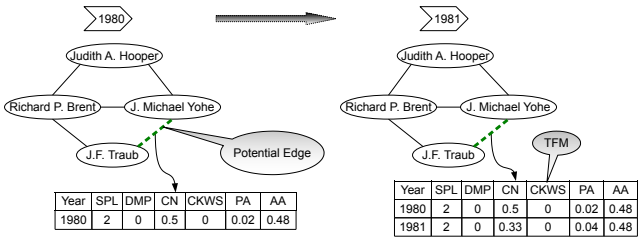
Feature Engineering for Supervised Link Prediction on Dynamic Social Networks

Jeyanthi Narasimhan[1], **and Lawrence Holder**[1]
[1] School of Electrical Engineering and Computer Science, Washington State University,
Pullman, WA 99164-2752, USA

Abstract— *Link prediction is an important network science problem in many domains such as social networks, chem/bio-informatics, etc. Most of these networks are dynamic in nature with patterns evolving over time. In such cases, it is necessary to incorporate time in the mining process in a seamless manner to aid in better prediction performance. We propose a two-step solution strategy to the link prediction problem in dynamic networks in this work. The first step involves a novel yet simple feature construction approach using a combination of domain and topological attributes of the graph. In the second phase, we perform unconstrained edge selection to identify potential candidates for prediction by any generic two-class learner. We design various experiments on a real world collaboration network and show the effectiveness of our approach.*

Keywords: Dynamic Graph Mining, Supervised Learning, Link Prediction, Feature Extraction, SVD



Year	SPL	DMP	CN	CKWS	PA	AA
1980	2	0	0.5	0	0.02	0.48

Year	SPL	DMP	CN	CKWS	PA	AA
1980	2	0	0.5	0	0.02	0.48
1981	2	0	0.33	0	0.04	0.48

Fig. 1: Time-Feature matrix (TFM) and a toy example with real DBLP authors. SPL: Shortest Path Length, DMP: Degree Mixing Probability (1), CN: Common Neighbors, CKWS: Common Keywords, PA: Preferential Attachment, AA: Adamic-Adar measure. Refer to Table 5 for related details. *J.Michael Yohe* and *J.F. Traub* are yet to collaborate during the years *1980* and *1981*, so *(J.Michael Yohe, J.F. Traub)* becomes a potential edge and its TFM is gathered over years till it forms.

1. Introduction

One of the graph mining tasks is Relationship Prediction or more commonly, Link Prediction (LP). It refers to predicting the likelihood of existence of a link between two entities of a network based on the existing links, node attribute information and other relevant details [1]. Instances of the LP problem can be found in application domains such as sociology, bio-chemistry, and online social networks. Communication networks can be studied under this context to disclose existing but missing communication between two people. Given the past network information of "follows" relationships from Twitter, it is possible to predict the future important person(s) in the network which is key to many business tactics like viral marketing [2]. Generally, any ecosystem, whether physical or abstract, can be mapped as networks to study the relationship formation patterns, and we are interested in finding an answer to the question: *"Is it possible to build features from the graph topological measures and also time information in such a way that any supervised learner is able to perform better on the LP task?"*

The existing approaches to the LP problem using supervised learning use a direct feature construction approach where each constituent element of a feature vector is found by measuring a global or local graph/node/edge metric. Since we are interested in dynamic networks, we offer a feature construction approach (Section 2) that is indirect in nature, yet easy to compute and includes time as an inherent component (referred hereafter as `MetaFeatures`). We also found that the existing work (Section 5) restrict their prediction to a specific set of nodes. For example, PathPredict [3] selects authors based on the number of papers and limit the training set to nodes that are directly reachable. We observe from figure 3 (Section 4.1) that majority of the edges that form are between initially unreachable nodes. Few other works ([4], [5]) restrict the nodes for potential edge prediction to those seen in training years, but since we are looking at dynamic graphs, imposing this constraint is not suitable here. We have attempted to provide a generic solution with these issues in mind. The contributions of this work include: 1) Novel meta-feature vector construction for the LP problem based on a well-established Linear Algebra technique. 2) Events like node arrival, edge formation, and deletion, with associated time information give rise to dynamic graphs, and such graphs can grow or shrink over time. We show the applicability of the meta-feature vector to dynamic graphs through experiments (Section 4). 3) We design several experiments to evaluate the above feature vector as a predictor of future link occurrences and compare with the state-of-the-art.

2. Model: Indirect discriminative feature construction

Our motivation for working with simple graph topological measures of a dynamic graph and using supervised learning for the LP problem is discussed below. Firstly, we think that there is still room for improvement in the homogeneous graph approach as invariably existing approaches use heuristics to work with only a small portion of the graph. Secondly, an ensemble of graph topological features is observed to be more effective for the problem at hand than using them separately [3]–[5]. Lastly, the importance of discriminative features in supervised learning cannot be stressed more, as any powerful learning algorithm tends to fail when not fed with good features. The necessity to combine time with the above three requirements, leads to our design of a time-feature matrix for meta-feature construction.

2.1 Features: An overview

Graph growth at the macro level is attributed to the node and edge arrivals at the micro level with continuous evolution of features. Each potential edge between any two nodes that have been in the network for some time carries with it what we call *track/historical information* (TI). This TI is maintained in a matrix format with the time in rows and static graph topological easy-to-construct measures in columns. A comprehensive list of static features suitable for the LP problem in general is given by [6]. Figure 1 shows the feature matrix for a small evolving graph between two specific authors from DBLP. We have used six features in most of the experiments - Some of the experiments use fewer features and their related details are explained in section 4.

Below is a brief overview of how each feature in this work is computed. Let $\mathcal{G}$ be the given undirected graph and $\mathcal{N}$ the set of all nodes at a given instant. Let $x, y \in \mathcal{N}$ be the authors or nodes of interest in $\mathcal{G}$. Let $\Gamma(x)$ be the set of neighbors of node x and $\mathcal{D}(x)$ its degree.

- **Common Neighbors**: $CN(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\mathcal{N}|}$
- **Preferential Attachment**: $PA(x,y) = \frac{|\Gamma(x)| * |\Gamma(y)|}{|\mathcal{N}|}$
- **Adamic-Adar**: $AA(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log|\Gamma(z)|}$
- **Common Keywords**: After stopping and stemming [7]: $CKWS(x,y) = \frac{|x_{kws} \cap y_{kws}|}{|x_{kws} \cup y_{kws}|}$
- **Degree mixing probability**: Adapted from Networkx [8]. Defined as the joint probability of degrees of two nodes under consideration. For any two degrees, $\mathcal{D}_i$ and $\mathcal{D}_j$, this value is calculated as:

$$DMP(\mathcal{D}_i, \mathcal{D}_j) = \frac{\sum_{\substack{\forall x \\ \mathcal{D}(x)=\mathcal{D}_i}} \sum_{\substack{\forall y \in \Gamma(x) \\ \mathcal{D}(y)=\mathcal{D}_j}} |(\mathcal{D}(x), \mathcal{D}(y))|}{\sum_{\forall x} \sum_{\forall y \in \Gamma(x)} |(\mathcal{D}(x), \mathcal{D}(y))|} \quad (1)$$

- **Shortest Path Length**: It is the shortest distance (in hops) between any two reachable nodes in $\mathcal{G}$.

Table 1: Feature derivation from a Time-Feature Matrix (TFM). In column 2, $7 := 6\ (features) + 1\ (time)$

Graph growth-Span	TFM Shape	Constructed Vector
1 (second/minutes/. . .years)	1×7	1×6
2 (second/minutes/. . .years)	2×7	1×6
3 (second/minutes/. . .years)	3×7	1×6

Fig. 2: Typical supervised framework for solving the LP problem. Approaches differ predominantly in the way of subsetting the nodes for edge prediction. This leads to analysis on an induced subgraph.

2.2 Time-feature matrix (*TFM*): Theory

The novel meta-feature vector of our system is constructed in two steps. In the first stage, a matrix of features is calculated for each time unit and each potential edge as shown in Figure 1. Such a matrix can also be viewed as a *Multivariate Time Series* [9] holding the evolution of features of a potential edge. In the second stage, we compute the *dominant right singular vector* (meta-features) of the matrix calculated in stage one, after slicing out the time column. This way we convert multidimensional feature values to single-dimensional ones. Refer to Table 1 for a prototypical construction of feature vectors from the TF matrix. To see why we take only the dominant singular vector, let us revisit the SVD steps.

Any $t \times d$ real matrix A can be decomposed as $A = U\Sigma V^T$, where U is a $t \times t$ orthogonal matrix whose columns are the eigenvectors of AA^T, Σ is a $t \times d$ diagonal matrix with its diagonal entries in descending order (the diagonal entries are $\sigma_1, \sigma_2, \ldots$), and V is a $d \times d$ orthogonal matrix whose columns are the eigenvectors of A^TA. Let $\mathcal{R}(A)$ be the row space of A and $r(A)$ be its rank. From our experiments, through the *SVD* of TFMs, we found that irrespective of their shape (Table 1), $r(TFM) = 1$ and $\sigma_1 >> \sigma_2 \simeq 0$. In such cases, the following holds true:

$$\mathcal{R}(A) \subseteq a\boldsymbol{v}_1 \subset \Re^d,\ a > 0 \quad (2)$$

$$dim(\mathcal{R}(A)) = r(A) = 1 \quad (3)$$

where $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d]$, $\boldsymbol{v}_i \in \Re^d$ and dim is the dimension of a vector space. It can be seen that the $\mathcal{R}(A)$

Table 2: Highlights of LP using supervised learning. *Any type of learning algorithm. +More than one dataset.

Highlights	Classification algorithm	Heterogeneous network	Feature construction	Dynamic network	Dataset
PathPredict	Logistic Regression	✓	✓	×	DBLP
[4]	SVM-RBF	×	✓	×	DBLP+
[5]	Random Forest	×	✓	×	Condmat+
MetaFeatures	*	×	✓	✓	DBLP

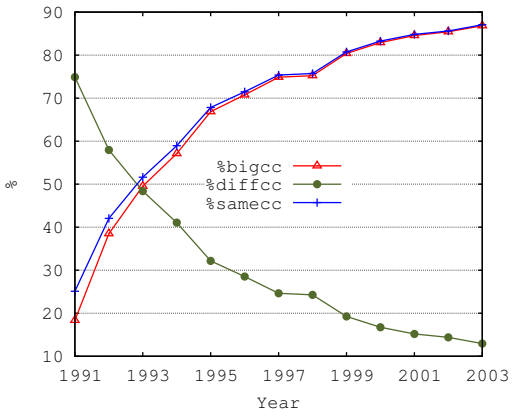


Fig. 3: Positive edges (with TI) Vs CCs in DBLP. *%bigcc*: proportion of positive edges that form in the BigCC, *%diffcc*: proportion of positive edges connecting two different CCs. Because of the affinity of BigCC with small CCs, *%samecc* curve is seen merging with *%bigcc*.

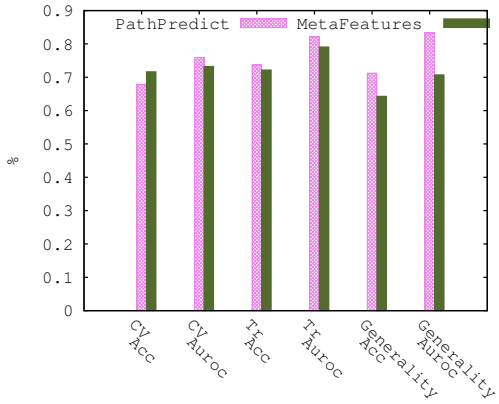


Fig. 4: Performance of meta-features based on Table 3 settings. The measures are CV: 10-fold cross-validation accuracy (Acc) and area under ROC ($Auroc$), Tr: Training set acc and auroc, and generalization performance on test years

is spanned by the first r columns of V, where $r = 1$ in our case. This clearly indicates that despite the shape of a TF matrix, all its rows are linearly dependent and they can be replaced by the single basis vector, $\boldsymbol{v}_1$. By utilizing $\boldsymbol{v}_1$, we get uniform lower-dimensional representation of all TFMs independent of their shape. We have also come across cases of an entire column being zero leading to a rank-deficient matrix, and we discuss this in our future work section

We hypothesize and validate by experiments that this TF matrix and its meta-features capture the discriminative information necessary for LP when cast as a binary classification problem. It should be noted that the time could be in any unit: seconds, minutes, years, etc., but should be uniform across all edges. Because of the SVD step, our method does indirect feature construction as opposed to the related work. In Table 1, for a graph of age $\breve{t}$, we construct TFM for all the potential edges. For those edges that appear during $\breve{t}-$th timestamp, we segregate them into one class and the potential edges become the second class yielding a two-class dataset. For these two classes of TFMs, we do feature extraction as explained in the previous paragraphs.

3. Evaluation

The two-class formulation of the LP problem in existing work [3]–[5] is given by Figure 2. Though our approach is more general, applicable to both static and dynamic graphs, for the sake of comparison, we replicated the experiments of one of the above works [3], except for a small change (discussed in section 4.1). We also used the same learning algorithms and metrics to compare the performance. We report our results on the DBLP [10] co-authorship network and compare with PathPredict [3]. We did some experiments for comparison with [4]and [5], but since these works do not address the generalization issue, which is important for us in a dynamic setting, we do not report the results here for brevity - TFM's performance was comparable with these works as well.

3.1 Experimental setup

Implementation is done in Python(Weave) [11] and C/C++ and uses Networkx graph library, LAPACK and LIBSVM [12] tools for related operations. Since the code involves multiprocessing routines, the execution is done on a 16-core 3.6GHz machine.

DBLP statistics: For the years 1937 to 2012, there are about 800K articles and 1.1M in-proceedings. The authors and edges are 1.1M and 4.1M in number. In the full grown graph, there are about 94K connected components and the largest component has 84% of the total nodes. We construct an undirected and weighted graph from this dataset for feature extraction. The graph growth is dynamic, and we allow for node and edge arrival, but not their deletion, hence this graph grows over time monotonically.

4. Results and Analysis

We conducted a series of experiments (Sections 4.1, 4.2), checking the generality and the skewed class distribution suitability to compare our results with PathPredict. The number of features in the TF matrix differs for each experiment and the reasons behind this are explained in the respective sections. For the curious reader, we provide table 2 showing the contrast between the link prediction solution strategies referenced in this work that use the supervised learning paradigm.

Table 3: Experimental Setup for Section 4.1

	Train years	Test Years	Graph type
PathPredict	1989 - 2002	1996 - 2009	Subgraph
MetaFeatures	1981 - 1988	1985 - 1992	BigCC

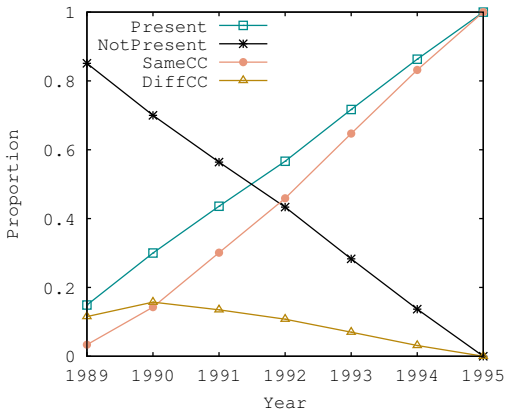


Fig. 5: DBLP (89 - 02). Training set author join patterns. $Present + Notpresent =$ total *#source authors* at the end of training years. Non-zero *diffCC* indicates many rank-deficient TFMs.

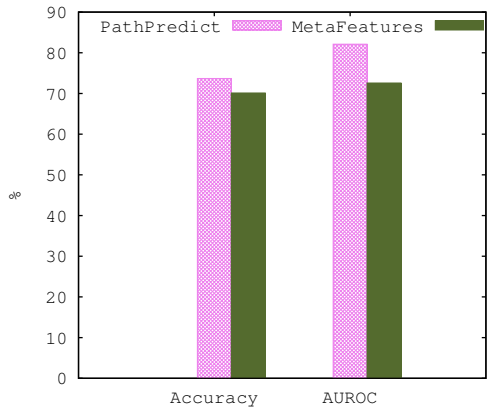


Fig. 6: DBLP (89 - 02). Performance comparison based on Table 4 settings. Accuracy and Auroc are calculated over the years shown in Figure 5 by adding the individual timestamp performance shown in Figures 7 & 8.

4.1 Experiment 1

Table 3 shows the differences in the setup between MetaFeatures and PathPredict for this experiment. We use the Logistic Regression implementation from LibLinear [13] to use the same learner as PathPredict. The approaches differ in the node selection process. While we do not restrict the source and target authors when considering potential edges, PathPredict picks close-to-prolific authors and their 2 or 3-hop authors as targets (indicated as subgraph in the Table 3 as this essentially reduces the graph to a subgraph of those nodes). BigCC in the table stands for the biggest connected component based on longitudinal growth (restricted to nodes in the bigCC of timestamp-1). To avoid negative entries in the TF table when calculating the SPL, we restrict the edges selection to bigCC. Implications of this are discussed in detail in future sections.

The importance of the bigCC in this work is stressed in Figure 3. The figure shows the pattern of edges that form in the graph for which we were able to collect the TI in relation to the connected components(*CCs*). It is clear from this figure that as the graph matures over years the positive edges form predominantly between two authors in the same CC, and all those components eventually get merged with the bigCC. As expected, the positive edges connecting two different CCs diminish quickly in number, but are significant in the initial stages. Hence, to accommodate all edges that form, however to avoid negative entries, we chose the bigCC based approach.

4.1.1 Results

Figure 4 shows cross-validation and training years accuracy, the area under the ROC curve and the model generality performance on DBLP. As can be seen, MetaFeatures stay close to current work in prediction despite our unconstrained source/target author selection, i.e., we consider two authors who are even more than 2 or 3-hops away for potential edge prediction.

4.2 Experiment 2

Here we set out to compare training set prediction performance between the two approaches, *MetaFeatures* and *PathPredict*. Table 4 shows the experimental setup. At this point, we would like to emphasize that it is common in the literature [3]–[5] to restrict the performance analysis to just training set based metrics like cross-validation accuracy. We believe that it is important to measure a solution to the LP problem by considering generalization performance as it is always possible to get good results by over-training on the training set. Whereas in experiment 1, we used 6 features including *SPL*, because of the disconnectivity of authors of potential edges, we had to remove the *SPL* feature from the TF table in this experiment to avoid feeding negative values to the SVD step. It is not only meaningless to have negative entries in SVD, the result of such a decomposition is hard to interpret since those negative values do not have physical significance. Indeed, we enter negative values to indicate that nodes are in different CCs and have a very large distance, but since mathematically negative values are smaller than their positive counter parts, their purpose is not served.

Refer to Figure 5 to see the proportion of authors of potential edges joining the graph in different CCs during the training years. Since we operate on growing graphs, the *present* and *notpresent* represent the potential authors that have joined the graph or not. The non-zero value of

Table 4: Experimental Setup. $*$: unconstrained node selection.

	Train years	Graph type	Learner
PathPredict	1989 - 2002	Subgraph	Logistic regression
MetaFeatures	1989 - 2002	$*$	Logistic regression

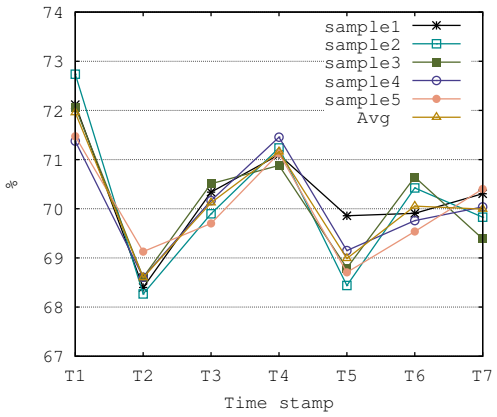


Fig. 7: DBLP (89 - 02). Accuracy Performance of Meta-Features over individual timestamps. Figure 6 shows the overall behavior.

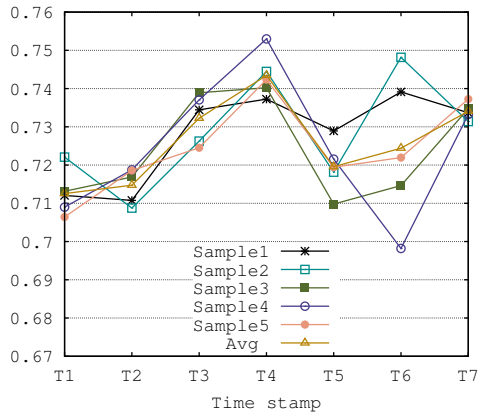


Fig. 8: DBLP (89 - 02). Auroc Performance of MetaFeatures over 7 timestamps shown in Figure 5.

the *DiffCC* curve shows that in the beginning a number of authors join the graph connecting different CCs. For example, about 14% of total #source authors are $present$ in the graph in the year 1989. Those TFMs are of shape 7x5. Of those, about 78% are in different CCs. Owing to the way we construct feature matrices, this is a significant detail because for many potential edges this leads to rank-deficient and sparse TFMs. The *sameCC* line in figure 5 represents the pattern of authors joining the graph in the same CC. Thus, *#present = #sameCC + #diffCC*.

4.2.1 Results

Figures 7 and 8 show the training set performance of MetaFeatures for the listed years. Since time information is an inherent component of our model, the figure shows the performance for each time stamp (in other words, potential edges have different TI). T1 indicates timestamp 1. Of all the edges of interest, we categorize the positive and negative edges into 7 categories depending on the number of years of information we collected on them. The five set of samples shown are created by under-sampling majority class edges, with positive edges (minority class) replicated each time (totally 5), thus these samples have equal number of edges from both classes in all the 5 subsets (this also reduces bias and allows for broader range of majority samples to be included in learning). For each sample, we set the hyperparameters after 10-fold cross validation. The LP problem is an ideal case of the extreme-skew class distribution based binary classification problem, and we conduct some experiments to this end in Section 4.3. Refer to Figure 6 for overall comparison. With just 5 features, MetaFeatures performs comparably (recall *SPL* is removed here). This shows that choosing the right graph topological measure helps to improve the performance. We plan to investigate this in detail with more complex features.

4.3 Experiment 3

This experiment was designed to solve the LP problem preserving its imbalanced nature. Figure 9 captures the acute level of imbalance between the two classes of edges - those that form (minority) and those that do not.

Traditionally, though the existing work recognizes this problem as such, the solution strategies are invariably provided after random sampling (under- sampling) of majority class samples - in this case, the edges that do not form. Both the experiments 1 and 2 (sections 4.1 and 4.2) follow this line of solving the problem with the TF table, but this experiment was designed to see if retaining the original distribution of classes helps for better prediction performance. Figure 10 shows that the positive edges are only a small portion of edges that form (which we know is a negligible portion of those that do not form). This indicates that a significant portion of edges form because of exogenous reasons (e.g., an author shifting to a different university) and their TI cannot be collected. Of those positive edges, a small portion are slings. In this work, we do not consider such edges for prediction, as they do not include two separate authors. For example, in figure 10, during the year 1990, of the total number of edges that formed, only about 3.5% were not slings and had TI associated with them.

Since we find various types of classification algorithms used in conjunction with the LP problem like Bagging with Random Forests [5] and SVMs with RBF [4], we experimented with the *polynomial kernel* (Figure 12) in combination with SVMs. The hyperparameters were set based on 10-fold stratified cross validation experiments. To make the SVM learner unbiased towards the majority class, we did cost-sensitive learning with minority class misclassifications highly penalized. We used LIBSVM for all the trials under this section. The DBLP years we used were from 1980 to 1985 and the metric is training set accuracy. The class distribution is shown in Figure 11 (skewed class

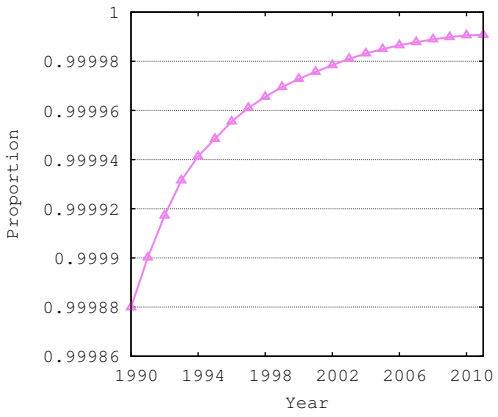


Fig. 9: DBLP over years. Increase in sparsity of the edges in the graph. This directly controls the % of positive edges with TI in Figure 10.

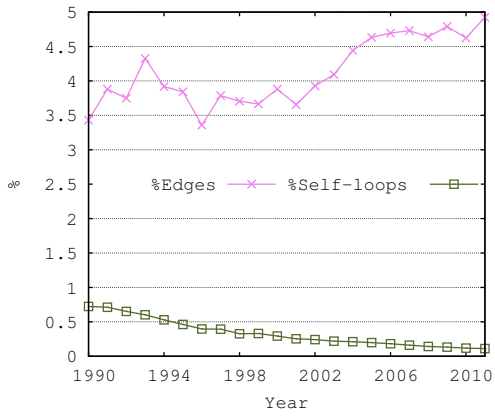


Fig. 10: DBLP. Fraction of edges that form with TI. *Edges*: have two distinct authors. *Self-loops*: Single author papers.

distribution is apparent in the figure).

In addition to the 6 features discussed in section 4.1, we also used *communicability* measure [8] in this experiment. Because *SPL* and *communicability* are path-based measures, we restricted our graph to the bigCC based longitudinal growth.

4.3.1 Results

Figure 12 shows the individual class based accuracy. Sometimes, we had to set the SVM hyperparameter C to even 1000 with misclassification cost of *pedges* class to 100. Though the accuracy values are high, we plan to investigate further here with other years for two reasons: one, there are variations in the results for the *pedges* case and two, it is a well-known fact that a high C will overfit the data. The accuracy for *medges* could be because of C or their sheer count. The last two pairs of bars show the effect of increased penalty to the minority class.

5. Related work

We broadly classify the solution strategies of the LP problem into 5 main categories: 1) Technique, 2) static/dynamic graphs, 3) heterogeneous networks, 4) LP flavor (candidate edges for prediction) and 5) the domain of application. Some of the literature is discussed in Section 1 and throughout the paper. For a solution using a regularized matrix factorization of the adjacency matrix of the graph to learn the latent features using stochastic gradient descent, refer to [14]. This solution is on static homogeneous graphs and domain of application is generic, however they restrict the prediction to nodes that are only two hops away which will not work in scenarios shown in figure 3. Techniques that use spectral theory [15] use graph kernels and model link prediction as spectral transformation of the Laplacian matrix. This approach is on static homogeneous graphs and works only with the bigCC edges. One main issue with the matrix methods ([14], [15]) is the difficulty in accommodating new nodes. While most of the works concentrate on finding new links, Huang and Lin [16] use a time series methodology to find the repetitive links. MRIP [17] solves the problem on heterogeneous networks and does temporal analysis, however their feature construction technique does not involve time directly. Our work aligns with the solution strategies that extracts proximity features from a homogeneous graph to apply supervised learning, but differs from the literature (Table 5) because we solve the problem for dynamic graphs. The table also shows the list of features used in MetaFeatures and in the related works. For a survey on the LP problem, refer to [1], [18].

Table 5: Representative list of features in this and other relevant work for Link Prediction using binary classification.

PathPredict [3]	[5]	[4]	MetaFeatures
• Path Count •Random Walk • Normalized Path Count	• Degree • Common Neighbors • PropFlow • Preferential-Attachment	• Sum of papers • Shortest Distance • Sum of neighbors • Second Shortest Distance	• Shortest-path length • Preferential Attachment • Adamic- Adar • Common Keywords

6. Conclusion and Future Work

We presented an indirect way of constructing feature vectors encapsulating time and tested its suitability for the LP problem framed in the binary classification setting. Through various experiments, we could see that having a correct set of features is important for better classification performance. We found that many TF matrices have zeros in an entire column, leading to singular matrices. This is a direct consequence of potential edges forming by connecting different connected components. We will have this problem as long as we include path and neighbor based measures.

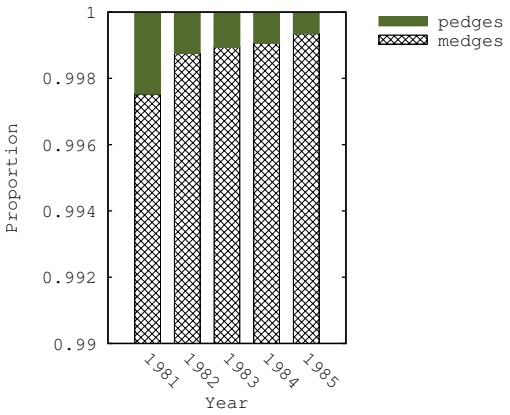


Fig. 11: Class distribution over the years 1980 - 1985 showing dominance of negative edges, edges with TI that never form (*medges*). Greater the age of the graph, larger the skew in class distributions because of sparsity (Figure 9). *pedges*: edges with TI that eventually form.

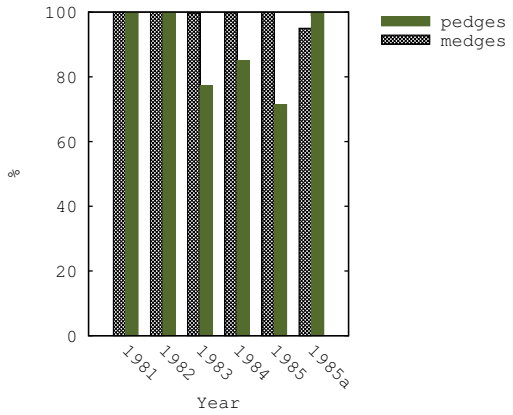


Fig. 12: Accuracy using MetaFeatures on skewed data. Despite the imbalance in class distribution (Figure 11), MetaFeatures perform well on the majority *medges* and comparably on *pedges*. *1985a*: with higher penalty for minority class misclassifications, shows improvement over no-penalty case (*1985*).

MetaFeatures performed comparably in most of the cases despite the unconstrained node selection, making it suitable for generic link prediction problems in social networks. Though we have predominantly aimed at comparing our results with the past work in this paper, it is important to note that even without any separate label acquisition period, our model can still verify link occurrence in the future in a pure dynamic setting. In that case, labels are acquired as the graph grows, giving us a training dataset for any of the future link predictions.

In our future work, in addition to verifying the performance of MetaFeatures with dynamic graphs from other domains, we plan to investigate the problem of rank-deficient TF matrices. There are two ways by which this problem could be addressed - one is to remove those features and complement them with new feature(s) that are robust against this problem (such a feature would have a non-zero value irrespective of the presence of authors in different CCs) or conduct experiments by segregating edges that connect and do not connect CCs.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 1318913 and 1318957. The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

References

[1] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, Dec. 2005.

[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proceedings of eighth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 2002.

[3] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, "Co-author relationship prediction in heterogeneous bibliographic networks," in *ASONAM*, 2011.

[4] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[5] R. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *KDD*, 2010.

[6] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*, ser. CIKM '03, 2003.

[7] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[8] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Aug. 2008, pp. 11–15.

[9] K. Yang and C. Shahabi, "A PCA-based similarity measure for multivariate time series," in *Proceedings of the 2nd ACM international workshop on Multimedia databases*, 2004.

[10] M. Ley. (1993) Dblp.uni-trier.de: Computer science bibliography.

[11] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001.

[12] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, 2008.

[14] A. K. Menon and C. Elkan, "Link prediction via matrix factorization," in *ECML/PKDD*, 2011.

[15] J. Kunegis and A. Lommatzsch, "Learning spectral graph transformations for link prediction," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 561–568.

[16] Z. Huang and D. K. J. Lin, "The time-series link prediction problem with applications in communication surveillance," *INFORMS Journal on Computing*, pp. 286–286, 2009.

[17] Y. Yang and N. V. Chawla, "Link prediction in heterogeneous networks: Influence and time matters," in *Proceedings of the 2012 IEEE International Conference on Data Mining*, 2012.

[18] L. Lu and T. Zhou, "Link prediction in complex networks: A survey," *CoRR*, vol. abs/1010.0725, 2010.