0. 檔案介紹

/ (根目錄)

訓練用的資料集

double\_quotes\_train.csv => 以「""」填補空缺的欄位

NA\_train.csv => 以「N/A」填補空缺的欄位

測試用的資料集 (惡意)

double\_quotes\_mali.csv => 以「""」填補空缺的欄位

NA\_mali.csv => 以「N/A」填補空缺的欄

攻擊用的資料集 (由2\_check\_malicious.py產出，對\*\_mali.csv中為被分類成惡意憑證做攻擊)

double\_quotes\_attack.csv

ps. 如果data沒有feature，可能會造成攻擊中斷3\_textattack1.py

\*.ipynb => 紀錄BERT model訓練過程

program/

0\_preprocessing.py => 生成csv資料集

1\_train\_model.py => 訓練 BERT model (cased or uncased)

2\_check\_malicious.py => 挑出部分壞憑證資料，並讓model將其分類，分類正確的資料將用於adversary

3\_textattack1.py => 框架1 (有bug...)

4\_textattack2.py => 框架2 (有bug...)

func\_def.py => 定義的function

1.

unhandled field

{'jurisdictionC': 10, 'jurisdictionST': 9, 'jurisdictionL': 1, 'businessCategory': 8, 'serialNumber': 8,

'subjectAltName': 4, 'name': 2, 'initials': 2, 'dnQualifier': 2, 'street': 2}

影響的cert

benign: 10

malicious: 4

# 最後因為考量只選了四個欄位

2.

因為有些O當中有使用',' 所以不能用','當作分隔符，不然會檢查錯誤。

3.

benign missing percentage:

{'C': 0.9475, 'ST': 0.947875, 'L': 0.948125, 'O': 0.997125, 'OU': 0.9985, 'CN': 0.0515, 'emailAddress': 1.0}

{'C': 7580, 'ST': 7583, 'L': 7585, 'O': 7977, 'OU': 7988, 'CN': 412, 'emailAddress': 8000}

{'C': 0.0, 'ST': 0.997625, 'L': 0.999125, 'O': 0.94625, 'OU': 0.996875, 'CN': 0.94725, 'emailAddress': 1.0}

{'C': 0, 'ST': 7981, 'L': 7993, 'O': 7570, 'OU': 7975, 'CN': 7578, 'emailAddress': 8000}

malicous missing percentag:

{'C': 0.584, 'ST': 0.685, 'L': 0.6995, 'O': 0.5905, 'OU': 0.793, 'CN': 0.1105, 'emailAddress': 0.969}

{'C': 1168, 'ST': 1370, 'L': 1399, 'O': 1181, 'OU': 1586, 'CN': 221, 'emailAddress': 1938}

{'C': 0.244, 'ST': 0.666, 'L': 0.689, 'O': 0.589, 'OU': 0.8075, 'CN': 0.455, 'emailAddress': 0.97}

{'C': 488, 'ST': 1332, 'L': 1378, 'O': 1178, 'OU': 1615, 'CN': 910, 'emailAddress': 1940}

4.

train 1: 使用六個欄位去訓練 subject["C", "O", "CN"]、issuer["C", "O", "CN"]，但是準確率不高(低於0.8)

train 2: 選擇一張憑證中的四個欄位去訓練 subject["CN"]、issuer["C", "O", "CN"]

train 3: epoch 20, batch\_size 64，結果依然不好

train 4: 採用 N/A方式處理，accuracy高於0.95，至於uncased和cased模型則沒差。

=> 可能資料的編碼方式嚴重影響準確率

5. textattack安裝 (本地)

https://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype

有一個很煩的套件要另外裝(pycld2)，要去下載檔案，而且要符合版本cp。

另外numpy 版本可能要調整，pip install numpy==1.20.3

6. 解析憑證問題

有時候會產生base64編碼或著是其他的非ascii碼(ex. unicode)，無法顯示到憑證中。

# 另外可以考慮加入不可見的文字或是中文到憑證中，去測試是否容易被分類成好的或壞的。

7. BERT model file (存放在雲端)

(1) BERT model uncased (dataset: double\_quotes\_train.csv, epoch: 10)

https://drive.google.com/file/d/1BGXclV06ADpHSQlMLWF1BfhdyVwQ-HRL/view?usp=sharing

uncased ID => 1BGXclV06ADpHSQlMLWF1BfhdyVwQ-HRL

# BERT model太大且是用GPU訓練的，所以將其放到雲端，並選擇使用colab載入model(因為我電腦沒網卡)，

用其做text attack。

8. colab 指令

在colab移動路徑 => %cd <dir>

在colab刪除檔案 => !rm -rf <dir>

!rm -rf Project

9. 目前任務

(1) 將資料夾內的malicous\_dataset中的cert分類，取出預測正確的。(生成target\_dataset.csv)

malicious\_dataset.csv => 136 data, 正確預測的有127筆資料，將用於adversary attack。

# accuracy 0.93

(2) 學會使用textattack

learning...

GPT生成的code都有bug...

# 參考資料

https://blog.csdn.net/weixin\_36378508/article/details/128035441?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522169752309716800184113899%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request\_id=169752309716800184113899&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~first\_rank\_ecpm\_v1~rank\_v31\_ecpm-3-128035441-null-null.142^v96^pc\_search\_result\_base4&utm\_term=textattack%20BERT&spm=1018.2226.3001.4187

10. attack result

(0) model: bert-base-uncased (10 epoch)

+---------------------------------------------------+

Epoch 10/10

Validation Accuracy: 0.98125

mean\_squared\_error: 0.01875

precision recall f1-score support

0 0.99 0.95 0.97 639

1 0.98 0.99 0.99 1441

accuracy 0.98 2080

macro avg 0.98 0.97 0.98 2080

weighted avg 0.98 0.98 0.98 2080

+---------------------------------------------------+

(1) PWWSRen2019

+-------------------------------+--------+

| Attack Results | |

+-------------------------------+--------+

| Number of successful attacks: | 89 |

| Number of failed attacks: | 414 |

| Number of skipped attacks: | 1 |

| Original accuracy: | 99.8% |

| Accuracy under attack: | 82.14% |

| Attack success rate: | 17.69% |

| Average perturbed word %: | 31.58% |

| Average num. words per input: | 6.17 |

| Avg num queries: | 40.69 |

+-------------------------------+--------+

攻擊模式 => 盡量將內容變成完整(常見)的句子、且會將濃縮的大寫單詞拉長。

ex.

[[domain]].com [[US]] My [[Company]] [[Name]] LTD. [[domain]].com

[[demesne]].com [[America]] My [[companionship]] [[diagnose]] LTD. [[arena]].com

[[example]].com [[GB]] [[Global]] Security [[example]].com

[[lesson]].com [[gigabit]] [[world-wide]] Security [[illustration]].com

disaaxpalallow.me [[GB]] COMODO [[CA]] Limited COMODO RSA [[Domain]] Validation Secure Server [[CA]]

disaaxpalallow.me [[sarin]] COMODO [[Calif]]. Limited COMODO RSA [[world]] Validation Secure Server [[Ca]]

(2) TextFoolerJin2019

+-------------------------------+--------+

| Attack Results | |

+-------------------------------+--------+

| Number of successful attacks: | 131 |

| Number of failed attacks: | 372 |

| Number of skipped attacks: | 1 |

| Original accuracy: | 99.8% |

| Accuracy under attack: | 73.81% |

| Attack success rate: | 26.04% |

| Average perturbed word %: | 32.28% |

| Average num. words per input: | 6.17 |

| Avg num queries: | 51.57 |

+-------------------------------+--------+

bert\_uncased

|  |  |  |
| --- | --- | --- |
| dataset\attack\_method | TextFoolerJin2019 | PWWSRen2019 |
| double\_quotes | 22.6% | 13.52% |
| NA |  |  |
|  |  |  |