Link prediction in social networks using machine learning methods

Simon Janežič

University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia

Abstract

We tackle the task of link prediction in social networks as a supervised learning problem. We construct node-based, neighborhood-based, path-based and community detection features, as well as a more recent *node2vec* approach that is based on random walks. We feed these features into multiple machine learning algorithms and evaluate the results on both directed and undirected social networks. We additionally attempt to explain the fitted models and analyze whether random sampling of edges can really give us a good model for predicting future edges, which is usually the desired task for the purpose of friends recommendation systems.

Keywords: link prediction, social recommendations, social network, machine learning.

1. Introduction

Online social networks have grown incredibly popular in the 21st century. According to Facebook[1] more than one billion users use their network on a daily basis and almost two billion people use it at least once a month. These huge networks understandably became interesting for the field of network analysis. Link prediction methods attempt to predict future or missing links in a network[1, 2]. Given an user in a social network, these methods can find most likely future pairings (e.g. friendships) with other users. This corresponds to recommender system that helps users find people they might be interested in or know in real life. We tackle this task as a supervised machine learning problem, where every node pair is potential training example and it's label corresponds to presence or absence of an edge. Features are obtained only from network's structure and don't include any domain specific knowledge, which makes our approach general to any network.

First we talk about related work in this area of research. We then talk about features that were constructed from the networks as well as provide some information about used machine learning methods and experimental setup. After that we present two undirected and two directed social network subgraphs that were used as our datasets. These are Facebook, Livemocha, Twitter and Digg networks. We then present and analyze the performance of used methods as well as attempt to provide reasonable explanations behind fitted machine learning models.

2. Related work

There has been quite some work done in this field of research. Perhaps one of the earliest and well-known is the work of Hasan et al. [3] where they explore link prediction as a supervised learning task on coauthorship networks. The paper focuses on features that are based on graph topology as well as domain-specific features. Some topological features used in this paper such as shortest distance were also used in our approach. The paper also compares performances of multiple machine learning algorithms for this task and shows that only a small subset of features are important for predicting coauthorships.

Email address: `sj8495@student.uni-lj.si` (Simon Janežič)

[1]`https://newsroom.fb.com/company-info/`

Even more relevant to our problem is the work of Lieben-Nowell and Kleineberg [4] in which they explore the link prediction problem for social networks. They experiment on coauthorship datasets, however as opposed to Hasan et al. [3] they only deal with features that are based on graph topology and experiment with a large number of those. Advantage of those features is that they are general to any network. They explore different node-based, neighborhood-based and path-based features and show correlation between many of these features and future links.

A more recent work by Grover and Leskovec [5] introduces *node2vec* method that learns mapping of nodes in a network to low-dimensional vectors using neural networks and random walks. They show that the method's mapping can be very successfully used as features on various node classification tasks as well as link prediction. We also experiment with this approach in our work.

3. Methods

We tackle link prediction task as a supervised learning problem. First we describe our machine leaning framework. We construct our test set by randomly selecting some portion of existing edges to represent our positive examples. We also select equal number of non-edges (pairs of nodes that are not connected) to represent our negative examples. Selected edges are removed from the original network. We also remove zero degree nodes in a resulting network, since it would be practically impossible to predict anything useful for those. Resulting network is used to compute features for our test set. To get a training set the same procedure is repeated again.

For the purposes of recommender system application we would like to train our model to predict future edges. We assume that by removing random edges we are simulating network's behavior over time. This assumptions can be way off and give us inconclusive results. Better approach would be to select edges based on the time they were formed rather than randomly. Unfortunately only Digg dataset includes temporal information. We use that network to compare link prediction performance using random selection of edges versus time-based selection. That way we get an estimate of how accurate the results for the rest of the networks (without temporal information) are for the task of future prediction.

3.1. Feature extraction

Extraction of informative features is perhaps the most important and time-consuming task of many machine learning problems. Our task is to extract features, given a network and a pair of nodes that we are interested in. We focus only on features that are based on graph topology since we do not have any domain-specific information. The rest of this section describes all the features we use.

- **Preferential attachment** [6]. Measure that is based on the concept that high degree nodes are more likely to connect than low degree nodes. It is defined as

  $$|\Gamma(x)| \cdot |\Gamma(y)| \tag{1}$$

  where $\Gamma(x)$ is a set of neighbors of node x.

- **Clustering index** [7]. It is defined as

  $$c(x) \cdot c(y) \tag{2}$$

  where $c(x)$ is clustering coefficient of node x. In social networks $c(x)$ is the probability that two of x's friends are also friends between each other. This feature might not be informative by itself but in combination with other features.

- **Common neighbors** [8]. As the name suggests this feature is simply defined as

  $$|\Gamma(x) \cap \Gamma(y)| \tag{3}$$

  For social networks the intuition behind this is that two people with many common friends are likely friends themselves.

- **Jaccard index** [4]. Normalized number of common neighbors defined as

$$\frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{4}$$

- **Adamic-Adar index** [9]. Another common neighbors feature defined as

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|} \tag{5}$$

  It is based on intuition that having a common friend with low degree is more significant than having one with high degree. This feature has been shown to be very informative for social networks [4].

- **Length of shortest path** [4, 3]. As the name suggest this feature is computed by finding shortest path from node x to node y and taking it's length. Intuitively, nodes with longer shortest paths are unlikely to connect.

- **Community structure index** [10]. Given a partitioning of a network into communities this measure is defined as

$$s_{xy} = \begin{cases} \frac{m_{c_x}}{\binom{n_x}{2}} & \text{if } c_x = c_y \\ -\infty & otherwise \end{cases} \tag{6}$$

  where c_x is community of node x, while n_x and m_{c_x} are number of nodes and number of edges within community c_x. Two nodes that belong to very dense community are more likely to connect than two nodes that belong to sparse community. We used Louvain's method [11] as our community detection algorithm.

- **Community resource allocation index** [12]. Community detection measure defined as

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{f(z)}{|\Gamma(z)|} \tag{7}$$

  where $f(z)$ is 1 if z belongs to the same community as x and y and 0 otherwise. This feature combines community detection and common neighbors features.

- **Within inter cluster (WIC)** [13]. Also combines community detection and common neighbors features and is defined as

$$\frac{\sum_{z \in \Gamma(x) \cap \Gamma(y)} f(z)}{\sum_{z \in \Gamma(x) \cap \Gamma(y)} 1 - f(z)} \tag{8}$$

  which corresponds to ratio between common neighbors that belong to the same community as a given pair of nodes and the rest of common neighbors.

- ***node2vec*** [5]. This method is based on *word2vec* [14]. Given a corpus of sentences, *word2vec* learns low-dimensional vector representation for every word. These latent vectors represent word's context. The learning is done using artificial neural networks and basically corresponds to non-linear dimensionality reduction. If we use nodes instead of words and short random walks instead of sentences, we get an approach called *DeepWalk* [15]. *node2vec* is essentially a generalization of *DeepWalk* that allows more flexibility with choosing random walk strategy (biased random walks). Result of this approach is a low-dimensional vector for every node. To use it for link prediction we need to combine vectors of two candidate nodes. Grover and Leskovec [5] found that multiplying corresponding vector components gave best results in most cases.

3.2. *Features for directed networks*

Shortest path length and *node2vec* are also defined for directed networks. However, since shortest path length is asymmetrical for directed networks, we have to add reverse shortest path length as another feature. For the rest of the features we construct them as if the networks were undirected. We also include a couple of directed versions of the features: four versions of preferential attachment based on in/out degrees of nodes and four versions of common neighbors based on in/out common neighbors. We also add a simple binary feature that indicates if there exists an edge between two nodes in the opposite direction.

3.3. *Machine learning algorithms*

We use logistic regression, random forest and SVM with Gaussian kernel as our machine learning algorithms. This selection of methods covers one linear and two different non-linear approaches. Advantage of linear models is their simplicity and clear explanation of prediction, while non-linear models achieve better performance when separating function is highly non-linear, but can overfit the data.

4. Results

In this section we present and discuss results of our methods. We have evaluated our approach on four online social networks [16, 17, 18, 19], listed in Table 1. In undirected networks edges represent two-way friendships. In directed networks edges represent one-way friendships/followerships. These datasets only represent subgraphs of otherwise huge networks. To add some variety to our analysis, we have intentionally picked two web services that have social network as their core functionality (Facebook and Twitter) as well as two web services where social network is part of the extra functionality (Livemocha and Digg).

Network	**Number of nodes**	**Number of edges**	**Type**
Facebook	63,731	817,035	undirected
Livemocha	104,103	2,193,083	undirected
Twitter	81,306	1,768,149	directed
Digg	279,630	1,731,653	directed

Table 1: Used social networks, along with basic statistics.

We have run machine learning algorithms on selected networks, using 1% of edges and equal number of non-edges as a test set and additional 2% of edges and equal number of non-edges as a training set. We repeated this procedure three times and averaged the results to account for randomness. We also included two baseline link prediction methods (Adamic-Adar index and preferential attachment index) for comparison.

Network	**LR**	**SVM**	**RF**	**AA**	**PA**
Facebook	0.982	0.980	**0.983**	0.951	0.890
Livemocha	0.953	0.948	**0.960**	0.790	0.943
Twitter	**0.999**	**0.999**	**0.999**	**0.999**	0.898
Digg	0.985	0.987	**0.995**	0.900	0.982
Digg*	0.971	0.970	**0.990**	0.880	0.975
Digg**	0.968	0.965	**0.989**	0.880	0.975

Table 2: Test set AUC for methods: Logistic regression (LR), support vector machines (SVM), random forest (RF), Adamic-Adar index (AA) and preferential attachment index (PF). First four rows represent average AUC over three train/test runs with random edge sampling. Digg* represents results using temporal-based sampling of edges. Digg** represents results using temporal-based sampling of test set and random sampling of train set.

Results can be observed in Table 2. We notice that the baseline predictions have quite a high AUC and can not be improved in some cases. Most obvious example is Twitter with unusually high AUC. Random forest consistently achieves best AUC. This might mean that there are some useful non-linearities between attributes, although SVM does not seem to model them, as it's performance is about equal to that of logistic regression. It is interesting that we are able to predict directed networks better than undirected. One would expect that added dimensionality of predicting directed links would be a harder problem. This might just be due to coincidental selection of datasets.

Table 2 also includes results for experiments that we ran with temporal edge sampling on Digg network, since we also had temporal information for that network. Both experiments give us slightly worse results than with random sampling of edges. Logistic regression and SVM perform very poorly in these two experiments as they are evaluated worse than the baseline methods. Random forest performs best with noticeably better AUC than the baseline methods. The last experiment's results are most important as it was performed by taking test set based on temporal information and train set randomly after that. Random forest achieves very similar results compared to sampling both sets based on temporal information, which may indicate that models that are trained by randomly removing edges may actually also be good with future prediction, which is what we really want for a real-world application.

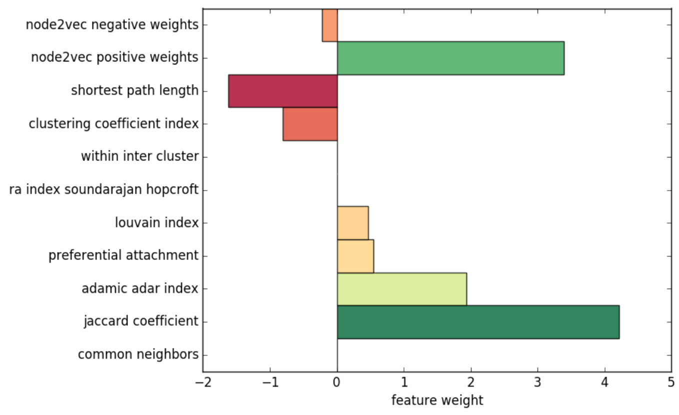
4.1. Explaining the models

If we consider the application of link prediction for recommendation systems, often we want to be able to explain our recommendations to the user. An example of this is a number of common friends indicator that Facebook shows along with it's recommendations. While random forest gave us better results (Table 2), we can not really find explanations of it's predictions easily because of it's complexity. With logistic regression however, this is a rather straight forward task. We can just look at learned attribute weights to see what contributes to model's prediction. We obviously had to standardize all features to have zero mean and unit variance in order to get comparable weights.

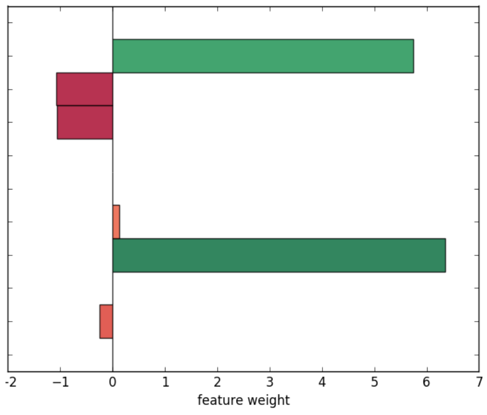
Logistic regression weights for all networks can be observed in Figure 1. Neighborhood-based features represent most important attribute groups for Facebook and Twitter, with Jaccard coefficient being most important. We notice that the weights of some features got squashed to zero. This is just the property of $L1$ logistic regression regularization that got us slightly better results than standard $L2$ regularization. In a way the method performs feature selection. Commmon neighbors feature for example, probably did not offer much extra information that Adamic-Adar index and Jaccard coefficient would not already include, so it got squashed to zero.

Preferential attachment features are more important for Livemocha and Digg. Unlike Facebook and Twitter, these two services do not have social network as their core functionality and users are mostly using anonymous usernames. That could be the reason for different linking process. Real-world friendships become less of a factor with these kind of networks compared to Facebook or Twitter. With online communities, users probably like linking to very active and outspoken people that already gained a lot of respect in that community, more so than inactive users with low friends/followership count. What stands out in Digg's network is out-in preferential attachment that gets much higher weight than undirected preferential attachment. This simply indicates that someone that follows a lot of people will likely link to someone that has a lot of followers. Out of all directed preferential attachments this one makes the most sense, so it's high weight is no surprise. What is also interesting about Digg is how much shortest path length feature stands out compared to other networks. It is in fact the most influential feature. Weight is obviously negative since larger shortest path intuitively corresponds to smaller probability of a future link.

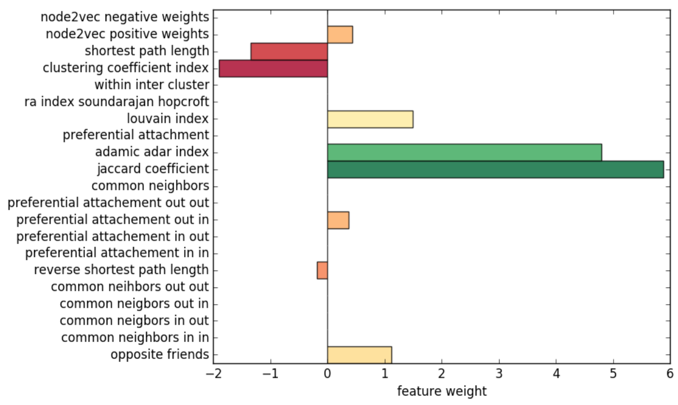
Consistent negative weight of clustering coefficient index is also interesting. Most notably in Twitter's network where it's influence is quite noticeable. If we pick two people at random they likely will not have many friends in common, however if their clustering coefficient is low they are more likely to connect outside of their circle (with each other) and vice-versa. Therefore negative weight could definitely make sense here.



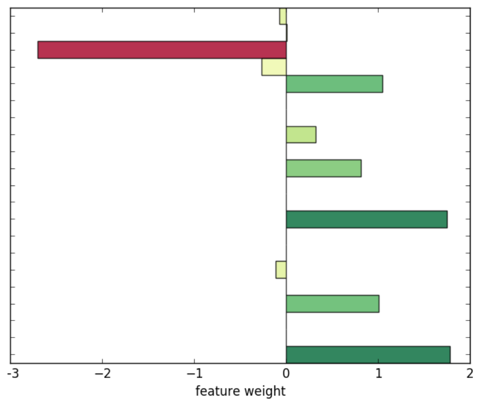
(a) Facebook



(b) Livemocha



(c) Twitter



(d) Digg

Figure 1: Logistic regression weights for undirected (upper figures) and directed (lower figures) networks. *node2vec* features were aggregated into sum of those with positive and negative influence.

node2vec features are quite significant for both undirected networks and insignificant for both directed networks. Since we are multiplying corresponding components of *node2vec* vectors we are getting random walk context similarity vector between two nodes as a result. It's kind of expected that nodes with similar *node2vec* vectors will likely link to each other so large positive influence is not surprising. It is rather hard to interpret, why directed networks do not benefit as much from these features. Perhaps there exist some useful non-linearities between them that logistic regression is obviously not able to model.

Lastly, community detection features do not stand out extremely in any network but have some noticeable influence in directed networks. In undirected networks all of them are zero or near-zero. It is possible that Louvain's method did not find useful communities because of potential existence of small communities and Louvain's method resolution problem when dealing with those.

5. Conclusion

We have shown that machine learning methods can successfully be used for link prediction in social networks. We were able to beat high accuracy baseline methods for most networks with random forest being the most successful method. Although the accuracies we achieved were high, these results likely are not representative of the entire networks. Facebook and Twitter subnetworks that we used are just a speck compared to whole networks. One would have to experiment on whole networks but that obviously introduces big data challenges. We have also shown that random forest model, whose training data is obtained by random edge sampling, can also be used for predicting future edges better than baseline methods. This might be useful for a social network service that wants to integrate recommender system into it's service, but does not store temporal information. Lastly, we have explained the fitted models by looking at logistic regression weights and reasoning about them. By doing that we have found some differences in linking process of the selected networks, where two of them link more so according to preferential attachment and the other two more so according to common neighbors/friends.

References

[1] L. Lü, T. Zhou, Link prediction in complex networks: A survey, Physica A: Statistical Mechanics and its Applications 390 (6) (2011) 1150–1170.

[2] M. Al Hasan, M. J. Zaki, A survey of link prediction in social networks, in: Social network data analytics, Springer, 2011, pp. 243–275.

[3] M. Al Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning, in: SDM06: workshop on link analysis, counter-terrorism and security, 2006.

[4] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, journal of the Association for Information Science and Technology 58 (7) (2007) 1019–1031.

[5] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.

[6] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, T. Vicsek, Evolution of the social network of scientific collaborations, Physica A: Statistical mechanics and its applications 311 (3) (2002) 590–614.

[7] D. J. Watts, S. H. Strogatz, Collective dynamics of small-worldnetworks, nature 393 (6684) (1998) 440–442.

[8] M. E. Newman, Clustering and preferential attachment in growing networks, Physical review E 64 (2) (2001) 025102.

[9] L. A. Adamic, E. Adar, Friends and neighbors on the web, Social networks 25 (3) (2003) 211–230.

[10] B. Yan, S. Gregory, Finding missing edges and communities in incomplete networks, Journal of Physics A: Mathematical and Theoretical 44 (49) (2011) 495102.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment 2008 (10) (2008) P10008.

[12] S. Soundarajan, J. Hopcroft, Using community information to improve the precision of link prediction methods, in: Proceedings of the 21st International Conference on World Wide Web, ACM, 2012, pp. 607–608.

[13] J. C. Valverde-Rebaza, A. de Andrade Lopes, Link prediction in complex networks based on cluster information, in: Advances in Artificial Intelligence-SBIA 2012, Springer, 2012, pp. 92–101.

[14] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.

[15] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 701–710.

[16] B. Viswanath, A. Mislove, M. Cha, K. P. Gummadi, On the evolution of user interaction in Facebook, in: Proc. Workshop on Online Social Networks, 2009, pp. 37–42.

[17] R. Zafarani, H. Liu, Social computing data repository at ASU (2009).
URL `http://socialcomputing.asu.edu`

[18] J. Leskovec, J. J. Mcauley, Learning to discover social circles in ego networks, in: Advances in neural information processing systems, 2012, pp. 539–547.

[19] T. Hogg, K. Lerman, Social dynamics of Digg, EPJ Data Science 1 (5).