VE444 Mile Stone

Group 18

Nov. 2020

Ge Jintian	517021911142
Ye Roushuang	516370910241
Xia Binyu	517370910012

1 Introduction

Nowadays, relationships between people and people become more and more important. Sometimes, if there are two groups, we may be interested in whether these two groups can be connected in some ways. For example, assume that there are two classes in JI, class No.1 and class No.2. Then, all students in class 1 will form a group, while all students in class 2 will form another group. Usually, some students in class 1 should have already known some students in class 2. Now, if the class assistant wants to introduce students in class 1 to students in class 2, how can he increase the probability that two students who are introduced to each other will become friends? If we can find out some potential friendships between students in class 1 and students in class 2, then it will be easier to make sure that two students who are introduced to each other will become friends. Such a technique can be applied on many other fields. For instance, in online communication platform, it can be used to recommend possible friends to someone.

This problem can be solved by using knowledge from network. From the perspective of network, we use vertices to represent individuals and use edges to indicate whether two individuals have a certain relationship. This relationship depends on the type of network. For instance, vertices in social network stand for people and edges will indicate that two connected people are friends, as shown in Figure 1. Then, this problem can be abstracted into a general problem in network: assume there are two networks which are of the same type, but only some individuals in one network are connected to some individuals in another network. Then, according to these edges, we want to find out all possible edges between these two networks. We call this as "merging two networks together according to existing edges".

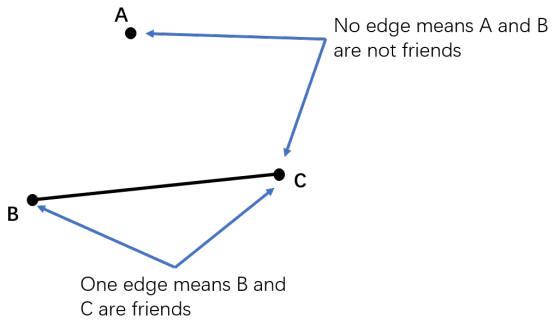

Figure 1: Social network

The problem is described in detail in Figure 2. There are two networks of the same type, A and B. We can see that there are only a few edges between A and B. The problem is that how can we merge A and B together. During this process, if vertex a_1 and b_1 have potential relationship (indicated by a virtual line), we will capture it and link them together. If two vertices don't have a potential relationship, like node a_2 and node b_2, we will try to avoid linking them together. Our project aims at building an algorithm to solve such a problem. We will use node embedding method to provide information for machine learning, and use machine learning to solve this problem.

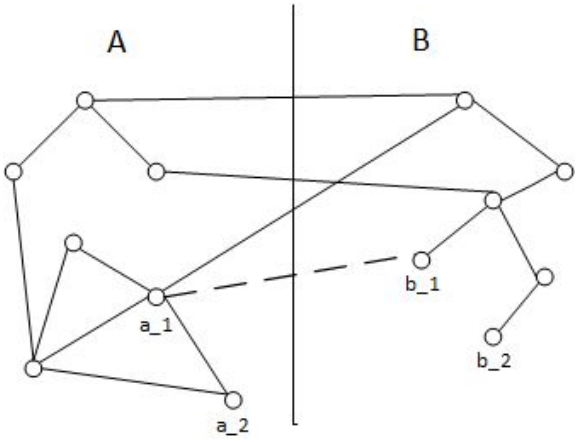

Figure 2: Network description

We plan to evaluate our algorithm using a dataset from Facebook. This dataset will be described clearly later. The dataset contains a whole network. We plan to divide this network into two separated networks. We will break all edges between these two networks and remember them. Then we will try to merge these two networks together and see if we restore original edges successfully. The evaluation method will be described in detail in Algorithm Proposal part.

2 Related works

2.1 Efficient Estimation of Word Representations in Vector Space. (Mikolov, 2013)

This paper introduces the famous "Word2Vec" Skip-gram model originally designed for word embedding by implicitly factorizing a matrix of shifted pointwise mutual information of word co-occurrence statistics. This model is proved to have a high quality with good accuracy and low computational cost with very large data sets. Since the model inspires several algorithms focus on neighborhood-preserving node embedding, it can help us have a better understanding of the node embedding and make it easier for us to furtherly learn other related algorithms and models.

2.2 Node2vec: Scalable Feature Learning for Networks. (Grover, 2016)

This paper introduces "Node2Vec" – a more familiar algorithmic framework for us for learning continuous feature representations of nodes in networks. Node2Vec gives a mapping of nodes to a low-dimensional space of features maximizing the likelihood of preserving network neighborhoods of nodes. We can learn a flexible notion of nodes' network neighborhood and a biased random walk procedure, which helps efficiently explores diverse neighborhoods. After reading the paper, we can better capture the diversity of connectivity patterns in the networks and have a deeper understanding of nodes representations, which will help us advance in node classification, link prediction and other possible problems facing us in the project.

2.3 Line: Large-scale Information Network Embedding. (Tang, 2015)

This paper provides another network embedding method "Line" to solve the problem of embedding very large real world information networks into low-dimensional vector spaces, which is useful in node classification, link prediction and network visualization. And the method is suitable for arbitrary types of networks, including undirected, directed, weighted an so on, which is proved to be effective on various information networks such as social networks, language networks and citation networks. The paper gives us a reference to the solutions of our project and it can also be used as a comparison to experiment our own algorithm.

2.4 Attributed Social Network Embedding. (Liao, 2018)

Embedding network data into a low-dimensional vector space performs well for node classification and entity retrieval by leveraging network structure. But for our proposal, such model and its related methods fail to sufficiently help us address the problems in our project since we obviously need more detailed information besides only network structure to find out the certain relationships among different individuals. This paper, in the focus of social networks, leverages not only the network structure but also the rich information about social actors (nodes) such as the user profiles of friendship networks – that is focusing on the attribute information of social networks to improve network embedding. In real world networks as well as the networks in our project, nodes often have attributes, and nodes with similar attributes are often more likely to be connected. Learning attributed social network embedding from this paper can enlighten us on the solutions of our project.

2.5 Multi-Scale Attributed Node Embedding. (Rozemberczki, 2019)

This paper presents many network embedding algorithms while delivering their own attributed algorithms, both pooled and multi-scale. It gives a useful method for a diverse range of applications, including latent feature identification across disconnected networks with similar attributes by capturing attribute-neighborhood relationships over multiple scales. The method both follows an approach similar to skip-gram while leveraging the attributes of nodes and the algorithms are proved to be outperforming comparable models on social networks. So the methods provided by this paper are worthy of study and reference for us to address our project problems..

3 Data Sets

3.1 Facebook Large Page-Page Network

3.1.1 Description

This webgraph is a page-page graph of verified Facebook sites. Nodes represent official Facebook pages while the links are mutual likes between sites. Node features are extracted from the site descriptions that the page owners created to summarize the purpose of the site. This graph was collected through the Facebook Graph API in November 2017 and restricted to pages from 4 categories which are defined by Facebook. These categories are: politicians, governmental organizations, television shows and companies. The task related to this dataset is multi-class node classification for the 4 site categories.

3.1.2 Feature

As Table 1 shows, the network is not directed and the nodes are assigned with certain features from the their description. The density of this network is around 0.001. From our perspective, this dataset is good for training for multi-class node classification, link prediction, community detection and network visualization.

Directed:	No.
Node features:	Yes.
Edge features:	No.
Node labels:	Yes. Binary−labeled.
Temporal:	No.
Nodes:	22,470
Edges:	171,002
Density:	0.001
Transitivity:	0.232

Table 1: Feature of Facebook Large Page-Page Network

4 Proposal Algorithms

Given two isolated groups (the connection between the nodes from each group is unknown), we want to predict whether a edge exist given two nodes from one of each groups. At first, we will use node embedding algorithm to estimate vertices features in two networks. Then, we use edges between these two networks to train our prediction machine learning algorithm, which is a neural network. Finally, we will use the trained machine learning algorithm to predict whether there is a potential edge between two vertex in these two networks. If so, we will link them together. By repeating this process, finally we can merge two networks together.

Algorithm 1: Merging Algorithm

Input: $Net_1(V_1, E_1)$,$Net_2(V_2, E_2)$:Network to be merged
$C_i(v_1, v_2), v_1 \in V_1' \subset V_1, v_2 \in V_2' \subset V_2$: The known connectivity between some nodes from Net_1, Net_2.

Output: $C_o(v_1, v_2), v_1 \in V_1 - V_1', v_2 \in V_2 - V_2'$: The predicted connectivity between the rest nodes from Net_1, Net_2.

$F(Net_1)$ =**Embedding** Net_1;//Feature of Net_1
$F(Net_2)$ =**Embedding** Net_2;//Feature of Net_2
M =**Machine Learning** $F(Net_1), F(Net_2), C_i$ //The trained model
for $v_i \in V_1 - V_1'$ **do**
  for $v_j \in V_2 - V_2'$ **do**
    $C_0(v_i, v_j)$=**Predict Connectivity**$F(Net_1, v_1), F(Net_2, v_2), M$
  endfor
endfor

To doing so, we may use the data from the previous section. To test our algorithms' performance while training, we will divide the original data set into two parts and then apply the embedding algorithms raised by Benedek and Carl, and train two groups individually. We will end up with two set of embedded features of each group. Then we may apply a Machine Learning algorithms to predict the connectivity of the nodes of two groups. We will suppose we have known about the 50% of their connectivity and then we may predict the rest of them based on the model trained.

Finally, we have a merged graph from two graph.

5 Future works

Now we have successfully implemented the node embedding algorithm to produce attributes for vertex and machine learning algorithm for prediction. Figure 3 shows the result of the node embedding algorithm. Next, we are trying to separate the dataset network and evaluate our algorithm. The difficulty mainly focuses on the threshold of separation. For now we will leave fifty percentages of edges between the two networks, and hide the rest for validation. But I think that fifty percentages might be too high to simulate real world data. But lower this threshold will reduce the efficiency of our prediction. We will do some experiments on it and find out the minimum threshold which can keep our prediction algorithm effective.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_10	x_11	x_12
2	0	-0.02679	0.016977	0.049024	-0.1615	0.13425	-0.10586	-0.22424	0.063134	0.012019	0.080763	0.137177	0.134277	0.140516
3	1	-0.02557	-0.00446	-0.00936	-0.14197	0.084177	-0.04011	-0.19012	0.054108	0.034803	0.078354	0.118137	0.060689	0.079433
4	2	-0.01804	-0.08832	-0.1951	-0.15233	0.082587	0.014095	-0.30176	0.185032	-0.04409	-0.10961	0.186739	-0.11421	-0.08973
5	3	-0.02421	-0.00205	-0.00053	-0.14813	0.105146	-0.05961	-0.17945	0.053396	0.030096	0.047848	0.113778	0.067871	0.082928
6	4	-0.04148	0.041228	0.061569	-0.16672	0.118603	-0.06966	-0.21442	0.027962	0.047091	0.125749	0.116546	0.149846	0.139995
7	5	-0.03518	-0.01577	0.006217	-0.14494	0.112257	-0.08361	-0.1896	0.037217	0.024571	0.082034	0.1322	0.103348	0.120661
8	6	-0.20555	0.035837	0.168743	-0.30501	0.192912	-0.02741	-0.1789	0.128612	0.073407	0.169341	0.257119	0.041803	0.17083
9	7	-0.01874	0.010813	0.015144	-0.23544	0.073886	-0.07302	-0.29702	0.124274	0.063078	0.155332	0.181627	0.104363	0.204199
10	8	-0.06412	0.003264	0.022279	-0.14168	0.122787	-0.07475	-0.17463	0.062598	0.012924	0.0655	0.122835	0.089029	0.111096
11	9	-0.02141	0.02455	0.034408	-0.10427	0.072885	-0.09007	-0.27868	0.050952	0.067401	0.073027	0.098608	0.083708	0.127109
12	10	-0.02018	-0.00151	0.003957	-0.11659	0.064853	-0.07567	-0.23132	0.068736	0.056156	0.045027	0.103521	0.050089	0.047352
13	11	-0.08159	0.107499	0.204451	-0.40333	0.110113	-0.09506	-0.38788	0.173483	0.142004	0.319387	0.319223	0.161166	0.415704
14	12	-0.21082	0.029341	0.059454	-0.30989	0.068042	0.095327	-0.21741	0.164665	-0.01436	0.125359	0.194105	-0.01912	0.079024
15	13	-0.03163	-0.00588	0.007195	-0.1509	0.096946	-0.06392	-0.20348	0.032533	0.044273	0.083889	0.111422	0.086007	0.107825
16	14	0.068263	-0.0249	-0.05674	-0.03064	0.084993	-0.04061	-0.41626	0.058935	0.056437	0.087672	0.139097	4.64E-05	0.037289
17	15	-0.05633	-0.04573	0.032479	-0.15632	0.147258	-0.13824	-0.18718	0.098337	0.022781	0.054736	0.122184	0.144258	0.078896
18	16	-0.01505	0.037746	0.053432	-0.12469	0.08383	-0.1037	-0.27623	0.062455	0.047278	0.112431	0.131753	0.132275	0.160404

Figure 3: Node Embedding results

6 Reference

1. Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In International Conference on Learning Representations, Workshop Track Proceedings, 2013.

2. Liao, L., He, X., Zhang, H. and Chua, T. Attributed social network embedding. IEEE Transactions on Knowledge and Data Engineering, 20(12): 2257-2270, 2018.

3. Grover, A. and Leskovec, J. Node2Vec: Scalable feature learning for networks. In International Conference on Knowledge Discovery and Data Mining, 2016.

4. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-Scale Information Network Embedding. In International Conference on World Wide Web, 2015.

5. Rozemberczki, B., Allen, C., Sarkar, R. Multi-Scale Attributed Node Embedding. arXiv preprint arXiv:1909.13021, 2019