Log: Ssl Dynamic Graph

Anak Wannaphaschaiyong

November 8, 2021

Contents

1 Research Date: *[2021-10-12 Tue]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

1.1 Error Log 1: Error is observed after running

1.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

ran the following command in anaconda prompt (gpu can be accessed) cause the error

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

The first error is as followed

```
C:/cb/pytorch_1000000000000/work/aten/src/ATen/native/cuda/IndexKernel.cu:142: block:
C:/cb/pytorch_1000000000000/work/aten/src/ATen/native/cuda/IndexKernel.cu:142: block:
```

The second error is as followed

```
Traceback (most recent call last):
  File "train_self_supervised.py", line 313, in <module>
    test_ap, test_auc = eval_edge_prediction(model=tgn,
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\evaluation\evaluation.py", line
    pos_prob, neg_prob = model.compute_edge_probabilities(sources_batch, destinations_
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\model\tgn.py", line 210, in comp
    source_node_embedding, destination_node_embedding, negative_node_embedding = self.
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\model\tgn.py", line 148, in comp
    node_embedding = self.embedding_module.compute_embedding(memory=memory,
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\modules\embedding_module.py", li
    source_embedding = self.aggregate(n_layers, source_node_features,
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\modules\embedding_module.py", li
    source_embedding, _ = attention_model(source_node_features,
  File "C:\Users\terng\anaconda3\envs\py38\lib\site-packages\torch\nn\modules\module.p
    result = self.forward(*input, **kwargs)
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\model\temporal_attention.py", li
    neighbors_padding_mask[invalid_neighborhood_mask.squeeze(), 0] = False
RuntimeError: CUDA error: device-side assert triggered
```

2 Research Date: *[2021-10-14 Thu]*

- ref

  - https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F

2.1 Research Log 1: Minor error in tgn is detected in evaluation of validating model with new nodes. Does error has any sideffect?

2.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I was making notes on the code and detect this minor error that during evaluation of new nodes validation set, $\text{val}_{\text{randsampler}}$ were passed to $\text{eval}_{\text{edgeprediction}}$ instead of $\text{nn}_{\text{valrandsampler}}$. I have also check with the original code that that this error exists in the original code base as well. To further confirm the detected bug, the model performs slightly worse in the correct validated data compared to the incorrect validation data.

before

```
INFO:root:start 0 epoch
INFO:root:epoch: 0 took 178.85s
INFO:root:Epoch mean loss: 0.4696121758298996
INFO:root:val auc: 0.9782346543861894, new node val auc: 0.93634020991667
INFO:root:val ap: 0.978809639955437, new node val ap: 0.9390815218886828
```

after

```
INFO:root:start 0 epoch
INFO:root:epoch: 0 took 180.36s
INFO:root:Epoch mean loss: 0.4696121758298996
INFO:root:val auc: 0.9782346543861894, new node val auc: 0.9277168485494387
INFO:root:val ap: 0.978809639955437, new node val ap: 0.9296338834432695
```

3 Research Date: *[2021-10-15 Fri]*

- ref
  - https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F

3.1 Error Log 1: memory is incorrectly updated when runing `sliding_window_evaluation` with `memory_update_at_end` flag == false

3.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I run with the program with the following command.

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
at this point in time, I haven't refactor sliding_window_evaluation into a function ye

The error below show that updating memory is not working correctly
#+BEGIN_SRC markdown
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 57.62s
INFO:root:Epoch mean loss: 0.5761979354879795
INFO:root:val auc: 0.974073302790504
INFO:root:val ap: 0.9742595324036754
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 58.29s
INFO:root:Epoch mean loss: 0.5718959999985029
INFO:root:val auc: 0.972725
INFO:root:val ap: 0.9704519025842379
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 58.77s
INFO:root:Epoch mean loss: 0.5814582438725034
INFO:root:val auc: 0.983
INFO:root:val ap: 0.9833502508908536
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 61.49s
INFO:root:Epoch mean loss: 0.5955527337650991
```

```
INFO:root:val auc: 0.9725874999999999
INFO:root:val ap: 0.9752678302020195
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 64.80s
INFO:root:Epoch mean loss: 0.5719773534077163
INFO:root:val auc: 0.9775125
INFO:root:val ap: 0.9775497565693843
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 64.64s
INFO:root:Epoch mean loss: 0.5794334606000572
INFO:root:val auc: 0.984
INFO:root:val ap: 0.9843943041643091
DEBUG:root:-ws = 6
DEBUG:root:--epoch = 0
Traceback (most recent call last):
  File "train_self_supervised.py", line 447, in <module>
    pos_prob, neg_prob = tgn.compute_edge_probabilities(sources_batch, destinations_ba
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\model\tgn.py", line 210, in comp
    source_node_embedding, destination_node_embedding, negative_node_embedding = self.
  File "C:\Users\terng\OneDrive\Documents\Working\tgn\model\tgn.py", line 166, in comp
    assert torch.allclose(memory[positives], self.memory.get_memory(positives), atol=1
AssertionError: Something wrong in how the memory was updated
```

3.2 Error Log 2: EarlyStopping is incorrectly implemented for `sliding_window_evaluation`

3.2.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I run with the program with the following command

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

I detected error that earlystopping shouldn't be call because earlystopping should only be used when number of epoch are greater than 1.

```
INFO:root:Namespace(aggregator='last', backprop_every=1, bs=200, data='reddit', differ
The dataset has 672447 interactions, involving 10984 different nodes
The training dataset has 389989 interactions, involving 9574 different nodes
The validation dataset has 100867 interactions, involving 9839 different nodes
The test dataset has 100867 interactions, involving 9615 different nodes
The new node validation dataset has 19446 interactions, involving 3491 different nodes
The new node test dataset has 21470 interactions, involving 3515 different nodes
1098 nodes were used for the inductive testing, i.e. are never seen during training
INFO:root:num of training instances: 389989
INFO:root:num of batches per epoch: 1950
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.37s
INFO:root:Epoch mean loss: 0.6701792616492662
INFO:root:val auc: 0.9560339441899208
INFO:root:val ap: 0.9621079926903555
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.27s
INFO:root:Epoch mean loss: 0.6677183579849068
INFO:root:val auc: 0.955675
INFO:root:val ap: 0.9545854824876753
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.75s
INFO:root:Epoch mean loss: 0.6809701967955856
INFO:root:val auc: 0.956825
INFO:root:val ap: 0.9587151165945682
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.55s
INFO:root:Epoch mean loss: 0.6915415738582003
INFO:root:val auc: 0.9525375
INFO:root:val ap: 0.9545045599044651
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.35s
INFO:root:Epoch mean loss: 0.6720186233976666
INFO:root:val auc: 0.9542875
```

```
INFO:root:val ap: 0.9591196464551901
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.40s
INFO:root:Epoch mean loss: 0.6741445491268376
INFO:root:val auc: 0.963675
INFO:root:val ap: 0.9609251681134591
INFO:root:No improvement over 5 epochs, stop training
INFO:root:Loading the best model at epoch 0
INFO:root:Loaded the best model at epoch 0 for inference
```

3.3 Research Log 1: Evaluate tgn with sliding window see no obvious improvement overtime.

3.3.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

3.3.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

No improvement is easily observed from evaluating with sliding window

```
INFO:root:Namespace(aggregator='last', backprop_every=1, bs=200, data='reddit', differ
The dataset has 672447 interactions, involving 10984 different nodes
The training dataset has 389989 interactions, involving 9574 different nodes
The validation dataset has 100867 interactions, involving 9839 different nodes
The test dataset has 100867 interactions, involving 9615 different nodes
The new node validation dataset has 19446 interactions, involving 3491 different nodes
```

```
The new node test dataset has 21470 interactions, involving 3515 different nodes
1098 nodes were used for the inductive testing, i.e. are never seen during training
INFO:root:num of training instances: 389989
INFO:root:num of batches per epoch: 1950
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 18.97s
INFO:root:Epoch mean loss: 0.6701792616492662
INFO:root:val auc: 0.9560339441899208
INFO:root:val ap: 0.9621079926903555
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.63s
INFO:root:Epoch mean loss: 0.6677183579849068
INFO:root:val auc: 0.955675
INFO:root:val ap: 0.9545854824876753
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.51s
INFO:root:Epoch mean loss: 0.6809701967955856
INFO:root:val auc: 0.956825
INFO:root:val ap: 0.9587151165945682
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.63s
INFO:root:Epoch mean loss: 0.6915415738582003
INFO:root:val auc: 0.9525375
INFO:root:val ap: 0.9545045599044651
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.62s
INFO:root:Epoch mean loss: 0.6720186233976666
INFO:root:val auc: 0.9542875
INFO:root:val ap: 0.9591196464551901
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.41s
INFO:root:Epoch mean loss: 0.6741445491268376
INFO:root:val auc: 0.963675
```

```
INFO:root:val ap: 0.9609251681134591
DEBUG:root:-ws = 6
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.75s
INFO:root:Epoch mean loss: 0.663450689412862
INFO:root:val auc: 0.963975
INFO:root:val ap: 0.9644888677043832
DEBUG:root:-ws = 7
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.82s
INFO:root:Epoch mean loss: 0.6593993042596867
INFO:root:val auc: 0.961275
INFO:root:val ap: 0.9589147598135823
DEBUG:root:-ws = 8
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.21s
INFO:root:Epoch mean loss: 0.6775721033590699
INFO:root:val auc: 0.9550249999999999
INFO:root:val ap: 0.9567142759053265
DEBUG:root:-ws = 9
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 16.90s
INFO:root:Epoch mean loss: 0.6900044053377459
INFO:root:val auc: 0.9646
INFO:root:val ap: 0.9698732681887545
DEBUG:root:-ws = 10
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.52s
INFO:root:Epoch mean loss: 0.6587084134168263
INFO:root:val auc: 0.9661375
INFO:root:val ap: 0.9691376413033301
DEBUG:root:-ws = 11
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.56s
INFO:root:Epoch mean loss: 0.6933906199281044
INFO:root:val auc: 0.9513125
INFO:root:val ap: 0.9530318558394366
DEBUG:root:-ws = 12
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.16s
```

```
INFO:root:Epoch mean loss: 0.6702749680238541
INFO:root:val auc: 0.9615125
INFO:root:val ap: 0.963401992827749
DEBUG:root:-ws = 13
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.17s
INFO:root:Epoch mean loss: 0.6783014771165643
INFO:root:val auc: 0.9407999999999999
INFO:root:val ap: 0.9437721917021403
DEBUG:root:-ws = 14
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.08s
INFO:root:Epoch mean loss: 0.6738769938648498
INFO:root:val auc: 0.94555
INFO:root:val ap: 0.9511214821192583
DEBUG:root:-ws = 15
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 17.09s
INFO:root:Epoch mean loss: 0.6751367596335381
INFO:root:val auc: 0.9475749999999998
INFO:root:val ap: 0.9529235421313578
```

3.3.3 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

I suspect that the way memory is updated/cleared/stored are not implemented correctly for $\text{sliding}_{\text{window evaluation}}$.

It is also possible that (hyper-)parameters must be adjusted to clearly show model performance over time.

4 Research Date: *[2021-10-18 Mon]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

4.1 =IMPLEMENTATION MISTAKE= Research Log 1: tgn performance improve epoch wise, but it may improve window sliding wise. I have to plot it for longer doesn't improve window sliding wise.

4.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I modified tgn code to start with 1 percent data and epoch = 5 and 50.

4.1.2 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

It is clear that tgn performance improve epoch wise, but I still need to plot it visually to see if performance improve window sliding wise and perform longer runs. However performance still plateau at 97 percent.

at epoch = 5

```
INFO:root:Namespace(aggregator='last', backprop_every=1, bs=200, data='reddit', differ
The dataset has 672447 interactions, involving 10984 different nodes
The training dataset has 389989 interactions, involving 9574 different nodes
The validation dataset has 100867 interactions, involving 9839 different nodes
The test dataset has 100867 interactions, involving 9615 different nodes
The new node validation dataset has 19446 interactions, involving 3491 different nodes
The new node test dataset has 21470 interactions, involving 3515 different nodes
```

```
1098 nodes were used for the inductive testing, i.e. are never seen during training
INFO:root:num of training instances: 389989
INFO:root:num of batches per epoch: 1950
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 1.09s
INFO:root:Epoch mean loss: 1.1142270684242248
INFO:root:val auc: 0.8646999999999999
INFO:root:val ap: 0.8677555140505844
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.48s
INFO:root:Epoch mean loss: 0.9375462263822556
INFO:root:val auc: 0.8795
INFO:root:val ap: 0.8931062365069626
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.45s
INFO:root:Epoch mean loss: 0.8739588379859924
INFO:root:val auc: 0.8984000000000001
INFO:root:val ap: 0.9099307766474001
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.45s
INFO:root:Epoch mean loss: 0.8326897919178009
INFO:root:val auc: 0.9001000000000001
INFO:root:val ap: 0.9120455157804328
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.44s
INFO:root:Epoch mean loss: 0.8357435047626496
INFO:root:val auc: 0.9203
INFO:root:val ap: 0.9247411671031869
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.52s
INFO:root:Epoch mean loss: 1.1422362157276698
INFO:root:val auc: 0.89575
INFO:root:val ap: 0.9074792811869621
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.50s
INFO:root:Epoch mean loss: 0.9477060777800423
INFO:root:val auc: 0.8992500000000001
```

```
INFO:root:val ap: 0.9164590722911732
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.49s
INFO:root:Epoch mean loss: 0.8796056168419975
INFO:root:val auc: 0.910125
INFO:root:val ap: 0.925225386907099
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.51s
INFO:root:Epoch mean loss: 0.8459391253335136
INFO:root:val auc: 0.928425
INFO:root:val ap: 0.9373177663544129
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.49s
INFO:root:Epoch mean loss: 0.8262992245810372
INFO:root:val auc: 0.9253250000000001
INFO:root:val ap: 0.9357822473937303
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.55s
INFO:root:Epoch mean loss: 1.0918263494968414
INFO:root:val auc: 0.875275
INFO:root:val ap: 0.9029663734414153
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.55s
INFO:root:Epoch mean loss: 0.8991197916594419
INFO:root:val auc: 0.905125
INFO:root:val ap: 0.9224104242714677
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.51s
INFO:root:Epoch mean loss: 0.8507793627002023
INFO:root:val auc: 0.9258
INFO:root:val ap: 0.9331773488925642
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.51s
INFO:root:Epoch mean loss: 0.832731076262214
INFO:root:val auc: 0.929375
INFO:root:val ap: 0.9374399999135906
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.52s
INFO:root:Epoch mean loss: 0.819046901030974
```

```
INFO:root:val auc: 0.929375
INFO:root:val ap: 0.9378003211554748
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.55s
INFO:root:Epoch mean loss: 1.13708366000134
INFO:root:val auc: 0.889
INFO:root:val ap: 0.9065481622966958
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.53s
INFO:root:Epoch mean loss: 0.9459260209746982
INFO:root:val auc: 0.903375
INFO:root:val ap: 0.9193141365355181
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.52s
INFO:root:Epoch mean loss: 0.875484269598256
INFO:root:val auc: 0.9101500000000001
INFO:root:val ap: 0.9212875826823976
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.52s
INFO:root:Epoch mean loss: 0.8401498950046041
INFO:root:val auc: 0.9136249999999999
INFO:root:val ap: 0.9248470812624878
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.54s
INFO:root:Epoch mean loss: 0.8349218809086344
INFO:root:val auc: 0.9213
INFO:root:val ap: 0.9325655600194183
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.61s
INFO:root:Epoch mean loss: 1.0927202130357425
INFO:root:val auc: 0.8864249999999999
INFO:root:val ap: 0.9025927672076126
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.55s
INFO:root:Epoch mean loss: 0.9124124969045321
INFO:root:val auc: 0.9043
INFO:root:val ap: 0.9172227198599441
DEBUG:root:--epoch = 2
```

```
INFO:root:epoch: 2 took 0.55s
INFO:root:Epoch mean loss: 0.8663089250524839
INFO:root:val auc: 0.916225
INFO:root:val ap: 0.9268765327194748
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.55s
INFO:root:Epoch mean loss: 0.8144349455833435
INFO:root:val auc: 0.9303750000000001
INFO:root:val ap: 0.9380847764492686
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.54s
INFO:root:Epoch mean loss: 0.7972120394309362
INFO:root:val auc: 0.9368249999999999
INFO:root:val ap: 0.9388699560507461
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.62s
INFO:root:Epoch mean loss: 1.0684358429908754
INFO:root:val auc: 0.8601999999999999
INFO:root:val ap: 0.8838599463136125
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.57s
INFO:root:Epoch mean loss: 0.8886247181892395
INFO:root:val auc: 0.888025
INFO:root:val ap: 0.9038433470441051
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.57s
INFO:root:Epoch mean loss: 0.8564816427230835
INFO:root:val auc: 0.8952999999999999
INFO:root:val ap: 0.9068050643905683
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.65s
INFO:root:Epoch mean loss: 0.8300222730636597
INFO:root:val auc: 0.8997249999999999
INFO:root:val ap: 0.9105965323086614
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.68s
INFO:root:Epoch mean loss: 0.8021341323852539
INFO:root:val auc: 0.907325
INFO:root:val ap: 0.9171881357353258
```

```
DEBUG:root:-ws = 6
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.66s
INFO:root:Epoch mean loss: 1.048623814032628
INFO:root:val auc: 0.911875
INFO:root:val ap: 0.9284353294416356
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.68s
INFO:root:Epoch mean loss: 0.9013866025667924
INFO:root:val auc: 0.92305
INFO:root:val ap: 0.937064472519639
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.62s
INFO:root:Epoch mean loss: 0.857620528111091
INFO:root:val auc: 0.9320999999999999
INFO:root:val ap: 0.9419461947847318
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.61s
INFO:root:Epoch mean loss: 0.8112900646833273
INFO:root:val auc: 0.9333
INFO:root:val ap: 0.9434705665362639
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.59s
INFO:root:Epoch mean loss: 0.8136325478553772
INFO:root:val auc: 0.935075
INFO:root:val ap: 0.9454940454291834
DEBUG:root:-ws = 7
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.64s
INFO:root:Epoch mean loss: 1.0794085286281727
INFO:root:val auc: 0.9096749999999999
INFO:root:val ap: 0.9271424050127471
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.61s
INFO:root:Epoch mean loss: 0.9020434551768832
INFO:root:val auc: 0.9315249999999999
INFO:root:val ap: 0.9421140379517917
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.61s
INFO:root:Epoch mean loss: 0.8332129893479524
```

```
INFO:root:val auc: 0.943425
INFO:root:val ap: 0.9512939486005327
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.62s
INFO:root:Epoch mean loss: 0.8195670445760092
INFO:root:val auc: 0.9465
INFO:root:val ap: 0.9535169603235449
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.62s
INFO:root:Epoch mean loss: 0.8031924344875194
INFO:root:val auc: 0.951025
INFO:root:val ap: 0.9547285364673678
DEBUG:root:-ws = 8
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.66s
INFO:root:Epoch mean loss: 1.0891311998878206
INFO:root:val auc: 0.9124
INFO:root:val ap: 0.9237784923484818
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.67s
INFO:root:Epoch mean loss: 0.8951979598828724
INFO:root:val auc: 0.9291999999999999
INFO:root:val ap: 0.9377181281618658
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.63s
INFO:root:Epoch mean loss: 0.8392845839262009
INFO:root:val auc: 0.9383
INFO:root:val ap: 0.9455613033105866
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.65s
INFO:root:Epoch mean loss: 0.8116127316440854
INFO:root:val auc: 0.9391499999999999
INFO:root:val ap: 0.9477476122654526
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.64s
INFO:root:Epoch mean loss: 0.7837122955492565
INFO:root:val auc: 0.943525
INFO:root:val ap: 0.9531621426751444
DEBUG:root:-ws = 9
DEBUG:root:--epoch = 0
```

```
INFO:root:epoch: 0 took 0.67s
INFO:root:Epoch mean loss: 1.0880023754876236
INFO:root:val auc: 0.877225
INFO:root:val ap: 0.8941089308742844
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.71s
INFO:root:Epoch mean loss: 0.9138705381031694
INFO:root:val auc: 0.8910249999999998
INFO:root:val ap: 0.905096340758637
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.67s
INFO:root:Epoch mean loss: 0.8378391882468914
INFO:root:val auc: 0.91445
INFO:root:val ap: 0.9212813088227907
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.64s
INFO:root:Epoch mean loss: 0.8168713244898566
INFO:root:val auc: 0.9191999999999999
INFO:root:val ap: 0.9244153842637846
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.70s
INFO:root:Epoch mean loss: 0.8000208344952814
INFO:root:val auc: 0.9188500000000001
INFO:root:val ap: 0.9240888176088151
DEBUG:root:-ws = 10
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.75s
INFO:root:Epoch mean loss: 1.0396650652090709
INFO:root:val auc: 0.8778
INFO:root:val ap: 0.8924838880317147
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.70s
INFO:root:Epoch mean loss: 0.870083232720693
INFO:root:val auc: 0.8987749999999999
INFO:root:val ap: 0.9063776163159412
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.68s
INFO:root:Epoch mean loss: 0.8177959104379018
INFO:root:val auc: 0.91445
INFO:root:val ap: 0.9230616843538288
```

```
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.66s
INFO:root:Epoch mean loss: 0.8089526017506917
INFO:root:val auc: 0.9195249999999999
INFO:root:val ap: 0.92795328050554
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.66s
INFO:root:Epoch mean loss: 0.7833456019560496
INFO:root:val auc: 0.918725
INFO:root:val ap: 0.9265936018534029
```

epoch = 50

```
INFO:root:Namespace(aggregator='last', backprop_every=1, bs=200, data='reddit', differ
The dataset has 672447 interactions, involving 10984 different nodes
The training dataset has 389989 interactions, involving 9574 different nodes
The validation dataset has 100867 interactions, involving 9839 different nodes
The test dataset has 100867 interactions, involving 9615 different nodes
The new node validation dataset has 19446 interactions, involving 3491 different nodes
The new node test dataset has 21470 interactions, involving 3515 different nodes
1098 nodes were used for the inductive testing, i.e. are never seen during training
INFO:root:num of training instances: 389989
INFO:root:num of batches per epoch: 1950
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 1.07s
INFO:root:Epoch mean loss: 1.1142270684242248
INFO:root:val auc: 0.8646999999999999
INFO:root:val ap: 0.8677555140505844
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.47s
INFO:root:Epoch mean loss: 0.9375462263822556
INFO:root:val auc: 0.8795
INFO:root:val ap: 0.8931062365069626
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.46s
INFO:root:Epoch mean loss: 0.8739588379859924
INFO:root:val auc: 0.8984000000000001
```

```
INFO:root:val ap: 0.9099307766474001
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.46s
INFO:root:Epoch mean loss: 0.8326897919178009
INFO:root:val auc: 0.9001000000000001
INFO:root:val ap: 0.9120455157804328
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.46s
INFO:root:Epoch mean loss: 0.8357435047626496
INFO:root:val auc: 0.9203
INFO:root:val ap: 0.9247411671031869
DEBUG:root:--epoch = 5
INFO:root:epoch: 5 took 0.45s
INFO:root:Epoch mean loss: 0.8182762324810028
INFO:root:val auc: 0.9256000000000001
INFO:root:val ap: 0.9276109461268157
DEBUG:root:--epoch = 6
INFO:root:epoch: 6 took 0.49s
INFO:root:Epoch mean loss: 0.8065412074327469
INFO:root:val auc: 0.9231
INFO:root:val ap: 0.9284842607684816
DEBUG:root:--epoch = 7
INFO:root:epoch: 7 took 0.46s
INFO:root:Epoch mean loss: 0.789261183142662
INFO:root:val auc: 0.9308
INFO:root:val ap: 0.9279925873062627
DEBUG:root:--epoch = 8
INFO:root:epoch: 8 took 0.46s
INFO:root:Epoch mean loss: 0.7801310509443283
INFO:root:val auc: 0.9327
INFO:root:val ap: 0.9325033356793793
DEBUG:root:--epoch = 9
INFO:root:epoch: 9 took 0.45s
INFO:root:Epoch mean loss: 0.7792910724878311
INFO:root:val auc: 0.9329999999999999
INFO:root:val ap: 0.9310856613186225
DEBUG:root:--epoch = 10
INFO:root:epoch: 10 took 0.45s
INFO:root:Epoch mean loss: 0.7806094139814377
INFO:root:val auc: 0.9299000000000001
```

```
INFO:root:val ap: 0.9317939126198809
DEBUG:root:--epoch = 11
INFO:root:epoch: 11 took 0.45s
INFO:root:Epoch mean loss: 0.7667293578386307
INFO:root:val auc: 0.9361
INFO:root:val ap: 0.9376028801803121
DEBUG:root:--epoch = 12
INFO:root:epoch: 12 took 0.45s
INFO:root:Epoch mean loss: 0.7507588684558868
INFO:root:val auc: 0.937
INFO:root:val ap: 0.937203709000963
DEBUG:root:--epoch = 13
INFO:root:epoch: 13 took 0.46s
INFO:root:Epoch mean loss: 0.7538762390613556
INFO:root:val auc: 0.9392
INFO:root:val ap: 0.9390106961714739
DEBUG:root:--epoch = 14
INFO:root:epoch: 14 took 0.45s
INFO:root:Epoch mean loss: 0.7447947978973388
INFO:root:val auc: 0.9403
INFO:root:val ap: 0.9393560776940532
DEBUG:root:--epoch = 15
INFO:root:epoch: 15 took 0.48s
INFO:root:Epoch mean loss: 0.7376965999603271
INFO:root:val auc: 0.9491
INFO:root:val ap: 0.947905135810321
DEBUG:root:--epoch = 16
INFO:root:epoch: 16 took 0.45s
INFO:root:Epoch mean loss: 0.7306154996156693
INFO:root:val auc: 0.9549
INFO:root:val ap: 0.9514620724739241
DEBUG:root:--epoch = 17
INFO:root:epoch: 17 took 0.45s
INFO:root:Epoch mean loss: 0.7582407504320144
INFO:root:val auc: 0.9537
INFO:root:val ap: 0.9510895201575147
DEBUG:root:--epoch = 18
INFO:root:epoch: 18 took 0.45s
INFO:root:Epoch mean loss: 0.7337055385112763
INFO:root:val auc: 0.9517
```

```
INFO:root:val ap: 0.9483391033984477
DEBUG:root:--epoch = 19
INFO:root:epoch: 19 took 0.45s
INFO:root:Epoch mean loss: 0.723554077744484
INFO:root:val auc: 0.9481999999999999
INFO:root:val ap: 0.9486109538338079
DEBUG:root:--epoch = 20
INFO:root:epoch: 20 took 0.46s
INFO:root:Epoch mean loss: 0.7325401097536087
INFO:root:val auc: 0.9601000000000001
INFO:root:val ap: 0.958928485598702
DEBUG:root:--epoch = 21
INFO:root:epoch: 21 took 0.45s
INFO:root:Epoch mean loss: 0.7203178882598877
INFO:root:val auc: 0.9523
INFO:root:val ap: 0.9492341789766078
DEBUG:root:--epoch = 22
INFO:root:epoch: 22 took 0.46s
INFO:root:Epoch mean loss: 0.7329257875680923
INFO:root:val auc: 0.953
INFO:root:val ap: 0.9491897856155596
DEBUG:root:--epoch = 23
INFO:root:epoch: 23 took 0.46s
INFO:root:Epoch mean loss: 0.7138326376676559
INFO:root:val auc: 0.9542
INFO:root:val ap: 0.9522482062482092
DEBUG:root:--epoch = 24
INFO:root:epoch: 24 took 0.45s
INFO:root:Epoch mean loss: 0.7142266869544983
INFO:root:val auc: 0.9541999999999999
INFO:root:val ap: 0.9530705556034346
DEBUG:root:--epoch = 25
INFO:root:epoch: 25 took 0.48s
INFO:root:Epoch mean loss: 0.7080656498670578
INFO:root:val auc: 0.9532
INFO:root:val ap: 0.9492066298481284
DEBUG:root:--epoch = 26
INFO:root:epoch: 26 took 0.45s
INFO:root:Epoch mean loss: 0.7016981899738312
INFO:root:val auc: 0.9571
```

```
INFO:root:val ap: 0.9528441313285743
DEBUG:root:--epoch = 27
INFO:root:epoch: 27 took 0.45s
INFO:root:Epoch mean loss: 0.7047642260789871
INFO:root:val auc: 0.95
INFO:root:val ap: 0.9488657456078735
DEBUG:root:--epoch = 28
INFO:root:epoch: 28 took 0.44s
INFO:root:Epoch mean loss: 0.6986190736293793
INFO:root:val auc: 0.9542
INFO:root:val ap: 0.9530986584496893
DEBUG:root:--epoch = 29
INFO:root:epoch: 29 took 0.44s
INFO:root:Epoch mean loss: 0.6806379020214081
INFO:root:val auc: 0.9524
INFO:root:val ap: 0.9455160616159393
DEBUG:root:--epoch = 30
INFO:root:epoch: 30 took 0.44s
INFO:root:Epoch mean loss: 0.7036882609128952
INFO:root:val auc: 0.9630000000000001
INFO:root:val ap: 0.9585163050828558
DEBUG:root:--epoch = 31
INFO:root:epoch: 31 took 0.44s
INFO:root:Epoch mean loss: 0.6920652687549591
INFO:root:val auc: 0.9564
INFO:root:val ap: 0.9479363942880463
DEBUG:root:--epoch = 32
INFO:root:epoch: 32 took 0.45s
INFO:root:Epoch mean loss: 0.6823025643825531
INFO:root:val auc: 0.9553
INFO:root:val ap: 0.9461869966256359
DEBUG:root:--epoch = 33
INFO:root:epoch: 33 took 0.45s
INFO:root:Epoch mean loss: 0.6692858219146729
INFO:root:val auc: 0.9561
INFO:root:val ap: 0.9457629758723517
DEBUG:root:--epoch = 34
INFO:root:epoch: 34 took 0.44s
INFO:root:Epoch mean loss: 0.6881103754043579
INFO:root:val auc: 0.9572
```

```
INFO:root:val ap: 0.9417181003246693
DEBUG:root:--epoch = 35
INFO:root:epoch: 35 took 0.46s
INFO:root:Epoch mean loss: 0.686672231554985
INFO:root:val auc: 0.9510000000000001
INFO:root:val ap: 0.938765606758409
DEBUG:root:--epoch = 36
INFO:root:epoch: 36 took 0.44s
INFO:root:Epoch mean loss: 0.6775761723518372
INFO:root:val auc: 0.9619000000000001
INFO:root:val ap: 0.9517109882872032
DEBUG:root:--epoch = 37
INFO:root:epoch: 37 took 0.45s
INFO:root:Epoch mean loss: 0.6790220454335213
INFO:root:val auc: 0.9584
INFO:root:val ap: 0.952467188535179
DEBUG:root:--epoch = 38
INFO:root:epoch: 38 took 0.45s
INFO:root:Epoch mean loss: 0.668178790807724
INFO:root:val auc: 0.9518
INFO:root:val ap: 0.9476795610264501
DEBUG:root:--epoch = 39
INFO:root:epoch: 39 took 0.45s
INFO:root:Epoch mean loss: 0.6959656566381455
INFO:root:val auc: 0.9535
INFO:root:val ap: 0.9470627910776206
DEBUG:root:--epoch = 40
INFO:root:epoch: 40 took 0.44s
INFO:root:Epoch mean loss: 0.6766523122787476
INFO:root:val auc: 0.9602
INFO:root:val ap: 0.9521326099757211
DEBUG:root:--epoch = 41
INFO:root:epoch: 41 took 0.45s
INFO:root:Epoch mean loss: 0.677765354514122
INFO:root:val auc: 0.9541
INFO:root:val ap: 0.9479907691613434
DEBUG:root:--epoch = 42
INFO:root:epoch: 42 took 0.45s
INFO:root:Epoch mean loss: 0.6816614508628845
INFO:root:val auc: 0.9534
```

```
INFO:root:val ap: 0.9470424525818912
DEBUG:root:--epoch = 43
INFO:root:epoch: 43 took 0.45s
INFO:root:Epoch mean loss: 0.6675903767347335
INFO:root:val auc: 0.966
INFO:root:val ap: 0.9562516041728377
DEBUG:root:--epoch = 44
INFO:root:epoch: 44 took 0.46s
INFO:root:Epoch mean loss: 0.6558136612176895
INFO:root:val auc: 0.9655
INFO:root:val ap: 0.9572726285844014
DEBUG:root:--epoch = 45
INFO:root:epoch: 45 took 0.47s
INFO:root:Epoch mean loss: 0.6543817013502121
INFO:root:val auc: 0.9622999999999999
INFO:root:val ap: 0.9504215774898183
DEBUG:root:--epoch = 46
INFO:root:epoch: 46 took 0.45s
INFO:root:Epoch mean loss: 0.6623854905366897
INFO:root:val auc: 0.9598
INFO:root:val ap: 0.9474473863966614
DEBUG:root:--epoch = 47
INFO:root:epoch: 47 took 0.45s
INFO:root:Epoch mean loss: 0.6525968670845032
INFO:root:val auc: 0.9586
INFO:root:val ap: 0.9475192639215037
DEBUG:root:--epoch = 48
INFO:root:epoch: 48 took 0.44s
INFO:root:Epoch mean loss: 0.6549943253397942
INFO:root:val auc: 0.9618
INFO:root:val ap: 0.9504917868648144
DEBUG:root:--epoch = 49
INFO:root:epoch: 49 took 0.46s
INFO:root:Epoch mean loss: 0.6504715844988823
INFO:root:val auc: 0.9584
INFO:root:val ap: 0.9476373998274553
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.51s
INFO:root:Epoch mean loss: 1.1508782080241613
```

```
INFO:root:val auc: 0.89805
INFO:root:val ap: 0.9102506809800469
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.50s
INFO:root:Epoch mean loss: 0.9271293055443537
INFO:root:val auc: 0.8944749999999999
INFO:root:val ap: 0.9129634381703282
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.49s
INFO:root:Epoch mean loss: 0.8990285027594793
INFO:root:val auc: 0.9158
INFO:root:val ap: 0.9280526282549573
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.48s
INFO:root:Epoch mean loss: 0.8543144010362171
INFO:root:val auc: 0.9258249999999999
INFO:root:val ap: 0.9364461974370846
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.48s
INFO:root:Epoch mean loss: 0.8387996270543053
INFO:root:val auc: 0.9357999999999999
INFO:root:val ap: 0.9438122552554649
DEBUG:root:--epoch = 5
INFO:root:epoch: 5 took 0.48s
INFO:root:Epoch mean loss: 0.8265136877695719
INFO:root:val auc: 0.9347249999999999
INFO:root:val ap: 0.9424165798934188
DEBUG:root:--epoch = 6
INFO:root:epoch: 6 took 0.49s
INFO:root:Epoch mean loss: 0.8021257775170463
INFO:root:val auc: 0.93955
INFO:root:val ap: 0.9483712431423004
DEBUG:root:--epoch = 7
INFO:root:epoch: 7 took 0.50s
INFO:root:Epoch mean loss: 0.7824988081341698
INFO:root:val auc: 0.9402999999999999
INFO:root:val ap: 0.9470556337793561
DEBUG:root:--epoch = 8
INFO:root:epoch: 8 took 0.49s
INFO:root:Epoch mean loss: 0.7899840190297082
```

```
INFO:root:val auc: 0.9416500000000001
INFO:root:val ap: 0.9468837635169876
DEBUG:root:--epoch = 9
INFO:root:epoch: 9 took 0.46s
INFO:root:Epoch mean loss: 0.7718663016955057
INFO:root:val auc: 0.93705
INFO:root:val ap: 0.9452391411332342
DEBUG:root:--epoch = 10
INFO:root:epoch: 10 took 0.48s
INFO:root:Epoch mean loss: 0.7669964177267892
INFO:root:val auc: 0.9421250000000001
INFO:root:val ap: 0.9481251509988853
DEBUG:root:--epoch = 11
INFO:root:epoch: 11 took 0.48s
INFO:root:Epoch mean loss: 0.7541449382191613
INFO:root:val auc: 0.940175
INFO:root:val ap: 0.9450849724461144
DEBUG:root:--epoch = 12
INFO:root:epoch: 12 took 0.49s
INFO:root:Epoch mean loss: 0.7448542543819973
INFO:root:val auc: 0.9422499999999999
INFO:root:val ap: 0.9478082556861789
DEBUG:root:--epoch = 13
INFO:root:epoch: 13 took 0.47s
INFO:root:Epoch mean loss: 0.7490229663394746
INFO:root:val auc: 0.9431249999999999
INFO:root:val ap: 0.9486259086770463
DEBUG:root:--epoch = 14
INFO:root:epoch: 14 took 0.48s
INFO:root:Epoch mean loss: 0.7560982221648807
INFO:root:val auc: 0.9434
INFO:root:val ap: 0.9498383874499255
DEBUG:root:--epoch = 15
INFO:root:epoch: 15 took 0.47s
INFO:root:Epoch mean loss: 0.7372378281184605
INFO:root:val auc: 0.9429249999999999
INFO:root:val ap: 0.9453990874260293
DEBUG:root:--epoch = 16
INFO:root:epoch: 16 took 0.48s
INFO:root:Epoch mean loss: 0.7436673726354327
```

```
INFO:root:val auc: 0.94075
INFO:root:val ap: 0.9434589999982885
DEBUG:root:--epoch = 17
INFO:root:epoch: 17 took 0.47s
INFO:root:Epoch mean loss: 0.7540551934923444
INFO:root:val auc: 0.9401
INFO:root:val ap: 0.9456721538116089
DEBUG:root:--epoch = 18
INFO:root:epoch: 18 took 0.47s
INFO:root:Epoch mean loss: 0.7335320909818014
INFO:root:val auc: 0.9472999999999999
INFO:root:val ap: 0.9511195681347252
DEBUG:root:--epoch = 19
INFO:root:epoch: 19 took 0.48s
INFO:root:Epoch mean loss: 0.7360767523447672
INFO:root:val auc: 0.938775
INFO:root:val ap: 0.9446987319483475
DEBUG:root:--epoch = 20
INFO:root:epoch: 20 took 0.48s
INFO:root:Epoch mean loss: 0.741033000605447
INFO:root:val auc: 0.9392
INFO:root:val ap: 0.9436939835504585
DEBUG:root:--epoch = 21
INFO:root:epoch: 21 took 0.48s
INFO:root:Epoch mean loss: 0.7079271333558219
INFO:root:val auc: 0.946125
INFO:root:val ap: 0.9470291041650883
DEBUG:root:--epoch = 22
INFO:root:epoch: 22 took 0.48s
INFO:root:Epoch mean loss: 0.6974603562127977
INFO:root:val auc: 0.9441499999999999
INFO:root:val ap: 0.9450807942360793
DEBUG:root:--epoch = 23
INFO:root:epoch: 23 took 0.49s
INFO:root:Epoch mean loss: 0.701942001070295
INFO:root:val auc: 0.943325
INFO:root:val ap: 0.9445960880195303
DEBUG:root:--epoch = 24
INFO:root:epoch: 24 took 0.51s
INFO:root:Epoch mean loss: 0.7146192391713461
```

```
INFO:root:val auc: 0.9461
INFO:root:val ap: 0.9477820278457252
DEBUG:root:--epoch = 25
INFO:root:epoch: 25 took 0.48s
INFO:root:Epoch mean loss: 0.7214535616693043
INFO:root:val auc: 0.9526249999999999
INFO:root:val ap: 0.9541278906669513
DEBUG:root:--epoch = 26
INFO:root:epoch: 26 took 0.50s
INFO:root:Epoch mean loss: 0.6939295274870736
INFO:root:val auc: 0.9496749999999999
INFO:root:val ap: 0.9518017280739952
DEBUG:root:--epoch = 27
INFO:root:epoch: 27 took 0.47s
INFO:root:Epoch mean loss: 0.7124776726677304
INFO:root:val auc: 0.95115
INFO:root:val ap: 0.9514711991243
DEBUG:root:--epoch = 28
INFO:root:epoch: 28 took 0.49s
INFO:root:Epoch mean loss: 0.701434782573155
INFO:root:val auc: 0.9507
INFO:root:val ap: 0.9517499619800665
DEBUG:root:--epoch = 29
INFO:root:epoch: 29 took 0.47s
INFO:root:Epoch mean loss: 0.7131512675966535
INFO:root:val auc: 0.9491
INFO:root:val ap: 0.9478697669825008
DEBUG:root:--epoch = 30
INFO:root:epoch: 30 took 0.48s
INFO:root:Epoch mean loss: 0.6965945448194232
INFO:root:val auc: 0.942075
INFO:root:val ap: 0.944027457273495
DEBUG:root:--epoch = 31
INFO:root:epoch: 31 took 0.48s
INFO:root:Epoch mean loss: 0.6902143926847548
INFO:root:val auc: 0.9447000000000001
INFO:root:val ap: 0.9480609060382532
DEBUG:root:--epoch = 32
INFO:root:epoch: 32 took 0.52s
INFO:root:Epoch mean loss: 0.699785209837414
```

```
INFO:root:val auc: 0.941425
INFO:root:val ap: 0.9417607644244308
DEBUG:root:--epoch = 33
INFO:root:epoch: 33 took 0.49s
INFO:root:Epoch mean loss: 0.6934038741247994
INFO:root:val auc: 0.9492250000000001
INFO:root:val ap: 0.9495362948277792
DEBUG:root:--epoch = 34
INFO:root:epoch: 34 took 0.49s
INFO:root:Epoch mean loss: 0.7012697827248346
INFO:root:val auc: 0.949175
INFO:root:val ap: 0.9499228532841639
DEBUG:root:--epoch = 35
INFO:root:epoch: 35 took 0.48s
INFO:root:Epoch mean loss: 0.6793368345215207
INFO:root:val auc: 0.953675
INFO:root:val ap: 0.9569011912838941
DEBUG:root:--epoch = 36
INFO:root:epoch: 36 took 0.50s
INFO:root:Epoch mean loss: 0.6887028926894778
INFO:root:val auc: 0.95425
INFO:root:val ap: 0.954754719998281
DEBUG:root:--epoch = 37
INFO:root:epoch: 37 took 0.53s
INFO:root:Epoch mean loss: 0.6888956541106814
INFO:root:val auc: 0.9511000000000001
INFO:root:val ap: 0.9526303269825549
DEBUG:root:--epoch = 38
INFO:root:epoch: 38 took 0.49s
INFO:root:Epoch mean loss: 0.680636051155272
INFO:root:val auc: 0.955425
INFO:root:val ap: 0.954809934635066
DEBUG:root:--epoch = 39
INFO:root:epoch: 39 took 0.50s
INFO:root:Epoch mean loss: 0.6739710285550072
INFO:root:val auc: 0.954575
INFO:root:val ap: 0.9527532643971751
DEBUG:root:--epoch = 40
INFO:root:epoch: 40 took 0.49s
INFO:root:Epoch mean loss: 0.6767874899364653
```

```
INFO:root:val auc: 0.950075
INFO:root:val ap: 0.95056709109543
DEBUG:root:--epoch = 41
INFO:root:epoch: 41 took 0.48s
INFO:root:Epoch mean loss: 0.6565616726875305
INFO:root:val auc: 0.9563250000000001
INFO:root:val ap: 0.9563241172620255
DEBUG:root:--epoch = 42
INFO:root:epoch: 42 took 0.47s
INFO:root:Epoch mean loss: 0.6654709691093081
INFO:root:val auc: 0.952275
INFO:root:val ap: 0.9538597104427355
DEBUG:root:--epoch = 43
INFO:root:epoch: 43 took 0.48s
INFO:root:Epoch mean loss: 0.6604141820044744
INFO:root:val auc: 0.946125
INFO:root:val ap: 0.9428192106912625
DEBUG:root:--epoch = 44
INFO:root:epoch: 44 took 0.47s
INFO:root:Epoch mean loss: 0.668881200608753
INFO:root:val auc: 0.9470000000000001
INFO:root:val ap: 0.9503942037049224
DEBUG:root:--epoch = 45
INFO:root:epoch: 45 took 0.47s
INFO:root:Epoch mean loss: 0.66869018191383
INFO:root:val auc: 0.9595250000000001
INFO:root:val ap: 0.9606817593517398
DEBUG:root:--epoch = 46
INFO:root:epoch: 46 took 0.47s
INFO:root:Epoch mean loss: 0.6629092494646708
INFO:root:val auc: 0.9534999999999999
INFO:root:val ap: 0.9522238924792052
DEBUG:root:--epoch = 47
INFO:root:epoch: 47 took 0.47s
INFO:root:Epoch mean loss: 0.6626323489915757
INFO:root:val auc: 0.9481999999999999
INFO:root:val ap: 0.9499731961686495
DEBUG:root:--epoch = 48
INFO:root:epoch: 48 took 0.48s
INFO:root:Epoch mean loss: 0.6583377804074969
```

```
INFO:root:val auc: 0.9521
INFO:root:val ap: 0.9504556830660533
DEBUG:root:--epoch = 49
INFO:root:epoch: 49 took 0.48s
INFO:root:Epoch mean loss: 0.6641509646461123
INFO:root:val auc: 0.953275
INFO:root:val ap: 0.9557377514693728
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:epoch: 0 took 0.53s
INFO:root:Epoch mean loss: 1.0943367643789812
INFO:root:val auc: 0.874925
INFO:root:val ap: 0.8808133528304708
DEBUG:root:--epoch = 1
INFO:root:epoch: 1 took 0.55s
INFO:root:Epoch mean loss: 0.9028595875610005
INFO:root:val auc: 0.911425
INFO:root:val ap: 0.9241305843618483
DEBUG:root:--epoch = 2
INFO:root:epoch: 2 took 0.51s
INFO:root:Epoch mean loss: 0.8623342974619432
INFO:root:val auc: 0.9283750000000001
INFO:root:val ap: 0.9340680314436075
DEBUG:root:--epoch = 3
INFO:root:epoch: 3 took 0.50s
INFO:root:Epoch mean loss: 0.8304535421458158
INFO:root:val auc: 0.926625
INFO:root:val ap: 0.9356712826837081
DEBUG:root:--epoch = 4
INFO:root:epoch: 4 took 0.50s
INFO:root:Epoch mean loss: 0.8112695623527874
INFO:root:val auc: 0.9323
INFO:root:val ap: 0.932007241794164
DEBUG:root:--epoch = 5
INFO:root:epoch: 5 took 0.50s
INFO:root:Epoch mean loss: 0.8120430897582661
INFO:root:val auc: 0.9296500000000001
INFO:root:val ap: 0.9354729928980177
DEBUG:root:--epoch = 6
INFO:root:epoch: 6 took 0.49s
```

```
INFO:root:Epoch mean loss: 0.7919468771327626
INFO:root:val auc: 0.93735
INFO:root:val ap: 0.940208791816891
DEBUG:root:--epoch = 7
INFO:root:epoch: 7 took 0.49s
INFO:root:Epoch mean loss: 0.7739886478944258
INFO:root:val auc: 0.9387
INFO:root:val ap: 0.9417045589157607
DEBUG:root:--epoch = 8
INFO:root:epoch: 8 took 0.49s
INFO:root:Epoch mean loss: 0.7877854948694055
INFO:root:val auc: 0.941825
INFO:root:val ap: 0.945530142706554
DEBUG:root:--epoch = 9
INFO:root:epoch: 9 took 0.51s
INFO:root:Epoch mean loss: 0.7799345011060889
INFO:root:val auc: 0.94595
INFO:root:val ap: 0.9488234307517888
DEBUG:root:--epoch = 10
INFO:root:epoch: 10 took 0.51s
INFO:root:Epoch mean loss: 0.776393779299476
INFO:root:val auc: 0.944425
INFO:root:val ap: 0.9454490808941172
DEBUG:root:--epoch = 11
INFO:root:epoch: 11 took 0.53s
INFO:root:Epoch mean loss: 0.7681501751596277
INFO:root:val auc: 0.943575
INFO:root:val ap: 0.9438913812636096
DEBUG:root:--epoch = 12
INFO:root:epoch: 12 took 0.50s
INFO:root:Epoch mean loss: 0.7605301277203993
INFO:root:val auc: 0.944125
INFO:root:val ap: 0.9456689414760246
DEBUG:root:--epoch = 13
INFO:root:epoch: 13 took 0.53s
INFO:root:Epoch mean loss: 0.7555187452923168
INFO:root:val auc: 0.943425
INFO:root:val ap: 0.9459265427838232
DEBUG:root:--epoch = 14
INFO:root:epoch: 14 took 0.51s
```

```
INFO:root:Epoch mean loss: 0.765173920176246
INFO:root:val auc: 0.948925
INFO:root:val ap: 0.9482102030711226
DEBUG:root:--epoch = 15
INFO:root:epoch: 15 took 0.50s
INFO:root:Epoch mean loss: 0.7628280926834453
INFO:root:val auc: 0.9470749999999999
INFO:root:val ap: 0.9486239601990133
DEBUG:root:--epoch = 16
INFO:root:epoch: 16 took 0.50s
INFO:root:Epoch mean loss: 0.7262498004869982
INFO:root:val auc: 0.9496749999999999
INFO:root:val ap: 0.9498017918050922
DEBUG:root:--epoch = 17
INFO:root:epoch: 17 took 0.53s
INFO:root:Epoch mean loss: 0.7351956232027574
INFO:root:val auc: 0.948075
INFO:root:val ap: 0.9511056010881188
DEBUG:root:--epoch = 18
INFO:root:epoch: 18 took 0.49s
INFO:root:Epoch mean loss: 0.733869964426214
INFO:root:val auc: 0.9450000000000001
INFO:root:val ap: 0.9470740977708808
DEBUG:root:--epoch = 19
INFO:root:epoch: 19 took 0.54s
INFO:root:Epoch mean loss: 0.728515085848895
INFO:root:val auc: 0.948875
INFO:root:val ap: 0.9511230424586729
DEBUG:root:--epoch = 20
INFO:root:epoch: 20 took 0.50s
INFO:root:Epoch mean loss: 0.7325451346960935
INFO:root:val auc: 0.956025
INFO:root:val ap: 0.9546747896264054
DEBUG:root:--epoch = 21
INFO:root:epoch: 21 took 0.50s
INFO:root:Epoch mean loss: 0.7320527136325836
INFO:root:val auc: 0.95765
INFO:root:val ap: 0.9577511217723825
DEBUG:root:--epoch = 22
INFO:root:epoch: 22 took 0.50s
```

```
INFO:root:Epoch mean loss: 0.718647222627293
INFO:root:val auc: 0.949275
INFO:root:val ap: 0.9502125806072602
DEBUG:root:--epoch = 23
INFO:root:epoch: 23 took 0.50s
INFO:root:Epoch mean loss: 0.7000040168111975
INFO:root:val auc: 0.955875
INFO:root:val ap: 0.9566666164791613
DEBUG:root:--epoch = 24
INFO:root:epoch: 24 took 0.50s
INFO:root:Epoch mean loss: 0.7020668712529269
INFO:root:val auc: 0.954625
INFO:root:val ap: 0.9542322241938086
DEBUG:root:--epoch = 25
INFO:root:epoch: 25 took 0.50s
INFO:root:Epoch mean loss: 0.7042659195986661
INFO:root:val auc: 0.951825
INFO:root:val ap: 0.9506161934376934
DEBUG:root:--epoch = 26
INFO:root:epoch: 26 took 0.51s
INFO:root:Epoch mean loss: 0.6983385438268835
INFO:root:val auc: 0.95695
INFO:root:val ap: 0.9564245150990434
DEBUG:root:--epoch = 27
INFO:root:epoch: 27 took 0.50s
INFO:root:Epoch mean loss: 0.700971554626118
INFO:root:val auc: 0.952525
INFO:root:val ap: 0.9489945105353675
DEBUG:root:--epoch = 28
INFO:root:epoch: 28 took 0.49s
INFO:root:Epoch mean loss: 0.6873618120496924
INFO:root:val auc: 0.9560249999999999
INFO:root:val ap: 0.9522888623169722
DEBUG:root:--epoch = 29
INFO:root:epoch: 29 took 0.54s
INFO:root:Epoch mean loss: 0.6817551986737684
INFO:root:val auc: 0.9533750000000001
INFO:root:val ap: 0.9539159920115351
DEBUG:root:--epoch = 30
INFO:root:epoch: 30 took 0.50s
```

```
INFO:root:Epoch mean loss: 0.7085827453569933
INFO:root:val auc: 0.9635
INFO:root:val ap: 0.9611891433019144
DEBUG:root:--epoch = 31
INFO:root:epoch: 31 took 0.50s
INFO:root:Epoch mean loss: 0.6855492700229991
INFO:root:val auc: 0.9528
INFO:root:val ap: 0.9533939983082755
DEBUG:root:--epoch = 32
INFO:root:epoch: 32 took 0.55s
INFO:root:Epoch mean loss: 0.6594131236726587
INFO:root:val auc: 0.961475
INFO:root:val ap: 0.9623644917024854
DEBUG:root:--epoch = 33
INFO:root:epoch: 33 took 0.56s
INFO:root:Epoch mean loss: 0.6990084566853263
INFO:root:val auc: 0.959975
INFO:root:val ap: 0.961435656147925
DEBUG:root:--epoch = 34
INFO:root:epoch: 34 took 0.57s
INFO:root:Epoch mean loss: 0.6916343515569513
INFO:root:val auc: 0.96315
INFO:root:val ap: 0.9618798857599633
DEBUG:root:--epoch = 35
INFO:root:epoch: 35 took 0.54s
INFO:root:Epoch mean loss: 0.6860063834623857
INFO:root:val auc: 0.9594999999999999
INFO:root:val ap: 0.9603510893478658
DEBUG:root:--epoch = 36
INFO:root:epoch: 36 took 0.53s
INFO:root:Epoch mean loss: 0.6760019958019257
INFO:root:val auc: 0.9544
INFO:root:val ap: 0.9525532547352636
DEBUG:root:--epoch = 37
INFO:root:epoch: 37 took 0.52s
INFO:root:Epoch mean loss: 0.666663874279369
INFO:root:val auc: 0.9655499999999999
INFO:root:val ap: 0.9662657438763307
DEBUG:root:--epoch = 38
INFO:root:epoch: 38 took 0.51s
```

```
INFO:root:Epoch mean loss: 0.6834593523632396
INFO:root:val auc: 0.953625
INFO:root:val ap: 0.9517793681083785
DEBUG:root:--epoch = 39
INFO:root:epoch: 39 took 0.51s
INFO:root:Epoch mean loss: 0.6757948019287803
INFO:root:val auc: 0.9614750000000001
INFO:root:val ap: 0.9577951065613551
DEBUG:root:--epoch = 40
INFO:root:epoch: 40 took 0.51s
INFO:root:Epoch mean loss: 0.6654394702477888
INFO:root:val auc: 0.956425
INFO:root:val ap: 0.9551919818863073
DEBUG:root:--epoch = 41
INFO:root:epoch: 41 took 0.51s
INFO:root:Epoch mean loss: 0.689492174170234
INFO:root:val auc: 0.959475
INFO:root:val ap: 0.9593954996753545
DEBUG:root:--epoch = 42
INFO:root:epoch: 42 took 0.51s
INFO:root:Epoch mean loss: 0.6611424792896617
INFO:root:val auc: 0.96465
INFO:root:val ap: 0.9635917241883323
DEBUG:root:--epoch = 43
INFO:root:epoch: 43 took 0.51s
INFO:root:Epoch mean loss: 0.6726490584286776
INFO:root:val auc: 0.961775
INFO:root:val ap: 0.9615965078067547
DEBUG:root:--epoch = 44
INFO:root:epoch: 44 took 0.51s
INFO:root:Epoch mean loss: 0.6655654067342932
INFO:root:val auc: 0.96565
INFO:root:val ap: 0.9651783552876344
DEBUG:root:--epoch = 45
INFO:root:epoch: 45 took 0.51s
INFO:root:Epoch mean loss: 0.6700492799282074
INFO:root:val auc: 0.96365
INFO:root:val ap: 0.9586298246220298
DEBUG:root:--epoch = 46
INFO:root:epoch: 46 took 0.51s
```

```
INFO:root:Epoch mean loss: 0.6541122035546736
INFO:root:val auc: 0.956625
INFO:root:val ap: 0.9548830218850607
DEBUG:root:--epoch = 47
INFO:root:epoch: 47 took 0.51s
INFO:root:Epoch mean loss: 0.6513235623186285
INFO:root:val auc: 0.965475
INFO:root:val ap: 0.9642057857064718
DEBUG:root:--epoch = 48
INFO:root:epoch: 48 took 0.51s
INFO:root:Epoch mean loss: 0.6490750800479542
INFO:root:val auc: 0.9582250000000001
INFO:root:val ap: 0.9584988026360699
DEBUG:root:--epoch = 49
INFO:root:epoch: 49 took 0.59s
INFO:root:Epoch mean loss: 0.6496444940567017
INFO:root:val auc: 0.960275
INFO:root:val ap: 0.961748872845985
```

4.1.3 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

is there a way to show that "property" of time series changes overtimes. (to reflect why tgn performance fluguate window slideing wise.)

5 Research Date: *[2021-10-23 Sat]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

5.1 =IMPLEMENTATION MISTAKE= Research Log 1: Performance of tgn improve more clearly window sliding wise when initialization of tgn class is moved from outside of window sliding loop and inside of run loop.

5.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

intialization of tgn inside of run loop and outside of windoe sliding loop

5.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

improvment of tgn performance is more clear window sliding wise.

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

6 Research Date: *[2021-10-25 Mon]*

- ref
  - https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F

6.1 Research Log 1: I fixed the error that I found from incorrectly implemented the window sliding. Re-running the model (size of validation data is equivalent of BATCH_SIZE and window slide by BATCH_SIZE each time.) resutls in 88.97 % auc and 86.49 % absolute precision on the first epoch of the first window slides of the first run.

6.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I spotted many subtle error which results in totoally incorrect evaluation process.

Parameters configuration are as followed.

- epoch = 50
- `BATCH_SIZE` = 200
- size of validation is `BATCH_SIZE`
- window slide by `BATCH_SIZE`
- size of initial training set = 6725 instances (which is 1 percent of training set)

6.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Results are produced by following command

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

below is result of the beginging of the run

```
INFO:root:Namespace(aggregator='last', backprop_every=1, bs=200, data='reddit', differ
The dataset has 672447 interactions, involving 10984 different nodes
The training dataset has 389989 interactions, involving 9574 different nodes
The validation dataset has 100867 interactions, involving 9839 different nodes
The test dataset has 100867 interactions, involving 9615 different nodes
The new node validation dataset has 19446 interactions, involving 3491 different nodes
The new node test dataset has 21470 interactions, involving 3515 different nodes
1098 nodes were used for the inductive testing, i.e. are never seen during training
INFO:root:num of training instances: 672447
INFO:root:num of batches per epoch: 3363
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.45s
INFO:root:Epoch mean loss: 1.052928791326635
INFO:root:val auc: 0.8897469558906308
INFO:root:val ap: 0.86485 DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 7.08s
INFO:root:Epoch mean loss: 0.9887493542012047
INFO:root:val auc: 0.8688078079711408
INFO:root:val ap: 0.858375
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 7.07s
INFO:root:Epoch mean loss: 0.9637184610553816
INFO:root:val auc: 0.8596108140253464
INFO:root:val ap: 0.8611500000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.82s
INFO:root:Epoch mean loss: 0.941917825709371
INFO:root:val auc: 0.8994615511610304
INFO:root:val ap: 0.8934750000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.77s
```

```
INFO:root:Epoch mean loss: 0.9261942600502687
INFO:root:val auc: 0.9023812423418279
INFO:root:val ap: 0.8968750000000001
```

the last few epoch at window sliding = 0 shows that model imporve epoch wise.

```
DEBUG:root:--epoch = 43
INFO:root:start validation...
INFO:root:epoch: 43 took 6.96s
INFO:root:Epoch mean loss: 0.7361521935040619
INFO:root:val auc: 0.929309066981998
INFO:root:val ap: 0.92225
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 6.81s
INFO:root:Epoch mean loss: 0.7336613066640555
INFO:root:val auc: 0.931861767629921
INFO:root:val ap: 0.92065
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 6.71s
INFO:root:Epoch mean loss: 0.7312600785089881
INFO:root:val auc: 0.9459098870336862
INFO:root:val ap: 0.9393999999999999
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 6.64s
INFO:root:Epoch mean loss: 0.7289862949983945
INFO:root:val auc: 0.9245228029680058
INFO:root:val ap: 0.9183750000000002
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 6.78s
INFO:root:Epoch mean loss: 0.7265213445632481
INFO:root:val auc: 0.9266184409014651
INFO:root:val ap: 0.9198000000000001
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 6.64s
INFO:root:Epoch mean loss: 0.7240259257935676
```

```
INFO:root:val auc: 0.927404060343775
INFO:root:val ap: 0.91485
DEBUG:root:--epoch = 49
INFO:root:start validation...
INFO:root:epoch: 49 took 6.65s
INFO:root:Epoch mean loss: 0.7213836356646874
INFO:root:val auc: 0.9304827199930047
INFO:root:val ap: 0.921325
```

at the begining of at window sliding = 9, we can see that performance has been improve window slides wise.

```
DEBUG:root:-ws = 9
INFO:root:start validation...
INFO:root:epoch: 0 took 8.71s
INFO:root:Epoch mean loss: 0.31605055720307107
INFO:root:val auc: 0.948271087922951
INFO:root:val ap: 0.93985
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 8.65s
INFO:root:Epoch mean loss: 0.32300567869530167
INFO:root:val auc: 0.942160147013112
INFO:root:val ap: 0.937225
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 8.84s
INFO:root:Epoch mean loss: 0.31925479894460634
INFO:root:val auc: 0.9489616222541905
INFO:root:val ap: 0.939925
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 8.59s
INFO:root:Epoch mean loss: 0.31782242326542387
INFO:root:val auc: 0.929089589439169
INFO:root:val ap: 0.92645
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 8.64s
INFO:root:Epoch mean loss: 0.3150137609520624
```

```
INFO:root:val auc: 0.9229070102548259
INFO:root:val ap: 0.9208500000000001
```

at the last few epoch of window sliding = 9, we can see that performance has improve when comparing each sliding windows performance epoch wise,

```
INFO:root:epoch: 43 took 8.63s
INFO:root:Epoch mean loss: 0.2948349748106071
INFO:root:val auc: 0.9440148323733668
INFO:root:val ap: 0.9344000000000001
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 8.74s
INFO:root:Epoch mean loss: 0.29469804887392725
INFO:root:val auc: 0.9459571109224829
INFO:root:val ap: 0.9358749999999999
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 8.56s
INFO:root:Epoch mean loss: 0.29437613250316697
INFO:root:val auc: 0.9461892901230279
INFO:root:val ap: 0.936875
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 8.28s
INFO:root:Epoch mean loss: 0.29392520030836017
INFO:root:val auc: 0.9369656878789392
INFO:root:val ap: 0.928825
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 8.37s
INFO:root:Epoch mean loss: 0.29369322981959745
INFO:root:val auc: 0.9489585615301237
INFO:root:val ap: 0.9356749999999999
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 8.34s
INFO:root:Epoch mean loss: 0.29348102484586175
INFO:root:val auc: 0.9422066845215218
INFO:root:val ap: 0.931975
DEBUG:root:--epoch = 49
```

```
INFO:root:start validation...
INFO:root:epoch: 49 took 8.58s
INFO:root:Epoch mean loss: 0.29309554672171906
INFO:root:val auc: 0.9461782362276291
INFO:root:val ap: 0.9359875000000001
```

6.1.3 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

There could still be few more minor error with various side effect (range from no impact to outcome to completely incorrect evaluation process.), but, as of now, I have certain confident of my understanding of the models (which can still be wrong.).

6.2 Research Log 2: Using the same model, I reduced size of initial training set to 0.1 percent which 673 instances and test data is 200 instances (~BATCH_SIZE~ * 1). Performance of the first epoch of the first window slides of the first run is 66.56 % auc and 69.13 ap. (almost random as Dr. zhu has expected.)

6.2.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Parameters configuration are as followed.

- epoch = 50
- BATCH_SIZE = 200

- size of validation is `BATCH_SIZE`
- window slide by `BATCH_SIZE`
- size of initial training set = 673 instances (which is 0.1 percent of training set)

6.2.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Results are produced by following command (command is the same because I modified the code without changning command line options)

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

below is result of the first 10 epochs of the run

```
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.41s
INFO:root:Epoch mean loss: 1.373808354139328
INFO:root:val auc: 0.665561098684202
INFO:root:val ap: 0.6912750000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.55s
INFO:root:Epoch mean loss: 1.3438538759946823
INFO:root:val auc: 0.7353635028218445
INFO:root:val ap: 0.7559125
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.58s
INFO:root:Epoch mean loss: 1.315415730079015
INFO:root:val auc: 0.8308778189752446
```

```
INFO:root:val ap: 0.81885
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.59s
INFO:root:Epoch mean loss: 1.2901920676231384
INFO:root:val auc: 0.8001855307437327
INFO:root:val ap: 0.8147249999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.58s
INFO:root:Epoch mean loss: 1.2648576319217681
INFO:root:val auc: 0.77991865704801
INFO:root:val ap: 0.809325
DEBUG:root:--epoch = 5
INFO:root:start validation...
INFO:root:epoch: 5 took 2.58s
INFO:root:Epoch mean loss: 1.24538508305947
INFO:root:val auc: 0.782943395766339
INFO:root:val ap: 0.7909125
DEBUG:root:--epoch = 6
INFO:root:start validation...
INFO:root:epoch: 6 took 2.68s
INFO:root:Epoch mean loss: 1.2294273184878486
INFO:root:val auc: 0.8174629034094566
INFO:root:val ap: 0.816575
DEBUG:root:--epoch = 7
INFO:root:start validation...
INFO:root:epoch: 7 took 2.64s
INFO:root:Epoch mean loss: 1.218052553012967
INFO:root:val auc: 0.8116568226075354
INFO:root:val ap: 0.8242499999999999
DEBUG:root:--epoch = 8
INFO:root:start validation...
INFO:root:epoch: 8 took 2.65s
INFO:root:Epoch mean loss: 1.2063783158858616
INFO:root:val auc: 0.7968244604958872
INFO:root:val ap: 0.8065875
DEBUG:root:--epoch = 9
INFO:root:start validation...
INFO:root:epoch: 9 took 2.66s
```

```
INFO:root:Epoch mean loss: 1.1951952621340751
INFO:root:val auc: 0.8230072051789133
INFO:root:val ap: 0.8368125
DEBUG:root:--epoch = 10
INFO:root:start validation...
INFO:root:epoch: 10 took 2.56s
INFO:root:Epoch mean loss: 1.1847205554897136
INFO:root:val auc: 0.815269468953453
INFO:root:val ap: 0.8404499999999999
```

the last few epoch at window sliding = 0 shows that model imporve epoch wise.

```
DEBUG:root:--epoch = 43
INFO:root:start validation...
INFO:root:epoch: 43 took 2.65s
INFO:root:Epoch mean loss: 0.9855934185060587
INFO:root:val auc: 0.7977299829913729
INFO:root:val ap: 0.8101749999999999
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 2.65s
INFO:root:Epoch mean loss: 0.9820751511388355
INFO:root:val auc: 0.8214032606694464
INFO:root:val ap: 0.823625
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 2.66s
INFO:root:Epoch mean loss: 0.9779674125754315
INFO:root:val auc: 0.7929033486807462
INFO:root:val ap: 0.8094250000000001
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 2.74s
INFO:root:Epoch mean loss: 0.9746559037173048
INFO:root:val auc: 0.8173192547642147
INFO:root:val ap: 0.8178749999999999
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 2.84s
INFO:root:Epoch mean loss: 0.9707115128015479
```

```
INFO:root:val auc: 0.8301512345509248
INFO:root:val ap: 0.8203750000000001
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 3.00s
INFO:root:Epoch mean loss: 0.9673184725094814
INFO:root:val auc: 0.7603498150434573
INFO:root:val ap: 0.8083749999999998
DEBUG:root:--epoch = 49
INFO:root:start validation...
INFO:root:epoch: 49 took 2.82s
INFO:root:Epoch mean loss: 0.9633100417256355
INFO:root:val auc: 0.8116656302925396
INFO:root:val ap: 0.8152750000000001
```

at the begining of at window sliding = 6, we can see that performance has been improve window slides wise.

```
DEBUG:root:-ws = 6
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.36s
INFO:root:Epoch mean loss: 0.5169417142868042
INFO:root:val auc: 0.8310620908292949
INFO:root:val ap: 0.821825
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.44s
INFO:root:Epoch mean loss: 0.5263854444026947
INFO:root:val auc: 0.8397020361015
INFO:root:val ap: 0.8283999999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.41s
INFO:root:Epoch mean loss: 0.528552093108495
INFO:root:val auc: 0.8370854859744108
INFO:root:val ap: 0.83905
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.34s
INFO:root:Epoch mean loss: 0.5329772986471653
```

```
INFO:root:val auc: 0.820963999442816
INFO:root:val ap: 0.815225
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.48s
INFO:root:Epoch mean loss: 0.5357914137840271
INFO:root:val auc: 0.8509630806455368
INFO:root:val ap: 0.8362250000000001
DEBUG:root:--epoch = 5
INFO:root:start validation...
INFO:root:epoch: 5 took 3.38s
INFO:root:Epoch mean loss: 0.5340422347187996
INFO:root:val auc: 0.8482639004153758
INFO:root:val ap: 0.82795
DEBUG:root:--epoch = 6
INFO:root:start validation...
INFO:root:epoch: 6 took 3.30s
INFO:root:Epoch mean loss: 0.5327632542167391
INFO:root:val auc: 0.8692719526204099
INFO:root:val ap: 0.8501000000000001
DEBUG:root:--epoch = 7
INFO:root:start validation...
INFO:root:epoch: 7 took 3.42s
INFO:root:Epoch mean loss: 0.5276553623378277
INFO:root:val auc: 0.864684765357338
INFO:root:val ap: 0.8494999999999999
DEBUG:root:--epoch = 8
INFO:root:start validation...
INFO:root:epoch: 8 took 3.38s
INFO:root:Epoch mean loss: 0.5222941017813153
INFO:root:val auc: 0.8778004480112852
INFO:root:val ap: 0.8566250000000002
```

at the last few epoch of window sliding = 6, we can see that performance has improve when comparing each sliding windows performance epoch wise,

```
DEBUG:root:--epoch = 41
INFO:root:start validation...
INFO:root:epoch: 41 took 3.43s
INFO:root:Epoch mean loss: 0.44555269231398903
INFO:root:val auc: 0.8666109716124597
```

```
INFO:root:val ap: 0.8478249999999999
DEBUG:root:--epoch = 42
INFO:root:start validation...
INFO:root:epoch: 42 took 3.52s
INFO:root:Epoch mean loss: 0.44494890834009926
INFO:root:val auc: 0.8658646465611006
INFO:root:val ap: 0.8432750000000001
DEBUG:root:--epoch = 43
INFO:root:start validation...
INFO:root:epoch: 43 took 3.38s
INFO:root:Epoch mean loss: 0.4451101179827343
INFO:root:val auc: 0.8320970567211693
INFO:root:val ap: 0.825875
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 3.48s
INFO:root:Epoch mean loss: 0.444454809361034
INFO:root:val auc: 0.8618565337630704
INFO:root:val ap: 0.8388
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 3.53s
INFO:root:Epoch mean loss: 0.4436413047754246
INFO:root:val auc: 0.8540459099566079
INFO:root:val ap: 0.8484
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 3.59s
INFO:root:Epoch mean loss: 0.4428245441076603
INFO:root:val auc: 0.8749871314804621
INFO:root:val ap: 0.853
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 3.59s
INFO:root:Epoch mean loss: 0.4415894846742352
INFO:root:val auc: 0.8439036608907651
INFO:root:val ap: 0.8331999999999999
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 3.49s
```

```
INFO:root:Epoch mean loss: 0.44040886224532616
INFO:root:val auc: 0.8487141821420114
INFO:root:val ap: 0.82995
DEBUG:root:--epoch = 49
INFO:root:start validation...
INFO:root:epoch: 49 took 3.52s
INFO:root:Epoch mean loss: 0.438747676551342
INFO:root:val auc: 0.8888354997482291
INFO:root:val ap: 0.8732249999999999

from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635187985.5239232.log')
plot_path = str(base_path / 'plot/1635187985.5239232.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
```

```
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])

# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)

fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
```

```
axs[0].legend()
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)

plot_path
```

6.3 Research Log 3: Using the same model, I reduced size of initial training set to 0.1 percent which 673 instances and increase test data to be 14000 instances (BATCH_SIZE * 70). Performance of the first epoch of the first window slides of the first run is 65.11 % auc and 63.61 ap. (almost random as Dr. zhu has expected.)

6.3.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Parameters configuration are as followed.

- epoch = 50
- BATCH_SIZE = 200
- size of validation is BATCH_SIZE * 70
- window slide by BATCH_SIZE
- size of initial training set = 673 instances (which is 0.1 percent of training set)

6.3.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?

– Describe what you have done and how you have done it.

Results are produced by following command (command is the same because I modified the code without changning command line options)

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

below is result of the first 10 epochs of the run

```
INFO:root:val auc: 0.6511335343309306
INFO:root:val ap: 0.6360903928571429
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 4.45s
INFO:root:Epoch mean loss: 1.3384163230657578
INFO:root:val auc: 0.6512451425948134
INFO:root:val ap: 0.6137665382653061
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 4.40s
INFO:root:Epoch mean loss: 1.3104274074236553
INFO:root:val auc: 0.668279682217932
INFO:root:val ap: 0.6144394591836735
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 4.36s
INFO:root:Epoch mean loss: 1.2870160341262817
INFO:root:val auc: 0.6812942614612717
INFO:root:val ap: 0.6224783494897959
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 4.62s
INFO:root:Epoch mean loss: 1.269434529542923
INFO:root:val auc: 0.6952138000592115
INFO:root:val ap: 0.6348522015306122
DEBUG:root:--epoch = 5
INFO:root:start validation...
INFO:root:epoch: 5 took 4.91s
INFO:root:Epoch mean loss: 1.2513272265593212
INFO:root:val auc: 0.702692208997267
INFO:root:val ap: 0.6526283954081633
```

```
DEBUG:root:--epoch = 6
INFO:root:start validation...
INFO:root:epoch: 6 took 5.13s
INFO:root:Epoch mean loss: 1.2329048833676748
INFO:root:val auc: 0.7211337052497626
INFO:root:val ap: 0.6732404566326531
DEBUG:root:--epoch = 7
INFO:root:start validation...
INFO:root:epoch: 7 took 4.44s
INFO:root:Epoch mean loss: 1.219748007133603
INFO:root:val auc: 0.7321618606305541
INFO:root:val ap: 0.6845407066326531
DEBUG:root:--epoch = 8
INFO:root:start validation...
INFO:root:epoch: 8 took 4.95s
INFO:root:Epoch mean loss: 1.205672002500958
INFO:root:val auc: 0.7274293267106745
INFO:root:val ap: 0.6813673979591837
DEBUG:root:--epoch = 9
INFO:root:start validation...
INFO:root:epoch: 9 took 4.65s
INFO:root:Epoch mean loss: 1.1930748984217643
INFO:root:val auc: 0.7284264869512782
INFO:root:val ap: 0.6881935918367347
DEBUG:root:--epoch = 10
INFO:root:start validation...
INFO:root:epoch: 10 took 4.69s
INFO:root:Epoch mean loss: 1.182747701352293
INFO:root:val auc: 0.7354694166872723
INFO:root:val ap: 0.6933646709183673
```

the last few epoch at window sliding = 0 shows that model imporve epoch wise.

```
DEBUG:root:--epoch = 41
INFO:root:start validation...
INFO:root:epoch: 41 took 2.81s
INFO:root:Epoch mean loss: 0.9908000631701379
INFO:root:val auc: 0.8445459661355534
INFO:root:val ap: 0.834775
DEBUG:root:--epoch = 42
```

```
INFO:root:start validation...
INFO:root:epoch: 42 took 2.74s
INFO:root:Epoch mean loss: 0.9874055517274279
INFO:root:val auc: 0.8354327492512973
INFO:root:val ap: 0.8295250000000001
DEBUG:root:--epoch = 43
INFO:root:start validation...
INFO:root:epoch: 43 took 2.95s
INFO:root:Epoch mean loss: 0.9855934185060587
INFO:root:val auc: 0.7977299829913729
INFO:root:val ap: 0.8101749999999999
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 2.82s
INFO:root:Epoch mean loss: 0.9820751511388355
INFO:root:val auc: 0.8214032606694464
INFO:root:val ap: 0.823625
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 2.72s
INFO:root:Epoch mean loss: 0.9779674125754315
INFO:root:val auc: 0.7929033486807462
INFO:root:val ap: 0.8094250000000001
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 2.87s
INFO:root:Epoch mean loss: 0.9746559037173048
INFO:root:val auc: 0.8173192547642147
INFO:root:val ap: 0.8178749999999999
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 2.78s
INFO:root:Epoch mean loss: 0.9707115128015479
INFO:root:val auc: 0.8301512345509248
INFO:root:val ap: 0.8203750000000001
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 2.69s
INFO:root:Epoch mean loss: 0.9673184725094814
INFO:root:val auc: 0.7603498150434573
```

```
INFO:root:val ap: 0.8083749999999998
DEBUG:root:--epoch = 49
INFO:root:start validation...
INFO:root:epoch: 49 took 2.67s
INFO:root:Epoch mean loss: 0.9633100417256355
INFO:root:val auc: 0.8116656302925396
INFO:root:val ap: 0.8152750000000001
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.73s
INFO:root:Epoch mean loss: 1.0040823698043824
INFO:root:val auc: 0.812749390880110
```

at the begining of at window sliding = 12, we can see that performance has been improve window slides wise.

```
INFO:root:start validation...
INFO:root:epoch: 1 took 4.34s
INFO:root:Epoch mean loss: 0.45914822444319725
INFO:root:val auc: 0.8848313098855033
INFO:root:val ap: 0.8616250000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 4.23s
INFO:root:Epoch mean loss: 0.46083338807026547
INFO:root:val auc: 0.8856304509881854
INFO:root:val ap: 0.872275
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 4.33s
INFO:root:Epoch mean loss: 0.45658676885068417
INFO:root:val auc: 0.8927446567736237
INFO:root:val ap: 0.8863249999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 4.18s
INFO:root:Epoch mean loss: 0.45376761369407176
INFO:root:val auc: 0.8767108598990043
INFO:root:val ap: 0.862975
DEBUG:root:--epoch = 5
```

```
INFO:root:start validation...
INFO:root:epoch: 5 took 4.28s
INFO:root:Epoch mean loss: 0.4565013349056244
INFO:root:val auc: 0.8937644460087458
INFO:root:val ap: 0.8794000000000001
DEBUG:root:--epoch = 6
INFO:root:start validation...
INFO:root:epoch: 6 took 4.54s
INFO:root:Epoch mean loss: 0.4542215139205967
INFO:root:val auc: 0.8706529419784408
INFO:root:val ap: 0.858875
DEBUG:root:--epoch = 7
INFO:root:start validation...
INFO:root:epoch: 7 took 4.82s
INFO:root:Epoch mean loss: 0.45427905046381056
INFO:root:val auc: 0.8566159866128196
INFO:root:val ap: 0.852125
DEBUG:root:--epoch = 8
INFO:root:start validation...
INFO:root:epoch: 8 took 4.81s
INFO:root:Epoch mean loss: 0.4532749371396171
INFO:root:val auc: 0.8953518350014628
INFO:root:val ap: 0.8728000000000001
DEBUG:root:--epoch = 9
INFO:root:start validation...
INFO:root:epoch: 9 took 5.00s
INFO:root:Epoch mean loss: 0.4521314548328519
INFO:root:val auc: 0.872496321027658
INFO:root:val ap: 0.858925
DEBUG:root:--epoch = 10
INFO:root:start validation...
INFO:root:epoch: 10 took 4.35s
INFO:root:Epoch mean loss: 0.44915330105207185
INFO:root:val auc: 0.8898807702906841
INFO:root:val ap: 0.86975
```

at the last few epoch of window sliding = 12, we can see that performance has improve when comparing each sliding windows performance epoch wise,

```
DEBUG:root:--epoch = 42
INFO:root:start validation...
```

```
INFO:root:epoch: 42 took 4.72s
INFO:root:Epoch mean loss: 0.4143759681475024
INFO:root:val auc: 0.8972788577844573
INFO:root:val ap: 0.87925
DEBUG:root:--epoch = 43
INFO:root:start validation...
INFO:root:epoch: 43 took 4.36s
INFO:root:Epoch mean loss: 0.41432577587494795
INFO:root:val auc: 0.8800932350334045
INFO:root:val ap: 0.8612000000000001
DEBUG:root:--epoch = 44
INFO:root:start validation...
INFO:root:epoch: 44 took 4.35s
INFO:root:Epoch mean loss: 0.41374201288239826
INFO:root:val auc: 0.8740375556667777
INFO:root:val ap: 0.8629875
DEBUG:root:--epoch = 45
INFO:root:start validation...
INFO:root:epoch: 45 took 4.36s
INFO:root:Epoch mean loss: 0.4127914353840701
INFO:root:val auc: 0.8619367852319543
INFO:root:val ap: 0.8559499999999999
DEBUG:root:--epoch = 46
INFO:root:start validation...
INFO:root:epoch: 46 took 4.44s
INFO:root:Epoch mean loss: 0.4118439966733468
INFO:root:val auc: 0.8864523498447134
INFO:root:val ap: 0.8687874999999999
DEBUG:root:--epoch = 47
INFO:root:start validation...
INFO:root:epoch: 47 took 4.34s
INFO:root:Epoch mean loss: 0.4109473192171815
INFO:root:val auc: 0.875179607880175
INFO:root:val ap: 0.85575
DEBUG:root:--epoch = 48
INFO:root:start validation...
INFO:root:epoch: 48 took 4.31s
INFO:root:Epoch mean loss: 0.41058344730385105
INFO:root:val auc: 0.8626506222513223
INFO:root:val ap: 0.8478000000000001
```

```
DEBUG:root:--epoch = 49
INFO:root:start validation...
INFO:root:epoch: 49 took 4.30s
INFO:root:Epoch mean loss: 0.4100404874049127
INFO:root:val auc: 0.8754560017300959
INFO:root:val ap: 0.8584999999999998
```

1635191185

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635191185.8549829.log')
plot_path = str(base_path / 'plot/1635191185.8549829.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
```

```
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])

# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)

fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
```

```
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)

plot_path
```

6.3.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

Comparing `=Reserach Log 2=` and `=Research Log 3=`, performance of `=Research Log 2=` improve faster than `=Research Log 3=`. This may be because temporal patterns are more useful to predict short term learning.

6.4 Research Log 4: similar to research log 2 except that the window slides by 10 instances and epoch is reduced from 50 to 5. (because we are concerned about window wise performance ratehr than epoch wise performance)

6.4.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Experiment is suggested by dr zhu in this email.
Parameters configuration are as followed.

- epoch = 5
- `BATCH_SIZE` = 200
- size of validation is `BATCH_SIZE`
- window slide by 10
- size of initial training set = 673 instances (which is 0.1 percent of training set)

6.4.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Results are produced by following command (command is the same because I modified the code without changning command line options)

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

window slides from 0 - 11

```
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.68s
INFO:root:Epoch mean loss: 1.373808354139328
INFO:root:val auc: 0.665561098684202
INFO:root:val ap: 0.6912750000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.49s
INFO:root:Epoch mean loss: 1.3438538759946823
INFO:root:val auc: 0.7353635028218445
INFO:root:val ap: 0.7559125
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.29s
INFO:root:Epoch mean loss: 1.315415730079015
INFO:root:val auc: 0.8308778189752446
INFO:root:val ap: 0.81885
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.46s
```

```
INFO:root:Epoch mean loss: 1.2901920676231384
INFO:root:val auc: 0.8001855307437327
INFO:root:val ap: 0.8147249999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.72s
INFO:root:Epoch mean loss: 1.2648576319217681
INFO:root:val auc: 0.77991865704801
INFO:root:val ap: 0.809325
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.17s
INFO:root:Epoch mean loss: 1.1583638787269592
INFO:root:val auc: 0.7842413101310546
INFO:root:val ap: 0.79475
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.57s
INFO:root:Epoch mean loss: 1.1500537246465683
INFO:root:val auc: 0.8269340798429601
INFO:root:val ap: 0.828675
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.84s
INFO:root:Epoch mean loss: 1.141145646572113
INFO:root:val auc: 0.8470407359245166
INFO:root:val ap: 0.8452875
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.03s
INFO:root:Epoch mean loss: 1.1363963931798935
INFO:root:val auc: 0.8319009778625907
INFO:root:val ap: 0.8261375
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.41s
INFO:root:Epoch mean loss: 1.1250496447086333
INFO:root:val auc: 0.8109861352206092
INFO:root:val ap: 0.8249875
```

```
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.86s
INFO:root:Epoch mean loss: 1.0831964612007141
INFO:root:val auc: 0.7958236511350533
INFO:root:val ap: 0.8223125000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.73s
INFO:root:Epoch mean loss: 1.0737642720341682
INFO:root:val auc: 0.8056329811184938
INFO:root:val ap: 0.8208749999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.26s
INFO:root:Epoch mean loss: 1.061115692059199
INFO:root:val auc: 0.814760354172205
INFO:root:val ap: 0.8287249999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.08s
INFO:root:Epoch mean loss: 1.0548642314970493
INFO:root:val auc: 0.8195999573498309
INFO:root:val ap: 0.810875
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.10s
INFO:root:Epoch mean loss: 1.040177944302559
INFO:root:val auc: 0.7939528265177259
INFO:root:val ap: 0.8136499999999999
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.86s
INFO:root:Epoch mean loss: 1.0137425065040588
INFO:root:val auc: 0.8496990695346767
INFO:root:val ap: 0.8436125000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
```

```
INFO:root:epoch: 1 took 5.76s
INFO:root:Epoch mean loss: 1.026831865310669
INFO:root:val auc: 0.7614639805790604
INFO:root:val ap: 0.7922
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.06s
INFO:root:Epoch mean loss: 1.0137006143728893
INFO:root:val auc: 0.8279466139210793
INFO:root:val ap: 0.8408500000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.83s
INFO:root:Epoch mean loss: 1.0059393793344498
INFO:root:val auc: 0.8286436812407252
INFO:root:val ap: 0.8421500000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.70s
INFO:root:Epoch mean loss: 1.004154509305954
INFO:root:val auc: 0.8145646755528133
INFO:root:val ap: 0.8298
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.62s
INFO:root:Epoch mean loss: 0.9660821855068207
INFO:root:val auc: 0.8569493697732077
INFO:root:val ap: 0.8509125000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 6.31s
INFO:root:Epoch mean loss: 0.9751399531960487
INFO:root:val auc: 0.777994721936543
INFO:root:val ap: 0.816075
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.54s
INFO:root:Epoch mean loss: 0.9659339437882105
INFO:root:val auc: 0.8244802230299247
```

```
INFO:root:val ap: 0.83045
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.53s
INFO:root:Epoch mean loss: 0.9808125235140324
INFO:root:val auc: 0.8468145721220256
INFO:root:val ap: 0.8406
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.63s
INFO:root:Epoch mean loss: 0.9794301450252533
INFO:root:val auc: 0.7891446821697532
INFO:root:val ap: 0.8029999999999999
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.55s
INFO:root:Epoch mean loss: 0.9929684102535248
INFO:root:val auc: 0.821682398760074
INFO:root:val ap: 0.8188
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.99s
INFO:root:Epoch mean loss: 0.970871165394783
INFO:root:val auc: 0.8250275058675668
INFO:root:val ap: 0.8282499999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.88s
INFO:root:Epoch mean loss: 0.9673895090818405
INFO:root:val auc: 0.8181926302013917
INFO:root:val ap: 0.8308000000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.51s
INFO:root:Epoch mean loss: 0.9537641443312168
INFO:root:val auc: 0.8232621260926585
INFO:root:val ap: 0.8257
DEBUG:root:--epoch = 4
INFO:root:start validation...
```

```
INFO:root:epoch: 4 took 5.29s
INFO:root:Epoch mean loss: 0.948519092798233
INFO:root:val auc: 0.8023793947877066
INFO:root:val ap: 0.8083750000000001
DEBUG:root:-ws = 6
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.77s
INFO:root:Epoch mean loss: 0.9294135868549347
INFO:root:val auc: 0.808663605913903
INFO:root:val ap: 0.816525
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.51s
INFO:root:Epoch mean loss: 0.9158882796764374
INFO:root:val auc: 0.7991101135022662
INFO:root:val ap: 0.8086749999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.54s
INFO:root:Epoch mean loss: 0.9219316492478052
INFO:root:val auc: 0.7455683785170508
INFO:root:val ap: 0.7703749999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.09s
INFO:root:Epoch mean loss: 0.9227612391114235
INFO:root:val auc: 0.817477605510263
INFO:root:val ap: 0.8177625000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.35s
INFO:root:Epoch mean loss: 0.916709166765213
INFO:root:val auc: 0.8596765462350913
INFO:root:val ap: 0.84045
DEBUG:root:-ws = 7
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.52s
INFO:root:Epoch mean loss: 0.8801154494285583
```

```
INFO:root:val auc: 0.8235985394742491
INFO:root:val ap: 0.82375
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.43s
INFO:root:Epoch mean loss: 0.8918719813227654
INFO:root:val auc: 0.7917512343924529
INFO:root:val ap: 0.7995000000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.59s
INFO:root:Epoch mean loss: 0.8876953025658926
INFO:root:val auc: 0.8536143168952812
INFO:root:val ap: 0.83165
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.35s
INFO:root:Epoch mean loss: 0.8839072212576866
INFO:root:val auc: 0.8168125535494811
INFO:root:val ap: 0.8197250000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.52s
INFO:root:Epoch mean loss: 0.8898903161287308
INFO:root:val auc: 0.7824664274031693
INFO:root:val ap: 0.785575
DEBUG:root:-ws = 8
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.21s
INFO:root:Epoch mean loss: 0.8735684752464294
INFO:root:val auc: 0.7945608901024718
INFO:root:val ap: 0.7988249999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.74s
INFO:root:Epoch mean loss: 0.8806914687156677
INFO:root:val auc: 0.8151094496108132
INFO:root:val ap: 0.80285
DEBUG:root:--epoch = 2
```

```
INFO:root:start validation...
INFO:root:epoch: 2 took 5.34s
INFO:root:Epoch mean loss: 0.8785523821910223
INFO:root:val auc: 0.8148555892118198
INFO:root:val ap: 0.80805
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.52s
INFO:root:Epoch mean loss: 0.878622192889452
INFO:root:val auc: 0.8214304666402381
INFO:root:val ap: 0.8088375000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.04s
INFO:root:Epoch mean loss: 0.8779767394065857
INFO:root:val auc: 0.836749811896152
INFO:root:val ap: 0.8338
DEBUG:root:-ws = 9
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.48s
INFO:root:Epoch mean loss: 0.8631740212440491
INFO:root:val auc: 0.8189712791755639
INFO:root:val ap: 0.821725
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.71s
INFO:root:Epoch mean loss: 0.855125792324543
INFO:root:val auc: 0.8130551985417914
INFO:root:val ap: 0.81725
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 5.51s
INFO:root:Epoch mean loss: 0.8509291559457779
INFO:root:val auc: 0.8181002841719129
INFO:root:val ap: 0.820425
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.34s
INFO:root:Epoch mean loss: 0.8542343378067017
```

```
INFO:root:val auc: 0.8144746338671741
INFO:root:val ap: 0.8153500000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.77s
INFO:root:Epoch mean loss: 0.8510933250188828
INFO:root:val auc: 0.7980211188889015
INFO:root:val ap: 0.8177000000000001
DEBUG:root:-ws = 10
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.95s
INFO:root:Epoch mean loss: 0.8450995832681656
INFO:root:val auc: 0.7957611356695007
INFO:root:val ap: 0.798875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.69s
INFO:root:Epoch mean loss: 0.8419748544692993
INFO:root:val auc: 0.8255452652656996
INFO:root:val ap: 0.819675
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.00s
INFO:root:Epoch mean loss: 0.8368527988592783
INFO:root:val auc: 0.8089988048468102
INFO:root:val ap: 0.8071999999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 5.70s
INFO:root:Epoch mean loss: 0.8405385315418243
INFO:root:val auc: 0.7925962592227653
INFO:root:val ap: 0.7878750000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 5.41s
INFO:root:Epoch mean loss: 0.832576060295105
INFO:root:val auc: 0.7883837774692198
INFO:root:val ap: 0.799325
DEBUG:root:-ws = 11
```

```
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 5.25s
INFO:root:Epoch mean loss: 0.7873219102621078
INFO:root:val auc: 0.8508060903136667
INFO:root:val ap: 0.8346875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 5.47s
INFO:root:Epoch mean loss: 0.8013995960354805
INFO:root:val auc: 0.8284296645488999
INFO:root:val ap: 0.819974
```

window slide from 29 - 31

```
DEBUG:root:-ws = 29
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.81s
INFO:root:Epoch mean loss: 0.5996578514575959
INFO:root:val auc: 0.8410694398618941
INFO:root:val ap: 0.8402
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.72s
INFO:root:Epoch mean loss: 0.5784491449594498
INFO:root:val auc: 0.8260065313362408
INFO:root:val ap: 0.8290875
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.77s
INFO:root:Epoch mean loss: 0.5758655389149984
INFO:root:val auc: 0.8333711551650275
INFO:root:val ap: 0.8472875
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.73s
INFO:root:Epoch mean loss: 0.5679163485765457
INFO:root:val auc: 0.8289472782001035
INFO:root:val ap: 0.844975
DEBUG:root:--epoch = 4
```

```
INFO:root:start validation...
INFO:root:epoch: 4 took 2.71s
INFO:root:Epoch mean loss: 0.5673521149158478
INFO:root:val auc: 0.8183706401918789
INFO:root:val ap: 0.8224625
DEBUG:root:-ws = 30
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.77s
INFO:root:Epoch mean loss: 0.5616489946842194
INFO:root:val auc: 0.871077002200196
INFO:root:val ap: 0.8551000000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.16s
INFO:root:Epoch mean loss: 0.5558686226606369
INFO:root:val auc: 0.8124931163389428
INFO:root:val ap: 0.81195
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.87s
INFO:root:Epoch mean loss: 0.5602826714515686
INFO:root:val auc: 0.8533982829459295
INFO:root:val ap: 0.8423250000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.74s
INFO:root:Epoch mean loss: 0.5588474214076996
INFO:root:val auc: 0.8370496377276873
INFO:root:val ap: 0.8349000000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.97s
INFO:root:Epoch mean loss: 0.5598188519477845
INFO:root:val auc: 0.8656066646065688
INFO:root:val ap: 0.855825
DEBUG:root:-ws = 31
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.86s
```

```
INFO:root:Epoch mean loss: 0.5559526562690735
INFO:root:val auc: 0.807369320734824
INFO:root:val ap: 0.811175
INFO:root:start validation...
INFO:root:epoch: 1 took 2.75s
INFO:root:Epoch mean loss: 0.5351231098175049
INFO:root:val auc: 0.82883
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.88s
INFO:root:Epoch mean loss: 0.5410954693953196
INFO:root:val auc: 0.823632907654786
INFO:root:val ap: 0.834075
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.76s
INFO:root:Epoch mean loss: 0.5381185799837113
INFO:root:val auc: 0.8179015647270385
INFO:root:val ap: 0.810225
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.95s
INFO:root:Epoch mean loss: 0.5388071215152741
INFO:root:val auc: 0.8337929440813407
INFO:root:val ap: 0.8291999999999999
DEBUG:root:-ws = 32
```

window slides from 120 - 123

```
DEBUG:root:-ws = 120
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.39s
INFO:root:Epoch mean loss: 0.38192642033100127
INFO:root:val auc: 0.861731593687741
INFO:root:val ap: 0.848975
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.26s
INFO:root:Epoch mean loss: 0.3769414186477661
INFO:root:val auc: 0.8472613213219015
```

```
INFO:root:val ap: 0.8399499999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.40s
INFO:root:Epoch mean loss: 0.37224142452081044
INFO:root:val auc: 0.853990351928941
INFO:root:val ap: 0.84355
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.45s
INFO:root:Epoch mean loss: 0.37095538303256037
INFO:root:val auc: 0.8670513395962873
INFO:root:val ap: 0.846125
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.42s
INFO:root:Epoch mean loss: 0.3699114257097244
INFO:root:val auc: 0.8838070719273978
INFO:root:val ap: 0.859725
DEBUG:root:-ws = 121
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.47s
INFO:root:Epoch mean loss: 0.36818104088306425
INFO:root:val auc: 0.8648194384449341
INFO:root:val ap: 0.8384750000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.48s
INFO:root:Epoch mean loss: 0.3605219177901745
INFO:root:val auc: 0.807539415221221
INFO:root:val ap: 0.815825
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.38s
INFO:root:Epoch mean loss: 0.36318827817837396
INFO:root:val auc: 0.8608213571867187
INFO:root:val ap: 0.835675
DEBUG:root:--epoch = 3
INFO:root:start validation...
```

```
INFO:root:epoch: 3 took 3.32s
INFO:root:Epoch mean loss: 0.3612583946436644
INFO:root:val auc: 0.878383251264836
INFO:root:val ap: 0.8689625
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.41s
INFO:root:Epoch mean loss: 0.36249289482831953
INFO:root:val auc: 0.8649418004120646
INFO:root:val ap: 0.835725
DEBUG:root:-ws = 122
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.40s
INFO:root:Epoch mean loss: 0.3795012265443802
INFO:root:val auc: 0.8550835261198685
INFO:root:val ap: 0.834225
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.37s
INFO:root:Epoch mean loss: 0.35670317262411116
INFO:root:val auc: 0.8571725816476352
INFO:root:val ap: 0.846975
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.38s
INFO:root:Epoch mean loss: 0.3598775307337443
INFO:root:val auc: 0.8784374039959257
INFO:root:val ap: 0.8561000000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.34s
INFO:root:Epoch mean loss: 0.36038096696138383
INFO:root:val auc: 0.8594430902292025
INFO:root:val ap: 0.838475
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.39s
INFO:root:Epoch mean loss: 0.36391469955444333
INFO:root:val auc: 0.8575324400834499
```

```
INFO:root:val ap: 0.853775
DEBUG:root:-ws = 123
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.41s
INFO:root:Epoch mean loss: 0.38957885801792147
INFO:root:val auc: 0.8465492607744067
INFO:root:val ap: 0.8301749999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.41s
INFO:root:Epoch mean loss: 0.37701243460178374
INFO:root:val auc: 0.8725687191639167
INFO:root:val ap: 0.8426
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.35s
INFO:root:Epoch mean loss: 0.370086869597435
INFO:root:val auc: 0.8463279634362182
INFO:root:val ap: 0.837825
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.35s
INFO:root:Epoch mean loss: 0.3679609358310699
INFO:root:val auc: 0.8873735190628589
INFO:root:val ap: 0.8622249999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.41s
INFO:root:Epoch mean loss: 0.3611508721113205
INFO:root:val auc: 0.8605547011182978
INFO:root:val ap: 0.8564750000000001
DEBUG:root:-ws = 124
DEBUG:root:--epoch = 0


from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
```

```
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635285856.5109835.log')
plot_path = str(base_path / 'plot/1635285856.5109835.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]
```

```
loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)

fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)

plot_path
```

window slides from 240 - 246 (This is equivalent to window slides from 120 - 123 in `=Reserach Log 5=` where window slide is 2 times higher than `=Research Log 4=`)

```
DEBUG:root:-ws = 240
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.45s
INFO:root:Epoch mean loss: 0.5096345860511065
INFO:root:val auc: 0.8862982154401386
INFO:root:val ap: 0.8689000000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 6.64s
INFO:root:Epoch mean loss: 0.5619527790695429
INFO:root:val auc: 0.8856094266898602
INFO:root:val ap: 0.8709250000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.74s
```

```
INFO:root:Epoch mean loss: 0.5478687596817812
INFO:root:val auc: 0.8642435667192068
INFO:root:val ap: 0.8506874999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 7.05s
INFO:root:Epoch mean loss: 0.5540620340034366
INFO:root:val auc: 0.8806385864619679
INFO:root:val ap: 0.8685749999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 7.49s
INFO:root:Epoch mean loss: 0.5430546270683407
INFO:root:val auc: 0.878520421990195
INFO:root:val ap: 0.8692000000000001
DEBUG:root:-ws = 241
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.71s
INFO:root:Epoch mean loss: 0.5042583476752043
INFO:root:val auc: 0.8850654841130015
INFO:root:val ap: 0.8748500000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 7.63s
INFO:root:Epoch mean loss: 0.5228401487693191
INFO:root:val auc: 0.8718922781467886
INFO:root:val ap: 0.8648125
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 7.59s
INFO:root:Epoch mean loss: 0.5114204228545228
INFO:root:val auc: 0.8767899695829481
INFO:root:val ap: 0.8600749999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.67s
INFO:root:Epoch mean loss: 0.5147292001638561
INFO:root:val auc: 0.8683341803271313
INFO:root:val ap: 0.8487624999999999
```

```
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.50s
INFO:root:Epoch mean loss: 0.5227945471182466
INFO:root:val auc: 0.8913919575902094
INFO:root:val ap: 0.873875
DEBUG:root:-ws = 242
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 6.70s
INFO:root:Epoch mean loss: 0.5169753022491932
INFO:root:val auc: 0.8700364502066325
INFO:root:val ap: 0.851875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 7.08s
INFO:root:Epoch mean loss: 0.5039303083904088
INFO:root:val auc: 0.870125859872114
INFO:root:val ap: 0.86135
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.58s
INFO:root:Epoch mean loss: 0.5113234653448065
INFO:root:val auc: 0.8785753075409466
INFO:root:val ap: 0.8653875000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.75s
INFO:root:Epoch mean loss: 0.502818655455485
INFO:root:val auc: 0.8976865596768346
INFO:root:val ap: 0.8787500000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.69s
INFO:root:Epoch mean loss: 0.5251276953145861
INFO:root:val auc: 0.8760274423737942
INFO:root:val ap: 0.8673250000000001
DEBUG:root:-ws = 243
DEBUG:root:--epoch = 0
INFO:root:start validation...
```

```
INFO:root:epoch: 0 took 7.20s
INFO:root:Epoch mean loss: 0.4996498618274927
INFO:root:val auc: 0.8853774610871264
INFO:root:val ap: 0.8696875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 7.08s
INFO:root:Epoch mean loss: 0.5147803849540651
INFO:root:val auc: 0.8879660319579803
INFO:root:val ap: 0.8658999999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.40s
INFO:root:Epoch mean loss: 0.5214052181690931
INFO:root:val auc: 0.8840315817655899
INFO:root:val ap: 0.867075
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.72s
INFO:root:Epoch mean loss: 0.5155797195620835
INFO:root:val auc: 0.8857779166002606
INFO:root:val ap: 0.8733500000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 7.35s
INFO:root:Epoch mean loss: 0.520146269351244
INFO:root:val auc: 0.8908891032659161
INFO:root:val ap: 0.86715
DEBUG:root:-ws = 244
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 6.86s
INFO:root:Epoch mean loss: 0.4736635033041239
INFO:root:val auc: 0.9024234954625925
INFO:root:val ap: 0.884175
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 6.44s
INFO:root:Epoch mean loss: 0.48307386273518205
INFO:root:val auc: 0.8913249450226426
```

```
INFO:root:val ap: 0.880725
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.89s
INFO:root:Epoch mean loss: 0.48893146123737097
INFO:root:val auc: 0.8965780326825642
INFO:root:val ap: 0.8827625
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.69s
INFO:root:Epoch mean loss: 0.49534035567194223
INFO:root:val auc: 0.8881958224908626
INFO:root:val ap: 0.8713625000000002
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.73s
INFO:root:Epoch mean loss: 0.4996651880443096
INFO:root:val auc: 0.8847179465032524
INFO:root:val ap: 0.874625
DEBUG:root:-ws = 245
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.47s
INFO:root:Epoch mean loss: 0.5653141112998128
INFO:root:val auc: 0.8712817447556112
INFO:root:val ap: 0.8525625
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 6.70s
INFO:root:Epoch mean loss: 0.5147116822190583
INFO:root:val auc: 0.8862973717154333
INFO:root:val ap: 0.8672125
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 7.16s
INFO:root:Epoch mean loss: 0.4766196177030603
INFO:root:val auc: 0.8476972105905733
INFO:root:val ap: 0.8466750000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
```

```
INFO:root:epoch: 3 took 7.30s
INFO:root:Epoch mean loss: 0.4671670999377966
INFO:root:val auc: 0.8726264786659991
INFO:root:val ap: 0.8529749999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 7.19s
INFO:root:Epoch mean loss: 0.4689589524641633
INFO:root:val auc: 0.9019660221202399
INFO:root:val ap: 0.878275
DEBUG:root:-ws = 246
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 6.82s
INFO:root:Epoch mean loss: 0.5146969771012664
INFO:root:val auc: 0.8917710918215933
INFO:root:val ap: 0.8771249999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 7.06s
INFO:root:Epoch mean loss: 0.508500530384481
INFO:root:val auc: 0.881755105482315
INFO:root:val ap: 0.8609249999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 6.71s
INFO:root:Epoch mean loss: 0.5011651267608007
INFO:root:val auc: 0.8969738466381543
INFO:root:val ap: 0.8782249999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 6.83s
INFO:root:Epoch mean loss: 0.5109477767255157
INFO:root:val auc: 0.8690930186174426
INFO:root:val ap: 0.8527874999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 6.99s
INFO:root:Epoch mean loss: 0.5168606143444776
INFO:root:val auc: 0.8931484288605691
```

```
INFO:root:val ap: 0.873625
DEBUG:root:-ws = 247
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 7.04s
INFO:root:Epoch mean loss: 0.5149209666997194
INFO:root:val auc: 0.9004014624487391
INFO:root:val ap: 0.8794875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 6.49s
INFO:root:Epoch mean loss: 0.5311510358005762
INFO:root:val auc: 0.8934856160178621
INFO:root:val ap: 0.8775875000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 7.06s
INFO:root:Epoch mean loss: 0.5506821464126309
INFO:root:val auc: 0.8969435375981248
INFO:root:val ap: 0.883225
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 8.67s
INFO:root:Epoch mean loss: 0.5448547655250877
INFO:root:val auc: 0.8996237187702545
INFO:root:val ap: 0.876525
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 7.93s
INFO:root:Epoch mean loss: 0.5380803488194943
INFO:root:val auc: 0.8926365397897882
INFO:root:val ap: 0.875125
```

6.4.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?

- Are graphs appropriately labeled and legible?

For the first 31 window slides, performance quickly increase from around 70 to 80-85 % on auc and ap. There exists no clear trend to indicate that model can learn.

But letting the program run up to 123 windows slide, the performance at this window slide sustaintially more than performance of the first 31 window slides.

But letting the program run up to 246 windows slide (246 window slides of `=Research Log 4=` is equivalent to 123 windows slides of `=Research Log 5=`), the performance at this window slide sustaintially more than performance of the first 123 window slides.

In conclusion, it shows that the models learn to predict better as window slides forward.

6.5 Research Log 5: similar to research log 4 except that the window slides by 20 instances.

6.5.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Experiment is suggested by dr zhu in this email.
Parameters configuration are as followed.

- epoch = 5
- `~BATCH_SIZE~` = 200
- size of validation is `~BATCH_SIZE~`
- window slide by 20
- size of initial training set = 673 instances (which is 0.1 percent of training set)

6.5.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Results are produced by following command (command is the same because I modified the code without changning command line options)

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

window slide from 0 - 5

```
INFO:root:run = 0
DEBUG:root:-ws = 0
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.96s
INFO:root:Epoch mean loss: 1.373808354139328
INFO:root:val auc: 0.665561098684202
INFO:root:val ap: 0.6912750000000001
INFO:root:epoch: 1 took 2.59s
INFO:root:Epoch mean loss: 1.1486801132559776
INFO:root:val auc: 0.784134403742594
INFO:root:val ap: 0.794075
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.54s
INFO:root:Epoch mean loss: 1.138505185643832
INFO:root:val auc: 0.8272550166687205
INFO:root:val ap: 0.824075
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.61s
INFO:root:Epoch mean loss: 1.125243417918682
INFO:root:val auc: 0.7892550180643526
```

```
INFO:root:val ap: 0.7957000000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.56s
INFO:root:Epoch mean loss: 1.1177918136119842
INFO:root:val auc: 0.7921565318891389
INFO:root:val ap: 0.799275
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.50s
INFO:root:Epoch mean loss: 1.0926084518432617
INFO:root:val auc: 0.8147188311035067
INFO:root:val ap: 0.8289500000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.60s
INFO:root:Epoch mean loss: 1.0779781714081764
INFO:root:val auc: 0.789629226901583
INFO:root:val ap: 0.8033625000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.56s
INFO:root:Epoch mean loss: 1.0687131931384404
INFO:root:val auc: 0.8372930353350166
INFO:root:val ap: 0.834875
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.61s
INFO:root:Epoch mean loss: 1.0750660710036755
INFO:root:val auc: 0.8285080704039063
INFO:root:val ap: 0.82985
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.55s
INFO:root:Epoch mean loss: 1.06872196495533
INFO:root:val auc: 0.8814538823715338
INFO:root:val ap: 0.8625750000000001
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
```

```
INFO:root:start validation...
INFO:root:epoch: 0 took 2.55s
INFO:root:Epoch mean loss: 1.012108102440834
INFO:root:val auc: 0.7983235222552066
INFO:root:val ap: 0.8281625
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.52s
INFO:root:Epoch mean loss: 1.0274480804800987
INFO:root:val auc: 0.8323616248311384
INFO:root:val ap: 0.8372625
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.61s
INFO:root:Epoch mean loss: 1.0233630935351055
INFO:root:val auc: 0.8334778476791689
INFO:root:val ap: 0.820925
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.59s
INFO:root:Epoch mean loss: 1.023796670138836
INFO:root:val auc: 0.7967995655358606
INFO:root:val ap: 0.8110124999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.58s
INFO:root:Epoch mean loss: 1.0241481959819794
INFO:root:val auc: 0.7880332234600147
INFO:root:val ap: 0.81155
DEBUG:root:-ws = 4
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.55s
INFO:root:Epoch mean loss: 0.9803871065378189
INFO:root:val auc: 0.8135372924619606
INFO:root:val ap: 0.8382250000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.67s
INFO:root:Epoch mean loss: 0.9937547370791435
```

```
INFO:root:val auc: 0.825041850979637
INFO:root:val ap: 0.828875
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.57s
INFO:root:Epoch mean loss: 0.9903728912274042
INFO:root:val auc: 0.8320450104922785
INFO:root:val ap: 0.8235249999999998
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 2.62s
INFO:root:Epoch mean loss: 0.9885445386171341
INFO:root:val auc: 0.7765998774123382
INFO:root:val ap: 0.8076749999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.59s
INFO:root:Epoch mean loss: 0.9845545679330826
INFO:root:val auc: 0.8229747285734814
INFO:root:val ap: 0.8318249999999999
DEBUG:root:-ws = 5
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.59s
INFO:root:Epoch mean loss: 0.9676448404788971
INFO:root:val auc: 0.8174855524102805
INFO:root:val ap: 0.8204750000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.61s
INFO:root:Epoch mean loss: 0.9630855321884155
INFO:root:val auc: 0.8136024398466474
INFO:root:val ap: 0.8224000000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.62s
INFO:root:Epoch mean loss: 0.9628886232773463
INFO:root:val auc: 0.785081101097099
INFO:root:val ap: 0.81945
DEBUG:root:--epoch = 3
```

```
INFO:root:start validation...
INFO:root:epoch: 3 took 2.59s
INFO:root:Epoch mean loss: 0.9618354290723801
INFO:root:val auc: 0.8698816010463123
INFO:root:val ap: 0.849575
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.59s
INFO:root:Epoch mean loss: 0.9541525483131409
INFO:root:val auc: 0.8064313678090249
INFO:root:val ap: 0.8146250000000002
```

window slide from 29 - 31

```
DEBUG:root:-ws = 29
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 2.94s
INFO:root:Epoch mean loss: 0.6438962987491063
INFO:root:val auc: 0.8267968509263952
INFO:root:val ap: 0.8247749999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 2.98s
INFO:root:Epoch mean loss: 0.6366550539221082
INFO:root:val auc: 0.8635386633395387
INFO:root:val ap: 0.8584499999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 2.96s
INFO:root:Epoch mean loss: 0.6210107434363592
INFO:root:val auc: 0.846588709975499
INFO:root:val ap: 0.8372499999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.00s
INFO:root:Epoch mean loss: 0.6166786445038659
INFO:root:val auc: 0.8255160396626551
INFO:root:val ap: 0.8377
DEBUG:root:--epoch = 4
INFO:root:start validation...
```

```
INFO:root:epoch: 4 took 3.08s
INFO:root:Epoch mean loss: 0.610985847881862
INFO:root:val auc: 0.8277437240753927
INFO:root:val ap: 0.8170249999999999
DEBUG:root:-ws = 30
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.09s
INFO:root:Epoch mean loss: 0.6015564927033016
INFO:root:val auc: 0.7928005291151575
INFO:root:val ap: 0.8044250000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.11s
INFO:root:Epoch mean loss: 0.5912862462656838
INFO:root:val auc: 0.8142441705264007
INFO:root:val ap: 0.816425
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.05s
INFO:root:Epoch mean loss: 0.5832368447667077
INFO:root:val auc: 0.8179434767619138
INFO:root:val ap: 0.8249749999999999
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.07s
INFO:root:Epoch mean loss: 0.5829793087073735
INFO:root:val auc: 0.8363408897006763
INFO:root:val ap: 0.844175
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 2.97s
INFO:root:Epoch mean loss: 0.5737274093287331
INFO:root:val auc: 0.814306648912961
INFO:root:val ap: 0.826125
DEBUG:root:-ws = 31
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.06s
INFO:root:Epoch mean loss: 0.5809557821069445
```

```
INFO:root:val auc: 0.8284858915141013
INFO:root:val ap: 0.8310749999999999
```

widows slide from 120 - 123

```
DEBUG:root:-ws = 120
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 4.13s
INFO:root:Epoch mean loss: 0.57504802942276
INFO:root:val auc: 0.8703298473178701
INFO:root:val ap: 0.8731625000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 4.09s
INFO:root:Epoch mean loss: 0.5430253874510527
INFO:root:val auc: 0.8860511154171905
INFO:root:val ap: 0.8769250000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 4.20s
INFO:root:Epoch mean loss: 0.5457504652440548
INFO:root:val auc: 0.8583491807344695
INFO:root:val ap: 0.8505750000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 4.02s
INFO:root:Epoch mean loss: 0.5526332529261708
INFO:root:val auc: 0.8706971551532664
INFO:root:val ap: 0.8527999999999999
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 4.06s
INFO:root:Epoch mean loss: 0.5835676431655884
INFO:root:val auc: 0.860246974650442
INFO:root:val ap: 0.8475
DEBUG:root:-ws = 121
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 4.09s
INFO:root:Epoch mean loss: 0.5604596864432096
```

```
INFO:root:val auc: 0.8575781258445585
INFO:root:val ap: 0.8654000000000002
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 4.30s
INFO:root:Epoch mean loss: 0.5774283641949296
INFO:root:val auc: 0.8960500463546441
INFO:root:val ap: 0.8852249999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 4.06s
INFO:root:Epoch mean loss: 0.5594887205710014
INFO:root:val auc: 0.8546231374230625
INFO:root:val ap: 0.8612875
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 4.03s
INFO:root:Epoch mean loss: 0.5528432731516659
INFO:root:val auc: 0.8461750046242986
INFO:root:val ap: 0.8504375
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 4.18s
INFO:root:Epoch mean loss: 0.5562572464346885
INFO:root:val auc: 0.9023323848861673
INFO:root:val ap: 0.884425
DEBUG:root:-ws = 122
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 4.10s
INFO:root:Epoch mean loss: 0.5503575094044209
INFO:root:val auc: 0.9019077297070347
INFO:root:val ap: 0.8854249999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 4.07s
INFO:root:Epoch mean loss: 0.6294386070221663
INFO:root:val auc: 0.8569027338344009
INFO:root:val ap: 0.854
DEBUG:root:--epoch = 2
```

```
INFO:root:start validation...
INFO:root:epoch: 2 took 4.14s
INFO:root:Epoch mean loss: 0.5774525546779236
INFO:root:val auc: 0.8990420353438267
INFO:root:val ap: 0.878325
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 4.08s
INFO:root:Epoch mean loss: 0.5907126846723258
INFO:root:val auc: 0.8877898901336683
INFO:root:val ap: 0.8704000000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 4.07s
INFO:root:Epoch mean loss: 0.5966454695910215
INFO:root:val auc: 0.8398419282383518
INFO:root:val ap: 0.8473124999999999
DEBUG:root:-ws = 123
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 4.04s
INFO:root:Epoch mean loss: 0.5103132352232933
INFO:root:val auc: 0.8774551924851247
INFO:root:val ap: 0.8809124999999999

from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635276458.063925.log')
plot_path = str(base_path / 'plot/1635276458.063925.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1
```

```
with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])
```

```
# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)

fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)
# plt.show()

plot_path
```

6.5.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

For the first 31 window slides, performance quickly increase from around 70 to 80-85 % on auc and ap. There exists no clear trend to indicate that model can learn.

But letting the program run up to 123 windows slide, the performance at this window slide sustaintially more than performance of the first 31 window slides.

In conclusion, it shows that the models learn to predict better as window slides forward.

6.5.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

To clearly see trend of model performance increase window slide wise, plotting is required.

6.6 Research Log 6: similar to research log 4 except that the validation set size is 2000 instances.

6.6.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Experiment is suggested by dr zhu in this email.
Parameters configuration are as followed.

- epoch = 5
- `~BATCH_SIZE~` = 200
- size of validation is 2000 (`~BATCH_SIZE~` * 10)
- window slide by 10
- size of initial training set = 673 instances (which is 0.1 percent of training set)

6.6.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Results are produced by following command (command is the same because I modified the code without changning command line options)

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_ru
```

window slides from 0 - 3

```
INFO:root:start validation...
INFO:root:epoch: 2 took 3.28s
INFO:root:Epoch mean loss: 1.3115784724553425
INFO:root:val auc: 0.7474429977090162
INFO:root:val ap: 0.725465375
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.63s
INFO:root:Epoch mean loss: 1.282339170575142
INFO:root:val auc: 0.731949122687042
INFO:root:val ap: 0.715311
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.34s
INFO:root:Epoch mean loss: 1.2582542896270752
INFO:root:val auc: 0.7476566541859365
INFO:root:val ap: 0.7357467499999999
DEBUG:root:-ws = 1
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.25s
INFO:root:Epoch mean loss: 1.1421709060668945
INFO:root:val auc: 0.7615665211320573
```

```
INFO:root:val ap: 0.743176875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.22s
INFO:root:Epoch mean loss: 1.1598860621452332
INFO:root:val auc: 0.762054911167359
INFO:root:val ap: 0.74255475
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.17s
INFO:root:Epoch mean loss: 1.1455095609029133
INFO:root:val auc: 0.770884377598583
INFO:root:val ap: 0.7581475
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.40s
INFO:root:Epoch mean loss: 1.1284798830747604
INFO:root:val auc: 0.7737840479019044
INFO:root:val ap: 0.765745
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.58s
INFO:root:Epoch mean loss: 1.1209552884101868
INFO:root:val auc: 0.792245455787415
INFO:root:val ap: 0.7770835
DEBUG:root:-ws = 2
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.43s
INFO:root:Epoch mean loss: 1.064228430390358
INFO:root:val auc: 0.7869888060810495
INFO:root:val ap: 0.794144875
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.44s
INFO:root:Epoch mean loss: 1.0667132139205933
INFO:root:val auc: 0.7986790013881927
INFO:root:val ap: 0.795262625
DEBUG:root:--epoch = 2
INFO:root:start validation...
```

```
INFO:root:epoch: 2 took 3.29s
INFO:root:Epoch mean loss: 1.0590859750906627
INFO:root:val auc: 0.7889203854466766
INFO:root:val ap: 0.781478125
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.31s
INFO:root:Epoch mean loss: 1.0455990433692932
INFO:root:val auc: 0.7763797858193688
INFO:root:val ap: 0.7799875000000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.33s
INFO:root:Epoch mean loss: 1.0429620385169982
INFO:root:val auc: 0.7921042302939977
INFO:root:val ap: 0.789086375
DEBUG:root:-ws = 3
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.41s
INFO:root:Epoch mean loss: 1.000748634338379
INFO:root:val auc: 0.7828607538223339
INFO:root:val ap: 0.786933375
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.31s
INFO:root:Epoch mean loss: 1.0044655501842499
INFO:root:val auc: 0.7749504527317143
INFO:root:val ap: 0.778422
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.45s
INFO:root:Epoch mean loss: 1.0033698678016663
INFO:root:val auc: 0.7780766195005349
INFO:root:val ap: 0.7793961250000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.24s
INFO:root:Epoch mean loss: 1.0033747851848602
INFO:root:val auc: 0.7804256952527345
```

```
INFO:root:val ap: 0.7824841250000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.53s
INFO:root:Epoch mean loss: 0.9975038766860962
INFO:root:val auc: 0.7751902146067262
INFO:root:val ap: 0.781201375
```

window slide from 34 - 38

```
DEBUG:root:-ws = 34
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.77s
INFO:root:Epoch mean loss: 0.7193483859300613
INFO:root:val auc: 0.7763243270326323
INFO:root:val ap: 0.7792365000000001
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.94s
INFO:root:Epoch mean loss: 0.701469120879968
INFO:root:val auc: 0.7522057484776283
INFO:root:val ap: 0.7561846249999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.59s
INFO:root:Epoch mean loss: 0.6932896905475192
INFO:root:val auc: 0.7549391418591935
INFO:root:val ap: 0.76490525
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.78s
INFO:root:Epoch mean loss: 0.6828512648741404
INFO:root:val auc: 0.7967860239596929
INFO:root:val ap: 0.7997602500000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.79s
INFO:root:Epoch mean loss: 0.6700180724263192
INFO:root:val auc: 0.7843179958299468
INFO:root:val ap: 0.7889215
```

```
DEBUG:root:-ws = 35
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.94s
INFO:root:Epoch mean loss: 0.7037995755672455
INFO:root:val auc: 0.7747758065862718
INFO:root:val ap: 0.7893804999999999
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.71s
INFO:root:Epoch mean loss: 0.7018633435169855
INFO:root:val auc: 0.80223110965111
INFO:root:val ap: 0.80935375
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.75s
INFO:root:Epoch mean loss: 0.6831860674752129
INFO:root:val auc: 0.8006861753579384
INFO:root:val ap: 0.8091645000000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.78s
INFO:root:Epoch mean loss: 0.671836165090402
INFO:root:val auc: 0.795143916072075
INFO:root:val ap: 0.801453875
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.68s
INFO:root:Epoch mean loss: 0.6552501966555914
INFO:root:val auc: 0.800620523314914
INFO:root:val ap: 0.806810875
DEBUG:root:-ws = 36
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.94s
INFO:root:Epoch mean loss: 0.6153814097245535
INFO:root:val auc: 0.7815555669398756
INFO:root:val ap: 0.792893
DEBUG:root:--epoch = 1
INFO:root:start validation...
```

```
INFO:root:epoch: 1 took 3.77s
INFO:root:Epoch mean loss: 0.6198568940162659
INFO:root:val auc: 0.7884020985264713
INFO:root:val ap: 0.7931170000000001
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 4.05s
INFO:root:Epoch mean loss: 0.6175177031093173
INFO:root:val auc: 0.783006104363591
INFO:root:val ap: 0.7932618750000001
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.77s
INFO:root:Epoch mean loss: 0.616212065021197
INFO:root:val auc: 0.7934862287953195
INFO:root:val ap: 0.8035486250000001
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.95s
INFO:root:Epoch mean loss: 0.6162412305672963
INFO:root:val auc: 0.7900001329205116
INFO:root:val ap: 0.799147625
DEBUG:root:-ws = 37
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.73s
INFO:root:Epoch mean loss: 0.5840299328168234
INFO:root:val auc: 0.7977117603700505
INFO:root:val ap: 0.803559125
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.59s
INFO:root:Epoch mean loss: 0.5758757367730141
INFO:root:val auc: 0.8091223446898834
INFO:root:val ap: 0.8111079999999999
DEBUG:root:--epoch = 2
INFO:root:start validation...
INFO:root:epoch: 2 took 3.70s
INFO:root:Epoch mean loss: 0.5644812666707568
INFO:root:val auc: 0.7842730038183883
```

```
INFO:root:val ap: 0.79537775
DEBUG:root:--epoch = 3
INFO:root:start validation...
INFO:root:epoch: 3 took 3.57s
INFO:root:Epoch mean loss: 0.5500421561300755
INFO:root:val auc: 0.798493973760281
INFO:root:val ap: 0.800874125
DEBUG:root:--epoch = 4
INFO:root:start validation...
INFO:root:epoch: 4 took 3.57s
INFO:root:Epoch mean loss: 0.5534079144398372
INFO:root:val auc: 0.7825338743661722
INFO:root:val ap: 0.7932275
DEBUG:root:-ws = 38
DEBUG:root:--epoch = 0
INFO:root:start validation...
INFO:root:epoch: 0 took 3.89s
INFO:root:Epoch mean loss: 0.5583125700553259
INFO:root:val auc: 0.7792751071423917
INFO:root:val ap: 0.7917055
DEBUG:root:--epoch = 1
INFO:root:start validation...
INFO:root:epoch: 1 took 3.75s
INFO:root:Epoch mean loss: 0.5634230698148409
INFO:root:val auc: 0.781136
```

window slides fom 120 - 123
window slides from 240 - 246
window slides from 395 - 400

```
2021-10-27 05:45:49,289 - root - DEBUG - -ws = 395
2021-10-27 05:45:49,289 - root - DEBUG - --epoch = 0
2021-10-27 05:45:52,193 - root - INFO - start validation...
2021-10-27 05:45:54,776 - root - INFO - epoch: 0 took 5.49s
2021-10-27 05:45:54,776 - root - INFO - Epoch mean loss: 0.31383699427048367
2021-10-27 05:45:54,776 - root - INFO - val auc: 0.9119400921707221
2021-10-27 05:45:54,776 - root - INFO - val ap: 0.9003925
2021-10-27 05:45:54,777 - root - DEBUG - --epoch = 1
2021-10-27 05:45:57,749 - root - INFO - start validation...
2021-10-27 05:46:00,453 - root - INFO - epoch: 1 took 5.68s
2021-10-27 05:46:00,453 - root - INFO - Epoch mean loss: 0.31902350578457117
```

```
2021-10-27 05:46:00,454 - root - INFO - val auc: 0.8985237688723426
2021-10-27 05:46:00,454 - root - INFO - val ap: 0.88670925
2021-10-27 05:46:00,454 - root - DEBUG - --epoch = 2
2021-10-27 05:46:03,387 - root - INFO - start validation...
2021-10-27 05:46:06,095 - root - INFO - epoch: 2 took 5.64s
2021-10-27 05:46:06,095 - root - INFO - Epoch mean loss: 0.30693080379731125
2021-10-27 05:46:06,095 - root - INFO - val auc: 0.9018421929987471
2021-10-27 05:46:06,096 - root - INFO - val ap: 0.88649125
2021-10-27 05:46:06,096 - root - DEBUG - --epoch = 3
2021-10-27 05:46:09,055 - root - INFO - start validation...
2021-10-27 05:46:11,685 - root - INFO - epoch: 3 took 5.59s
2021-10-27 05:46:11,686 - root - INFO - Epoch mean loss: 0.2961217219320436
2021-10-27 05:46:11,686 - root - INFO - val auc: 0.8973219907687386
2021-10-27 05:46:11,686 - root - INFO - val ap: 0.881797125
2021-10-27 05:46:11,686 - root - DEBUG - --epoch = 4
2021-10-27 05:46:14,649 - root - INFO - start validation...
2021-10-27 05:46:17,309 - root - INFO - epoch: 4 took 5.62s
2021-10-27 05:46:17,310 - root - INFO - Epoch mean loss: 0.29265690594911575
2021-10-27 05:46:17,310 - root - INFO - val auc: 0.9019378695675153
2021-10-27 05:46:17,310 - root - INFO - val ap: 0.8898928749999999
2021-10-27 05:46:17,407 - root - DEBUG - -ws = 396
2021-10-27 05:46:17,407 - root - DEBUG - --epoch = 0
2021-10-27 05:46:20,335 - root - INFO - start validation...
2021-10-27 05:46:22,982 - root - INFO - epoch: 0 took 5.58s
2021-10-27 05:46:22,982 - root - INFO - Epoch mean loss: 0.3291622058798869
2021-10-27 05:46:22,982 - root - INFO - val auc: 0.9026938195308385
2021-10-27 05:46:22,982 - root - INFO - val ap: 0.8900005000000002
2021-10-27 05:46:22,982 - root - DEBUG - --epoch = 1
2021-10-27 05:46:25,970 - root - INFO - start validation...
2021-10-27 05:46:28,595 - root - INFO - epoch: 1 took 5.61s
2021-10-27 05:46:28,596 - root - INFO - Epoch mean loss: 0.31689996365457773
2021-10-27 05:46:28,596 - root - INFO - val auc: 0.8907106891278899
2021-10-27 05:46:28,596 - root - INFO - val ap: 0.87687475
2021-10-27 05:46:28,596 - root - DEBUG - --epoch = 2
2021-10-27 05:46:31,542 - root - INFO - start validation...
2021-10-27 05:46:34,160 - root - INFO - epoch: 2 took 5.56s
2021-10-27 05:46:34,160 - root - INFO - Epoch mean loss: 0.3116990331974294
2021-10-27 05:46:34,161 - root - INFO - val auc: 0.9040530702305996
2021-10-27 05:46:34,161 - root - INFO - val ap: 0.8888640000000001
2021-10-27 05:46:34,161 - root - DEBUG - --epoch = 3
```

```
2021-10-27 05:46:37,074 - root - INFO - start validation...
2021-10-27 05:46:39,655 - root - INFO - epoch: 3 took 5.49s
2021-10-27 05:46:39,655 - root - INFO - Epoch mean loss: 0.31238439423032105
2021-10-27 05:46:39,656 - root - INFO - val auc: 0.901865620099245
2021-10-27 05:46:39,656 - root - INFO - val ap: 0.88608925
2021-10-27 05:46:39,656 - root - DEBUG - --epoch = 4
2021-10-27 05:46:42,619 - root - INFO - start validation...
2021-10-27 05:46:45,246 - root - INFO - epoch: 4 took 5.59s
2021-10-27 05:46:45,246 - root - INFO - Epoch mean loss: 0.30935903197775283
2021-10-27 05:46:45,246 - root - INFO - val auc: 0.895542455475056
2021-10-27 05:46:45,247 - root - INFO - val ap: 0.8808800000000001
2021-10-27 05:46:45,337 - root - DEBUG - -ws = 397
2021-10-27 05:46:45,338 - root - DEBUG - --epoch = 0
2021-10-27 05:46:48,197 - root - INFO - start validation...
2021-10-27 05:46:50,865 - root - INFO - epoch: 0 took 5.53s
2021-10-27 05:46:50,866 - root - INFO - Epoch mean loss: 0.3225811750938495
2021-10-27 05:46:50,866 - root - INFO - val auc: 0.9049911877659891
2021-10-27 05:46:50,866 - root - INFO - val ap: 0.889537
2021-10-27 05:46:50,866 - root - DEBUG - --epoch = 1
2021-10-27 05:46:53,728 - root - INFO - start validation...
2021-10-27 05:46:56,242 - root - INFO - epoch: 1 took 5.38s
2021-10-27 05:46:56,243 - root - INFO - Epoch mean loss: 0.3133492503936092
2021-10-27 05:46:56,243 - root - INFO - val auc: 0.9020607337059487
2021-10-27 05:46:56,243 - root - INFO - val ap: 0.888657
2021-10-27 05:46:56,243 - root - DEBUG - --epoch = 2
2021-10-27 05:46:59,116 - root - INFO - start validation...
2021-10-27 05:47:01,724 - root - INFO - epoch: 2 took 5.48s
2021-10-27 05:47:01,724 - root - INFO - Epoch mean loss: 0.3051986125194364
2021-10-27 05:47:01,725 - root - INFO - val auc: 0.8954830561201224
2021-10-27 05:47:01,725 - root - INFO - val ap: 0.8815553749999999
2021-10-27 05:47:01,725 - root - DEBUG - --epoch = 3
2021-10-27 05:47:04,658 - root - INFO - start validation...
2021-10-27 05:47:07,257 - root - INFO - epoch: 3 took 5.53s
2021-10-27 05:47:07,258 - root - INFO - Epoch mean loss: 0.29979205240185064
2021-10-27 05:47:07,258 - root - INFO - val auc: 0.9075584756275086
2021-10-27 05:47:07,258 - root - INFO - val ap: 0.8929605
2021-10-27 05:47:07,258 - root - DEBUG - --epoch = 4
2021-10-27 05:47:10,106 - root - INFO - start validation...
2021-10-27 05:47:12,747 - root - INFO - epoch: 4 took 5.49s
2021-10-27 05:47:12,747 - root - INFO - Epoch mean loss: 0.2947026109943787
```

```
2021-10-27 05:47:12,747 - root - INFO - val auc: 0.9039429106972696
2021-10-27 05:47:12,747 - root - INFO - val ap: 0.88834075
2021-10-27 05:47:12,835 - root - DEBUG - -ws = 398
2021-10-27 05:47:12,836 - root - DEBUG - --epoch = 0
2021-10-27 05:47:15,825 - root - INFO - start validation...
2021-10-27 05:47:18,420 - root - INFO - epoch: 0 took 5.58s
2021-10-27 05:47:18,421 - root - INFO - Epoch mean loss: 0.3225757585217555
2021-10-27 05:47:18,421 - root - INFO - val auc: 0.8978924421073103
2021-10-27 05:47:18,421 - root - INFO - val ap: 0.8851232499999999
2021-10-27 05:47:18,421 - root - DEBUG - --epoch = 1
2021-10-27 05:47:21,274 - root - INFO - start validation...
2021-10-27 05:47:23,850 - root - INFO - epoch: 1 took 5.43s
2021-10-27 05:47:23,851 - root - INFO - Epoch mean loss: 0.31578218067685765
2021-10-27 05:47:23,851 - root - INFO - val auc: 0.8922695598245065
2021-10-27 05:47:23,851 - root - INFO - val ap: 0.878046875
2021-10-27 05:47:23,851 - root - DEBUG - --epoch = 2
2021-10-27 05:47:26,778 - root - INFO - start validation...
2021-10-27 05:47:29,417 - root - INFO - epoch: 2 took 5.57s
2021-10-27 05:47:29,418 - root - INFO - Epoch mean loss: 0.30303874301413697
2021-10-27 05:47:29,418 - root - INFO - val auc: 0.9065492749983068
2021-10-27 05:47:29,418 - root - INFO - val ap: 0.891265
2021-10-27 05:47:29,418 - root - DEBUG - --epoch = 3
2021-10-27 05:47:32,304 - root - INFO - start validation...
2021-10-27 05:47:34,886 - root - INFO - epoch: 3 took 5.47s
2021-10-27 05:47:34,886 - root - INFO - Epoch mean loss: 0.2974980802585681
2021-10-27 05:47:34,886 - root - INFO - val auc: 0.8968472508993649
2021-10-27 05:47:34,886 - root - INFO - val ap: 0.879466
2021-10-27 05:47:34,887 - root - DEBUG - --epoch = 4
2021-10-27 05:47:37,814 - root - INFO - start validation...
2021-10-27 05:47:40,546 - root - INFO - epoch: 4 took 5.66s
2021-10-27 05:47:40,547 - root - INFO - Epoch mean loss: 0.2925099615007639
2021-10-27 05:47:40,547 - root - INFO - val auc: 0.8918195481104685
2021-10-27 05:47:40,547 - root - INFO - val ap: 0.880657
2021-10-27 05:47:40,637 - root - DEBUG - -ws = 399
2021-10-27 05:47:40,637 - root - DEBUG - --epoch = 0
2021-10-27 05:47:43,595 - root - INFO - start validation...
2021-10-27 05:47:46,208 - root - INFO - epoch: 0 took 5.57s
2021-10-27 05:47:46,208 - root - INFO - Epoch mean loss: 0.26040757820010185
2021-10-27 05:47:46,208 - root - INFO - val auc: 0.900333905821239
2021-10-27 05:47:46,209 - root - INFO - val ap: 0.88684875
```

```
2021-10-27 05:47:46,209 - root - DEBUG - --epoch = 1
2021-10-27 05:47:49,076 - root - INFO - start validation...
2021-10-27 05:47:51,635 - root - INFO - epoch: 1 took 5.43s
2021-10-27 05:47:51,636 - root - INFO - Epoch mean loss: 0.26303904317319393
2021-10-27 05:47:51,636 - root - INFO - val auc: 0.9006869254482733
2021-10-27 05:47:51,636 - root - INFO - val ap: 0.8867603750000002
2021-10-27 05:47:51,636 - root - DEBUG - --epoch = 2
2021-10-27 05:47:54,541 - root - INFO - start validation...
2021-10-27 05:47:57,110 - root - INFO - epoch: 2 took 5.47s
2021-10-27 05:47:57,111 - root - INFO - Epoch mean loss: 0.2616177995999654
2021-10-27 05:47:57,111 - root - INFO - val auc: 0.8913721479854231
2021-10-27 05:47:57,111 - root - INFO - val ap: 0.87842175
2021-10-27 05:47:57,111 - root - DEBUG - --epoch = 3
2021-10-27 05:48:00,005 - root - INFO - start validation...
2021-10-27 05:48:02,596 - root - INFO - epoch: 3 took 5.49s
2021-10-27 05:48:02,597 - root - INFO - Epoch mean loss: 0.2621427057310939
2021-10-27 05:48:02,597 - root - INFO - val auc: 0.8999058961056681
2021-10-27 05:48:02,597 - root - INFO - val ap: 0.8871543749999999
2021-10-27 05:48:02,597 - root - DEBUG - --epoch = 4
2021-10-27 05:48:05,496 - root - INFO - start validation...
2021-10-27 05:48:08,062 - root - INFO - epoch: 4 took 5.46s
2021-10-27 05:48:08,062 - root - INFO - Epoch mean loss: 0.2640071996798118
2021-10-27 05:48:08,062 - root - INFO - val auc: 0.8924547640113506
2021-10-27 05:48:08,063 - root - INFO - val ap: 0.87725525
2021-10-27 05:48:08,157 - root - DEBUG - -ws = 400
2021-10-27 05:48:08,157 - root - DEBUG - --epoch = 0
2021-10-27 05:48:11,085 - root - INFO - start validation...
2021-10-27 05:48:13,686 - root - INFO - epoch: 0 took 5.53s
2021-10-27 05:48:13,687 - root - INFO - Epoch mean loss: 0.2868427491436402
2021-10-27 05:48:13,687 - root - INFO - val auc: 0.9021711790490932
2021-10-27 05:48:13,687 - root - INFO - val ap: 0.88834325
2021-10-27 05:48:13,687 - root - DEBUG - --epoch = 1
2021-10-27 05:48:16,551 - root - INFO - start validation...
2021-10-27 05:48:19,105 - root - INFO - epoch: 1 took 5.42s
2021-10-27 05:48:19,105 - root - INFO - Epoch mean loss: 0.27642947652687627
2021-10-27 05:48:19,106 - root - INFO - val auc: 0.8936806064644667
2021-10-27 05:48:19,106 - root - INFO - val ap: 0.878453125
2021-10-27 05:48:19,106 - root - DEBUG - --epoch = 2
2021-10-27 05:48:22,026 - root - INFO - start validation...
2021-10-27 05:48:24,707 - root - INFO - epoch: 2 took 5.60s
```

```
2021-10-27 05:48:24,708 - root - INFO - Epoch mean loss: 0.2749719265848398
2021-10-27 05:48:24,708 - root - INFO - val auc: 0.9031046630325357
2021-10-27 05:48:24,708 - root - INFO - val ap: 0.8887600000000001
2021-10-27 05:48:24,708 - root - DEBUG - --epoch = 3
2021-10-27 05:48:27,850 - root - INFO - start validation...
2021-10-27 05:48:30,498 - root - INFO - epoch: 3 took 5.79s
2021-10-27 05:48:30,498 - root - INFO - Epoch mean loss: 0.2742931760537128
2021-10-27 05:48:30,499 - root - INFO - val auc: 0.9059814088025882
2021-10-27 05:48:30,499 - root - INFO - val ap: 0.890871
2021-10-27 05:48:30,499 - root - DEBUG - --epoch = 4
2021-10-27 05:48:33,427 - root - INFO - start validation...
2021-10-27 05:48:36,207 - root - INFO - epoch: 4 took 5.71s
2021-10-27 05:48:36,208 - root - INFO - Epoch mean loss: 0.2737107245872418
2021-10-27 05:48:36,208 - root - INFO - val auc: 0.9031955592223495
2021-10-27 05:48:36,208 - root - INFO - val ap: 0.8929564999999999
```

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635319704.7641783.log')
plot_path = str(base_path / 'plot/1635319704.7641783.png')


loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
```

```
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])

# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)
```

```
fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)

plot_path
```

6.6.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

performance stuck at 75-80 % on auc and ap with no clear trend to indicate that model can learn.

6.6.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

`=Research Log 6=` are similart to `=Research Log 4=`, so to make the performance perfectly comparable, I need to run up to 246 epochs.

7 Research Date: *[2021-10-26 Tue]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

7.1 Summary Log 1: `=INCORRECT OBSERVATION=` ~~**All things being equal, as number of validate size increases, number of epoch should increase otherwise model will not be able to learn and predict**~~**.**

7.1.1 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

1. first evidences

   from Research Data: *[2021-10-25 Mon]* under Research Log 4: similar to research log 2 except that the window slides by 10 instances and epoch is reduced from 50 to 5. (because we are concerned about window wise performance ratehr than epoch wise performance),

   I have assumed that epoch should be reduced from 50 to 5 to speed up the process (assuming there model can still be learning but some performance is loss). from Data and Observations section in `=Research Log 6=`, I have concluded that " observation performance of `=Research Log 4=`, `=Research Log 5=`, `=Research Log 6=`, number of epoch should increase as size of validation set increase."

2. second evidences From Research Data: *[2021-10-25 Mon]*

   Comparing `=Reserach Log 4=` and `=Research Log 6=`, although initial performance of `=Research Log 6=` is higher, (This was unexpected to me because validation set is 10 times bigger.) performance of `=Research Log 4=` plateau at 80-85 percents on auc and ap while

`=Research Log 6=` performance plateau at 75-80 percents on auc and ap.

This may be because temporal patterns aren't less useful to predict model too far into the future.

Compare `=Research Log 3=` and `=Research Log 6=` where num of epoch of `=Research Log 6=` are reduced from 50 to 5, and size of validation set is reduced from 70 times `BATCH_SIZE` to 10 times `BATCH_SIZE`, `=Research Log 3=` performance improve epoch wise as well as window sliding wise while performance of `=Research Log 6=` don't indicate that model can learn in paramaeters setting where epoch = 5 and validation set is 10 times the `BATCH_SIZE`.

In conclusion, observation performance of `=Research Log 3=`, `=Research Log 4=`, `=Research Log 6=`, number of epoch should increase as size of validation set increase.

7.2 Summary Log 2: At this point in time, I haven't investigate impact of size of window slides to model performance.

7.2.1 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

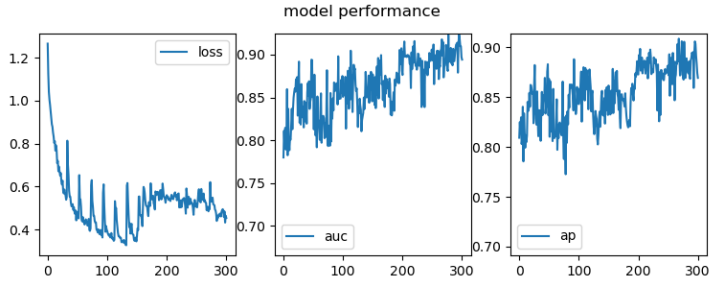
By observing `=Research Log 4=`, `=Research Log 5=` and `=Research Log6=`, I still unable to describe/detect impact of window slides to how well model can learn. This may proof to be imporant at some point in the future, but at this time of writing, I am unable to identify any effects with certainty.

8 Research Date: *[2021-10-27 Wed]*

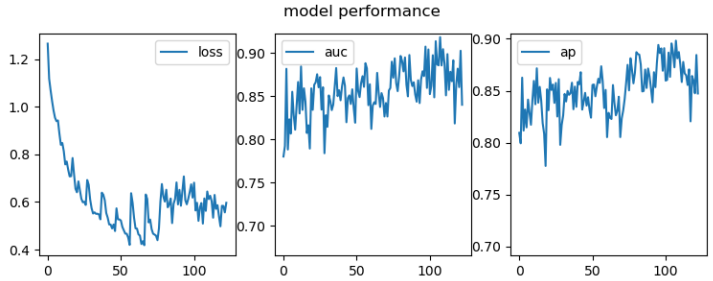
- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

8.1 [INSIGTHFUL] Summary Log 1: In Research Date: *[2021-10-25 Mon]*, Plotting of `=Research Log 4=` and `=Research 5=` hows that at first performance improve as window slides forwards. In addition, one can observed that `=Research Log 5=` takes half the time of `=Research Log 4=` to reach local minimum loss values This is because `=Research Log 5=` slides over 20 instances while `=Research Log 4=` slides over 10 instances.

`=Research Log 4=`

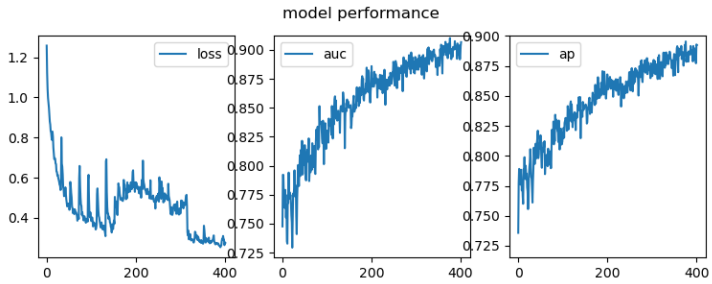


`=Research Log 5=`



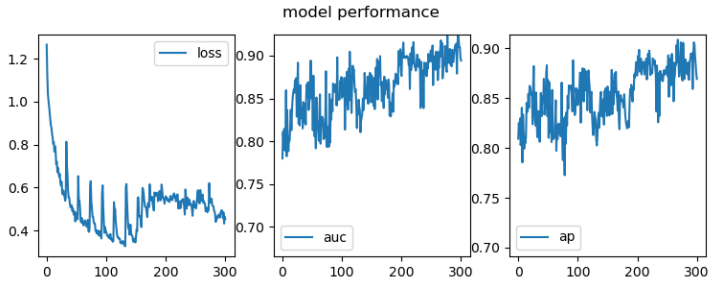
8.2 [INSIGHTFUL] Summary Log 2: In Research Date: *[2021-10-25 Mon]*, plotting of `=Researchs Log 6=` shows that loss function reaches first local minimum loss because loss value spike up for a period of time then reach even second local minimum loss (which is lower than the first one). Interestingly, auc and ap performances improve smoothly (in conflict with trend of loss values). Furthermore, one may say there during the period when loss function values spike up, auc and ap performance also spike up. (this is unexpected because loss and auc & ap should have reverse correlation.)

`=Research Log 6=`

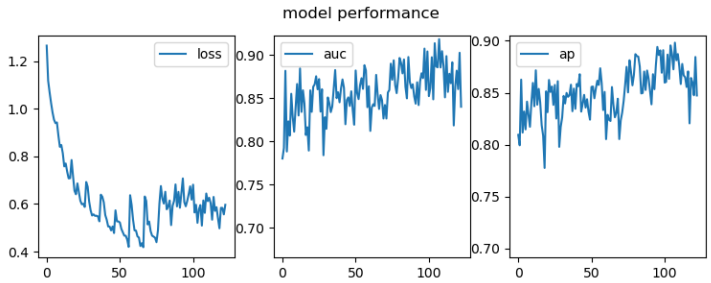


8.3 Summary Log 3: comparing `=Research Log 6=` to `=Research Log 4=` and `=Research Log 5=`, `=Research Log 6=` was trained around by sliding window 400 times to reach performance similar to `=Research Log4=` (slides over around 200 windows) and `=Research Log5=` (slides over around 100 windows) which range around 0.85 to 0.90 on boath auc and ap.

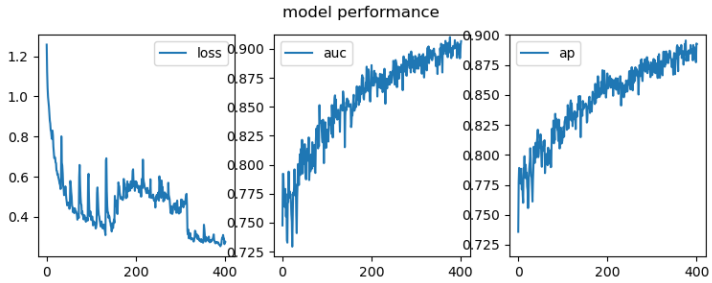
`=Research Log 4=`



=Research Log 5=



=Research Log 6=



8.4 Research Log 1: running link prediction with `train_self_supervised.py` with initial training set = 1 percent of , `BATCH_SIZE` = 200 number of epoch = 5, size validation set = `BATCH_SIZE` * 10, and size of window = `BATCH_SIZE` * 10.

8.4.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I spotted many subtle error which results in totoally incorrect evaluation process.

Parameters configuration are as followed.

- epoch = 5
- `BATCH_SIZE` = 200
- size of validation is `BATCH_SIZE` * 10

- window slide by `BATCH_SIZE` * 10
- size of initial training set = 6725 instances (which is 1 percent of all instances)

8.4.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635347571.7908711.log')
plot_path = str(base_path / 'plot/1635347571.7908711.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
```

```
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))

# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])

# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)
```

```
fig, axs = plt.subplots(1, 3, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
axs[2].plot(ws, ap[:,-1], label = 'ap')
axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
axs[1].legend()
axs[2].legend()
plt.savefig(plot_path)
# plt.show()

plot_path
```

8.5 Research Log 2: running default setup for node classification. (which use pretrain models by restoring model from checkpoint.)

8.5.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Default setting can be derived directly from parser. For this reason, I will omit re-writing the default setting here.

`=train_superivsed.py=` implements tgn model to solve node classification task while `=train_self_supervised.py=` implements tgn model to solve link prediction task.

According to my current understand of "Graph Self-Supervised Learning: A Survey", notion of "self supervised" may be abused in this context because tgn doesn't use any of the mentioned self-supervised training schemes. TGN approach of training cannot be considered as "unsupervised representation learning" because it evaluates using train-test-split (there is no freezing and unfreezing of parameters to retrain.).

On the other hands, using window sliding in place of train-test-split evaluation on default setting can be considered as "unsupervised representation learning" because each window step are equivalent of using freezed model's

parameters from the previous window as a initial parameters to train the current window.

output is generated by the following command line

```
python train_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_runs 10
```

8.5.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

I only ran a few sliding window to see that it works

```
INFO:root:Epoch 0: train loss: 0.015680748709190496, val auc: 0.6018773814583482, time
INFO:root:Epoch 1: train loss: 0.004703766344488684, val auc: 0.5976861625591623, time
```

9 Research Date: *[2021-10-29 Fri]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

9.1 [POSSIBLE MISTAKE] Research Log 1: Using train-val-test split evaluation methods, I modified node classification from using pre-train model (from link prediction) into training from scratch.

9.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

Originally, node classification was only implemented to use pre-training model only, so I modified the code to be also accept training mode from scratch.

I also observed that code for node classification tasks was modified, from my point of view, in an unnecessary ways (e.g. passing some parameters to `eval_node_classification` was unnecessary.) There are no comment that explains these strange implementation, so it is possible that I don't fully understand the code and incorrectly modified it.

I modified the code without changing command line behavior, so the same command line was run.

```
python train_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_runs 10
```

~~From what I am reading from the code, label of node classification are binary to determine whether or not the node will be connected by an edge in the next time step. Given that this is the task, performance of node classification should not be so different from link prediction (My intuition being made here could be wrong.)~~

"labels of the data is "$\text{state}_{\text{label}}$" column which has binary value where 0 if 'users are not banned after the post' otherwise 'users are banned after the post' " - Edit *<2021-11-01 Mon>*

Default Parameters configuration are as followed.

- epoch = 10
- `BATCH_SIZE` = 100
- train-test-split are 70%-15%-15%

9.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

```
2021-10-29 12:31:44,025 - root - INFO - Namespace(aggregator='last', backprop_every=1,
2021-10-29 12:31:50,857 - root - DEBUG - Num of training instances: 571580
```

```
2021-10-29 12:31:50,857 - root - DEBUG - Num of batches per epoch: 5716
2021-10-29 12:37:40,171 - root - INFO - Epoch 0: train loss: 0.015680748709190496, val
2021-10-29 12:43:25,914 - root - INFO - Epoch 1: train loss: 0.004703766344488684, val
2021-10-29 12:49:20,312 - root - INFO - Epoch 2: train loss: 0.004456446315055392, val
2021-10-29 12:55:10,235 - root - INFO - Epoch 3: train loss: 0.004356884730322363, val
2021-10-29 13:00:57,281 - root - INFO - Epoch 4: train loss: 0.004341939632570135, val
2021-10-29 13:06:49,123 - root - INFO - Epoch 5: train loss: 0.004198120632617056, val
2021-10-29 13:12:33,283 - root - INFO - Epoch 6: train loss: 0.004177507489677458, val
2021-10-29 13:18:16,126 - root - INFO - Epoch 7: train loss: 0.0041714576530825056, va
2021-10-29 13:23:57,147 - root - INFO - Epoch 8: train loss: 0.004178987974196398, val
2021-10-29 13:29:31,222 - root - INFO - Epoch 9: train loss: 0.004155165483568585, val
2021-10-29 13:29:31,223 - root - INFO - test auc: 0.5923509674471653
2021-10-29 13:29:32,158 - root - DEBUG - Num of training instances: 571580
2021-10-29 13:29:32,158 - root - DEBUG - Num of batches per epoch: 5716
2021-10-29 13:35:05,725 - root - INFO - Epoch 0: train loss: 0.013083785878262557, val
2021-10-29 13:40:39,486 - root - INFO - Epoch 1: train loss: 0.0048403758849680355, va
2021-10-29 13:46:12,806 - root - INFO - Epoch 2: train loss: 0.004630856380520815, val
2021-10-29 13:51:53,658 - root - INFO - Epoch 3: train loss: 0.004498479094219901, val
2021-10-29 13:57:37,527 - root - INFO - Epoch 4: train loss: 0.004474180551070048, val
2021-10-29 14:03:22,755 - root - INFO - Epoch 5: train loss: 0.004429378035828983, val
2021-10-29 14:09:11,014 - root - INFO - Epoch 6: train loss: 0.0043627313015625726, va
2021-10-29 14:15:12,568 - root - INFO - Epoch 7: train loss: 0.004305073541679438, val
2021-10-29 14:20:59,766 - root - INFO - Epoch 8: train loss: 0.004290463193223379, val
2021-10-29 14:27:13,071 - root - INFO - Epoch 9: train loss: 0.004272419925862897, val
2021-10-29 14:27:13,072 - root - INFO - test auc: 0.6015367697063402
2021-10-29 14:27:14,204 - root - DEBUG - Num of training instances: 571580
2021-10-29 14:27:14,204 - root - DEBUG - Num of batches per epoch: 5716
2021-10-29 14:33:37,045 - root - INFO - Epoch 0: train loss: 0.014651692523281613, val
2021-10-29 14:39:25,883 - root - INFO - Epoch 1: train loss: 0.004900022633157317, val
2021-10-29 14:45:13,549 - root - INFO - Epoch 2: train loss: 0.004660469374956713, val
2021-10-29 14:51:02,411 - root - INFO - Epoch 3: train loss: 0.004524011285374138, val
2021-10-29 14:56:46,595 - root - INFO - Epoch 4: train loss: 0.004505760545485991, val
2021-10-29 15:02:30,132 - root - INFO - Epoch 5: train loss: 0.004398693075604757, val
2021-10-29 15:08:13,608 - root - INFO - Epoch 6: train loss: 0.0043469518602649955, va
2021-10-29 15:14:01,058 - root - INFO - Epoch 7: train loss: 0.004368818391971913, val
2021-10-29 16:21:41,425 - root - INFO - Epoch 8: train loss: 0.004276090840649053, val
2021-10-29 16:30:36,942 - root - INFO - Epoch 9: train loss: 0.004251734715704296, val
```

9.1.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

Note that node classification were unfreeze from the same model that were trained for link prediction. In the other word, node classification model is pretrained while link prediction model is trained from scratch.

Node Classification Results from TGN paper.

From the results above, there was no improvment epoch wise. Furthermore auc results are not within 67.06 +- 0.9 as shown in the table 3 from TGN paper.

9.1.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

I strongly believe that I may incorrectly modified the code.

One possible mistake that I noticed was that it only took about 5-7 mins to produce results but when I ran supposedly the same model on *<2021-11-01 Mon>* overnight. The results took more than 4 hours and still haven't complete even 1 epoch.

Rough calculation of time is as followed. Each batch in an epoch takes about 0.4 sec. There are 5716 batches, so (5716 * 0.4 / 60) = 38 mins. Calculation are way off than 4 hours.

10 Research Date: *[2021-11-01 Mon]*

- ref

- https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F

10.1 Research Log 1: Sliding window for node classification with (epoch, batch size, size of validation,window slide, first window size) = (5, 100, 100, 100, 673).

10.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

About the dataset, labels of the data is from "$\text{state}_{\text{label}}$" column which has binary value where 0 if 'users are not banned after the post' otherwise 'users are banned after the post'

I modified train-val-test evaluation into sliding-window. Here, I tried to make adjustment as little as possible from default value.

#CAPTION: First positive sample is instance number 32632. First positive sample is instance number 32632.

- epoch = 5
- `~BATCH_SIZE~` = 100
- size of validation is 100 (`~BATCH_SIZE~` * 1)
- window slide by `BATCH_SIZE`
- size of initial training set = 673 instances (which is 0.1 percent of training set)

10.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635742197.4732468.log')
plot_path = str(base_path / 'plot/1635742197.4732468.png')

loss = []
auc = []
ap = []
epoch = []
ws = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    # if i == 100:
    #   exit()

    ws_val = line.split(" ")[-1].rstrip() if 'ws' in line and 'DEBUG' in line and 'roo
    epoch_val = line.split(" ")[-1].rstrip() if 'epoch' in line and 'DEBUG' in line el
    loss_val = line.split(" ")[-1].rstrip() if 'mean loss' in line and 'INFO' in line
    auc_val = line.split(" ")[-1].rstrip() if 'val auc' in line and 'INFO' in line els
    ap_val = line.split(" ")[-1].rstrip() if 'val ap' in line and 'INFO' in line else


    # reset param to None to prevent side effect
    if loss_val is not None:
      loss.append(float(loss_val))
    if auc_val is not None:
      auc.append(float(auc_val))
    if ap_val is not None:
      ap.append(float(ap_val))
    if epoch_val is not None:
      epoch.append(int(epoch_val))
    if ws_val is not None:
      ws.append(int(ws_val))
```

```
# non_missing_len = min(len(loss), len(auc), len(ap), len(epoch))
complete_ws_len = min(max(ws) * 5, len(epoch))

if complete_ws_len == max(ws) * 5:
  ws = ws[:-1]

loss = np.array(loss[:complete_ws_len]).reshape(-1, 5)
auc = np.array(auc[:complete_ws_len]).reshape(-1,5)
ap = np.array(ap[:complete_ws_len]).reshape(-1,5)
epoch = np.array(epoch[:complete_ws_len]).reshape(-1,5)


# plt.plot(loss[:,-1])
# plt.plot(auc[:,-1])
# plt.plot(ap[:,-1])

# plt.plot(loss.flatten())
# plt.plot(auc.flatten())
# plt.plot(ap.flatten())

# plt.plot(loss)
# plt.plot(auc)
# plt.plot(ap)

fig, axs = plt.subplots(1, 2, figsize=(9, 3))
axs[0].plot(ws, loss[:,-1], label = 'loss')
axs[1].plot(ws, auc[:,-1], label = 'auc')
axs[1].set_ylim(auc.min(), auc.max())
# axs[2].plot(ws, ap[:,-1], label = 'ap')
# axs[2].set_ylim(ap.min(), ap.max())
fig.suptitle('model performance')
axs[0].legend()
axs[1].legend()
# axs[2].legend()
plt.savefig(plot_path)
# plt.show()

plot_path
```

10.1.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

from picture above, one can see that loss function is improvment constantly while node classification performance fluctuate. This may be due to highly imbalance dataset where positive samples are only 336 instances which account to 0.05%.

Note that node classification were unfreeze from the same model that were trained for link prediction. TGN also train on top of JODIE embedding which are provided as edge instances features. In the other word, node classification model is pretrained while link prediction model is trained from scratch.

Node Classification Results from TGN paper.

#caption: node classification results from JODIE paper node classification results from JODIE paper

10.1.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

To make it fair to compare train-val-test results to sliding window, number of positive samples mucht be consider as well. In this case, sliding window has no positive instances up until around 32532th instances while train-val-test (75-15-15) have few positive instances in training set.

10.2 [RUN THIS NEXT] Research Log 2: Sliding window for node classification with (epoch, batch size, size of validation,window slide, first window size) = (50, 100, 100, 100, 673).

10.2.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I played around with parameters to see its impact on model performance.

- epoch = 50
- `~BATCH_SIZE~` = 100
- size of validation is 100 (`~BATCH_SIZE~` * 1)
- window slide by `BATCH_SIZE`
- size of initial training set = 673 instances (which is 0.1 percent of training set)

11 Research Date: *[2021-11-02 Tue]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

11.1 Research Log 1: In window sliding, burstiness of edges addition can be observed as window slides forward.

11.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

The goal is to visualize whether burtiness property of edges additions can be detected when window slides evaludation are performed.

11.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

Visualization step is done as followed

1. For every 100 instances (no overlapped), number of occurence of source and destination nodes are counted
2. the resulting list are sorted
3. the sorted list are concatenated.

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635945009.1619291.log')
plot_path = str(base_path / 'plot/1635945009.1619291.png')

nodes_freq = []
BATCH_SIZE = 100
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    nodes_freq_val = line.split("[")[1].split("]")[0].split(",") if 'list of sorted co

    # reset param to None to prevent side effect
    if nodes_freq_val is not None:
      nodes_freq_val = [int(ii) for ii in nodes_freq_val ]
      nodes_freq.extend(nodes_freq_val)
```

```
fig, axs = plt.subplots(1, 1, figsize=(9, 3))
axs.plot(nodes_freq, label = 'nodes_freq')
fig.suptitle('Bustiness visualization')
axs.legend()

plt.savefig(plot_path)
plot_path
```

11.1.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

From the above "Burstiness visualization" picture, burstiness behavior can be observed.

12 Research Date: *[2021-11-03 Wed]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

12.1 Research Log 1: =PAUSE= Using sliding window evaluation to evaluate nodes classifition where nodes are labeled 0 if in the next time step, it havek new edges, otherwise label is 0.

12.2 Research Log 2: =IMPLEMENTATION MISTAKE= With BATCH_SIZE = 100. Sliding window over each destination nodes of reddit data (which is post) to get temporal frequencies of interactions each post has.

12.2.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

BATCH_SIZE = 100 BATCH_SIZE set value for "window size." each window is not overlapped.

12.2.2 Experimental apparatus and procedures

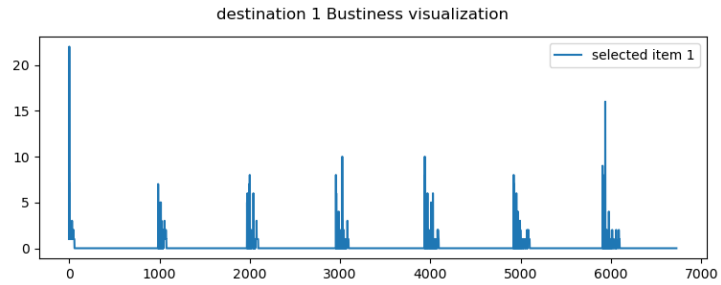
- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

I disable plt.savefig in the code below

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / 'log/1635950460.0076563.log')
plot_path = str(base_path / 'plot/1635950460.0076563.png')
```

```
items_freq = []
BATCH_SIZE = 100
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post count'

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
      items_freq.append(item_freq_val)



items_freq_np = np.array(items_freq).reshape(-1, len(items_freq))

for  ind, i in enumerate(items_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.plot(i, label = f'selected item {selected_item}')

  plot_path = str(base_path / f'plot/1635950460.0076563-destination={selected_item}.pn
  fig.suptitle(f'destination {selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  # plt.savefig(plot_path)
  plt.close()
# plot_path
```

exmample of the results is shown below



12.2.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

There exists periodic spike behavior of each items nodes. I am unable to reason over this unexpected behavior.

12.2.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

make sure that I provide gif of animated sliding over ordered nodes.

12.3 Research Log 3: `=IMPLEMENTATION MISTAKE=` With `BATCH_SIZE = 5000`. Sliding window over each destination nodes of reddit data (which is post) to get temporal frequencies of interactions each post has.

12.3.1 Background (information and description of goal)

- note

  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

`BATCH_SIZE` = 5000 `BATCH_SIZE` set value for "window size." each window is not overlapped.

12.3.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

I commented out plt.savefig()

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = '1636130420.8830836'
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / f'log/{log_file}.log')

items_freq = []
BATCH_SIZE = 5000
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post count'

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
```

```
      items_freq.append(item_freq_val)


items_freq_np = np.array(items_freq).reshape(-1, len(items_freq))

for  ind, i in enumerate(items_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.plot(i, label = f'selected item {selected_item}')

  plot_path = str(base_path / f'plot/{log_file}-destination={selected_item}.png')
  fig.suptitle(f'destination {selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  # plt.savefig(plot_path)
  plt.close()
# plot_path
print('done')
```

Example plot

12.3.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

~~The periodic spike behavior that observed in the following log Research Log 2: With `BATCH_SIZE = 100`. Sliding window over each destination nodes of reddit data (which is post) to get temporal frequencies of interactions each post has. are no longer existed when `BATCH_SIZE` is set to 5000.~~

After a more careful observation, when create animation from stack of destination nodes in order, data still seems to move to the left. (there exists some kind of temporal dependecies between ordered reddit's desitnation nodes.)

12.3.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

Given that there exists a period temporal frequency between ordered reddit's destination as observed, how does it effect performance of the model? if there exists any visible impact at all.

12.4 Research Log 4: `=IMPLEMENTATION MISTAKE=` With `BATCH_SIZE = 1000`. Sliding window over each destination nodes of reddit data (which is post) to get temporal frequencies of interactions each post has.

12.4.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

`BATCH_SIZE` = 1000 `BATCH_SIZE` set value for "window size." each window is not overlapped.

12.4.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

I commented out plt.show()

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = '1636130908.7385356'
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / f'log/{log_file}.log')

items_freq = []
BATCH_SIZE = 1000
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post count'

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
      items_freq.append(item_freq_val)



items_freq_np = np.array(items_freq).reshape(-1, len(items_freq))

for  ind, i in enumerate(items_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.plot(i, label = f'selected item {selected_item}')

  plot_path = str(base_path / f'plot/{log_file}-destination={selected_item}.png')
  fig.suptitle(f'destination {selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  # plt.savefig(plot_path)
  plt.close()
```

```
# plot_path
print('done')
```

Example plot

12.4.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

After a more careful observation, when create animation from stack of destination nodes in order, data still seems to move to the left. (there exists some kind of temporal dependecies between ordered reddit's desitnation nodes.)

13 Research Date: *[2021-11-04 Thu]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

13.1 Research Log 1: `=DEPRECATED=` Reordered reddit destination nodes validate that temporal dependencies of temporal frequency plot between ordered reddit destination nodes cannot occur by chance.

13.1.1 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

From Log 3 and 4 of Research log on *<2021-11-03 Wed>*, I reordered reddit destination nodes (because it seems that temporal depencies exists between ordered reddit destination nodes) to observer whether the temporal depencies can occurs by chance.

13.1.2 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
import matplotlib.image as mpimg

log_file = '1636130908.7385356'
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / f'log/{log_file}.log')

indices = list(range(984))
random.shuffle(indices)

items_freq = []
# BATCH_SIZE = 5000
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post count'

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
      items_freq.append(item_freq_val)
```

```
items_freq_np = np.array(items_freq).reshape(-1, len(items_freq))
```

```
for i in indices:
  plot_path = str(base_path / f'plot/{log_file}-destination={i}.png')
  img = mpimg.imread(plot_path)
  imgplot = plt.imshow(img)
  plt.show()
```

13.1.3 Data and Observations

- note
  - Does the logbook present the data and observations adequately?
  - How did you performed validation step as proof of correctness of your work?
  - Are graphs appropriately labeled and legible?

Temporal depencies between ordered reddit destination nodes seems to go away when I randomized order of reddit destination nodes.

13.1.4 Thought

- note
  - Record your thought at the time that you perform the experiments.
  - Are you drawing conclusion from your data?
  - Given what you have done and learned today, what should you do next and which ideas or concepts can be extended in the current and potention future researches?

Is it possible that the temporal depencies of temporal frequency plot between ordered reddit destination are caused by data collect?

14 Research Date: *[2021-11-07 Sun]*

- ref
  - `https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F`

14.1 Research Log 1: I re-do Research Log 3 from *<2021-11-03 Wed>* Fri> by plotting desitnation nodes of reddit data with window size = 5000.

14.1.1 Action Logs

- note
  - log every actions relevant to the current log
- I investigate whether temporal frequency can be correctly extracted `1636139834.1964872.log`
  - checking why when users data is extracted from the log into numpy array of (10000, 135), some of them the user column sum up to 0? This is in disagreement with freqency table generated by visidata.
    * I discovered that reshaping was used incorrectly and caused the data to have "temporal dependencies"

14.1.2 Background (information and description of goal)

- note
  - Why are you doing this? What is the movitation?
  - What is your expectation of the results?

I found that reshaping data was not correctly used and cause the data to have "temporal dependencies."

14.1.3 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

1. Reimplementation of Research Log 3 on *<2021-11-03 Wed>* where window size is set to 5000, but only show top 10 and bottom 1. I disable `plt.savefig` in the code below

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = "1636130420.8830836"
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / f'log/{log_file}.log')
plot_path = str(base_path / f'plot/{log_file}.png')

items_freq = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post co

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
      items_freq.append(item_freq_val)



items_freq_np = np.array(items_freq).T

top_10_item = [2,6,32,31,33,38,19,72,15,10]
# bottom_10_item = [2,6,32,31,33,38,19,72,15,10]
selected_items_freq_np = items_freq_np[top_10_item, :]

for  ind, i in enumerate(selected_items_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.bar(range(len(i)), i, label = f'selected item is top{selected_item}')
  # axs.plot(i, label = f'selected user {selected_item}')
```

```
  plot_path = str(base_path / f'plot/{log_file}-destinations=top{selected_item}.pn
  fig.suptitle(f'sources top{selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  # plt.savefig(plot_path)
  plt.close()
# plot_path
print('done')
# plot_path
```

top 1 top 2 top 3 top 4 top 5 top 6 top 7 top 8 top 9 top 10

For the bottom k, it doesn't work much sense to plot bottom 10. The following picture shows that number of count is less than 20.

bottom k of reddit destionation

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = "1636130420.8830836"
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')
log_path = str(base_path / f'log/{log_file}.log')
plot_path = str(base_path / f'plot/{log_file}.png')

items_freq = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    item_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit post co

    # # reset param to None to prevent side effect
    if item_freq_val is not None:
      item_freq_val = [int(ii) for ii in item_freq_val]
```

```
      items_freq.append(item_freq_val)



items_freq_np = np.array(items_freq).T

# bottom_10_item = [982,983,964,975,981,842,951,959,968,974]
bottom_k_item = [982]
selected_items_freq_np = items_freq_np[bottom_k_item, :]

for  ind, i in enumerate(selected_items_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.bar(range(len(i)), i, label = f'selected item is bottom{selected_item}')
  # axs.plot(i, label = f'selected user {selected_item}')

  plot_path = str(base_path / f'plot/{log_file}-destinations=bottom{selected_item
  fig.suptitle(f'sources bottom{selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  # plt.savefig(plot_path)
  plt.close()
# plot_path
print('done')
```

bottom 1

15 Research Date: *[2021-11-08 Mon]*

- ref
  - https://roamresearch.com/#/app/AdaptiveGraphStucture/page/pfT6XbX8F

15.1 Research Log 1: Using window size = 1000, visualize temporal frequency of top k (most active) and bottom k (least active) users.

15.1.1 Experimental apparatus and procedures

- note
  - have you provided sufficient detail in your description of experiment and apparatus to allow another person to reproduce your results or track down the source of any problems with the results?
  - have you included important detail that are not contained within the writeup?
  - Describe what you have done and how you have done it.

top k is plotted using the code below

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = "1636139834.1964872"
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')

log_path = str(base_path / f'log/{log_file}.log')
plot_path = str(base_path / f'plot/{log_file}.png')

users_freq = []
i = 1

with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    user_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit user count'

    # # reset param to None to prevent side effect
    if user_freq_val is not None:
      user_freq_val = [int(ii) for ii in user_freq_val]
      users_freq.append(user_freq_val)

users_freq_np = np.array(users_freq).T
```

```
# bottom_10_item = [982,983,964,975,981,842,951,959,968,974]
# bottom_k_item = [982]
top_k_users = [101, 554, 119, 150, 135, 529, 152, 576, 128, 241, 964]

selected_users_freq_np = users_freq_np[top_k_users, :]

for  ind, i in enumerate(selected_users_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.bar(range(len(i)), i, label = f'selected item is top{selected_item}')
  # axs.plot(i, label = f'selected user {selected_item}')

  plot_path = str(base_path / f'plot/{log_file}-sources=top{selected_item}.png')
  fig.suptitle(f'sources top{selected_item} Bustiness visualization')
  axs.legend()
  # plt.show()

  plt.savefig(plot_path)
  plt.close()
# plot_path
print('done')
```

top 1 top 2 top 3 top 4 top 5 top 6 top 7 top 8 top 9 top 10
bottomw k is plotted using the code below

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

log_file = "1636139834.1964872"
base_path = Path('/mnt/c/Users/terng/OneDrive/Documents/Working/tgn/')

log_path = str(base_path / f'log/{log_file}.log')
plot_path = str(base_path / f'plot/{log_file}.png')

users_freq = []
i = 1
```

```
with open(log_path, 'r') as f:
  for i, line in enumerate(f.readlines()): # 9341 lines in total
    user_freq_val = line.split("[")[1].split("]")[0].split(",") if 'reddit user count'

    # # reset param to None to prevent side effect
    if user_freq_val is not None:
      user_freq_val = [int(ii) for ii in user_freq_val]
      users_freq.append(user_freq_val)

users_freq_np = np.array(users_freq).T

bottom_k_users = [91, 107, 109, 147, 180, 208, 234, 249, 349, 400]
selected_users_freq_np = users_freq_np[bottom_k_users, :]

for  ind, i in enumerate(selected_users_freq_np):
  print(ind)
  fig, axs = plt.subplots(1, 1, figsize=(9, 3))
  selected_item = ind + 1
  axs.bar(range(len(i)), i, label = f'selected item is bottom{selected_item}')
  # axs.plot(i, label = f'selected user {selected_item}')

  plot_path = str(base_path / f'plot/{log_file}-sources=bottom{selected_item}.png')
  fig.suptitle(f'sources bottom{selected_item} Bustiness visualization')
  axs.legend()
  plt.show()

  # plt.savefig(plot_path)
  plt.close()
# plot_path
print('done')
```

bottom 1 bottom 2 bottom 3 bottom 4 bottom 5 bottom 6 bottom 7 bottom 8 bottom 9 bottom 10