Progess of the Project

Tsung-Min Pai

2023/8/30

Outline

- **GAT**
  - **Dataset**
  - **Experiment**
- **TRAM**
- **Future Work**

Graph Attention Network - GAT

Graph Attention Network - GAT

Model:

```
class GAT(nn.Module):
    def __init__(self, in_dim, hidden_dim, out_dim, num_heads, dropout_prob=0.2):
        super(GAT, self).__init__()
        # do not check the zero in_degree since we have all the complete graph
        self.layer1 = GATConv(in_dim, hidden_dim, num_heads=num_heads, activation=F.relu, allow_zero_in_degree=True)
        self.layer2 = GATConv(hidden_dim * num_heads, out_dim, num_heads=num_heads, allow_zero_in_degree=True)

        # Adding Dropout for regularization
        self.dropout = nn.Dropout(dropout_prob)

    def forward(self, g, h):
        # Apply GAT layers
        h = self.layer1(g, h)
        h = h.view(h.shape[0], -1)
        h = F.relu(h)
        h = self.dropout(h)
        h = self.layer2(g, h).squeeze(1)

        # Store the output as a new node feature
        g.ndata['h_out'] = h

        # Use mean pooling to aggregate this new node feature
        h_agg = dgl.mean_nodes(g, feat='h_out')
        return h_agg
```

- Use the **new** verison of the dataset

Dataset

Format:

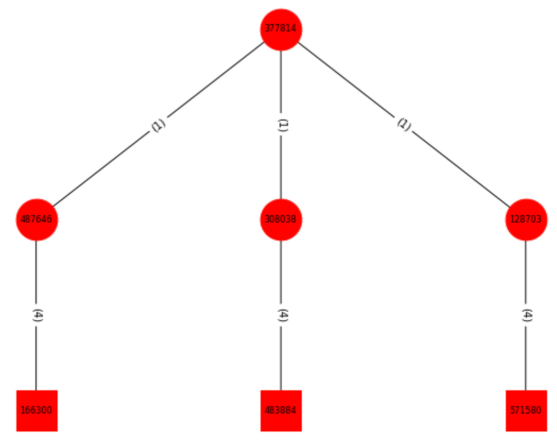

```
{"label": 10, "num_nodes": 3, "node_feat": [205565, 733769, 250773], "edge_attr": [23, 23], "edge_index": [[0, 0], [1, 2]]}
{"label": 11, "num_nodes": 3, "node_feat": [470650, 663446, 627322], "edge_attr": [23, 23], "edge_index": [[0, 0], [1, 2]]}
{"label": 15, "num_nodes": 2, "node_feat": [9863, 103498], "edge_attr": [23], "edge_index": [[0], [1]]}
{"label": 16, "num_nodes": 2, "node_feat": [157277, 753159], "edge_attr": [23], "edge_index": [[0], [1]]}
{"label": 22, "num_nodes": 36, "node_feat": [83068, 614681, 444724, 266227, 121794, 623948, 116790, 769462, 255741, 169794,
```


- Have 165 APs, each AP has 1000 variation → nodes are different but relations are same
- 0~99 test, 100~199 validation, 200~999 train → 1:1:8

- Use transR_50, transE_50, transH_50, secureBERT… as embedding → **8 versions**
- Benign → benign.txt

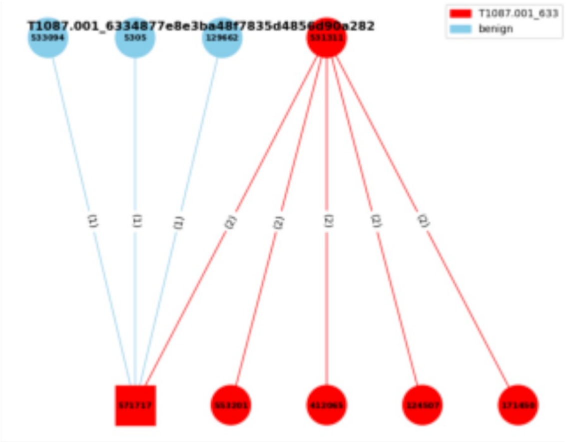
Experiment

- **Experiment 1:**
  - Dataset is 165 APs with 8 versions of embedding
  - Graph classification



- **Experiment 2:**
  - Experiment 1 + **benign** data
  - Benign made from benign.txt → 1000 graphs
  - Graph classification

- **Experiment 3:**
  - Consider the **neighbor** benign nodes
  - Edge classification



Experiment 1

- Total: 100 epochs
- About 30 epochs, they seem to be have the similar performance
- Except **secureBERT**, all about 40% test accuracy
- secureBERT early stopped at epoch 50 with 10% test accuracy
  - Since the dimension of the embedding is way more larger ?!
  - Dimension of SeureBERT's node embedding is from 768 → 250

- Record the training in a **log file**
- Also record the **classification report** supportedd by sklearn

Experiment 1

- Log file:

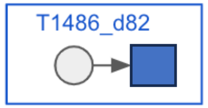
```
08/29/2023, 09:04:02# labels of Validation: tensor([ 96,   4, 129, 111,  84, 162, 102, 103, 132, 114,  32, 110,  57,  85,
        149,  74], device='cuda:3') torch.Size([16])
08/29/2023, 09:04:02# predicted of Validation: tensor([ 96, 153, 113, 153,  84, 153, 153, 113, 161, 113,  32, 113, 153,  85,
        149, 153], device='cuda:3') torch.Size([16])
08/29/2023, 09:04:03# Validation Loss: 2.6715 | Validation Accuracy: 0.3879
```

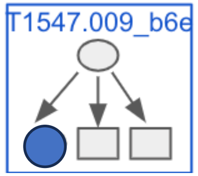
- Classification report:

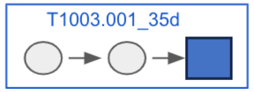
	precision	recall	f1-score	support
T1003.001_0ef4cc7b-611c-4237-b20b-db36b6906554	1.00	1.00	1.00	100
T1003.001_35d92515122effdd73801c6ac3021da7	1.00	1.00	1.00	100
T1003.002_5a484b65c247675e3b7ada4ba648d376	0.00	0.00	0.00	100
T1003.002_7fa4ea18694f2552547b65e23952cabb	1.00	1.00	1.00	100
T1003.003_9f73269695e54311dd61dc68940fb3e1	0.00	0.00	0.00	100
T1003.003_f049b89533298c2d6cd37a940248b219	0.00	0.00	0.00	100

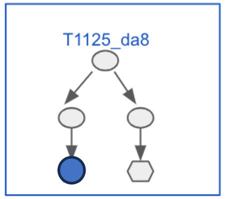
Similar with the MLP, RNN:
More triplets, more accurate

Experiment 2 & 3

- Experiment 2:
  - Haven't made the dataset of benign yet
  - Dataset would be
    - 400: Leaf nodes + their source nodes 

    - 300: Leaf nodes + their source nodes with source nodes' neighbor nodes 

    - 200: Leaf nodes + their 2 layer source nodes 

    - 100: Leaf nodes + their 2 layer source nodes with source nodes' neighbor nodes 

- Experiment 3:
  - After finishing the experiment 2, discuss with Euni

TRAM

Format

On TRAM:

1	[2] Enterprise T1219 Remote Access Software RTM has used a modified version of TeamViewer and Remote Utilities for remote access.
	[2] Enterprise T1204 .002 User Execution: Malicious File RTM has attempted to lure victims into opening e-mail attachments to execute malicious code.
1	[2] Enterprise T1102 .001 Web Service: Dead Drop Resolver RTM has used an RSS feed on Livejournal to update a list of encrypted C2 server names.

Technique Add...	Confidence
T1102 - Web Service	63.7%

Exported json file:


```
\n\n  1:39:16\n現正播放\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n          What's NEW in React
,"mappings":[{"id":17910,"attack_id":"T1087","name":"Account Discovery","confidence":"67.9"}]},
```


Json file includes: Text, attack_id, confidence

- Should be processed based on the confidence

Result

json files → csv files:

The post also directed users as to how they could avail of th	T1543
There has been no subsequent activity from <strong>corind	T1068
Do not be shy’ These posts were recovered th	T1070.004
A ‘Regular’ campaign will necessitate a decryption key for e	T1573

All csv files → xlsx file:

file_name	T1218	T1021.002	T1562.003
Report for Cyberattack-on-Ukrainian-state-authorities-using-the-Cobalt-.html	0	0	0
Report for Group-description-Cleaver.html	0	0	0
Report for Analysis-Report-FiveHands-Ransomware.html	0	0	0

- with all files and TTPs
- Record the # of the labels detected in the specific file
- 141 unique TTPs
- pdf: 109/751, html: 378/3937

Future Work

Future Work

- **GNN**
  - Do the experiment 2 and 3
  - Improve the performance of the model (if available)

- **TRAM**
  - Figure out the reason of low efficiency
  - Processed the data based on the confidence

Thanks!!

Appendix

Graph Convolutional Network - GCN

Graph Convolutional Network - GCN

Model:

```python
class GCN(nn.Module):
    def __init__(self, in_feats, hidden_size, num_classes):
        super(GCN, self).__init__()
        self.conv1 = GraphConv(in_feats, hidden_size)
        self.conv2 = GraphConv(hidden_size, num_classes)

    def forward(self, g, inputs):
        h = self.conv1(g, inputs)
        h = torch.relu(h)
        h = self.conv2(g, h)

        g.ndata['h'] = h
        hg = dgl.mean_nodes(g, 'h')
        return hg
```

- Use the **old** verison of the dataset
- Use **DGL** to be our library
- DGL data format:

```
batched_g is like:
Graph(num_nodes=96, num_edges=160, ndata_schemes={'feat': Scheme(shape=(1,), dtype=torch.int64)}, edata_schemes={})
num_nodes = 3*batch_size, num_edges = 5*batch_size

labels is like: tensor([ 76,   0,   0,   0,   0,   0,   0,   0,   0,  76,   0,  76,   0,   0,
                          0,   0,  76,   0,  30,  92,   0,   0,  76,   0,   0,   0,   0,   0,
                        116,   0,  76,  76])
```

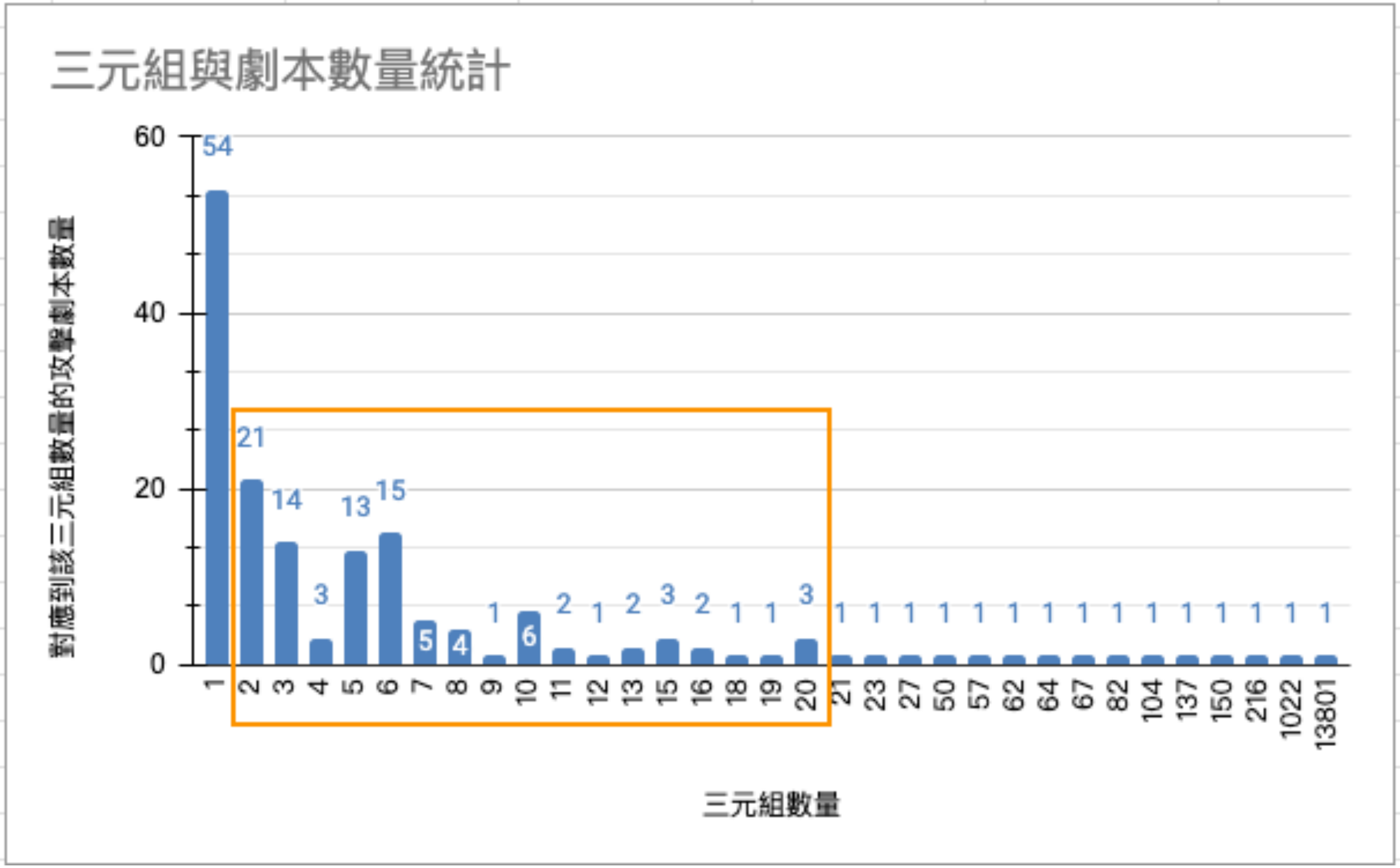
Result:

```
  0%|          | 0/120 [00:00<?, ?it/s]
Epoch 0 | Train Loss: 2625.5943 | Train Accuracy: 0.4763
  1%|          | 1/120 [00:56<1:52:21, 56.65s/it]
Validation Loss: 494.0275 | Validation Accuracy: 0.6642
 99%|█████████▉| 119/120 [1:51:06<00:55, 55.13s/it]
Validation Loss: 0.9964 | Validation Accuracy: 0.6642
Epoch 119 | Train Loss: 0.9625 | Train Accuracy: 0.6644
100%|██████████| 120/120 [1:52:03<00:00, 56.03s/it]
Validation Loss: 0.9965 | Validation Accuracy: 0.6642

Test Accuracy: 66 %
```

- GAT applied on the old data has the similar result

Appendix



三元組與劇本數量統計
對應到該三元組數量的攻擊劇本數量
60
40
20
0
54 21 14 3 13 15 5 4 1 6 2 1 2 3 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 18 19 20 21 23 27 50 57 62 64 67 82 104 137 150 216 1022 13801
三元組數量