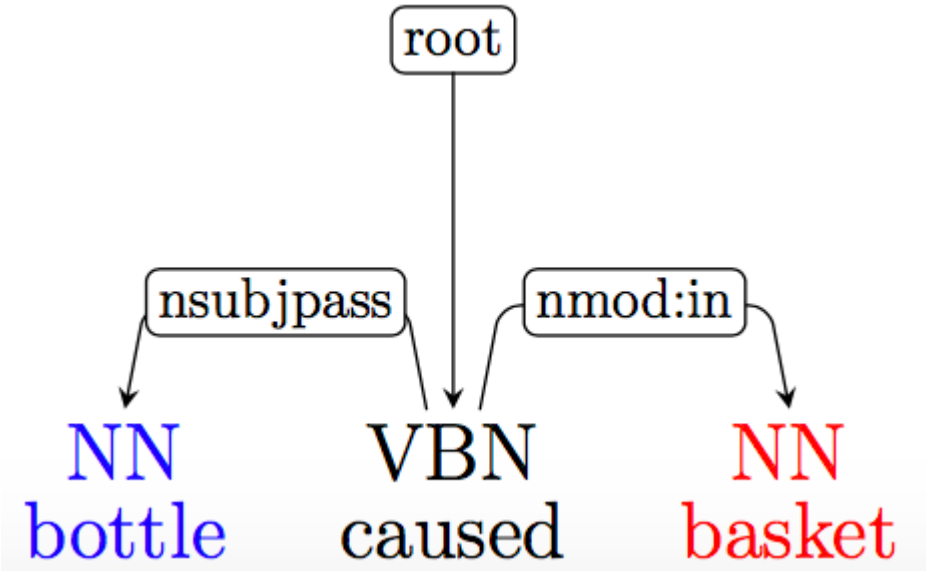
除此之外，對於由兩個以上token組成的name entity，為避免這些token本身形成樹狀結構無法取得單一的路徑，我們只保留位於dependency tree中較靠近root的token來作為路徑的起點或終點。由於dependency tree中的parent節點永遠都是一個dependency關係中的head，而child節點們則是對於head的modifier，此方法同時也能確保這些由多個tokens組成的name entity中較為重要的head可以被保存。



以另外一個句子"The <e1>glass bottle</e1> was contained in a <e2>wire mesh basket</e2> in case of explosion."為例，若在選擇entity2的端點時選擇basket以外的節點，仍不能包含entity2裡所有的節點。



因此僅選擇head為端點，最後的路徑為：



Word Embedding

在詞向量的部分，我們使用"Glove"，將給予的資料句子轉成向量表示，我們使用的是"Wikipedia2014"+"Gigaword5"所提供維度為300的向量表示。
針對英文詞部分，我們會先把有"_"和"-"的詞分開，並利用"nltk"的"lemmatization"將有"ed"、"s"等轉回原本詞的型態，最後換成小寫再用"Glove"中對應的詞向量表示，其中忽略找不到的字並"padding"至固定長度。

Classification

分類部分，為了讓模型有學到方向性，我們從九類擴展成十九類的分類問題，前九類作為順向、第十類為其他類、後九類則為逆向。

訓練過程中我們將每筆訓練資料都產生了兩個方向的"Dependency Tree"和詞向量並配上對應的分數(正負向)，而測試資料也同樣生出兩個方向的結果，並把兩個方向對於十九類相同的結果合併作為最後的答案，舉例來說(第一筆測試資料)，「<e1>audits</e1> were about <e2>waste</e2>」就會分別用關係樹和詞向量等並經過模型產生順向和反向的兩個"one hot"陣列

```
順向的 : [0.1,0,0,0,0,0,0,0,0,0,0.9,0,0,0,0,0,0,0,0]
反向的 : [0.8,0,0,0,0,0,0,0,0,0,0.2,0,0,0,0,0,0,0,0]
```

兩者根據相對應的答案相加，即順向的順向加上逆向的逆向、順向的逆向加上逆向的順向(0.1+0.2作為第一類、0.9+0.8作為第十一類的結果)，最後取最大值得到第十一類"Cause-Effect(e2,e1)"。

Experiment & Discussions

Model Description

Features | Model	Baseline+NN	Glove+NN	Lstm+NN	Lstm(dir)+NN
Glove(e1,e2)	–	V	–	–
Glove(sentence)	–	–	V	V
POS(onehot)	V	V	V	V
POS(sequence)	–	–	–	–
Relation(onehot)	V	V	V	V
Relation(sequence)	–	–	–	–

Features | Model	GRU+NN	poolGRU+NN	poolGRU+XGB	poolGRU+Extrat
Glove(e1,e2)	–	–	–	–
Glove(sentence)	V	V	V	V
POS(onehot)	V / -	V	V	V
POS(sequence)	V / -	–	–	–
Relation(onehot)	V / -	V	V	V
Relation(sequence)	V / -	–	–	–

Baseline Model

Assumption

只用上"one hot"的"pos tag"和"relation"並配上簡單的DNN model，得到一個簡單的baseline

Architecture

Input => Dense(2048) => Dense(1024) => Dense(19)

Result

Evaluation	–
Accuracy	28.82%
P	31.64%
R	32.57%
F1	32.10%

Discussion

baseline model只做得比random好一些些(大概10%)

LSTM+NN Model

Assumption

把e1、e2字的"word embedding"拉進來，和"pos tag"和"relation"接起來，可以讓正確率大概到40多，之後決定把e1到e2句子做 lstm預期得到更好的結果，然後在 lstm 的部分分別試了把"hidden layer"全部接起來和只拿最後一層

Architecture

Input => LSTM(256) => Concatenate => Dense(2048) => Dense(1024) => Dense(19)

Result

Evaluation	return_sequences,128	return_sequences,256	256
Accuracy	67.83%	68.46%	66.88%
P	68.79%	68.74%	66.92%
R	80.95%	82.10%	80.29%
F1	74.38%	74.83%	73.00%

Discussion

只拿最後一層的結果比全拿稍微差了一點，且可以看到256維的 lstm 表現較 128維來的好，但是

GRU+NN Model

Assumption

想說把 lstm 換成 GRU 256 維 ，一樣是把 GRU 過完後接"pos tag"和"relation"，看結果會不會比較好

Architecture

Input => GRU(256) => Concatenate => Dense(2048) => Dense(1024) => Dense(19)

Result

Evaluation	–
Accuracy	67.50%
P	68.31%
R	80.29%
F1	73.82%

Discussion

結果跟用 lstm 256維差不多，意料之內。

poolGRU+NN Model

Architecture

Input => GRU(256) => Maxpooling、Averagepooling => Concatenate => Dense(2048) => Dense(1024) => Dense(19)

Assumption

再換上更強一點的model - poolGRU，在過完 GRU 後分別做"max pooling"和"average pooling"並concate起來放在前面，其他一樣

Discussion

在仔細調了一下參數之後，結果上升了不少，再觀察結果之後發現其他類做得特別不好，於是在判定類別的時候設定一個threshold。

Assumption

在計算分數時，因為其他類不屬於前九類，所以學習到的效果可能有限，因此最後算出機率時若分數最高的那類少於一定的threshold時，我們就判定為其他類。

Result

Evaluation	poolGRU	poolGRU(other threshold)
Accuracy	73.54%	**78.43%**
P	73.78%	**80.96%**
R	87.54%	**85.29%**
F1	80.07%	**83.06%**

Discussion

結果就過 baseline 的，其實沒有很意外，因為原本做出來的其他類真的很差，400多個只找出10多個，在經過threshold的判定後找了200多個出來，而這個也是最好的結果

poolGRU+Xgboost Model

Assumption

利用傳統機器學習的 xgboost 之後用來做 ensemble

Result

Evaluation	n_estimators=120,max_depth=8	n_estimatoes=150,max_depth=8
Accuracy	71.81%	71.84%
P	72.01%	72.04%
R	85.95%	86.08%
F1	78.36%	78.44%

Discussion

在試過各種參數之後，發現xgboost在 150, 8 的時候最好，且和 NN 結果相差不遠

poolGRU+Extratree Model

Assumption

一樣是用來做 ensemble，用"sklearn"的"extratree"來實作

Result

Evaluation	n_estimators=1000,max_depth=23	n_estimatoes=1200,max_depth=1
Accuracy	72.36%	70.48%
P	72.46%	70.62%
R	86.47%	84.45%
F1	78.96%	76.92%

Discussion

試過各種參數之後發現extree的結果也是挺不錯的

Ensemble

Assumption

試了兩種方法，第一種是直接拿 GRU+NN 、 XGB 、 Extratree 三個分數加起來來選出分數最高的，第二種則是投票，若是意見歧異則選擇目前最好的 GRU+NN，應該要在進步一點點

Result

Evaluation	分數相加	**vote**
Accuracy	75.19%	77.70%
P	88.22%	80.73%
R	76.45%	84.62%
F1	81.91%	82.63%

Discussion

跟想法有些差距，並沒有更好的成績，推測是因為 model 數量不多且兩個"tree based" model 結果相差不大，再加上其他類的偏差，因此沒有得到更好的結果。