Comparing Machine Learning Algorithms For Link Prediction

Maryam Kheirkhahzadeh - markhe@kth.se, Pablo Laso Mielgo - plaso@kth.se, Lucas Trouessin - lucastr@kth.se

Abstract. We consider comparing different machine learning models for link prediction problem. Link prediction algorithms aim to extract information about future interactions in the networks. In this paper we apply some similarity functions on the node embedding which is calculated by the DeepWalk algorithm to train the ML algorithms for link prediction. We use grid search to tune the parameters and compare machine learning models based on their best set of parameters.

Keywords. data mining, link prediction, machine learning, DeepWalk

1. Introduction. Link prediction aims to predict whether two nodes in a network are likely to have a link (1). It can be used in many research fields such as movie recommendation (2), friendship recommendation (3), and knowledge graph completion (4). Some methods consider first-order heuristics like common neighbors (CN)(5). They consider one-hop neighbors of each pair of nodes. Some other methods consider second-order heuristics such as Adamic-Adar (AA)(3). High-order heuristics consider the whole network such as SimRank (SR) (6). There are some research that shows the high-order heuristics have better performance in comparison to the first and second-order ones (7). Our contribution are as follows: 1) find a link prediction algorithm based on similarity functions calculating on node embedding in the social networks. We use the DeepWalk algorithm to find the node embedding. 2) Comparing some machine learning approaches based on a number of regression and classification metrics. 3) We can show the difference results for two situations. First, considering only one similarity function for modeling and the second is regarding combination of similarity functions. We consider two similarity functions, namely Euclidean similarity and Cosine similarity function to train our models.

2. Related works. Liben Nowell et al. (1) first gave a definition of link prediction. They summarized the link prediction method by calculating the similarity based on the ensemble of all paths between node neighbors or node pairs.

There are many methods based on the similarity near the node, such as Common Neighbor (CN) (5), Jaccard's Coefficient (JC) (8), Adamic/Adar (AA) (3), Preferential Attachment (PA) (9), etc. For each pair of nodes, the higher the number of common adjacencies, the higher the similarity score. These algorithms have been applied to many networks and experiments have shown that better performance can be achieved. The simplest one is based on the Common Neighbor.

Hasan et al. (10) pointed out that feature set extraction depends on the actual application. They used bibliographic data set. They compared the results of various classification algorithms, including SVMs, decision trees, nearest neighbors, and more. The result shows that SVM is better than others. However, they used some different features from papers and authors and some other attributes from the network. Benchettara et al. (11) apply supervised machine learning to predict the links in twomode social networks. However, these studies don't consider the dynamic evolution and diversity of the networks. Although the classification model can improve perfor-

mance, how to choose the appropriate features is still a problem. Moreover, if the networks are sparse, the positive and negative classes for training are extremely imbalance, it will seriously affect the accuracy of link prediction.

There are some other papers consider the link prediction problem for signed networks. In (12) and (13) the authors try to use the balance theory in the combination of other features of the signed graph to predict the sign of a link in a signed network. However, we do not consider signed network in this paper.

3. Data Describtion. We use the Facebook data set. This link contains 5 different data sets. We implement different machine learning algorithms on three data sets consisting of politician_edges data set, company_edges data set, and government_edges data set. We describe the attributes of these data sets here. After calculating node embedding, we mapped the data to two dimensions via PCA. The picture of three diagrams are as follows.

Politician edges data set attributes:

- #edges: 41729
- #nodes: 5908
- Clustering coefficient: 0.193
- Average path length: 0.272

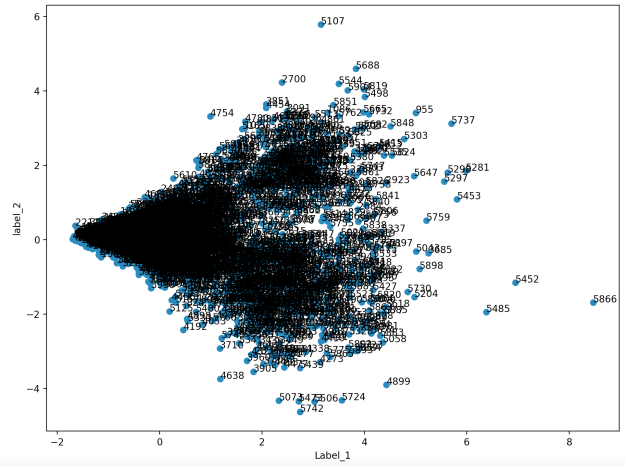

Figure 1. Politician edge data set

Company edges data set attributes:

- #edges: 52310
- #nodes: 14113
- Clustering coefficient: 0.120
- Average path length: 0.166

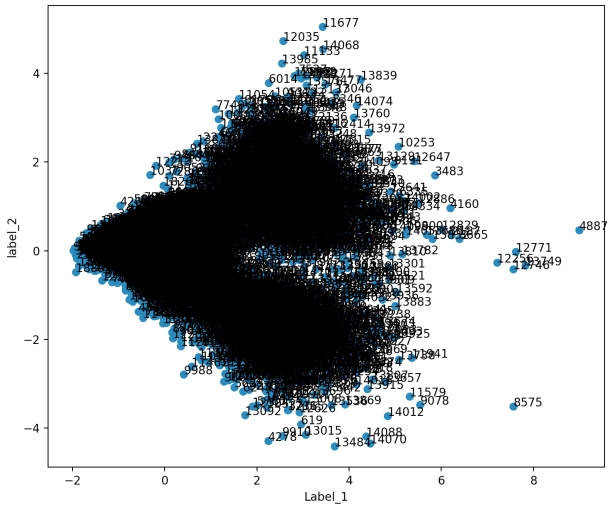

Figure 2. Company edges data set

Government edges data set attributes:

- #edges: 89455
- #nodes: 7057
- Clustering coefficient: 0.205
- Average path length: 0.816

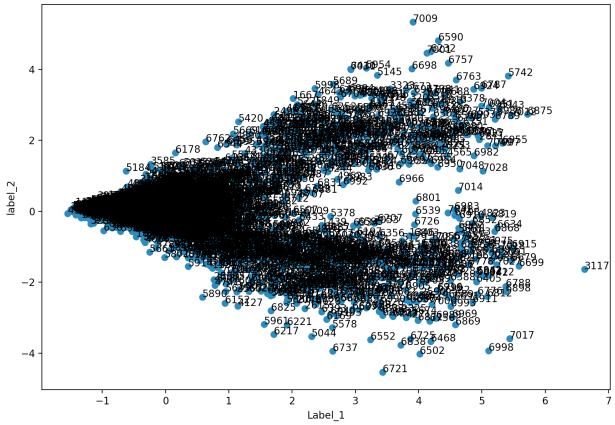

Figure 3. Government edges data set

4. Data pre-processing. Networks are based on relationships between nodes. To have a good representation of networks we need a data structure that reflects the properties of that network. So, we use graphs. DeepWalk learns embedding of a graph's vertices, by modelling a stream of short random walks (14).

4.1. DEEPWALK. DeepWalk is an algorithm proposed for learning latent representations of nodes in a network. DeepWalk utilizes the random path traversing technique to represent the graph as sequences. These sequences will be used to train a Skip-Gram Language Model. It takes a graph as input and produces a latent representation as an output.

4.2. NODE EMBEDDING. We use DeepWalk to find a proper node embedding.The Karate Club is an unsupervised machine learning extension library for NetworkX. It learns the graph structure and contains some useful libraries such as DeepWalk. We use this library to have embeddings for the nodes in the networks with different dimensions. We find three different embeddings with dimensions 16, 64, and 128. Then we can use these vectors to learn our models and link prediction.

5. The approach. We see the link prediction model as a classification problem. In fact, the pairs of nodes with an edge belong to class 1 and the remaining edges belong to class 2. We use cosine similarity and Euclidean similarity between node embedding vectors to train the models as we explain in this section.

We use the Gridsearch method to tune the parameters of each machine learning algorithm. Then, choose the best set of parameters to compare with other models.

5.1. TRAINING BASED ON SIMILARITY METHODS. We should train the models to predict 1 for each pair of nodes with edge and zero for nodes without edge. Considering memory and time complexity we cannot train the model with the node embedding vectors.

5.1.1. TRAIN SET. We choose 20% of edges randomly and keep them for the test phase. Therefore, 80% of the edges are used for train set. To prevent bias training, we train the model with balanced data from both classes. So, the train set consists of 80% of the edges with y_train value equal to 1, and the same amount of non edges with y_train value equal to zero. We choose the pairs of nodes without edges randomly from the list of non edges of the graph.

5.1.2. TEST SET. The test set consists of two categories of rows. First, The remaining 20% of edges, shapes the half of the test set with y_test value equal to 1. Then, we choose the same number of node pairs from non edges set randomly with y_test value equal to zero.

5.1.3. SIMILARITY FUNCTIONS. Each row of the train set corresponds to a pair of nodes, with or without edges. The corresponding row of y_train has 1 for edge and has zero when there is no edge. We calculate the similarity of this pair and save it as one row in the train set. We develop and test our approach with two different train sets. First, we consider only one column of similarity for Euclidean or Cosine similarity. Second, we build a training set with two columns corresponding to two similarity functions.

5.2. METRICS TO EVALUATE OUR ALGORITHM . We use two built-in functions predict_proba and predict to evaluate the supervised trained model. predict_proba accepts a single argument as a test set and returns an array containing the predicted label. However, the predict function gets the test set and predicts the actual class over the test set. So, we use two regression metrics

for evaluating the results of predict_proba and some classification metrics to evaluate the classes predicted by predict function.

5.2.1. REGRESSION METRICS. Regression metrics are as follows:

- Area Under the Curve. AUC represents how much the model is capable of distinguishing between classes
- Mean Squared Error. It is based on the average squared difference between the actual and predicted values.

5.2.2. CLASSIFICATION METRICS. Classification metrics are as follows:

- Accuracy. Classification accuracy is the ratio of number of correct predictions to the total number of input samples.
- Recall. The recall is calculated as the ratio between true positive to the total number of positive samples.
- Precision. The precision is calculated as the ratio between true positive to the summation of true positives and false positives.
- Brier score. The Brier score measures the mean squared difference between the predicted probability assigned to the possible outcomes for each item in test set and the actual outcome for that item.
- F1-score. The F1-score is a combining measure of precision and recall. In fact it is defined as harmonic mean of the model's precision and recall.

6. Results.

6.1. ONE SIMILARITY FUNCTION. ....

6.2. COMBINATION APPROACH. ....

7. General difficulties. The most important problem is that tuning parameters for different models is extremely time consuming. We use grid search for finding the best set of parameters. The built-in function GridSearchCV implement grid search. We have three data sets, with 6 machine learning models. we run them with three different dimensions. We implement node embedding for 16, 64, 128 dimensions. So running GridSearchCV for 6x6x3 cases is so expensive and takes time.

8. Conclusion & Discussion. In this project, ...
In evaluation part, ...

9. Future work. We have developed and tested our approach on directed data sets. It can be developed and tested on undirected data sets. In addition, since we had limited resources to implement our method, we could not test more machine learning algorithms and more similarity functions. So, we can consider these issues as future work.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[4] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[5] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025102, 2001.

[6] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 538–543.

[7] 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.

[8] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[9] Y.-B. Xie, T. Zhou, and B.-H. Wang, "Scale-free networks without growth," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 7, pp. 1683–1688, 2008.

[10] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *SDM06: workshop on link analysis, counter-terrorism and security*, vol. 30, 2006, pp. 798–805.

[11] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," in *2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2010, pp. 326–330.

[12] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon, "Exploiting longer cycles for link prediction in signed networks," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1157–1162.

[13] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 641–650.

[14] 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*, 2014, pp. 701–710.