Technical report on MNCI

Due to the page limit of the paper, we present more experiments in the technical report. In this report, we added several experiments, such as link prediction, parameter sensitivity, and ablation study.

1 DATASETS

Table 1: Description of the datasets

Datasets	DBLP	BITotc	BITalpha	ML1M	AMms	Yelp
Nodes	28,085	5,881	3,783	9,746	74,526	424,450
Edges	236,894	35,592	24,186	1,100,209	89,689	2,610,143
Labels	10	7	7	5	5	5

We list the statistical information of the following six real-world network datasets in Table 1.

DBLP [11]: This is a co-authorship network of Computer Science domain. We extracted ten research fields from DBLP to construct our dataset. If more than half of a researcher's last ten papers are published in a particular research field, we assume that the researcher belongs to this field.

BIT otc/alpha [4, 5]: There are two datasets from two bitcoin trading platforms OTC and Alpha, respectively. Members of platforms rate other members in a scale of -10 (total distrust) to +10 (total trust) in steps of 1. We classify every three scores into one category, and there are seven categories, e.g., users with scores -10, -9 and -8 are in the same category.

ML1M [6]: This is a widely used movie dataset for machine learning tasks, and we use the version MovieLens-1M that has 1 million user ratings. For each movie, we choose the score that people rated most as its label. Since each score is an integer between 1 and 5, we can divide all movies into five categories.

AMms [8]: This dataset is taken from the magazine subscription, which is a part of the Amazon website. For each magazine, we choose the score people rated most as its label. We also divide all magazines into five categories.

Yelp [11]: We derive the dataset from the Yelp Challenge dataset. Users and businesses are considered as nodes in a network, and commenting behaviors are taken as edges. Each business is assigned either one or more categories. We only retain business in the top-5 categories. If a business has more than one category, we assign the top one category as the business's label.

2 BASELINES

We compare AGLI with five state-of-the-art baselines.

DeepWalk [9]: This method performs a random walk procedure over the network to generate the node sequence and then employ Skip-Gram [7] to learn node embeddings.

node2vec [1]: This method extends DeepWalk and proposes a biased random walk procedure to maintain a balance between breadth-first and depth-first search strategy.

GraphSAGE [2]: This method is a general inductive framework which learns a function that generates node embedding by sampling and aggregating features from nodes' local neighborhood.

HTNE [11]: This method uses the Hawkes process to capture influence of historical neighbors on the current node to obtain node embeddings.

DyREP [10]: This method proposes a two-time scale deep temporal point process model, which captures the interleaved dynamics of the observed processes.

3 LINK PREDICTION

Based on the embeddings of two nodes, we can calculate their dot product to determine whether there is an edge between two nodes. For link prediction, we conduct experiments on six datasets and take the Area Under the ROC Curve (AUC) [3] and Accuracy to measure the prediction performance.

We sort all interactions in a dataset in order of interaction time. We select the top 80% of each dataset as our training set, and the rest 20% as the test set. If the same timestamp interactions are assigned to both training set and test set, we assign all interactions at this timestamp to the training set. We obtain all node embeddings by applying MNCI and baselines in training set. In the test set, we sample a certain number of node pairs connected by edges as positive samples and sample the same number of node pairs without edge as negative samples.

For AUC, we calculate the dot product of their embeddings for each pair of nodes and use the sigmoid function to normalize the dot product as the interaction probability. Then we sort all interaction probabilities in descending order and assume that there are edges between each node pair of the top-half. By comparing the truth on node pairs, we can obtain the AUC score. For Accuracy, we set a threshold value of 0.5 to evaluate the prediction result. When the normalize dot product of a node pair is greater than 0.5, we consider that there exists an edge between this node pair.

As shown in Table 2, it can be seen that MNCI has the best performance on all datasets, which demonstrates the ability of MNCI to capture interactive information in the network. It also shows that for prediction tasks, a method such as HTNE that exploits the interaction time are more effective than a method such as Deepwalk that only focuses on network structure.

4 PARAMETER SENSITIVITY

4.1 Embedding dimension size

Embedding dimension size d means the length of the generated node embedding, where each dimension in node embedding represents a value. Combining values of fixed lengths can express the important information contained in the network.

In this section, we will fix the other parameter settings and choose different values for d such as 32, 64, 128, 256, 512 to test the performance on MNCI and baselines. We evaluate these methods on node classification task and use Accuracy as metrics.

Table 2: Link prediction results of all methods on all datasets

Metric	method	DBLP	BITotc	BITalpha	ML1M	AMms	Yelp
AUC	DeepWalk	0.8253	0.5199	0.5558	0.4635	0.5483	0.7740
	node2vec	0.8173	0.5799	0.6245	0.5012	0.5228	0.8426
	GraphSAGE	0.8452	0.5967	0.6964	0.5055	0.5554	0.8553
	HTNE	0.8868	0.7145	0.7401	0.5021	0.5741	0.8821
	DyREP	0.8763	0.7112	0.7342	0.5069	0.5807	0.8664
	MNCI	**0.8966**	**0.7477**	**0.7432**	**0.5413**	**0.5934**	**0.8929**
Accuracy	DeepWalk	0.5225	0.5390	0.5399	0.5004	0.5067	0.5174
	node2vec	0.5009	0.5017	0.5031	0.5008	0.5000	0.5002
	GraphSAGE	0.6662	0.5539	0.5550	0.5049	0.5332	0.5023
	HTNE	0.7357	0.5912	0.6238	0.4639	0.5687	0.5290
	DyREP	0.7203	0.6049	0.6433	0.5023	0.5454	0.5148
	MNCI	**0.7804**	**0.6972**	**0.6936**	**0.5058**	**0.5700**	**0.5299**

According to Table 3, we find that the embedding size has little effect on the performance of MNCI and it performs the best overall. The results prove the robustness to embedding size of MNCI.

In addition, we can see almost all methods achieve the best performance when the node embedding size d=128. Where GraphSAGE works best when d=256 on DBLP, HTNE works best when d=512 on AMms. However, their results have a smaller difference compare with the perofrmance when d=128. We believe this is due to the acceptable error on different datasets. We can also see that GraphSAGE and HTNE achieve the best performance when d=128 on another dataset. To our knowledge, most network representation learning methods choose d=128 as the default value for training. Therefore, we set the embedding dimension size d to be 128 on MNCI and use default values of d in baselines (In fact, all d=128 of them).

4.2 Community number

For the community influence, we set the community number K, which is a hyperparameter. In this part, we will fix the other parameter settings and choose different value of K to test the performance on MNCI. We use node classification task and use Accuracy as metrics to evaluate the performance on several datasets, such as DBLP, BITotc, BITalpha, and Yelp.

As shown in Figure 1, MNCI obtain the best performance on BITotc and BITalpha when $K = 5$, and obtain the best performance on DBLP and Yelp when $K = 10$. According to Table 1, the real numbers of node labels are 5 for BITotc and BITalpha, 7 for Yelp, and 10 for DBLP, respectively. This is generally consistent with the optimal community number K on the different datasets, and the performance on Yelp is similar when $K = 5/10$. The results reflect that MNCI is able to mine the communities in real-world networks well.

5 ABLATION STUDY

In this section, we construct several variants of MNCI to study the effect of positional encoding, community and neighborhood influences on the performance improvements.

5.1 Positional Encoding

Different from existing NRL methods, we use positional encoding instead of random initialization to generate the initial embedding. We use positional encoding and random initialization, respectively, to obtain two variants of MNCI. We discover that there is no significant difference between the final performance of the two initialization methods through experiments. However, when comparing the loss function's convergence speed, positional encoding is much faster than random initialization. Since one epoch represents one complete training on the whole dataset, we use ***epoch number*** as a metric to compare the convergence speed.

On DBLP, when the performance is almost the same, random initialization requires ***30 epochs*** to converge, while positional encoding only needs ***10 epochs***. The convergence speed of the latter is 3 times as fast as the former. On BITotc and BITalpha, the convergence speeds of random initialization and positional encoding are ***20*** and ***5 epochs*** respectively. The convergence speed of the latter is 4 times as fast as the former. The results demonstrate that positional encoding can accelerate the convergence speed of MNCI without reducing performance, which is especially applicable for large-scale dataset.

5.2 Neighborhood and Community Influences:

MNCI aggregates information from three parts: current node embedding, neighborhood influence and community influence.

Let MNCI.z be a variant of MNCI which does not aggregate neighborhood and community influence, i.e., $\tilde{z}_u^{t_n} = \tanh[W_z z_u^{t_{n-1}} + b_z]$. MNCI.zn denotes a variant of MNCI that only aggregate neighborhood influence and node embedding, i.e., $\tilde{z}_u^{t_n} = \tanh[W_z(z_u^{t_{n-1}} + NRG_u^{t_n} \cdot NE_u^{t_n}) + b_z]$. MNCI.zc denotes a variant of MNCI that only aggregate community influence and node embedding, i.e., $\tilde{z}_u^{t_n} = \tanh[W_z(z_u^{t_{n-1}} + CRG_u^{t_n} \cdot CO_u^{t_n}) + b_z]$.

We evaluate these variants of MNCI on node classification task. As shown in Figure 2, when neighborhood and community influences are not used, the performance is the worst. The performance is improved when we use community influence or neighborhood influence. However, MNCI.zn with neighborhood influence is better

Table 3: Parameter sensitivity experiment of embedding dimension size d

Accuracy	method	d=32	d=64	d=128	d=256	d=512
DBLP	DeepWalk	0.5990	0.6057	**0.6140**	0.6129	0.6051
	node2vec	0.6212	0.6246	**0.6249**	0.6207	0.6249
	GraphSAGE	0.6239	0.6303	0.6331	**0.6340**	0.6305
	HTNE	0.6136	0.6255	**0.6347**	0.6328	0.6337
	JODIE	0.6209	0.6233	**0.6259**	0.6203	0.6187
	MNCI	0.6392	0.6399	**0.6435**	0.6407	0.6398
AMms	DeepWalk	0.5660	0.5757	**0.5780**	0.5711	0.5697
	node2vec	0.5612	0.5646	**0.5772**	0.5707	0.5749
	GraphSAGE	0.5685	0.5687	**0.5763**	0.5759	0.5703
	HTNE	0.5736	0.5743	0.5767	0.5728	**0.5768**
	JODIE	0.5680	0.5703	**0.5755**	0.5744	0.5731
	MNCI	0.5802	0.5811	**0.5874**	0.5809	0.5807

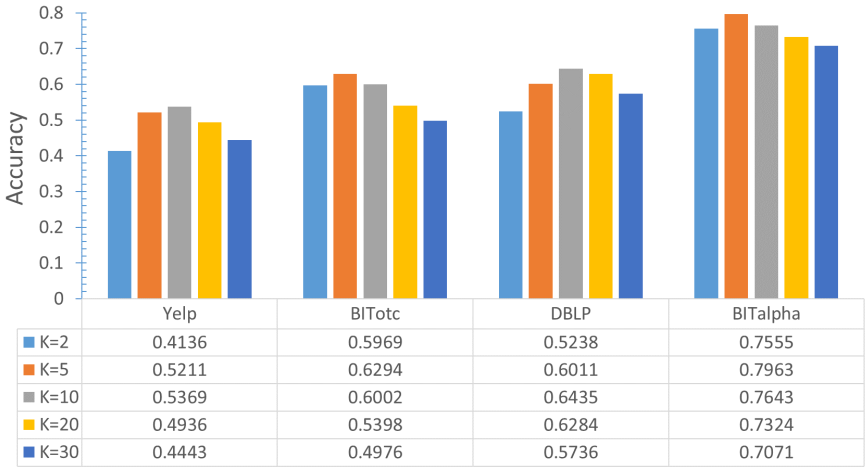

Figure 1: Parameter sensitivity experiment of community number K

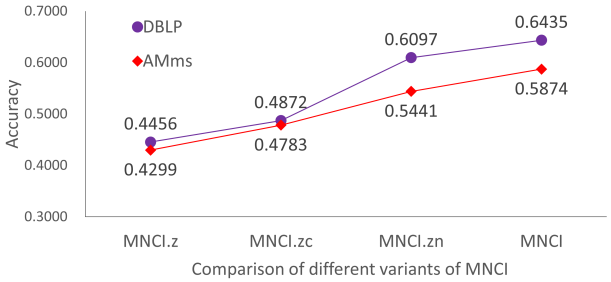

Figure 2: Ablation study of community and neighborhood influences on node classification

than MNCI.zc with community influence. It means that both neighborhood and community influences are effective, and neighborhood influence is more important than community influence.

Comparing the two datasets' performance, we discover that the performance improvement of neighborhood influence on DBLP is greater than that on AMms. We believe this phenomenon is due to the following reasons. As shown in Table 1, the average degree of each node in DBLP and AMms is 16.87 and 1.20, respectively. It means that nodes in DBLP are more sensitive to neighborhood neighborhood influence than nodes in AMms, because nodes in DBLP have more interactions with neighbors. Therefore, when we consider neighborhood influence, the performance improvement of MNCI on DBLP is greater than that on AMms.

REFERENCES

[1] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *KDD* (2016), 855–864.
[2] L. William Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *NIPS* (2017), 1024–1034.
[3] A. James Hanley and J. Barbara McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* (1982), 29–36.
[4] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 333–341.
[5] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *Data Mining*

(ICDM), 2016 IEEE 16th International Conference on. IEEE, 221–230.

[6] Jiacheng Li, Yujie Wang, and J. Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining Houston TX USA February, 2020* (2020), 322–330.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* (2013).

[8] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fined-Grained Aspects. *EMNLP/IJCNLP (1)* (2019), 188–197.

[9] Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. 2014. DeepWalk: online learning of social representations. *KDD* (2014), 701–710.

[10] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep - Learning Representations over Dynamic Graphs. *ICLR* (2019).

[11] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding Temporal Network via Neighborhood Formation. *KDD* (2018), 2857–2866.