Spherical Graph Embedding for Item Retrieval in Recommendation System

Wenqiao Zhu
zhuwenqiao.ai@bytedance.com
Bytedance Inc.
Beijing, China

Yesheng Xu
xuyesheng@bytedance.com
Bytedance Inc.
Beijing, China

Xin Huang
hx@bytedance.com
Bytedance Inc.
Beijing, China

Qiyang Min
minqiyang@bytedance.com
Bytedance Inc.
Beijing, China

Xun Zhou
zhouxun@bytedance.com
Bytedance Inc.
Beijing, China

ABSTRACT

One of the challenging problems in large-scale recommendation systems is to retrieve relevant candidates accurately and efficiently. Graph-based retrievals have been widely deployed in industrial recommendation systems. Previous graph-based methods depend on integrated graph infrastructures because of inherent data dependency in graph learning. However, it could be expensive to develop a graph infrastructure. In this paper, we present a simple and effective graph-based retrieval method, which does not need any graph infrastructures. We conduct extensive offline evaluations and online tests in a real-world recommendation system. The results show that the proposed method outperforms the existing methods. The source code of our algorithm is available online[1].

CCS CONCEPTS

• **Information systems → Novelty in information retrieval**.

KEYWORDS

Graph Embedding; Graph-Based Retrieval; Recommendation Systems

ACM Reference Format:
Wenqiao Zhu, Yesheng Xu, Xin Huang, Qiyang Min, and Xun Zhou. 2022. Spherical Graph Embedding for Item Retrieval in Recommendation System . In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557704

1 INTRODUCTION

Recommendation systems are important in various commercial platforms. The objective of these systems is to connect users to relevant items from a large corpus based on user features and behaviors. Due to the large size of users and items, many industrial systems consist of a retrieval stage and a ranking stage. The purpose of a retrieval stage is to retrieve a small fraction of relevant items at a low computational cost and latency, and a ranking stage is deployed to further refine the ranking of these items according to the user's interest.

Item-based collaborative filtering (Item-CF) [8] is one of the successful retrieval methods in recommendation systems. It considers the similarities between items and items based on their co-occurrence patterns. Recently, graph-based retrieval methods have been widely adopted in industrial recommendations. Pinterest deploys a data-efficient Graph Convolutional Network (GCN) algorithm called Pinsage [13], which combines efficient random walks and graph convolutions to generate embeddings of nodes. Alibaba proposes a Path-based Deep Network (PDN) [4], which incorporates both personalization and diversity to enhance retrieval performance. PDN models the user-item relationship with a 2-hop graph.

Graph-based retrieval methods learn a low-dimensional embedding for each node in a graph and consider the similarities between node embeddings. Previous graph embedding methods, such as DeepWalk [5], node2vec [2], and GraphSAGE [3], usually need to sample multi-hop neighbors to achieve better performances. Due to the multi-hop data dependency and large-scale graph data, industrial recommendation systems need to incorporate a graph service for graph learning. However, it could be expensive to develop such a graph service. One needs to consider efficiency, scalability as well as fault tolerance.

Capturing a node's position within a graph is crucial for recommendation systems. Two users that view a lot of similar items should have a small distance in the embedding space. Many previous graph embedding methods are structure-aware instead of distance-aware [14]. DeepWalk [5] and node2vec [2] are distance-aware methods. However, they do not explicitly model the distance metric in the objective function, which limits their distance-aware property. In our method, we design an objective function that explicitly considers the distance metric in spherical space.

In this paper, we present a simple graph-based retrieval method, which does not need a graph service and can support online graph embedding learning. To reduce the dependency of multi-hop neighbors in graph learning, we apply only one-hop neighbors for each node. Furthermore, we model the graph learning task in a spherical space to improve the performance. We empirically find that our

[1] https://github.com/WNQzhu/Q-align

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557704

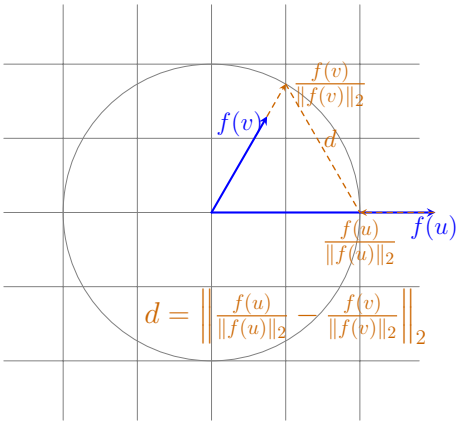

Figure 1: Previous graph embedding methods model the neighbor interaction via inner-product in Euclid space. Our method model the neighbor interaction via distance in spherical space.

method achieves better results than other multi-hop-based methods, such as DeepWalk and node2vec.

2 METHOD

2.1 Spherical Graph Embedding

We assume a user-item graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ are node set and edge set respectively. The node in graph $\mathcal{G}$ could be a user or an item. A graph embedding method aims to learn a mapping function $f : \mathcal{V} \rightarrow R^d$ that projects each node v_i to a vector $f(v_i)$ in a d-dimensional ($d \ll N$) space. We define $N_K(u)$ as a K-hop neighborhood of u.

Graph embedding methods aim to maximize the log-probability of observing a neighborhood $N_K(u)$ of node u given its representation $f(u)$:

$$\max_f \sum_{u \in \mathcal{V}} \log Pr\left(N_K(u) | f(u)\right). \tag{1}$$

Where

$$Pr\left(N_K(u) | f(u)\right) = \prod_{n_i \in N_K(u)} Pr(n_i | f(u)) \tag{2}$$

$$Pr(n_i | f(u)) = \frac{exp\left(f(n_i) \cdot f(u)\right)}{\sum_{v \in \mathcal{V}} exp\left(f(v) \cdot f(u)\right)} \tag{3}$$

Function (2) and (4) are known as conditional independence and symmetry assumptions.

Previous methods model objective functions in Euclid space and depend on multi-hop neighbor samplings. In this work, we model the objective function in spherical space. Many works [12] from other domains show that learning in spherical space achieves better results. As shown in Figure 1, we define the symmetry assumption function in spherical space as:

$$Pr(n_i | f(u)) = \frac{exp\left(-d^2(n_i, u)\right)}{\sum_{v \in \mathcal{V}} exp\left(-d^2(v, u)\right)} \tag{4}$$

where

$$d(n_i, u) = \left\| \frac{f(n_i)}{\|f(n_i)\|_2} - \frac{f(u)}{\|f(u)\|_2} \right\|_2$$

Under the conditional independence and symmetry assumptions, the objective (1) is equivalent to:

$$\min_f \sum_{u \in \mathcal{V}} \left[\alpha \sum_{n_i \in N_K(u)} d^2(n_i, u) + \beta \log Z_u \right] \tag{5}$$

$$Z_u = \sum_{v \in \mathcal{V}} exp\left(-d^2(u, v)\right) \tag{6}$$

Z_u is the per-node partition function. Following the parameter settings in [12], we introduce two hyper-parameters α and β in our objective function. To reduce the data dependency between nodes, we set $K = 1$ in our implementation, and each node can only sample its 1-hop neighbors, To minimize the function (5), we approximate it using the in-batch negative sampling method and optimize it using the stochastic gradient descent method.

2.2 System

Many industrial recommendation systems apply online learning methods. All the user behaviors are converted to training instances immediately for further training. Generally, a single training instance can be formulated as:

$$(\textit{user id}, \textit{item id}, \textit{features}, \textit{label})$$

The *features* consist of user features, item features, and context features. A *label* indicates a user's behavior on an item. For example, we can define *label* = 1 if a user clicks an item, and *label* = 0 if the user does not click an item.

Figure 2 shows the typical retrieval stage in an industrial recommendation system. Most industrial recommendation systems deploy multiple retrieval modules in parallel. We use *Other Retrievals* to denote other retrieval modules and detail the graph-based retrieval in Figure 2 since we focus on graph-based retrieval in this work. From the training instances or all users' behaviors in a time interval, one can generate a user-item heterogeneous graph via *Graph Builder*, which could be a MapReduce job. For example, if a user has clicked an item, there could be an edge between them in the graph. The *Graph Builder* usually runs daily to generate graph data since the real industrial graph data could be huge. The *Graph Service* module takes the generated graph data as input, and the *Trainer* module samples nodes and edges from the *Graph Service* for graph embedding learning.

However, the previous graph-based retrieval methods have a few limitations. First, one needs to implement a graph service with large memory and high bandwidth networks. The properties of fault tolerance also should be considered. The multi-hop data dependency and the graph scale lead to the need of a graph service.

In our method, we model the graph embedding learning in spherical space and restrict the data dependency to be one-hop. Based on this idea, we design a graph-based retrieval without resorting to a graph service, as shown in Figure 3. In the training instances, we introduce a new feature *recent item id*, which is the item a user recently clicked. In a positive training instance (label=1), both the *user id* and the *recent item id* could be viewed as a one-hop neighbor of the *item id* in an implicit graph. We filter out all the negative

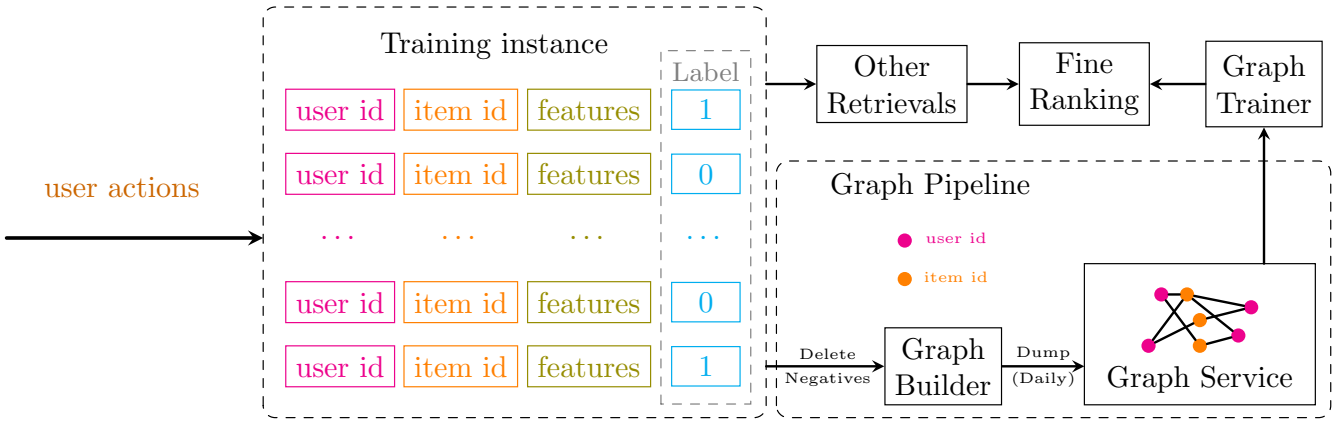

Figure 2: Previous graph embedding methods rely on a heavy graph pipeline.

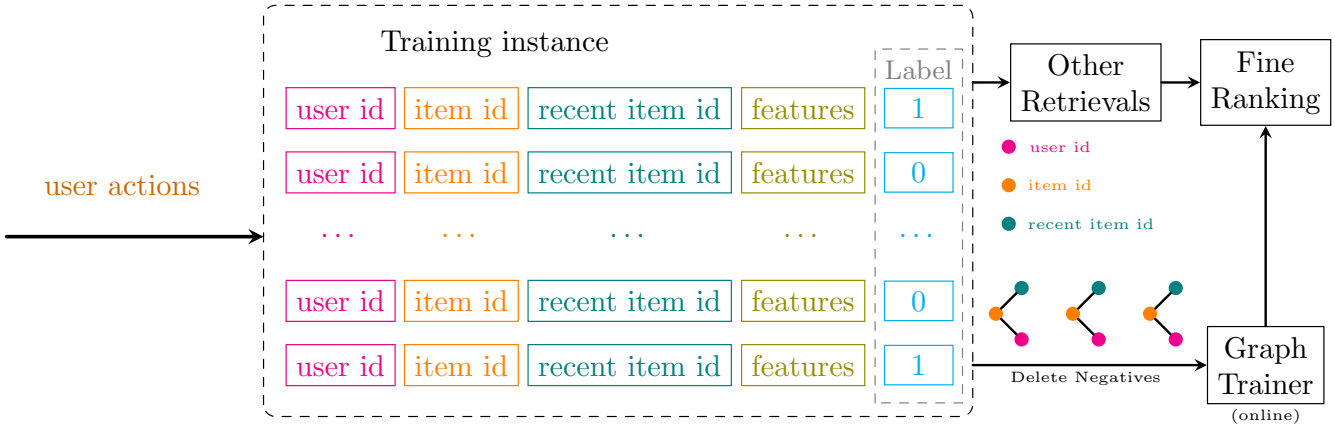

Figure 3: Our method supports online graph embedding learning.

training instances, and the graph embedding trainer updates its parameters based on the (*item id*, *user id*) and (*item id*, *recent item id*) pairs, which are the edges in an implicit graph. In this way, we do not need a graph service, and the learning method is online.

3 EXPERIMENTS ON PUBLIC DATASETS

In this section, we evaluate the proposed approach on the unsupervised node classification task to answer the following questions: How does our one-hop-based graph embedding method perform compared to other methods? The *CogDL* toolkit [1] and the following datasets are used for evaluation.

- BlogCatalog [15] is a social blogger graph. It has 10,312 nodes, 333,983 edges, and 39 different labels. The node labels are the Bloggers' interests.
- DBLP [10] is an academic citation graph where authors are treated as nodes. It has 51,264 nodes, 127,968 edges and 60 different labels. The node labels are the authors' dominant conferences.
- Flickr [11] is a user contact graph in Flickr. It has 80,513 nodes, 5,899,882 edges and 195 different labels. The node labels represent user's interest groups.

We use SGD with momentum as the optimizer. The learning rate is set with starting value of 0.025 and decays linearly to 0.0001. The embedding size is set to 128. Table 2 shows the other parameters used by our method. For the other methods, we used the parameters provided by the CogDL Toolkit [1] with the option *use_best_config=True*. When evaluating node2vec and DeepWalk, we set the walk length to 80. We randomly sample a portion of labeled nodes for training and use the rest for testing. For Blogcatalog, the training ratio is set to 50%. For DBLP and Flickr, the training ratio is set to 5%. We run our method 5 times, and report the mean result in table 1. From the results, we find that our method outperforms node2vec and DeepWalk. The node2vec and DeepWalk both are multi-hop-based methods. Compared with the one-hop-based LINE method, our method achieves a significant performance improvement.

We further compare our method with the matrix factorization(MF) based methods. From table 1, we can find that MF-based methods

Table 1: Micro-F_1 scores for multi-label node classification on Blogcatalog, DBLP and Flicker graphs.

Method	Blogcatalog (50%)	DBLP (5%)	Flickr (5%)
NetMF [7]	42.47±0.35	56.72±0.14	36.27±0.17
ProNE [16]	41.14±0.26	56.85±0.28	36.56±0.11
NetSMF [6]	40.62±0.35	59.76±0.41	35.49±0.07
node2vec [2]	40.16±0.29	57.36±0.39	36.13±0.13
LINE [9]	38.06±0.39	49.78±0.37	31.61±0.09
DeepWalk [5]	40.48±0.47	57.54±0.32	36.09±0.10
Our method	**42.63**±0.47	**60.32**±0.10	**36.75**±0.11

like NetMF[7] and ProNE[16] are very powerful as they achieve better results than previous graph embedding based methods. However, our method outperforms the MF-based methods on all datasets.

Table 2: The parameters used by our method.

Dataset	batch size	α	β
Blogcatalog	128	0.5	0.75
DBLP	256	0.5	0.1
Flickr	512	0.5	1.5

4 LIVE EXPERIMENTS

In this section, we show our experiments in a real-world news-feed application. It has hundreds of millions of users. We report both the offline evaluation and the online A/B testing results.

4.1 Offline Evaluation

For offline evaluation, we build a user-item graph for training from all users' behaviors in one day and construct a graph for testing from all users' behaviors the next day. After training, we apply the generated item embedding to evaluate the similarities between the items in the testing graph. Specifically, we use Mean Reciprocal Rank (MRR), which is a measure to evaluate systems that return a list of recommended items for query items:

$$MRR = \frac{1}{Q}\sum_{1}^{Q}\frac{1}{rank_i}$$

Due to the large pool of items, we roughly estimate the rank in the following way. For each item q, we sample a neighbor item p from the testing graph and randomly sample a batch of items $(n_1, n_2, \cdots, n_b)$ from the item pool, where $b = 512$. The rank is estimated by sorting the score list $sim(q, p), sim(q, n_1), \cdots, sim(q, n_b)$, where $sim(\cdot, \cdot)$ is a function to compute the inner product of the normalized item embeddings. We set the batch size to be 512 for all the methods. For node2vec, we use $p = 1$, $q = 1$, and set the walk length to be 20. For GraphSAGE, we use sample sizes S_1=10 and S_2 = 10 for first and second hop sampling, respectively.

Table 3 compares the performance of the various approaches using the MRR. Our method achieves the best performance at 0.0974 MRR. The results further show that the one-hop-based graph embedding method with a carefully designed objective function can perform as powerful as multi-hop-based graph embedding methods.

Table 3: MRR for our method and other baselines.

Method	MRR
node2vec [2]	0.0955
LINE [9]	0.0874
GraphSAGE [3]	0.0843
Our method	**0.0974**

Table 4: Online A/B testing results. An improvement of 0.2% in Duration/U is regarded as significant in our system.

Method	Read/U (%)	Duration/U (%)	Latency (%)
+LINE [9]	+0.198	-0.0457	+0.923
+node2vec [2]	+0.257	+0.0611	+0.962
+Our method	**+0.266**	**+0.254**	+0.953

4.2 Online A/B Experiment

We introduce a new graph-based retrieval module to our news-feed recommendation system and conduct an online A/B test. The introduced graph-based retrieval module is not to replace the existing retrieval modules. It works with them in parallel since there usually are many retrieval modules in the industrial recommendation system for different purposes.

For node2vec and LINE, we apply the graph pipeline shown in figure 2 since they need multi-hop sampling or global negative sampling. The graph data gets updated at 2:00 a.m. each day using all the users' behaviors during the past two days. The proposed method applies the pipeline shown in figure 3 since it only needs one-hop neighbor sampling. Furthermore, our approach can be independent of graph service and can support online graph learning. 20% of online traffic is applied to evaluate each method.

We show the effects of each approach on the business metric in Table 4. Read/U and Duration/U are two important business metrics in our system. Read/U represents the number of readings per user, and Duration/U means how long per user stays in our application. We can observe that introducing a graph-based retrieval in our system can improve the Read/U metric. However, the performances on the Duration/U metric are different. Our method achieves the best on both the Read/U and Duration/U metrics. We deploy the proposed approach in our recommendation system to improve the user experience.

5 CONCLUSIONS

In this paper, we present a graph-based retrieval method for recommendation systems. It is independent of graph service and ease-to-deploy. We conduct extensive experiments to evaluate our approach. Both the offline experiments and online A/B tests show the effectiveness of our method. In our experiments, we only use the user id and item id for graph embedding learning. For future work, we will utilize the features in the training instance.

REFERENCES

[1] Yukuo Cen, Zhenyu Hou, Yan Wang, Qibin Chen, Yizhen Luo, Xingcheng Yao, Aohan Zeng, Shiguang Guo, Peng Zhang, Guohao Dai, Yu Wang, Chang Zhou, Hongxia Yang, and Jie Tang. 2021. CogDL: Toolkit for Deep Learning on Graphs. *arXiv preprint arXiv:2103.00959* (2021).

[2] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks.. In *KDD*.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[4] Houyi Li, Zhihong Chen, Chenliang Li, Rong Xiao, Hongbo Deng, Peng Zhang, Yongchao Liu, and Haihong Tang. 2021. *Path-Based Deep Network for Candidate Item Matching in Recommenders.* Association for Computing Machinery, New York, NY, USA, 1493–1502. https://doi.org/10.1145/3404835.3462878

[5] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[6] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1509–1520. https://doi.org/10.1145/3308558.3313446

[7] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and Node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 459–467. https://doi.org/10.1145/3159652.3159706

[8] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[9] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. WWW - World Wide Web Consortium (W3C). https://www.microsoft.com/en-us/research/publication/line-large-scale-information-network-embedding/

[10] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA) *(KDD '08)*. Association for Computing Machinery, New York, NY, USA, 990–998. https://doi.org/10.1145/1401890.1402008

[11] Lei Tang and Huan Liu. 2009. Relational Learning via Latent Social Dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 817–826. https://doi.org/10.1145/1557019.1557109

[12] Tongzhou Wang and Phillip Isola. 2020. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *International Conference on Machine Learning*. PMLR, 9929–9939.

[13] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 974–983. https://doi.org/10.1145/3219819.3219890

[14] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7134–7143. https://proceedings.mlr.press/v97/you19b.html

[15] R. Zafarani and H. Liu. 2009. Social Computing Data Repository at ASU. http://socialcomputing.asu.edu

[16] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*. https://www.microsoft.com/en-us/research/publication/prone-fast-and-scalable-network-representation-learning/