Link and Annotation Prediction Using Topology and Feature Structure in Large Scale Social Networks

Burak Isikli[1,2], Fatih Erdogan Sevilgen[1], and Mustafa Kirac[2]

[1] Department of Computer Engineering Gebze Insititute of Technlogy, Gebze, Kocaeli, Turkey
sevilgen@gyte.edu.tr
[2] Turkcell, Inc. Istanbul, Turkey
{burak.isikli,mustafa.kirac}@turkcell.com.tr

Abstract. Repeated patterns observed in graph and network structures can be utilized for predictive purposes in various domains including cheminformatics, bioinformatics, political sciences, and sociology. In large scale network structures like social networks, graph theoretical link and annotation prediction algorithms are usually not applicable due to graph isomorphism problem, unless some form of approximation is applied. We propose a non-graph theoretical alternative to link and annotation prediction in large networks by flattening network structures into feature vectors. We extract repeated sub-network pattern vectors for the nodes of a network, and utilize traditional machine learning algorithms for estimating missing or unknown annotations and links in the network. Our main contribution is a novel method for extracting features from large scale networks, and evaluation of the benefit each extraction method provides. We applied our methodology for suggesting new Twitter friends. In our experiments, we observed 11-27% improvement in prediction accuracy when compared to the simple methodology of suggesting friends of friends.

Keywords: social networks, data mining and knowledge discovery, big data, business intelligence, link prediction, graph processing, graph mining.

1 Introduction

Importance of network structures has been recently increasing, due to advancements in data collection, storage, and processing technologies. Social networks, mobile call networks, biological networks, and World Wide Web are some examples of network structures that many end users of online services and researchers deal with. Network structures can be utilized for revealing information that is difficult to obtain from nodes directly. For instance, discovery of disease related proteins within a protein interaction network [5], targeting potential customers with ads by using social networks [4], and looking for similar graph patterns

S. Kozielski et al. (Eds.): BDAS 2014, CCIS 424, pp. 238–249, 2014.

in chemical molecular structures are some tasks that can only be accomplished using a network oriented analytic methodology.

A node-annotated network consists of i) a list of nodes, ii) links representing some relationship between nodes, and iii) features associated with nodes. For example, consider the Facebook network: Facebook is a node-annotated social network where its nodes are people, friendship activities amongst people are links, and liked/favorited items are node features. Since the association of Facebook users with content items is voluntarily accomplished by the users themselves, not all users reveal all pieces of information about themselves. The problem we primarily attack in this paper is the problem of completing missing or incomplete node annotations. We present a novel methodology for predicting unknown annotation of network nodes. We propose a solution to network link prediction problem [12] by reducing it into another form of node annotation prediction problem as well.

Recent research shows that homophily [3] and contagion [3] in real world networks suggest shared features between connected network nodes. In addition, correlation between existing node features is shown to be employed for predicting missing or unknown features of network nodes [15]. Such properties of real world networks are utilized by learning correlations between existing node annotations into a data mining model, and applying such model to network nodes with missing annotations. D. Liben et al [9] showed that features from network topology of a co-authorship network can be used for supervised learning effectively. This is the first and common approach for solving the link prediction problem. Taskar et al. [13] fulfilled the relational Markov network algorithm. They aim to predict missing links in a network of web pages and a social network. Although it is not directly related to link prediction problem, users are linked by their common interests. Natural language processing (NLP) and text mining methods [7] are used to predict interests by finding tweet similarity. Because of multitude of common interests, users can be associated with each other. There are several other researches can be related to our work such as Goldberg and Roth [2]. They employed the neighborhood as a reliable property of small networks (e.g. protein-protein interactions) for confidence. Clauset et al. [1] developed a prediction algorithm on social and biological networks.

Our approach for predicting missing node annotations in a network has the following steps. First, the network is preprocessed (i.e., extract-transform-load) to collect for each node all available annotation patterns in the network neighborhood of that node, and create an integer feature vector representing the counts of annotation patterns observed in the whole network. This approach is similar to text mining [7], where text documents are converted into term frequency vectors. Next, we create a table from the count vectors of network nodes, and feed into machine learning algorithms in order to obtain models that explain complex relationships between node annotations. Finally, we utilize resulting models learned from the network for estimating whether a particular network node is highly likely to be associated with a target feature annotation.

The process summarized above requires processing large amount of data. This process depends on the ability of forming and counting annotation patterns in a large network data. In real world cases, network data is so huge that it does not fit in the memory of a single computer with commodity hardware. One option is building an expensive supercomputer, and applying a trivial pattern counting algorithm. Another option is making use of a smaller size, sample data. In the latter case, sampling network structures partitions it into disconnected components, resulting in information loss. We propose a hybrid approach: We compute network patterns in a disk-based distributed processing environment, and convert network data into a node-based feature vector sets. Then, we run machine learning algorithms on a single computation framework using data sampled from the node-based feature sets.

Main contributions of this paper are 1) a systematic way of converting network structures into flat vector sets, 2) a scalable methodology for computing feature vectors in a parallel DBMS environment, and 3) a novel solution to link prediction problem by formulating it as a form of node annotation prediction problem.

2 Methods and Algorithms

Definition 1 (Network): A directed network $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$ is a structure that consists of a set of nodes $\mathbf{N}$, set of links $\mathbf{L}$ that connects two nodes ($\mathbf{l} \in \mathbf{L}$ then $\mathbf{l} : \mathbf{n_1} \rightarrow \mathbf{n_2}$ and $\mathbf{n_1}, \mathbf{n_2} \in \mathbf{N}$), set of features $\mathbf{F}$ that have information about node (e.g. person of interests, hobbies), and set of feature annotations $\mathbf{A}$ that associates nodes with features ($\mathbf{f} : \mathbf{n} \rightarrow \mathbf{a}, \mathbf{a} \in \mathbf{A}, \mathbf{n} \in \mathbf{N}$, and $\mathbf{f} \in \mathbf{F}$).

Example 1: Fig. 1 illustrates an example network structure. Nodes that represent social network profiles are numbered from 1 to 6 and each node is assigned some features, such as Sailing, Swimming and Baseball. Each annotation represents existence of feature such as if a person like sailing, sailing's annotation is 1. Nodes are connected with links that represent friendship relationship between nodes.

In this paper, we study Twitter Social Network presented in [6]. Twitter graph is created in the form of a two-column table of user pairs. An excerpt from a Twitter follower table is shown in Tab. 1. Each column in the Twitter table denotes a Twitter user node, and each row in the Twitter table represents a directed link. Hence a Twitter table is a list of links in the Twitter network. Twitter network and further details can be obtained via Twitter Application Programming Interface (API) [14]. The Twitter dataset we employ consists of approximately 40 million nodes and 1.5 billion links.

Problem 1 (Network Link Prediction): Real world networks continuously evolve, as they gain or lose nodes, links, and annotation. Hence, a network $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$ is a snapshot of a real world network at time $\mathbf{t_1}$, and the network

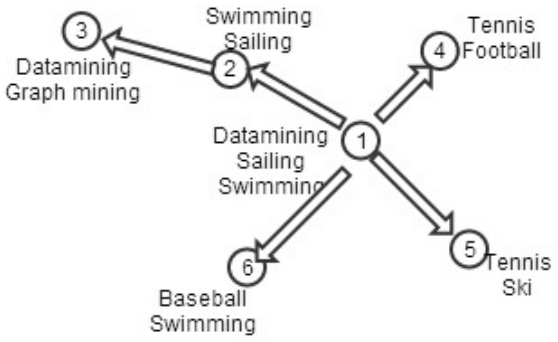

Fig. 1. Representing a graph. Each circle denotes a node, each arrow between them denotes a link and each node has a label which is representing person's interests.

Table 1. Twitter graph table. It consists of user ids that denote whom following. For example user 48954673, 49233593, 27433315, 60843485 are followers of user 43661838. User 27433315 is a follower of user 43661851.

To Id	*From Id*
43661838	48954673
43661838	49233593
43661838	27433315
43661838	60843485
43661851	27433315

evolves to $\mathbf{G'} = (\mathbf{N'}, \mathbf{L'}, \mathbf{F'}, \mathbf{A'})$ at $\mathbf{t_2}$ where $\mathbf{t_2} > \mathbf{t_1}$. **Network link prediction** problem is to obtain a new $\mathbf{l} = \mathbf{n_1}' \rightarrow \mathbf{n_2}$ $(\mathbf{n_1}, \mathbf{n_2} \in \mathbf{N}, \mathbf{n_1}, \mathbf{n_2} \in \mathbf{N'})$ such that $\mathbf{l} \notin \mathbf{L}, \mathbf{l} \in \mathbf{L}$.

Problem 2 (Network Annotation Prediction): Real world network data does not perfectly represent the underlying network. Node annotations are not complete, and existing annotations may not be correct due to data quality issues. Let $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$ is a real world network, and $\mathbf{G'} = (\mathbf{N'}, \mathbf{L'}, \mathbf{F'}, \mathbf{A'})$ is the known representation of $\mathbf{G}$ in the data obtained. Network annotation prediction problem is to obtain the annotations in $\mathbf{A} - \mathbf{A'}$ with some probability and confidence. The problem can also be configured as **a network annotation correction**, where the existing annotations in $\mathbf{A} - \mathbf{A'}$ can be assigned of being noise or error with some probability and confidence.

Definition 2 (Network Neighborhood): A node $n \in N$ in graph $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$ has **incoming direct neighbor of m** such that $\mathbf{m} \in \mathbf{N}$, and $\bigvee \mathbf{l}_1 \in \mathbf{L}$, $\mathbf{l}_1 = \mathbf{m} \rightarrow n$, and has an **outgoing direct neighbor o** such that $\mathbf{o} \in \mathbf{N}$, and $\bigvee \mathbf{l}_2 \in \mathbf{L}$, $\mathbf{l}_2 = \mathbf{n} \rightarrow \mathbf{o}$. A **directed path** of length $\mathbf{z}$ from $\mathbf{n} \in \mathbf{N}$ to $\mathbf{n}_z \in \mathbf{N}$ in $\mathbf{G}$ consists of links $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_z$ such that $\mathbf{l}_1 = \mathbf{n} \rightarrow \mathbf{n}_1$, $\mathbf{l}_2 = \mathbf{n}_1 \rightarrow \mathbf{n}_2$,

..., and $\mathbf{l}_z = \mathbf{n}_{z-1} \rightarrow \mathbf{n}_z$. All direct neighbors of a node **n** are connected to n through a directed path of length 1. **Network neighborhood** of a node n consists of nodes that have at least one directed path connected to **n**. When we limit the maximum length of directed paths between all nodes in the network neighborhood of node **n** and the node **n** by a fixed number **z**, we say that the **radius** of network neighborhood of **n** is **z**.

In this paper, we focus on predicting properties of Twitter user profiles. There are many ways of obtaining Twitter user interest. In [11], the text in the news feed of users, or the tweets of the users are processed through NLP and text mining to obtain the concepts the users interested in. In another work, physical locations and point of interests of the users are retrieved from their check in and shared locations. As locations reflect personal interests, it is also possible to generate location-based Twitter user interests [8]. In this work, we consider the celebrities that the Twitter users follow, as their interests. In Twitter, there is no distinction between celebrity and a regular user, as Facebook does (i.e., people and pages). Hence, we make use of a simple approach to make such distinction. When the follower count of a user is above some certain threshold, we classify the user as a celebrity. Activities of following celebrities are considered as interests of other regular users. In short, our aim in this paper is to predict interests (i.e., annotations) of regular users or equivalently, estimating whether regular users would follow (i.e., show interest) a celebrity user (e.g., a famous actress, a rock band, or an auto maker brand).

Definition 3 (Transformation of Network Link Prediction problem to Network Annotation Prediction problem): Real world networks follow a scale-free property [6] where some nodes have very high link counts whereas many nodes have quite few link counts. Usually, the nodes with very high link counts have a special real world meaning (i.e., celebrity people, or brands in the Twitter network). Hence, set of regular nodes (i.e., common users) are obtained by removing all nodes with very high link counts (i.e., above some certain threshold). Not to loose link information of removed nodes, the information is stored as node annotations; suppose that u is a celebrity node with dense link information, and v and w are common nodes with much less link counts. Further suppose that the network G has the links $v \rightarrow u$ and $w \rightarrow v$. The links $v \rightarrow u$ and $w \rightarrow v \rightarrow u$ are converted to two annotations of v and w as, v is assigned {outgoing_U}, and w is assigned {incoming_U} as new annotations. As we converted link information to annotation information, we can now formulate link prediction problem as an annotation prediction problem. In the beginning, a Twitter network dataset is a non-annotated directed graph. After we remove all the celebrity nodes from the network, and convert links of celebrity nodes into annotations of regular nodes, it becomes an annotated directed graph.

Definition 4 (Network Pattern): Given a large directed annotated $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$, a **network pattern** is a sub-graph $\mathbf{p} = (\mathbf{N}', \mathbf{L}', \mathbf{F}', \mathbf{A}')$ such that $\mathbf{N}' \subseteq \mathbf{N}$, $\mathbf{L}' \subseteq \mathbf{L}$, $\mathbf{F}' \subseteq \mathbf{F}$, $\mathbf{A}' \subseteq \mathbf{A}$, and **p** is a connected component where

there exists at least one directed path between every $\mathbf{n}_1, \mathbf{n}_2 \in \mathbf{N}'$. In the case when $\mathbf{p}$ is a much smaller sub-graph of $\mathbf{G}$, there can be multiple isomorphic mappings between $\mathbf{p}$ and $\mathbf{G}$. In that case, we say $\mathbf{p}$ is a **repeated network pattern** in $\mathbf{G}$.

Definition 5 (Network Pattern Flattening): A network pattern $\mathbf{p} = (\mathbf{N}', \mathbf{L}', \mathbf{F}', \mathbf{A}')$ in network $\mathbf{G} = (\mathbf{N}, \mathbf{L}, \mathbf{F}, \mathbf{A})$ can be flattened into a vector as follows; define annotation subset of a node $\mathbf{n}$ as $\mathbf{S}(\mathbf{n}) = \{\mathbf{s}_i | \mathbf{s}_i \rightarrow \mathbf{f}, \mathbf{s}_i \in \mathbf{A}, \mathbf{f} \in \mathbf{F}$, and $\mathbf{n} \in \mathbf{N}\}$. Then, we compute a list $\mathbf{P}$ of all annotation subsets of a node $\mathbf{n}$ and all other nodes in the neighborhood of $\mathbf{n}$ by radius $\mathbf{z}$. It is possible that p can have repeated annotation subsets. We can group p by distinct annotation subsets and record the number of repetitions as counts. A vector that consists of counts of such network patterns is called a **flattened vector of a network pattern** (See Tab. 2).

Example 2: In Example 1, node 4 has annotations Tennis (T) and Football (F) hence annotation subsets {T, F, TF}. Node 4 has an incoming link from node 1 and node 1 has annotations data mining (M), Sailing (S), and swimming (W), hence annotation sets {M, S, W, MS, MW, SW, MSW}. Network neighborhood of Node 4 on radius 1 can be flattened as {T:1, F:1, TF:1, M:1, S:1, W:1, MS:1, MW:1, SW:1, MSW:1} where T:1 denotes that pattern T appears only once. Length-1 patterns in this list are {T:1, F:1, M:1, S:1, W1}. Adding all other annotations (i.e., Graph Mining (G), Ski (SK), Baseball (B) from the network, final flattened vector becomes {T:1, F:1, M:1, S:1, W:1, G:0, SK:0, B:0}. It is also possible to separate a flattened vector to indicate whether a pattern came from an incoming neighbor or self and so on. A separated flattened vector for this example becomes { self:{T:1, F:1, G:0, SK:0, B:0}, incoming_radius_1: {M:1, S:1, W:1, G:0, S:0, B:0} }.

Pattern discovery and counting is the step where we flatten network neighborhoods of nodes in a network into feature vectors. First we create a table including two columns which names are from_id and to_id, representing the Twitter following graph (TWITTER_GRAPH) using the dataset in [6]. We define a celebrity node table using a predefined threshold on count of links (CELEBRITY_NODES). The threshold is decided by looking up to the count data. Then top-k celebrity nodes are selected (k piece) from the data created in the previous step using ordering of the count data.

These selected top-k celebrity nodes are used as feature using TWITTER_GRAPH network ($\mathbf{N}$, $\mathbf{L}$, $\mathbf{F}$, $\mathbf{A}$) and then feature annotations are extracted from network nodes and links, and created a table named FEATURE_TABLE. The difference between celebrity node and feature is that celebrity node can be famous persons (singer, artist...etc.), or ordinary person with high link count but features can't be a famous person which means we decide which one is used so that they are subset of celebrity nodes excluding the famous persons.

We can compute the number of annotation subsets as support. For example, in order to compute length-1 patterns in the outgoing neighbors (denoted by o1)

Table 2. This table is explaining the each different kind of pattern type regarding the neighborhood in demand

Type	***Description***
t1	Patterns that links has 1 length (radius is fixed, for instance set to 1)
i1	Patterns that incoming links has 1 length (radius is fixed, for instance set to 1)
i2	Patterns that incoming links has 2 length (radius is fixed, for instance set to 1)
o1	Patterns that outgoing links has 1 length (radius is fixed, for instance set to 1)
o2	Patterns that outgoing links has 2 length (radius is fixed, for instance set to 1)
c1	Patterns that links has length 1(radius is fixed, for instance set to 1, i.e. t1-i1-o1)
c2	Patterns that links has length 2 (radius is fixed, for instance set to 1, i.e. t1-i2-o2)
c3	Patterns that links has length 3 (radius is fixed, for instance set to 1, i.e. t1-i3-o3)
n0	Patterns that links has radius 0 (length is predefined, for instance set to 1)
n1	Patterns that links has radius 1 (length is predefined, for instance set to 1)
n2	Patterns that links has radius 2 (length is predefined, for instance set to 1)
n3	Patterns that links has radius 3 (length is predefined, for instance set to 1)

of all nodes, node itself, and total link of each node which is matched from the feature table is used excluding the celebrity nodes from the TWITTER_GRAPH set.

More complex patterns can be obtained, say annotation subsets of length-2 in the outgoing neighborhood of distance 1. The hint is combining two different features to use as a single feature using the same table (FEATURE_TABLE) twice and same rules we explained above, in the query such as feature_1 || '-' || feature_2.Thus these new-forming features are permutation of features.

SQL provides a viable and scalable option to flatten network structures into feature vectors. After a bunch of transformation SQLs, we convert Twitter graph into a set of records, where each record represent a network node, flattened feature vector of that node, and known annotations of that network node. This tabular representation allows us to use various machine learning algorithms to make predictions.

There are several powerful classification machine learning algorithms that can be used in this work. Although their performances are comparable, one of them usually works better than others. In this work, we compared four supervised machine learning (classification) algorithms. Support Vector Machine (SVM) [10], Naive Bayes (NB), Generalized Linear Model (GLM), Decision Tree (DT). They are compared using accuracy and lift. All the algorithms are available in Oracle Data Miner (ODM) Library. Default ODM parameter values are used since they perform quite good. For all the algorithms, we used k-fold cross validation.

3 Experimental Results

All the experiments are performed on Oracle Exadata V2 using 64 bit Red Hat Linux 2.4 and the Oracle 11g database. Preprocessing is programmed with Structured Query Language (SQL) and Procedural Language/Structured Query Language (PLSQL).

As part of our experiment, the nodes having more than 90000 links are considered as celebrity. The nodes with link count from 90000 to 100000 are selected as features. There are 34 such nodes. Using these features, tag patterns are stored in a tag pattern table including type, node, feature, and count of pattern that has links (support). However, it's hard to calculate the table due to large number (1785) of different patterns. So, only patterns having more than 10000 links are selected. The patterns are enumerated and the lookup table that contains the information about patterns for each type with their support is created. From this, another table is created summing of the support for each pattern. Because our study is aimed to predict the existence of a link, Boolean support (if support is greater than 0 then 1, otherwise 0) is used instead of actual support when it's as target. Thus, count vectors of network nodes that we use to feed the machine learning algorithm, are formed. The generated data is fed to classifiers (SVM, NB, GLM, DT). To choose the machine learning algorithm, 10-fold cross validation is employed. After the machine learning algorithm that is chosen, 70% of data is used for training and the rest of data is used in tests.

3.1 Choosing a Machine Learning Algorithm

In this experiment, we built a dataset from network nodes by converting annotation existence (or counts) as an integer vector. For instance, suppose that {A, B, C, D} as the 4 annotations available in whole dataset, and node v is assigned {A, C}, we build a feature vector for v as {A:1, B:0, C:1, D:0}. Then, for each annotation, we build a learning set by choosing an annotation as target, and omitting the target from the input set. For instance, for testing prediction ability of whether v has annotation A, we convert its vector to {B:0, C:1, D:0} by omitting A.

Table 3. Comparison table for machine learning algorithms using accuracy and lift

Algorithm	***Average Accuracy %***	***True Positive Accuracy %***	***True Negative Accuracy %***	***Lift Cumulative***
Support Vector Machine (SVM)	66.68	99.89	7.66	4.06
Naive Bayes (NB)	87.49	98.81	51.44	4.92
Generalized Linear Model (GLM)	89.18	99.07	45.43	4.88
Decision Tree (DT)	89.91	99.26	38.39	4.43

Next, we build a dataset for every annotation in the network, and test prediction accuracy via 10-fold cross validation. Then, we repeated this procedure for 4 machine learning algorithms, and compared prediction accuracy metrics. We observed that Generalized Linear Models and Decision Trees have higher average accuracy in comparison with other two algorithms (i.e., Support Vector Machines, and Naive Bayes). In addition, we observed a better lift curve for GLM, hence we utilized GLM algorithm for the rest of the experiments.

3.2 Prediction Performance Comparison by Annotation Position

In this experiment, we observe how much prediction capability is supplemented by the known annotations of a node (say vector t1), annotations of the neighbors connected via incoming links (say vector i1), and annotations of the neighbors connected via outgoing links (say vector o1). Finally we built 4 datasets to compare prediction performance: t1, i1, o1, and c1=t1|><|i1|><|o1.

We observed (see Fig. 2) that, incoming links provide more prediction information than outgoing links. We computed that augmenting t1, i1, and o1 (i.e., dataset c1) provided only 11% improvement over using i1 alone, while augmentation supplemented 27% improvement over o1. In addition, providing neighborhood information boosted prediction performance by 362%.

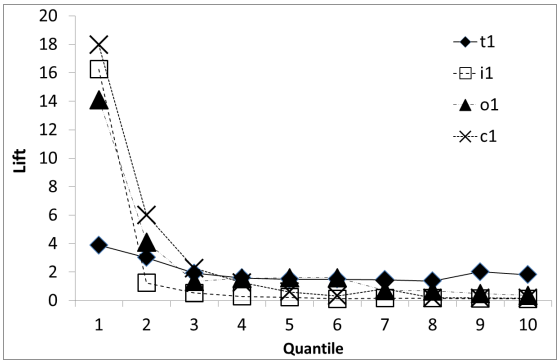

Fig. 2. Impact of annotation position (node itself, or incoming or outgoing neighbors) on prediction capability

3.3 Prediction Improvement by Increasing Pattern Complexity

In this experiment, we build feature vectors from annotations in the direct neighborhood of nodes (i.e., node itself and its direct incoming and outgoing links). We divide direct neighborhood annotation information into 3 distinct vectors. 1) Annotations of the node itself, namely t, 2) annotations of the node's incoming links, namely i, and 3) annotations of the nodes are outgoing links, namely o. For each vector, we build feature sets by computing frequency of annotation patterns. For example, if a node has three annotations, namely, A, B, and C,

we build the following feature vectors: {A, B, C} as length-1 patterns, {AB, BC, AC} as length-2 patterns, and {ABC} as a length-3 pattern. We name feature vectors of node and its direct links by their corresponding pattern lengths. For instance, t1, i1, and o1 correspond to length-1 pattern counts observed in a node v, observed among the nodes pointing to v, and observed among the nodes pointed by v, respectively. Similarly, t2, i2, and o2 represent length-2 patterns, and t3, i3, and o3 represent length-3 patterns.

Different learning sets are considered to compute the impact of pattern lengths in prediction capability:
$c1=t_1|><|i_1|><|o_1$, $c2=c1|><|t_2|><|i_2|><|o_2$, and $c3=c2|><|t_3|><|i_3|><|o_3$, where $|><|$ is the joining operator.

In other words, all feature vectors is generated corresponding to patterns of the same length together.

Note that, we omit all patterns that contain the target annotation from the t1, t2, and t3 portions of the learning sets. For example, if we are attempting to predict annotation A, and t2 vector of a node has patterns {AB, BC, AC}, we omit {AB, AC} from t2, as such patterns already indicate the existence of A.

Predictive capability of c1, c2, and c3 datasets are presented in Fig. 3. An improvement of 8% is observed when length-2 patterns' feature vectors are added, and improvement is only 3.5% when length-3 patterns feature vectors added (according to the lift values at the first quantile). We conclude that more complex patterns in the feature vectors definitely improve prediction capability. However, such improvement gets less significant while the complexity is increasing.

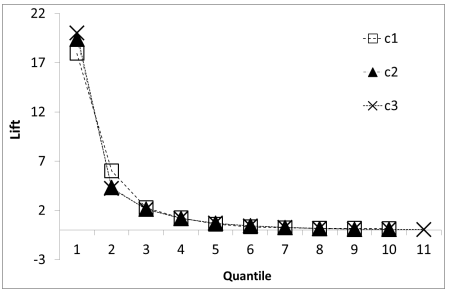

Fig. 3. Impact of pattern complexity on prediction capability

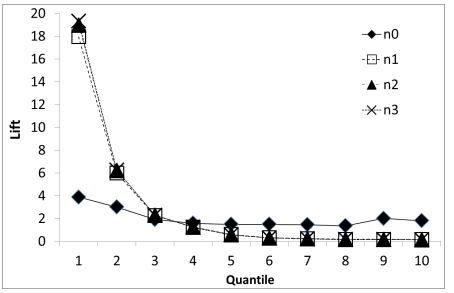

Fig. 4. Impact of network neighborhood radius on prediction capability

3.4 Prediction Information Coming from Indirect Neighbors

In this experiment, we study whether adding data from network neighborhoods with larger radiuses supplement prediction capability. In other words, we first build a feature vector using only the annotations from a node v (say, neighborhood n_0). Then we test with additional annotations coming from direct neighbors of v (say, neighborhood n_1 that is both incoming and outgoing neighbors). Next, we increase radiuses to n_2, n_3, and n_4 as well. In order to augment network

neighborhood annotations into a single vector, we inner join feature vectors of each radius. For instance n_3 means, self-annotations of node v (i.e., n_0), joined with annotation vector of direct neighbors of v (i.e., r_1), joined with indirect neighbors in distance 2 and 3 of v, namely r_2 and r_3: $n_i = n_0 \mid><\mid r_1 \mid><\mid \ldots \mid><\mid r_i$.

Note that we employ annotation patterns of length 1 for this experiment. In this experiment, we observed that, annotations of a node and annotations of direct neighbors provide the most of the prediction power. Radiuses 2 and higher do not bring additional prediction capability worth the computational effort (see Fig. 4). Lift improvement at the first quantile for n_3 over n2 is 1.7%, and n2 over n1 is 5.8%. For the second quantile, improvement for n_3 over n2 is 1%, and n2 over n1 is 1.04%.

4 Conclusions

In this paper, we proposed a methodology for converting network links into network annotations, and predicting missing links and annotations. Our approach is scalable in the sense that network data is preprocessed into a dense and simple feature vector before consumed by complicated machine learning algorithms. We observed in our experimental evaluation that augmenting network-based features to a simple correlation-based prediction algorithm provides 11-27% improvement in prediction accuracy. We plan to extend our work by evaluating use of more complex patterns, and evaluating scalability tradeoffs of those.

Acknowledgments. We would like to thank Zeki Erdem for his comments and support during the development of this work.

References

1. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical structure and the prediction of missing links in networks. Nature 453(7191), 98–101 (2008)
2. Goldberg, D., Roth, F.: Assessing experimentally derived interactions in a small world. Proc. Natl. Acad. Sci. U.S.A. (2003)
3. Golub, B., Jackson, M.O.: How homophily affects the speed of learning and best-response dynamics. Quarterly Journal of Economics (2012)
4. Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., Zadeh, R.: WTF: The Who to Follow Service at Twitter. In: Proceedings of the 22nd International Conference on World Wide Web, WWW 2013, pp. 505–514 (2013)
5. Kirac, M., Ozsoyoglu, G., Yang, J.: Annotating proteins by mining protein interaction networks. Bioinformatics 22(14) (2008)
6. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a Social Network or a News Media? In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 591–600 (2010)
7. Lee, D.: Document ranking and the vector-space model. IEEE Computer Society 14(2) (1997)

8. Lee, R., Sumiya, K.: Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In: LBSN (2010)
9. Liben-Nowell, D., Kleinberg, J.M.: The link prediction problem for social networks. In: LinkKDD (2004)
10. Milenova, B., Yarmus, J., Campos, M.: SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines. In: Very Large Databases, VLDB (2005)
11. Pennacchiotti, M., Popescu, A.M.: A machine learning approach to twitter user classification. In: AAAI Conference on Weblogs and Social Media (2011)
12. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3) (2008)
13. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: Proceeding of Neural Information Processing Systems (2003)
14. Twitter Inc.: Twitter rest api, `https://dev.twitter.com/docs/api`
15. Zhou, T., Lu, L., Zhang, Y.C.: Predicting missing links via local information. The European Physical Journal B 71(4) (2009)