DMSN final project: Improve LESSR model structure

TENG, LI-CHANG
Department of Electrical Engineering
National Cheng Kung University
n26091194@gs.ncku.edu.tw

TENG, LI-CHANG
Department of Electrical Engineering
National Cheng Kung University
n26091194@gs.ncku.edu.tw

TENG, LI-CHANG
Department of Electrical Engineering
National Cheng Kung University
n26091194@gs.ncku.edu.tw

TENG, LI-CHANG
Department of Electrical Engineering
National Cheng Kung University
n26091194@gs.ncku.edu.tw

Abstract

None

1 INTORDUCTION

None

2 RELAED WORK

None

3 PRELIMINARY

None

Preprint. Under review.

Table 1: statistics of dataset

Diginetica	
No. of Clicks	981,620
No. of Sessions	777,029
No. of Items	42,596
Average length	4.80

Table 2: Multi-head w/o pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
Head=1	52.65	18.25	25.85	-0.903594775
Head=2	52.58	18.27	25.84	-0.965396084
Head=4	52.6	18.28	25.85	-0.834321462
Head=8	52.63	18.28	25.87	-0.700394057
Head=16	52.62	18.28	25.86	-0.757891648
Head=32	52.64	18.29	25.87	-0.626817026

4 EXPERIMENTS

In this section, we will introduce experiment setting, dataset, and analyze the experiment result. We conducted several experiments to check out hypotheses and evaluate our model with choosen metric.

4.1 Dataset

We choose Diginetica dataset[1] following LESSR [1] paper, which is the CIKM cup 2016 dataset provided by DIGINETICA Crop. There are 6 files in Diginetica dataset, but we only need the transaction one. As [1], we used last week sessions as test data. We got the same training and test set by following preprocessing method described in [1]. Statistics of Diginetica dataset is shown in Table 1.

4.2 Baseline and metrics

We choose [1] as out baseline, then we tried to improve model structure in [1] by some changes. Comparing the metrics to [1], we could know the change is postive or negative influence. Following [1], the metrics we used are HR@20 (Hit Rate) and MRR@20 (Mean Reciprocal Rank).

4.3 Multi-Head Attention

MUTIHEADATTENTION[2] is a official implemented self-attention layer by pytorch. Here we replace GRU[3] layer in EOPA block in [1] by MUTIHEADATTENTION layer. All settings are the same but GRU now replaced by MUTIHEADATTENTION. We adjusted num of heads parameter in MUTIHEADATTENTION layer to see the influence of muti-head attention.

The pytorch official did not implemented positional encoding in MUTIHEADATTENTION layer, so there is no position information within layer. To handle this problem we need to do position encoding manually. We fonud a offical tutorial[4] that manually implemented position encoding, so we followed the encoding method here.

Table 2 and Table 3 are the experiment result without and with positional encoding respectively. It turns out no matter multi-head or positional encoding, can not improve the result. So in next section we decided to using more complex layer.

[1] `https://competitions.codalab.org/competitions/111610`

[2] `https://pytorch.org/docs/stable/generated/torch.nn.MultiheadAttention.html`

[3] `https://pytorch.org/docs/stable/generated/torch.nn.GRU.html`

[4] `https://pytorch.org/tutorials/beginner/transformer_tutorial.html`

Table 3: Multi-head with pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
Head=1	52.57	18.26	25.83	-1.077538484
Head=2	52.55	18.29	25.85	-0.874337766
Head=4	52.59	18.3	25.88	-0.628267962
Head=8	52.62	18.29	25.87	-0.664681471
Head=16	52.54	18.31	25.86	-0.745415003
Head=32	52.57	18.32	25.89	-0.518277422

Table 4: dim exp w/o pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
Dim = 2048	52.73	18.25	25.83	-0.82926773
Dim = 1024	52.9	18.31	25.86	-0.063854988
Dim = 512	52.97	18.35	26	0.827164961
Dim = 256	52.67	18.28	25.88	-0.586099799
Dim = 128	52.88	18.34	25.94	0.370737939
Dim = 64	52.84	18.39	26.01	0.83819067
Dim = 32	52.7	18.24	25.85	-0.863578471
Dim = 16	52.68	18.3	25.88	-0.457877958

4.4 Transformer Encoder

In this section, we use transformer encoder [5] to replace GRU. TransformerEncoder has a lot of hyperparameter, so we conducted 3 main experiments to tuning the model: 1) dim_feedforward 2) nhead 3) encoder_layer. Also, each main experiments have two sub experiments: 1) w/o pos encoding 2) with pos encoding.

4.4.1 Dim_Feedforward Experiment

In this experiment we fix all hyperprameters but dim_feedforward. Table 4 shown the result without positional encoding. Table 5 shown the result with positional encoding.

From Table 4 and Table 5, we fonud that the best dim_feedforward is seeting 512, whether with pos encoding or not, dimension 512 in both case has a good result, so we choose dimension 512 for our model in the later experiments.

[5] `https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html`

Table 5: dim exp with pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
Dim = 2048	52.74	18.28	25.88	-0.45357424
Dim = 1024	52.86	18.3	25.88	-0.117097951
Dim = 512	52.85	18.36	25.97	0.538926994
Dim = 256	52.7	18.26	25.86	-0.715723485
Dim = 128	52.89	18.37	25.95	0.592169956
Dim = 64	52.79	18.34	25.95	0.238913304
Dim = 32	52.74	18.29	25.9	-0.321798695
Dim = 16	52.6	18.36	25.92	-0.127205414

Table 6: multi-head exp w/o pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
nhead=1	52.97	18.35	26	0.827164961
nhead=2	52.77	18.37	25.95	0.364983285
nhead=4	52.98	18.35	26	0.846097184
nhead=8	52.87	18.37	25.96	0.592870879
nhead=16	52.78	18.37	25.97	0.461046244
nhead=32	52.92	18.41	25.97	0.944676596

Table 7: multi-head exp with pos encoding

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
nhead=1	52.85	18.36	25.97	0.538926994
nhead=2	52.7	18.31	25.91	-0.2496726
nhead=4	52.88	18.38	25.98	0.743578647
nhead=8	52.87	18.41	26.03	1.081407692
nhead=16	52.82	18.35	25.96	0.388920149
nhead=32	52.75	18.35	25.95	0.217829222

4.4.2 Multi-Head Experiment

Here we fixed all hyperprameters but nhead to see the influence. Also, The dim_feedforward set to 512. Result without positional encoding shown in Table 6 and result with positional encoding shown in Table 7.

Comparing Table 6 and Table 7, We found the metrics without positional encoding are usually better than the other one. So positional information might not a critical info in this scenario.

Note that best performance appeared when nhead set to 8 with postional encoding.

4.4.3 Num-Layers Experiment

Here all hyperprameters was fixed but num_layers will be change. The dim_feedforward was set to 512 and nhead was set to 1. Table 8 and Table 9 shown the result without and with positional encoding respectively.

We found that whether transformer encoder with positional encoding or not, it has similar trend that performance decreased as layers increased. And we found as layers increased, models's inference time also increased.

Table 8: num-layers exp w/o pos

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
layer=1	52.97	18.35	26	0.827164961
layer=2	52.72	18.41	25.93	0.41177067
layer=3	52.69	18.4	25.95	0.37745993
layer=4	52.69	18.33	25.89	-0.236445941
layer=6	52.65	18.13	25.72	-2.060682268
layer=8	52.24	17.96	25.48	-4.691433984
layer=16	52.11	17.77	25.34	-6.515719401

Table 9: num-layers exp with pos

AGG.TYPE	HR@20	MRR@20	NDCG@20	Total impv.
baseline	52.82	18.3	25.93	-
layer=1	52.85	18.36	25.97	0.538926994
layer=2	52.77	18.41	25.96	0.622127888
layer=3	52.72	18.34	25.88	-0.163569833
layer=4	52.84	18.32	25.9	0.031457958
layer=6	52.8	18.18	25.78	-1.272082675
layer=8	52.3	17.95	25.46	-4.709616194
layer=16	52.04	17.81	25.37	-6.313969619

5 CONCLUSION

In our experimets, no matter with postional encoding or not, the performance of MUTIHEADATTENTION is worse than GRU, although using MUTIHEADATTENTION is slightly fast. In muti-head experiment of TransformerEncoder, we found there is no distinct trend about num of head, also we found when TransformerEncoder stacking more layers, model's performance will decreased, this means the model is over-fitting, moreover, evaluate time will increased, this cause training time getting longer.

After several experimets, we found that TransformerEncoder can improve model performance by replacing GRU in EOPA layer, but the parameter of TransformerEncoder need in proper setting. In SGAT and Readout layer, we tried to change the attention mechanisms, but we didn't got a good result. This might bacause attention mechanisms is dependent on model structure and data, it won't be useful if we just replace attention mechanisms from different paper.

Finally the best result we got is when layer=1 and nhead=8 with positional encoding, total improvement of three metrics is 1.08%, althought there is still a space for improvements in this model. The result is a acceptable to us because we didn't change the model structure so much but just change a layer inside EOPA block. If we could modify SGAT and Readout layer's attention mechanisms to multi-head attention, we might get a better result.

References

[1] Tianwen Chen and Raymond Chi-Wing Wong. Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20)*, pages 1172—1180, 2020.