Life is short, use everything2vec

Yikai Wang
School of Data Science
Fudan University
15300180076@fudan.edu.cn

Dan Wu

Zheng Wei

I. Task Review

A. Task 1

Design clustering algorithms or community mining algorithms to cluster all the papers in the data set. Use visualize tools to show all fields (ie, communities, identify corresponding community research topics), and highlight the most influential scholars in each field.

B. Task 2

Realization of demonstrating the ego-network to any input scholar (refer to the function example on the ArnetMiner website).

C. Task 3

Use the data provided by DBLP and ArnetMiner to analyze and model more social relationships among scholars, such as predicting the cooperation or citation relationship between two scholars, and predicting which conference will a scholar publish papers on in the future.

II. Division of Work

1) Yikai Wang:
   In brief, he applies several methods in network representation learning and uses and expands a hierarchical representation learning algorithm(HARP) for Networks. For homogeneous networks, he mainly uses deepwalk, node2vec and LINE. For heterogeneous networks, he uses metapath2vec++. For task 1, he uses both homogeneous NRL and heterogeneous NRL methods to extract features for clustering and uses KMeans and Birch to complete the clustering task. For task 3, he uses heterogeneous NRL methods to generate the vector representation of scholar cooperation network, scholar citation network and scholar-conference network.
2) Dan Wu:
3) Zheng Wei:

III. Methodology

A. Network Representation Learning

Network is an important form of expressing the relationship between objects and objects. A key issue for the analysis of networks is to study how to reasonably represent feature information in the network. With the development of machine learning technology, feature learning for nodes in the network has become an emerging research task. Network representation learning algorithm transforms network information into low-dimensional dense real vector.

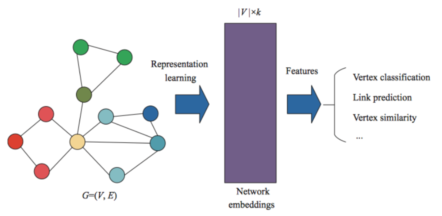

Fig. 1. Network representation learning flowchart

There are two types of networks in NRL, one supposes that all nodes in the network have the same type, and the other supposes that all nodes have different types. The former network is called homogeneous network and the other is called heterogeneous network.

Formally, let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. The goal of network representation learning is to develop a mapping function $\Phi : V \rightarrow R^{|V|*d}, d \ll |V|$, which defines the latent representation of each node $v \in V$. Here we briefly introduce some homogeneous network learning methods and detailed introduce a heterogeneous network learning method and an optimization algorithm on all these methods.

Methods for homogeneous network learning

1) DeepWalk:
   The idea of DeepWalk [1] is borrowed from word2vec. It uses short random walks to generate paths of a node, then just like word2vec, it uses the path to generate a probability of its neighbors, which could give each node a vertices in networks.
2) Node2vec:
   Node2vec [2] is an improvement for DeepWalk. In short, it regards the random walk as a searching problem. It provides a way of balancing the exploration-exploitation tradeoff that in turn leads to representations obeying a spectrum of equivalences from homophile to structural equivalence.
3) LINE:
   LINE [3] uses another way to realize network embedding. The main idea of this method is learning a low-dimensional embedding with preserving both the first-

order proximity and the second-order proximity between the vertices.

Methods for heterogeneous network learning

Different with homogeneous networks, heterogeneous networks involve more than one node types and relationships between the same type of nodes and/or different types of nodes. Thereafter, these networks cannot be handled by representation learning models that specifically designed for homogeneous networks. In this project, we use a heterogeneous skip-gram model, *metapath2vec*++ [4], to model the heterogeneous neighborhood of a node.

Specifically, in *metapath2vec*++, we enable skip-gram to learn effective node representations for a heterogeneous network $G = (V, E, T)$ with $|T_V| > 1$ by maximizing the probability of having the heterogeneous context $N_t(v), t \in T_V$ given a node v:

$$argmax_\theta \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} logp(c_t|v; \theta) \tag{1}$$

where $N_t(v)$ denotes v's neighborhood with the t^{th} types of nodes and $p(c_t|v;\theta)$ is commonly defined as a softmax function, that is: $p(c_t|v;\theta) = \frac{exp\{X_{c_t} * X_v\}}{\sum_{u_t \in V_t} exp\{X_{u_t} * X_v\}}$, where X_v is the v^{th} row of X, representing the embedding vector for node v. V_t is the node set of type t in the network.

However, the method will cost a terrible long time. To solve the problem, negative sampling was introduced by Mikolov et al [5]. Using this method, we just need to sample a small set of nodes from the network in order to construct softmax. Specifically, given a negative sample size M, we use following method to update Equation 1:

$$log\sigma(X_{c_t} * X_v) + \sum_{m=1}^{M} E_{u_t^m \sim P_t(u_t)}[log\sigma(-X_{u_t^M} * X_v)] \tag{2}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and $P(u)$ is the pre-defined distribution from which a negative node u^m is drew from for M times. In our model, we regard different types of nodes homogeneously and do not distinguish them when drawing negative nodes. The gradients of equation 2 are derived as follows:

$$\begin{aligned} \frac{\partial O(X)}{\partial X_{u_t^m}} &= (\sigma(X_{u_t^m} X_v - I_{c_t}[u_t^m]))X_v \\ \frac{\partial O(X)}{\partial X_v} &= \sum_{m=0}^{M} (\sigma(X_{u_t^m} X_v - I_{c_t}[u_t^m]))X_{u_t^m} \end{aligned} \tag{3}$$

Having defined the skip-gram model, the problem we need to solve is how to effectively transform the heterogeneous network into skip-gram. Similar to homogeneous network, we can use random walk to generate paths of multiple types of nodes. Different with homogeneous network, in heterogeneous network, we need to consider the type of the nodes in the random walk.

Formally, a meta-path scheme P is defined as a path that is denoted in the form of $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \dots V_t \xrightarrow{R_t} V_{t+1} \dots \xrightarrow{R_{l-1}} V_l$, wherein $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ defines the composite relations between node types V_1 and V_l. Thus we could show how to use meta-paths to guide heterogeneous random walkers. Given a heterogeneous network $G = (V, E, T)$ and a meta-path scheme P, the transition probability at step i is as follows:

$$p(v^{i+1}|v_t^i, P) = \begin{cases} \frac{1}{|N_{t+1}(v_t^l)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases} \tag{4}$$

where $v_t^i \in V_t$ and $N_{t+1}(v_t^i)$ denote the V_{t+1} type of neighborhood of node v_t^i. Further more, meta-paths are commonly used in a symmetric way, which means its first node type is the same as the last one. That is:

$$p(v^{i+1}|v_t^i) = p(v^{i+1}|v_1^i), \text{if } t = l \tag{5}$$

The complete algorithm is expressed as follows:

Algorithm 1: The metapath2vec++ Algorithm.

```
Input: The heterogeneous information network
       $G = (V, E, T)$, a meta-path scheme $P$,# walks
       per node $w$, walk length $l$, embedding
       dimension $d$,neighborhood size $k$
Output: The latent node embeddings $X \in R^{|V|*d}$
1  initialize $X$;
2  for $i = 1; i \le w$ do
3      for $v \in V$ do
4          MP=MetaPathRandomWalk($G, P, v, l$);
5          X=HeterogeneousSkipGram($X, k, MP$);
6  return $X$;
7  MetaPathRandomWalk($G, P, v, l$) MP[1]=v;
8  for $i = 1; i < l$ do
9      draw u according to Eq. 4;
10     MP[i+1]=u;
11 return MP;
12 HeterogneousSkipGram($X, k, MP$) for $i = 1; i \le l$
     do
13     v = MP[i];
14     for $j = max(0, i-k); j \le min(i+k, l); j \neq i$ do
15         $c_t = MP[j]$;
16         $X^{new} = X^{old} - \eta \frac{\partial O(X)}{\partial X}$(Eq. 3;
```

HARP

The network embedding method we introduced above is very useful in some situations. However, there are two main disadvantages for these methods:

1) all the models do not involve high-order network structural information
2) their stochastic optimization can fall victim to poor initialization

Thus, we change the traditional problem into hierarchical representation learning problem. The main idea is we seek to find a graph $G_s = (V_s, E_s)$ which captures the essential structure of G, but is much smaller(i.e. $|V_s| \ll |V|, |E_s| \ll |E|$). It is trivial that G_s is easier to embed. Since we have much less relationships, which means the mapping could be much smoother. Further more, since the G_s is much smaller, the models that focus on local structure now could have a better performance on global structure's representation.

Our method for multilevel network representation learning, HARP [6], consists of three parts-graph coarsening, graph embedding and representation refinement.

1) Graph Coarsening:
   Given a graph G, graph coarsening algorithms create a hierarchy of successively smaller graphs $G_0, G_1, \ldots, G_L$, where $G_0 = G$. The coarser graphs preserve the global structure of the original graph and have much fewer nodes and edges.
2) Graph Embedding on the Coarsest Graph:
   Using provided network embedding algorithms to to graph embedding.(DeepWalk, Node2vec, LINE, etc.)
3) Graph Representation Prolongation and Refinement:
   We prolong and refine the graph representation from the coarsest to the finest graph. For each graph G_i, we use the graph representation of G_{i+1} as its initial embedding and refine the graph embedding.

The complete algorithm is expressed as follows:

Algorithm 2: HARP(G,Embed())

```
Input: network $G = (V, E)$, arbitrary network
       embedding algorithm EMBED()
Output: node embeddings $\Phi \in R^{|V|*d}$
1  $G_0, G_1, \ldots, G_L \leftarrow$GRAPHCOARSENING(G);
2  Initial $\Phi'_{G_L}$ by assigning zeros;
3  $\Phi_{G_L} \leftarrow EMBED(G_L, \Phi'_{G_L})$ for $i = L-1; i \geq 0$ do
4      $\Phi'_{G_i} \leftarrow PROLONGATE(\Phi_{G_{i+1}}, G_{i+1}, G_i)$;
5      $\Phi_{G_i} \leftarrow EMBED(G_i, \Phi'_{G_i})$
6  return $\Phi_{G_0}$;
7  GraphGoarsening($G(V, E)$) $L \leftarrow 0$;
8  $G_0 \leftarrow G$;
9  while $|V_L| \geq threshold$ do
10     $L \leftarrow L + 1$;
11     $G_L \leftarrow$EDGECOLLAPSE(STARCOLLAPSE(G));
12 return $G_0, G_1, \ldots, G_L$;
```

In the algorithm, **Edge Collapse** is an algorithm for preserving first-order proximity when coarsening the network. For edges E, it will select $E' \subset E$, satisfying no two edges in the subset are incident to the same vertex. That is, for each $(u_i, v_i) \in E'$, it merges the pairs into a single node w_i, and merge the edges incident to u_i and v_i. **Star Collapse** is an algorithm for preserving second-order proximity when coarsening the network. It will merges nodes with the same neighbors into super-nodes. For example, in the Figure 2,$(v_1, v_2), (v_3, v_4)$ and (v_5, v_6) are merged into super-nodes as they share the same neighbors (v_7).

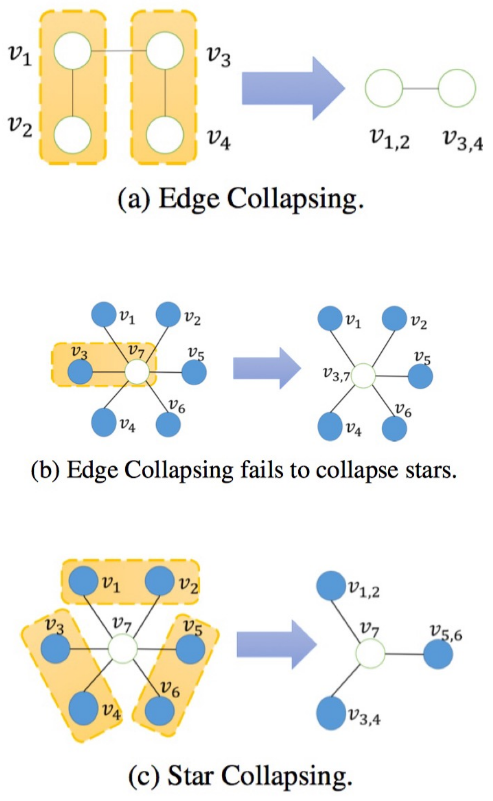

Fig. 2. Illustration of graph coarsening algorithms

It is worth mentioning that in the model of HARP, they just consider homogeneous networks. However, without much modify, it can be used in heterogeneous networks, too. Specifically, in the algorithm, we first do star collapse and then do edge collapse. For a heterogeneous network, since the edges are mostly between two different types of nodes, we could find that similar nodes with the same type will be regard as the same super-node, which is just what we hope to happen.

IV. Experiment

A. Data Set

Our data are selected from Arnetminer [7]. We choose all papers published on 21 conferences until 2017. A summary of our data are shown in Table I:

B. Task 1

The first part of task 1 is clustering. We need to cluster all the papers with the information involving authors' names, references, title, abstract and year. We tried several methods to make use of the information. Specifically, word2vec uses title and abstract to generate a vector. DeepWalk and LINE uses the citation relationship of papers. Metapath2vec++ regards papers and authors as node and generate vectors. The accuracy of our methods are shows in Table II. From the result we

Field	Conferences	Number of Papers	Number of scholars
Distributed & Parallel Computing	PPOPP,PACT,IPDPS,ICPP	Number of Papers	Number of scholars
Natural Language Processing	ACL,EACL,COLING, EMNLP		
Data Mining	ICDE,SIGMOD,KDD,ICDM		
Computer Education	AIED,ITS,ICALT		
Machine Learning	IJCAI,ICML,NIPS		
Operating Systems & Simulations	MASCOTS,SOSP,OSDI		

TABLE I
A SUMMARY OF DATA.

could find that the information between papers and authors are more important the relationship between papers and papers. Further, our HARP model could improve the accuracy of about 5 percent. It also shows that KMeans is better than Birch in some sense. And word2vec seems useless from our result.

Model/Cluster Algorithm	KMeans	Birch
word2vec	44.25%	45.13%
DeepWalk	53.08%	53.85%
LINE	50.44%	38.82%
metapath2vec++	66.39%	59.69%
HARP(metapath&DeepWalk)	**71.66%**	61.24%
HARP(metapath&line)	67.05%	53.49%
DeepWalk+word2vec	52.26%	52.24%
LINE+word2vec	51.21%	36.77%
(metapath2vec++)+word2vec	49.88%	58.12%
HARP(metapath&DeepWalk)+word2vec	51.72%	55.39%
HARP(metapath&line)+word2vec	60.55%	53.15%

TABLE II
ACCURACY OF MODELS IN CLUSTERING TASK.

C. Task 2

D. Task 3

For this task, we construct several networks. A trivial idea is that we use the whole network, which involves edges between papers and papers(reference), papers and authors, papers and conferences. Another idea is just consider the relationship we want to survey. In the cooperation task, we just construct a network which involves edges between papers and authors. In the citation task, we just construct a network which involves edges between papers and papers, papers and authors. And in publish task, we just construct a network with edges involve relationship between authors and conferences.

To quantify the accuracy of prediction algorithms. We use a method called area under the receiver operating characteristic curve(AUC) [8].

Definition of AUC: Provided the rank of all non-observed links, the AUC values can be interpreted as the probability that a randomly chosen missing link is given a higher score than a randomly chosen nonexistent link. To save time, we calculate the score of each non-observed link instead of giving the ordered list. At each time, we randomly pick a missing link and a nonexistent link to compare their scores. If among n independent comparisons, there are n_1 times the missing link having a higher score and n_2 times they have the same score, the AUC value is:

$$AUC = \frac{n_1 + 0.5n_2}{n} \tag{6}$$

If all the scores are generated from an independent and identical distribution, the AUC value should be about 0.5. Thereafter, the degree to which the value exceeds 0.5 indicates how much better the algorithm performs than pure chance.

In our experiments, we use the cosine distance between two vectors as the score of two nodes. The result is shown in Table III

Model/task	cooperation	citation	publish
metapath2vec++	75.39%	66.92%	77.86%
HARP(DeepWalk)	83.38%	**87.80 %**	87.76%
HARP(LINE)	83.73%	76.93%	67.32%
HARP(DeepWalk)(whole net)	**85.72%**	85.30%	**88.50%**
HARP(LINE)(whole net)	56.86%	56.43%	54.81%

TABLE III
AUC OF TASK3

V. CONCLUSION AND DISCUSSION

REFERENCES

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, pp. 701–710. [Online]. Available: http://doi.acm.org/10.1145/2623330.2623732

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[4] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[6] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," *arXiv preprint arXiv:1706.07845*, 2017.

[7] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 990–998.

[8] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.