R-UniMP: Solution for KDDCUP 2021 MAG240M-LSC

A Preprint

Yunsheng Shi
Team PGL
Baidu Inc., Shenzhen
China
shiyunsheng01@baidu.com

Zhengjie Huang
Team PGL
Baidu Inc., Shenzhen
China
huangzhengjie@baidu.com

Weibin Li
Team PGL
Baidu Inc., Shenzhen
China
liweibin02@baidu.com

Weiyue Su
Team PGL
Baidu Inc., Shenzhen
China
suweiyue@baidu.com

Shikun Feng
Team PGL
Baidu Inc., Shenzhen
China
fengshikun01@baidu.com

Abstract

The MAG240M-LSC track on KDD Cup 2021 invites participants to deploy a graph neural network for node-level prediction over a large-scale citation network. MAG240M-LSC is a heterogeneous academic graph extracted from the Microsoft Academic Graph (MAG) with multiple relations between papers, authors, and institutions. Participants are required to predict the topics corresponding to the publication. In this competition, we adopt the recent advanced technique UniMP Shi et al. [2020] which proposes to incorporate feature and label propagation at both training and inference time, making significant improvements across several node classification tasks. And we modify it into an R-UniMP version for a heterogeneous graph with "R" stands for "Relational". Besides, we provide a detailed recall of our key strategies and valuable findings during the entire competition. Our best single models can reach 73.71% in the official validation split. For final submission, we train our models with 5-Fold settings. And we make a bagging search for ensemble selections over our local 5-Fold splits. The final submission is bagged over 30 models' predictions. And it achieves 75.49% in the final test set. The source code is available at https://github.com/PaddlePaddle/PGL/tree/main/examples/kddcup2021/MAG240M/r_unimp.

1 Introduction

To empower the machine learning (ML) development on large-scale graph data, Hu et al. [2021] holds OGB Large-Scale Challenge (OGB-LSC) at KDD Cup 2021 which contains three large-scale real-world datasets corresponding to three common graph challenges: node prediction, link prediction, and graph prediction. MAG240M-LSC is one of the tasks asking participants to predict labels for nodes. MAG240M-LSC is extracted from Microsoft Academic Graph (MAG) Wang et al. [2020] which contains 244,160,499 nodes and 1,728,364,232 edges, the largest dataset among OGB-LSC. Nodes in MAG240M-LSC represent papers, authors, and institutes. And we have three types of edges: paper-cite-paper, author-write-paper, author-affiliated-institute. Among the 121M paper nodes, there are about 1.4M nodes are from ARXIV annotated with 153 ARXIV subject areas. Features for paper nodes are extracted by powerful pre-trained language model RoBERTA Liu et al. [2019] with concatenated title and abstract of the titles as inputs. The task is to predict the primary subject areas of the given ARXIV papers as an ordinary multi-class classification problem. The metric is classification accuracy.

2 Methods

2.1 R-GAT

Recent advanced approaches for node classification are mainly based on graph neural networks. Hu et al. [2021] provides early investigation on several classical methods including GCN, GAT, R-GCN, R-GAT. We simply follow the best baseline model R-GAT from them. And we make some modifications for further improvements.

2.1.1 Relation-wise Neighborhood Sampling

The original implementation of R-GAT takes all neighbors as a homogeneous graph during neighborhood sampling. However, they perform relation-wise GAT aggregation ignoring the sampling bias. After a full investigation of the neighborhood distribution, we find that the homogeneous sampling methods cannot guarantee that each relation has at least one neighborhood. For the reason that it recursively samples 25 neighbors in the first layer and 15 neighbors in the second layer, but there are more paper-cites-paper relations than author-writes-paper as shown in Table 1. Therefore, we perform relation-wise sampling with different amounts of sampled neighbor for each relation correspond to each layer.

Table 1: Statistic Relation

NodeType	Relation	Avg. Relation Num.	Max Relation Num.	Sample Num. 1st Layer	Sample Num. 2nd Layer
Paper	Paper	21.31	242655	25	15
	Author	3.17	6760	10	10
Author	Paper	3.15	4724	-	10
	Institute	0.36	78	-	5
Institute	Author	1733.70	395249	-	-

2.1.2 Relation-wise BatchNorm

Besides, normalization has been proved to be an effective technique for deep residual graph neural networks Li et al. [2019]. The original implementation of R-GAT simply plugs in BatchNorm modules after feature aggregation between different relations. And all the nodes share the same statistics of running mean and deviation. However, paper nodes are the majority in each training batch causing inaccurate statistics for author and institute nodes. To address this issue, as shown in Figure 1, we place BatchNorm layer just after neighborhood aggregation. And we prevent parameters and statistics sharing between node and edge types.

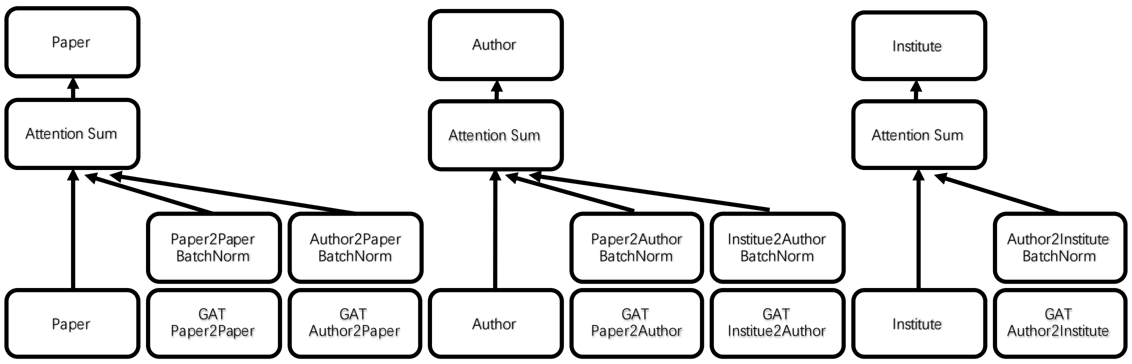

Figure 1: R-GAT with Relation-Wise BatchNorm and Relation-Wise Attention

2.1.3 Relation-wise Attention

Moreover, edge relations might have various contributions on different nodes. Therefore, in replace of feature summation between relations, we design a weighted attention sum layer for relation-wise feature aggregation as shown in Figure 1. The formulation of Relation-wise Attention is following:

$$\begin{aligned} \mathbf{H}_p^{k+1} &= \alpha_p \mathbf{H}_p^k + \alpha_{p2p}\mathbf{H}_{p2p}^{k+1} + \alpha_{a2p}\mathbf{H}_{a2p}^{k+1} \\ \mathbf{H}_a^{k+1} &= \alpha_a \mathbf{H}_a^k + \alpha_{p2a}\mathbf{H}_{p2a}^{k+1} + \alpha_{i2a}\mathbf{H}_{i2a}^{k+1}, \\ \mathbf{H}_i^{k+1} &= \alpha_i \mathbf{H}_i^k + \alpha_{a2i}\mathbf{H}_{a2i}^{k+1} \end{aligned} \tag{1}$$

where $\mathbf{H}_p, \mathbf{H}_a, \mathbf{H}_i$ denote feature for paper, author and institute nodes. And $\mathbf{H}_{p2p}, \mathbf{H}_{a2p}, \mathbf{H}_{p2a}, \mathbf{H}_{a2i}, \mathbf{H}_{i2a}$ are aggregated representation from different relations. Each node hidden is computed with attention weighted sum between residual connection and neighbors. The attention weight are computed as follows:

$$\begin{aligned} \alpha_p, \alpha_{p2p}, \alpha_{a2p} &= \text{softmax}(\mathbf{W}\mathbf{H}_p^k, \mathbf{W}\mathbf{H}_{p2p}^{k+1}, \mathbf{W}\mathbf{H}_{a2p}^{k+1}) \\ \alpha_a, \alpha_{p2a}, \alpha_{i2a} &= \text{softmax}(\mathbf{W}\mathbf{H}_a^k, \mathbf{W}\mathbf{H}_{p2a}^{k+1}, \mathbf{W}\mathbf{H}_{i2a}^{k+1}) \\ \alpha_i, \alpha_{a2i} &= \text{softmax}(\mathbf{W}\mathbf{H}_i^k, \mathbf{W}\mathbf{H}_{a2i}^{k+1}). \end{aligned} \tag{2}$$

2.2 Masked Label Prediction

Shi et al. [2020] proposes a model UniMP that combines label propagation and feature propagation in both the training and inference stage. And it points out that the utilization of observed labels in the training set can make huge improvements for predictions. Since labels might be ambiguous in node classification. For example, a paper can hold several different topics, which might be confusing for GNN models. The most direct evidence is that the accuracy of the training set can hardly achieve high accuracy. Therefore, we adopt the idea in UniMP and perform **Masked Label Prediction** as described in Shi et al. [2020]. At training time, we add label embeddings on all paper nodes that have observed labels except the target ones.

Random Label Input. Besides, we find that adding random labels on all paper nodes during training can make our model more robust in prediction. We guess that random label noises force our model to make predictions depending more on node feature but not labels, since the un-labeled nodes in validation connected to less labeled nodes than the one in the training set.

2.3 Post-Smoothing

After training our models and making predictions, there are still tricks for further improvements. **Correct and Smooth** (C&S) Huang et al. [2020] finds that simple post-processing steps via modifications to standard label propagation techniques can boost the performance. In our work, we find that after training with label information with UniMP techniques. Simple C&S post-processing step can have little impact on the prediction. However, we find that if we are performing smoothing on a constructed paper-paper graph with a higher homophily ratio, we can achieve 0.05%-0.1% improvements. Therefore, we investigate validation accuracy by simple voting strategies on the different paper graph as follows:

Table 2: Voting Accuracy with Different Graph Construction

Graph	Validation Accuracy By Voting	Coverage on Validation Nodes
Paper Cites Paper (Bidirected)	41.69%	91.17%
Paper more than 2 Coauthor	54.59%	88.28%
Top50 Coauthor Paper	60.99%	94.54%
Coauthor Jaccard $>$ 0.5	48.46%	77.16%

In Table 2, we find that simple voting on coauthor graph can achieve a higher prediction accuracy which brings hope for post-processing. Our smoothing strategy can be formulated as follows:

$$\mathbf{Y}^k = (1-\alpha)\mathbf{A}\mathbf{Y}^{k-1} + \alpha\mathbf{Y}^0 \tag{3}$$

$$\mathbf{Y}^0 = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n] = \begin{cases} \text{one-hot ground truth, if } node_i \text{ in training set} \\ \text{softmax prediction, if } node_i \text{ not in training set} \end{cases} \tag{4}$$

where α controls the importance between self and neighbors' prediction, k stands for the number of iterations. Unlike the C&S implement, we evaluate the validation accuracy after each iteration and perform early stopping on k and α. After a few steps (3 or 4) of post-smoothing with $\alpha = 0.8$, the prediction can reach a convergence. And surprisingly, Coauthor Jaccard Graph outperform all the others. The details can be found in the experiments.

2.4 Year of Publication as Position Embedding

We simply perform voting strategies on validation nodes by their neighborhoods according to the published time. An important insight from Table 3 is that papers published in similar years have a closer relationship (validation nodes are all papers published in 2019). To ensure our model can capture the relationship between the years of publications, we adopt the position embedding from Vaswani et al. [2017] and simply sum it together with the node features.

Table 3: Voting Accuracy from Different Years

Paper Published Years	Voting Accuracy on Validation	Coverage
2018	28.68%	63.63%
2017	20.68%	62.93%
2016	16.09%	60.26%
< 2016	18.72%	85.63%

2.5 Network Embeddings

There are difficulties for training deep graph neural networks such as exponential growth for both execution time and memory. Therefore, to utilize the structure information between nodes, we train network embeddings on these heterogeneous networks. Metapath2vec Dong et al. [2017] is one of the classical representation learning algorithms on heterogeneous networks. The metapath2vec model formalizes meta-path-based random walks to construct the heterogeneous neighborhood of a node and then trains a heterogeneous skip-gram model to perform node embeddings. We adopt the implementation from PGL[1] and train the network embeddings which are then concatenated besides the provided 768 semantic features.

2.6 Ensemble Settings

2.6.1 5-Fold Ensemble

As mentioned in Section 2.4, publications have similar topics with recent papers and the dataset split by years, which makes that the validation data more significant. To make full use of the validation data, we randomly split the validation data into 5-parts. For each fold, we take one of five for validation on both the training and post-smoothing stage. For each hyper-parameter setting, we train five models for the folds, then make a prediction on the test set and apply bagging for the results.

2.6.2 Bagging Beam Search for Final Predictions

Each hyper-parameter setting contributes 5 validation predictions on each fold and one bagging prediction on test data. To make the final prediction, we have a beam search strategy that greedily adds one prediction from different hyper-parameter settings to achieve higher validation accuracy, which can be regarded as weighted bagging.

3 Experiments

3.1 Implementation Details

All our implementation can be found at https://github.com/PaddlePaddle/PGL/tree/main/examples/kddcup2021/MAG240M/r_unimp. The code is implemented with Paddle Graph Learning (PGL) which is a graph neural network framework based on message passing paradigms. We train each of our models with four 32GB Tesla V100-SXM2 and 500GB CPU memories, using 24 hours. Since the graph data and the feature need a large amount of memory, we use PGL's shared memory graph to load the graph structure and node features. To speed up the training process, we use a multiprocessing dataloader for neighborhood sampling.

[1]metapath2vec can be found at https://github.com/PaddlePaddle/PGL/tree/static_stable/examples/metapath2vec

3.2 Best Single Model

Table 4 shows the process of the entire optimization of our models. During the exploration of model architectures, we keep the almost similar hyper-parameter settings for each architecture upgrade. And we haven't make extensive searching on hyper-parameters. Under our dedicated architecture design and the efforts from mentioned components in Section 2, the upgraded model lift to 73.66 % from 70.20%.

Table 4: The Birth of Best Single Model

No.	Model	Official Valiation
0	R-GAT (PyTorch)	70.02 %
1	R-GAT (Ours)	70.20 %
2	1 + Masked Label Prediction	70.90 %
3	1 + Metapath2vec	70.77 %
4	2 + 3 + Relation-wise BatchNorm & Sampling	71.67 %
5	4 + Years of Publication as Position Embedding	72.73 %
6	5 + Random Label Inputs	73.22 %
7	6 + Relation-wise Attention	73.50 %
8	7 + Hyper Parameter Tuning	**73.66 %**
9	8 + Post-Smoothing on Paper Cites Paper (Bidirectioned)	73.68 %
10	8 + Post-Smoothing on Paper More Than 2 Coauthor	73.70 %
11	8 + Post-Smoothing on Top50 Coauthor Paper	73.50 %
12	8 + Post-Smoothing on Coauthor Jaccard $>$ 0.5	**73.71 %**

3.3 Ensemble Models

Finally, we train our models with 6 different hyper parameters. The major difference and the 5-Fold performance are shown in Table 5. Besides, we find that concatenation the node representation from all layers for final prediction can achieve further improvement. However, we don't have enough time to run ablation on official validation split. Our final ensemble prediction are obtained by 30 models with validation accuracy as 77.73%. And each contribution ratio to the final prediction is shown in Table 5.

Table 5: 5-Fold Validation Performance and Model Ensemble

Models Settings	Metapath2vec Settings	5-Fold Validation Acc.	Ensemble Ratio
Last layer for prediction	4 epochs, 7 window size, 128 dim	76.97%	5%
- Post-Smoothing		77.12%	10%
Last layer for prediction	10 epochs, 3 window size, 128 dim	77.07%	5%
-Post-Smoothing		77.19%	15%
Last Layer for prediction	5 epochs, 3 window size, 64 dim	77.03%	5%
- Post-Smoothing		77.17%	5%
Concat. all layers for prediction	4 epochs, 7 window size, 128 dim	77.20%	5%
- Post-Smoothing		77.35%	15%
Concat. all layers for prediction	10 epochs, 3 window size, 128 dim	77.11%	5%
- Post-Smoothing		77.26 %	15%
Concat. all layers for prediction	5 epochs, 3 window size, 64 dim	77.15%	5%
- Post-Smoothing		77.29%	20%
Ensemble Models		77.73%	

References

Yunsheng Shi, Zhengjie Huang, Wenjin Wang, Hui Zhong, Shikun Feng, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.