Workshop at ISWC 2007 + ASWC 2007
FEWS on the Web with Semantics Finding Experts

Finding Experts by Link Prediction in Co-authorship Networks

Milen Pavlov[1,2], Ryutaro Ichise[2]

[1] University of Waterloo, Waterloo ON N2L 3G1, Canada
[2] National Institute of Informatics, Tokyo 101-8430, Japan

Abstract. Research collaborations are always encouraged, as they often yield good results. However, the researcher network contains massive amounts of experts in various disciplines and it is difficult for the individual researcher to decide which experts will match his own expertise best. As a result, collaboration outcomes are often uncertain and research teams are poorly organized. We propose a method for building link predictors in networks, where nodes can represent researchers and links - collaborations. In this case, predictors might offer good suggestions for future collaborations. We test our method on a researcher co-authorship network and obtain link predictors of encouraging accuracy. This leads us to believe our method could be useful in building and maintaining strong research teams. It could also help with choosing vocabulary for expert description, since link predictors contain implicit information about which structural attributes of the network are important with respect to the link prediction problem.

1 Introduction

Collaborations between researchers often have a synergistic effect. The combined expertise of a group of researchers can often yield results far surpassing the sum of the individual researchers' capabilities. However, creating and organizing such research teams is not a straightforward task. The individual researcher often has limited awareness of the existence of other researchers with which collaborations might prove fruitful. Furthermore, even in the presence of such awareness, it is difficult to predict in advance which potential collaborations should be pursued. As a result, an expert in a particular field might find himself uncertain as to who to collaborate with.

Experts' semantic descriptions might be very helpful here. If one knew to what extent each researcher is an expert in each field, one could potentially use this knowledge to find researchers with compatible expertise and suggest collaborations. However, semantic descriptions are often unavailable due to lack of supporting vocabulary.

Fortunately, semantic descriptions are not the only tool that can realize the goal of suggesting collaborations. We propose a method for link prediction in networks, where nodes can represent researchers and links - collaborations. Thus, if we could predict the appearance of new links with reasonable accuracy, links that have been predicted but do not appear might be good suggestions for future research collaborations.

Our method extracts structural attributes from the graph of past collaborations and uses them to train a set of predictors using supervised learning algorithms. These predictors can then be used to predict future links between existing nodes in the graph. We test our method on a co-authorship network and the results confirm that the appearance of new collaborations is dependent on past network structure and that supervised learning methods can exploit this dependence to make predictions with reasonable accuracy. The approach is not specific to the domain of co-authorship networks and can be easily applied to virtually all types of networks in which link prediction is desirable.

Section 2 briefly mentions this research and compares it to our method. Section 3 formally defines how we approach the link prediction problem. Section 4 describes the experiments we run and discusses their results. Finally, Section 5 concludes.

2 Related Work

Research in the area of network evolution models [14] is closely related to the problem of link prediction. Kashima et. al [9] attempt to fit a parametrized "copy-and-paste" model to a partial snapshot of a network and use this model to predict the full network structure. The same problem (though viewed in slightly differing ways) is tackled by [16] and [19]: they build partial networks from observable data and use them to infer links that are not directly observable but are likely to exist. These approaches all differ from our view of the problem, as they only work with a static snapshot of a network and do not consider network evolution over time.

A view similar to ours is employed in [12], where a collaboration network is cut into time slices and the network structure of one time slice is used to predict the structure in the next slice. A link is considered likely to appear if a structural attribute for the two end-nodes achieves a high score. The authors test and compare the predictive power of several such attributes. While generally achieving a low prediction rate, their approach is promising in the sense that it significantly outperforms a random pre-

dictor. This implies there is, in fact, a correlation between the structure of a collaboration network and the places where collaborations are likely to appear in the future. Our approach aims to build an improved method for link prediction by utilizing *multiple* structural attributes in the prediction of a single link. The following explains how.

3 Supervised Learning Method for Building Link Predictors

Consider a network G, defined in the following way. Let $G = \langle V, E, W \rangle$ be a weighted graph with nodes $v_i \in V$ and edges $(v_i, v_j) \in E$, $1 \leq i, j \leq |V|$, where $w_{ij} \in W$ denotes the weight of edge (v_i, v_j). For every pair of nodes in the graph we can compute various graph attributes (such as shortest path, common neighbours, etc.), which may be correlated with the probability that a link between the nodes will appear in the future. Once computed, the set of attribute values for a pair of nodes is said to form a *feature vector*. Let f_{ij} denote the feature vector for pair (v_i, v_j). Note that vectors f_{ij} are defined for *all* pairs of nodes in V, not only for the subset E.

There are many structural attributes that can be used to form a feature vector. [2] and [12] describe a multitude of candidate attributes. Clearly, the effectiveness of different attributes will depend on the nature of the network whose links they are utilized to predict. In our method we choose a variety of attributes that are not specific to any type of network but we think might be informative for a wide range of different types of networks. Our selection of attributes is enumerated in Section 3.1.

Let F denote the full set of feature vectors. Formally speaking, $F = \{f_{ij} | (v_i, v_j) \in V \times V\}$. The size of F varies, depending on whether the graph is directed or undirected and whether self-links are allowed. In co-authorship networks, self-links and directed edges are meaningless and so $|F| = \frac{|V| \cdot |V-1|}{2}$; in web page networks such as the Internet, $|F| = |V|^2$.

We want to learn how to use the information contained in each feature vector f_{ij} to predict its *label* y_{ij}. The label y_{ij} will equal *true* iff a new link appears, joining nodes v_i and v_j. We assume that some labels are already known and denote them by the set Y_r. Our goal is to predict the remaining labels Y_s. We train several supervised learning algorithms on the subset of feature vectors that correspond to labels in Y_r. Formally speaking, this *training set* is defined as $F_r = \{f_{ij} | y_{ij} \in Y_r\}$. Testing is done on the set of remaining feature vectors $F_s = F - F_r$, which we will refer to as the *test set*. Since there are many learning algorithms available (and

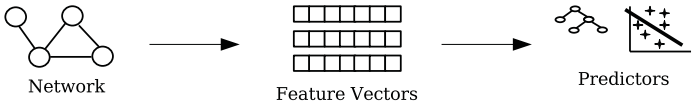

Fig. 1. Building link predictors from network structure. Step 1: Extract feature vectors from network. Step 2: Train predictors on extracted feature vectors using supervised learning algorithms.

Table 1. Attributes computed for each node pair (v_i, v_j).

Attribute name	*Formula*
Shortest path	$\min\{s \mid \text{paths}_{ij}^s > 0\}$
Common neighbours	$\lvert\Gamma(v_i) \cap \Gamma(v_j)\rvert$
Jaccard's coefficient	$\frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{\lvert\Gamma(v_i)\cup\Gamma(v_j)\rvert}$
Adamic/Adar	$\sum_{v_k \in \Gamma(v_i)\cap\Gamma(v_j)} \frac{1}{\log\lvert\Gamma(v_k)\rvert}$
Preferential attachment	$\lvert\Gamma(v_i)\rvert \cdot \lvert\Gamma(v_j)\rvert$
Katz_β	$\sum_{s=1}^{\infty} \beta^s \cdot \text{paths}_{ij}^s$
Weighted Katz_β	Same as above, but $\text{paths}_{ij}^1 = w_{ij}$
PageRank_d^{min}	$\min\{\text{PageRank}_d(v_i), \text{PageRank}_d(v_j)\}$
PageRank_d^{max}	$\max\{\text{PageRank}_d(v_i), \text{PageRank}_d(v_j)\}$
SimRank_γ	$\begin{cases} 1 & \text{if } v_i = v_j \\ \gamma \cdot \frac{\Sigma_{a\in\Gamma(v_i)}\Sigma_{b\in\Gamma(v_j)}\text{SimRank}_\gamma(a,b)}{\lvert\Gamma(v_i)\rvert\cdot\lvert\Gamma(v_j)\rvert} & \text{otherwise} \end{cases}$
Link value	w_{ij}

none of them are universally effective), we see it reasonable to test and compare the performance of several popular ones. Section 3.2 discusses our selection.

For an overview of the method, refer to Fig. 1. The network entity was discussed here. The feature vectors and predictors are discussed in the following subsections, respectively.

3.1 Attributes in a feature vector

A feature vector $f_{ij} \in F$ consists of a number of attributes, computed for the node pair (v_i, v_j). Table 1 shows the attributes we consider in our method. $\Gamma(v_i)$ denotes the set of all neighbours of v_i and paths_{ij}^s denotes the number of paths of length s connecting v_i to v_j. The following briefly describes the significance of each attribute.

The *shortest path* between two nodes v_i and v_j is defined as the minimum number of edges connecting them. If there is no such connecting path then the value of this attribute is generally assumed to be infinite. For many types of networks, the shorter the shortest path between two nodes is, the more likely the nodes are to become linked. *Common neighbours* counts the number of neighbours that the two nodes have in common. E.g., for two researchers in a collaboration network, this is the number of other researchers, with which both have had some collaboration. Arguably, if two researchers tend to collaborate with the same group of other researchers then they are likely to collaborate with each other as well. *Jaccard's coefficient* [18] is a normalized measure of common neighbours. It computes the ratio of common neighbours out of all neighbours (common or not). This is sometimes a better measure than the (unnormalized) common neighbours, especially when one end-node has a substantially larger neighbourhood than the other. *Adamic/Adar* [1] measures similarity between two nodes by weighing "rarer" common neighbours more heavily. The rationale is that two nodes that have a common neighbour that no other node has are often more "similar" than two nodes whose common neighbours are common for many other nodes. Arguably, the more similar nodes are, the more likely they are to be linked. *Preferential attachment* [13] says that new links will be more likely to connect higher-degree[3] nodes than lower-degree nodes. In a collaboration network, this means a new collaboration is more likely to occur between authors who collaborate more often (regardless of who they collaborate with). This (unnormalized) likelihood is reflected by the simple product of the nodes' degrees. *Katz* [10] is a refined measure of shortest path. It considers all paths between two nodes and weighs shorter ones more heavily. The "non-attenuation" parameter $\beta \in [0, 1]$ controls the aggressiveness of weighing. E.g., a very small β yields a measure which is similar to common neighbours, since paths^s-values for higher s will not contribute significantly to the summation. *Weighted Katz* uses the same core formula as Katz, but also observes the weight between linked nodes. Thus, two nodes that are connected by "heavier" paths will achieve a higher score. *PageRank* is the same core algorithm used by Google to rank search results [3]. Using our notation, $PageRank_d(v_i) = (1 - d) + d\Sigma_{v_m \in \Gamma(v_i)} \frac{PageRank_d(v_m)}{\Gamma(v_m)}$. In effect, the rank of a node in the graph is proportional to the probability that the node will be reached through a random walk on the graph. $d \in [0, 1]$ is a damping factor which specifies how likely the algorithm is to visit the

[3] The degree of a node is equal to the number of edges linking to it. Thus, $\deg(v_i) = |\Gamma(v_i)|$.

node's neighbours rather than starting over (refer to [3] for more details). Note that the original algorithm computes ranks over nodes. Since we need ranks over pairs, we take the minimum and maximum page ranks for the two nodes in a pair. *SimRank* [8] is also recursively defined. It states two nodes are similar to the extent they are connected to similar neighbours. $\gamma \in [0, 1]$ is a parameter that controls how fast the weight of connected nodes' SimRank decreases as they get farther away from the original nodes. Finally, the *link value* of a pair of nodes is simply the weight of the edge between them. If such an edge does not exist then the value is assumed to be zero.

3.2 Predictors

First, let us formally define what we mean by a *predictor*[4]. A predictor $p : F \rightarrow \{true, false\}$ is a function that maps feature vectors to the binary space. A good predictor is one for which $p(f_{ij}) = y_{ij}$ holds true for a large proportion of the test feature vectors $f_{ij} \in F_s$.

We build predictors by training several learning algorithms on the training set F_r. Each algorithm can produce different predictors, depending on the nature of the algorithm and the contents of F_r. The following briefly outlines the algorithms we employ in our method.

Support vector machines (SVMs) [4] can be used to solve a wide range of classification problems, including ours. The basic idea is as follows: a feature vector containing n attributes can be mapped to a point in n-dimensional space (where every dimension corresponds to an attribute). Thus, our feature vectors set F is represented by a set of points in this space. Each point then has its own binary label. The goal is to separate the points into two groups so that points with the same label are in the same group. A simple way to do this is by using a *linear* separator (i.e., an n-dimensional hyperplane) and this is the approach we adopt in our experiments. To minimize generalization error, the hyperplane is usually chosen in such a way as to maximize the margins on both of its sides. We use the sequential minimal optimization (SMO) training algorithm, since it is known to perform well with linear SVMs. [15]

Decision trees are also popular tools for solving classification problems. A decision tree is a tree whose terminal nodes are classification outcomes and non-terminal nodes - decisions. Therefore, for each feature vector, the label is obtained by traversing the tree from the root to a leaf

[4] Our definition of a *predictor* is very similar to what is known in the machine learning community as a *classifier*.

(terminal) node, following the branches specified by the decision nodes along the way. An advantage of this approach over SVM is that we can observe the learned decision tree directly and potentially make meaningful conclusions about which attributes are truly important for the prediction problem. To build decision trees we use the J48 (also known as C4.5) algorithm [17], which first chooses an attribute that best differentiates the feature vectors and uses it as a decision node in the tree. The set of feature vectors is then split in two, according to this decision node, and new decision nodes are chosen in the same way for both partitions. The process terminates when all nodes in a partition have the same label or when no attributes can split a partition any further.

A *Decision stump* is a decision tree with only one decision node. It is normally not a good classification tool on its own, but in conjunction with AdaBoost (described below) can sometimes yield good results.

Boosting [5] is a kind of meta-learning: It considers a set of "weak" predictors and computes a prediction based on their individual opinions. A popular boosting algorithm is AdaBoost [7]. AdaBoost trains a set of weak predictors successively, by presenting to each predictor training data on which the previous predictor performed poorly. In the end, the prediction label is decided by the weighted average of the votes of each predictor. AdaBoost can be used with any of the previously mentioned predictors, although it generally achieves a better performance boost when applied with simpler predictors (e.g., decision stump).

Finally, we consider the *random* predictor as a baseline for comparison. This predictor assigns a random label to each feature vector, regardless of attribute values. There is no training involved.

4 Experiments and Results

To evaluate the feasibility and practicality of our method, we conduct a set of experiments on a Japanese co-authorship network. The data comes from the Institute of Electronics Information and Communication Engineers (IEICE). The IEICE is considered by many Japanese scholars to be the Japanese analogue to the Institute of Electrical and Electronics Engineers [11]. The data is hosted by the National Institute of Informatics (NII), Tokyo. It contains information about 110,210 published articles by 86,696 authors, collected over the years from 1993 to (and including) 2006. By testing our method on this co-authorship network, we hope to gain valuable information about which structural attributes are most in-

dicative of future collaborations and to help with identifying potentially fruitful collaborations that have not happened.

Using the notation defined in Secion 3, we construct the network $G = \{V, E, W\}$ as follows. V is the set of all authors in the data set. $E = \{(v_i, v_j) \in V \times V | v_i \text{ and } v_j \text{ have co-authored at least one paper}\}$. Finally, the weight of a collaboration between two authors, w_{ij}, is equal to the number of co-authored papers between them. The full co-authorship network constructed this way is effectively an *undirected* graph, having 86,696 nodes and 672,332 edges.

However, this network contains 14 years of evolution history clumped together and it might be difficult to see (or predict) evolution patterns unless we consider some method of time slicing. We split the data set into two partitions, each consisting of seven years of evolution data, and perform the same experiment on both partitions independently. We want to compare the performances of both sets of predictors thus obtained and check if there are any differences in the way they use structural attributes to predict future links. Such differences might suggest that not only the network structure has evolved but that the evolution dynamics have also changed.

Before jumping into the experiments, let us define some performance metrics for predictor evaluation. For each feature vector, a predictor p can make either a *positive* (P) or a *negative* (N) prediction concerning the corresponding label.[5] In the positive case, if p is correct, the prediction is said to be *true-positive* (TP); otherwise it is *false-positive* (FP). Conversely, in the negative case a prediction can be either *true-negative* (TN) if correct or *false-negative* (FN) if wrong. We can now define the metric *recall* as the proportion of TP predictions out of all *true* labels. Recall will give us an idea of how well p is able to predict collaborations that will happen in the future. It might also be useful to define the metric *precision* as the proportion of TP predictions out of all positive predictions. Precision will be useful in determining how well p fits the whole data (as opposed to just always predicting *true*, which guarantees a 100% recall rate). Using both precision and recall we are able to numerically evaluate and compare predictors.

$$precision = \frac{|TP|}{|TP| + |FP|}, \quad recall = \frac{|TP|}{|TP| + |FN|}$$

[5] To disambiguate: A positive prediction means p thinks the label is *true*.

The following two subsections describe the two experiments and their results individually. Section 4.3 gives some general conclusions regarding both experiments.

4.1 The data from 1993 to 1999

The co-authorship network for the data set spanning the years 1993 to 1999 is constructed the same way as described above. However, before extracting feature vectors we draw attention to several things. First, we note that many collaborations between authors occur only once. This results in many edges of weight one. These "tenuous" edges do not seem to carry much information about collaboration tendencies between authors - in fact, they could often be attributed to chance[6]. For this reason, we find it reasonable to *filter out* all edges whose weight is equal to one.

From the seven years of evolution data we need to extract feature vectors and corresponding labels. We use the first four years for the former and the remaining three for the latter. We note, however, that some authors are only active during the first four years but stop publishing after that. Conversely, others only start publishing in the latter three years and are inactive in the preceding time period. In the graph, this results in a variable nodes set V, which leads to a mismatch between feature vectors and labels. To avoid this, we *trim* the set V to only contain authors that are active during both time periods. The resulting network contains 1,380 nodes and 1,620 edges.

We use the following parameter values for feature extraction: Katz's $\beta = 0.05$, PageRank's $d = 0.85$ and SimRank's $\gamma = 0.8$. These seem to be the popular values in the research community [3, 8, 10]. The count of thus computed feature vectors is upwards of a million. However, we note that many feature vectors are duplicates. The most prominent example is for a pair of nodes that are far apart in the graph, have no common paths between them, score very low on all attributes and end up never publishing together. Such pairs usually result in the same feature vectors. (We consider two feature vectors the same if none of their corresponding attribute values differ by more than 0.001.) We remove all duplicates and obtain a set of 29,437 *unique* feature vectors. This is the set F we will present to our supervised learning algorithms. Of the labels corresponding to vectors in this set, we note that only 1.6% are *true*.

[6] E.g., two authors might never personally collaborate, yet still have their names appear on a paper because of a third party. Unless this happens again, we consider the link between such authors accidental.

For training, we use a subset containing 90% of the vectors in F, sampled at random. Testing is performed on the remaining 10%. Implementations of the learning algorithms described in Section 3.2 are already available in the Weka software package [20]. We use these implementations for our experiments.

Let us look at the results shown in Fig. 2. The SMO, J48 and AdaBoost/SMO predictors display outstanding recall rates, with SMO even going so far as to predict 100% of the *true* test samples correctly. The equation used by SMO to accomplish this feat assigns relatively high weights to the Katz and preferential attachment attributes (respectively, -3.1 and 1.8) and low weights to the rest of the attributes (between -1 and 1). The shortest path, Jaccard's coefficient, PageRank$_{max}$ and SimRank are almost completely ignored, since each of their weights is less than |0.06|. The AdaBoost version of SMO uses a weighted average of ten "weak" SMO predictors. The first one is the same as above, however, the other nine seem to perform much worse and lower the overall performance of the compound predictor. The nine predictors seem to favour shortest path and link value attributes in their weight distributions. J48 performs almost as well as AdaBoost/SMO. We were hoping that the decision tree learned by J48 could give us some insight as to which attributes are most informative to the co-authorship prediction problem, but in actuality the tree was too large to even visualize. Similar to the case of AdaBoost/SMO, AdaBoost/J48 achieves a lower performance than J48 alone. Lastly, the performance of AdaBoost/DecisionStump should be thought of as the worst of all predictors. Though it achieves a good precision rate, the reason it does so is simply because it is overly conservative in its *true* predictions: This leads to a decrease in FP rates (boosting precision) at the cost of a considerable increase in FN rates (resulting in the abysmal recall performance).

4.2 The data from 2000 to 2006

We run this experiment the same way as described in the previous subsection. During this latter seven-year period however, the number of active authors, as well as their publishing rates, seem to have increased, resulting in a network of 3,603 nodes and 7,512 edges. We extract more than ten million feature vectors in total and take only the 1.6 million unique ones. Out of those, we note that only 0.019% have *true* labels.

We observe some differences when we compare the results from the previous subsection to those shown in Fig. 3. In the latter, SMO and J48 both score lower than before, on both measures. This might be an

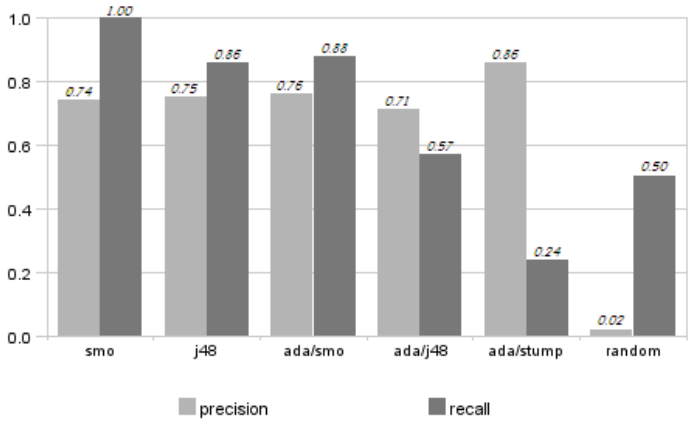

Fig. 2. Classification results for the years 1993 $\sim$ 1999.

indication that new collaborations are now less dependent on past network structure or that the dependence holds for shorter periods of time. AdaBoost does not improve the performance of J48, but it achieves a remarkable 100% recall rate when combined with DecisionStump. It seems here the case is reversed from before: Now this predictor is being overly generous in its *true* predictions, which leads to a decline in precision. All ten "weak" DecisionStumps base their one-node decisions on either the shortest path or link value attributes.

Weka failed to provide a result for the AdaBoost/SMO experiment. We suspect that the size of the training set might have been prohibitively large for the limited amount of memory (10GB) on the test machine.

4.3 General Discussion

First, we note that virtually all of our predictors outperform the random classifier. This is encouraging in itself, considering our predictions are solely based on structural information from the co-authorship network and do not take into account any node-specific properties (such as university affiliations, geographical locations, research areas, etc.). It also confirms the findings of [12], namely, that there is valuable information to be learned from the structure of a collaboration network, with respect to its evolution.

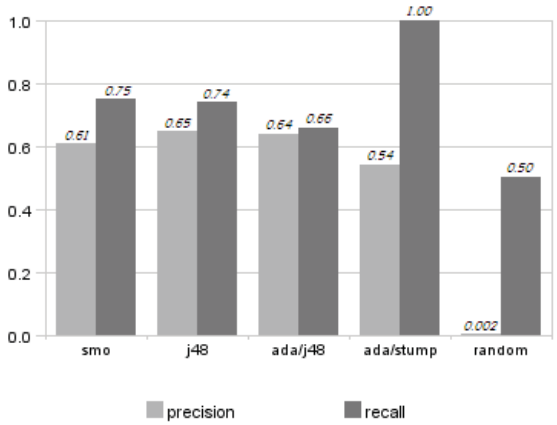

Fig. 3. Classification results for the years 2000 $\sim$ 2006.

Some of the predictors' recall rates might seem overly optimistic. This might indicate that the evolution dynamics of the tested co-authorship network are easy to predict or that the proportion of *true* test samples is too small to yield reliable metric readings.

Regardless, we observe that our predictors do make meaningful predictions for the data tested on. For example, the misclassified samples by J48 in the second experiment include pairs of authors who have a high number of common neighbours (up to 9), large preferential attachment value (up to 171) and have co-authored plenty of papers in the past (up to 23), yet end up not collaborating again when it seems likely they would. Even when making mistakes, the predictor is not entirely unreasonable. In fact, these "mistakes" might be a fair indication that such collaborations should be encouraged.

Overall, we see the results of both experiments as evidence that our method can be very effective in building accurate link predictors in co-authorship networks. Since the method itself relies solely on structural attributes of the underlying network and on general supervised learning algorithms, it should be easily extendible to any kinds of networks in which link prediction is desirable. Whether similar success can be achieved in such networks remains to be seen.

5 Conclusion

This paper presented a supervised learning method for building link predictors from structural attributes of the underlying network. In a network of researchers, where a link represents a collaboration, such predictors could be useful is suggesting unrealized collaborations and thus help in building and maintaining strong research teams. In addition, by analyzing the algorithmic structure of predictors build for a specific network, we could gain valuable information about which attributes are most informative for the link prediction problem and use this knowledge as a basis for specifying vocabularies for expert description.

There are many directions future research in this area might take. We think an important next step would be evaluating the generality of the method by testing it on various kinds of networks and comparing the predictive accuracy of the respective link predictors. It would be also interesting to see if predictors would rely on different structural attributes in different networks and if there is any evident correlation between the type of network and informative attributes. We also believe the process of time slicing in building network data should be investigated further. For example, in both of our experiments, we use the first four years of network history to predict the evolution in the following three. However, many networks do not evolve at a constant rate. It would be interesting to see if this rate varies significantly and if we can adjust the durations of the history and evolution time spans to take such variations into account. One way to do this would be by compiling different pairs of history-evolution data and comparing predictor performances on them.

References

1. L. Adamic, E. Adar. "Friends and neighbors on the web". *Social Networks*, 25(3), 2003.
2. K. Börner, S. Sanyal, A. Vespignani. "Network Science". In B. Cronin, ed., *Annual Review of Information Science and Technology*, Information Today, Inc./American Society for Information Science and Technology, Medford, NJ, in press.
3. S. Brin, L. Page. "The anatomy of a large-scale hypertextual Web search engine". In *Proceedings of the 7th international conference on World Wide Web*, 107-117. Brisbane, Australia, 1998.
4. C.J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". *Data Mining and Knowledge Discovery*, 2(2):1-47. 1998.
5. T.G. Dietterich. "Ensemble Methods in Machine Learning". In J. Kittler, F. Roli, eds., *Proceedings of the 1st International Workshop on Multiple Classifier Systems*, LNCS, vol. 1857, 1-15. Springer-Verlag, London, 2000.
6. R.O. Duda, P.E. Hart, D.G. Stork. "Pattern Classification" (2nd Edition). Wiley-Interscience, 2000.

7. Y. Freund, R.E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". *Journal of Computer and System Sciences*, 55(1):119-139, 1997.
8. G. Jeh, J. Widom. "SimRank: A measure of structural-context similarity". In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada. 2002.
9. H. Kashima, N. Abe. "A Parametrized Probabilistic Model of Network Evolution for Supervised Link Prediction". In *Proceedings of the 6th IEEE International Conference on Data Mining*, 2006.
10. L. Katz. "A new status index derived from sociometric analysis". *Psychometrika* 18, 39-43, 1953.
11. G. LaRowe, R. Ichise, K. Börner, "Analysis of Japanese Information Systems Coauthorship Data". In *Proceedings of the 11th International Conference on Information Visualization*, 459-464, 2007.
12. D. Liben-Nowell, J. Kleinberg. "The Link Prediction Problem for Social Networks". In *Proceedings of the 12th International Conference on Information and Knowledge Management*, 2003.
13. M.E.J. Newman. "Clustering and preferential attachment in growing networks". *Phys. Rev. E*, 64(2):025102, 2001.
14. M.E.J. Newman. "The structure and function of complex networks". *SIAM Review*, 45:167-256, 2003.
15. J.C. Platt. "Fast training of support vector machines using sequential minimal optimization". In B. Schölkopf, C. J. Burges, and A. J. Smola, eds., *Advances in Kernel Methods: Support Vector Learning*, 185-208. MIT Press, Cambridge, MA, 1999.
16. A. Popescul, L. Ungar. "Statistical relational learning for link prediction". In *Workshop on Learning Statistical Models From Relational Data*, 81-90. ACM Press, New York, 2003.
17. J.R. Quinlan. "C4.5: Programs for Machine Learning". Morgan Kaufmann Publishers, 1993.
18. G. Salton, M.J. McGill. "Introduction to Modern Information Retrieval". McGraw-Hill, Inc., New York, NY, 1986.
19. B. Taskar, M.-F. Wong, P. Abbeel, D. Koller. "Link prediction in relational data". In *Proceedings of the Neural Information Processing Systems*, 659-666. MIT Press, Cambridge, MA, 2003.
20. I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, S.J. Cunningham. "Weka: Practical machine learning tools and techniques with Java implementations". In H. Kasabov, K. Ko, eds., *ICONIP/ANZIIS/ANNES International Workshop*, Dunedin, 1999.