This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



# Discriminative Link Prediction using Local, Community and Global Signals

Abir De, Sourangshu Bhattacharya, Sourav Sarkar, Niloy Ganguly
{abir.de,sourangshu,sourav.sarkar,niloy}@cse.iitkgp.ernet.in
*IIT Kharagpur*

Soumen Chakrabarti
soumen@cse.iitb.ac.in
*IIT Bombay*

**Abstract**—Predicting plausible links that may emerge between pairs of nodes is an important task in social network analysis, with over a decade of active research. Here we propose a novel framework for link prediction. It integrates signals from node features, the existing local link neighborhood of a node pair, community-level link density, and global graph properties. Our framework uses a stacked two-level learning paradigm. At the lower level, the first two kinds of features are processed by a novel local learner. Its outputs are then integrated with the last two kinds of features by a conventional discriminative learner at the upper-level. We also propose a new stratified sampling scheme for evaluating link prediction algorithms in the face of an extremely large number of potential edges, out of which very few will ever materialize. It is not tied to a specific application of link prediction, but robust to a range of application requirements. We report on extensive experiments with seven benchmark datasets and over five competitive baseline systems. The system we present consistently shows at least 10% accuracy improvement over state-of-the-art, and over 30% improvement in some cases. We also demonstrate, through ablation, that our features are complementary in terms of the signals and accuracy benefits they provide.

**Index Terms**—Social Network, Link Prediction

## 1 Introduction

The link prediction (LP) problem [26] is to predict future relations that can form between nodes in a social network, given historical views of the network up to the present time. E.g., one may wish to predict that a user will like a restaurant or a book, or that two researchers will coauthor a paper, or that a user will endorse another on LinkedIn, or that two users will become "friends" on Facebook. Apart from the obvious recommendation motive, LP can be useful in social search, such as Facebook Graph Search[1], as well as ranking goods or services based on not only real friends' recommendations but also that of imputed social links. Driven by these strong motivations, LP has been intensively researched in recent years; Lu and Zhou [31] provide a comprehensive survey which also points to several scopes for improvement.

There are three major aspects of a LP task:

**Features** capturing complementary pieces of information from nodes, edges and other graph attributes.

**Model and algorithm** utilizing the features to score / rank the potential edges as future edges.

**Evaluation metric** which measures the utility of the technique towards a set of applications.

Here we make contributions in each of these three aspects. Before describing our contributions, we elaborate on the above.

### 1.1 Features

**Node features:** Given a potential edge between nodes $u$ and $v$ of a social network, the features we extract from nodes are usually guided by the problem domain. E.g., one may use the genre of movies in movie-movie networks, and title words or salient keywords in papers for citation networks.

Broadly, three sets of features can be derived from a network structure: (a) local features, (b) global features, and (c) community Features.

**Local features** are derived from linkage information in the immediate network vicinity of $u$ and $v$, such as the existence of many common neighbors. The well-known Adamic-Adar (AA) [1] predictor is based on the number of common neighbors between two nodes. Preferential attachment (PA) assigns more weight to potential links between high degree nodes [4].

**Global features** between two nodes are devised to capture non-local effects of links, such as effective conductance, hitting time, commute time [12], etc. These are often deeply related to the non-local community structure of the social network. In this work, we use the Katz score [21], a length-weighted count of the number of paths between two vertices, as a global feature. Other prominent features used in the literature are local and cumulative random walks [31] (LRW and CRW), which depend on the steady state visit probability of each node and its cumulative version over time (see Section 2.2). The random walk paradigm has also been combined [3] with edge features for enhanced accuracy, called supervised random walk (SRW). These approaches are described in Section 2 in greater detail and evaluated as baselines.

**Community features:** Communities or graph clusters are usually characterized by a larger density of links within a community and sparse links across communities. These show up as block structure in adjacency matrices, when rows and columns are suitably reordered, using methods such as co-clustering [11], cross-association [6], stochastic block models [17], etc. Co-clustering [11] is a popular technique, which exposes rich block structure in a dyadic

1. https://www.facebook.com/about/graphsearch

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



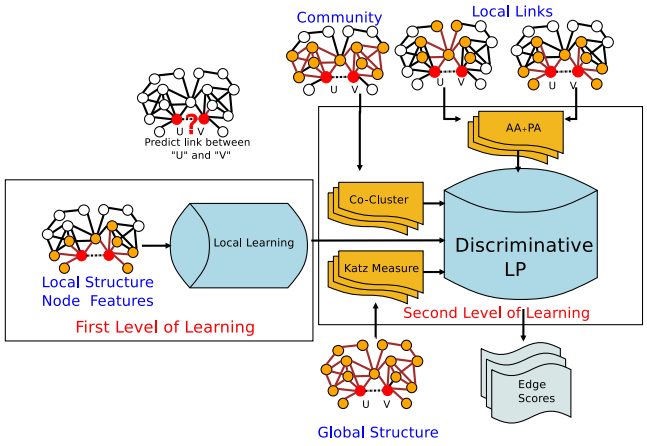


Figure 1: Overview of LCG: LCG is a two level link predictor. In this first level, it combines local structure and node features to obtain locality dependent feature weights and feature scores (LL). In the second level, LL is combined with community, local and global signals to learn the final discriminative link prediction model.

relation, expressed as a matrix. We will use co-clustering to derive community features.

### 1.2 Model outline

The main idea behind the technique proposed in this paper is to combine signals from all four feature types (node features, local neighborhood features, community level features, and global graph features) to learn a unified model. We combine the features in two levels. Figure 1 shows a high level block diagram of our LP system. At the lower level, we learn a local model for similarity of nodes across edges (and non-edges). At the higher level, a support vector machine combines the local signal with non-local signals suitably tuned into feature values: the output of co-clustering [11] capturing community level signal, and various global graph based signals, e.g. Katz score [21].

To the best of our knowledge, this is the first LP system which uses a two-level learning framework. Another salient feature of our system is the local dissimilarity model. We designed it after recognizing that node features have to be interpreted in the context of the local network to determine the similarity between a pair of nodes. For example, "romantic" movies may be closely tied to actors who work in them, while the director may be more important for "sci-fi" movies. The computation of node similarity is posed as a problem of learning weights of a locally constrained linear programming formulation at the lower level. We show that this local learning leads to improved performance on real life datasets. A third important feature is our use of community-level link density signals to further inform our LP algorithm.

### 1.3 Evaluation overview

We perform exhaustive experimentation with seven public datasets (NetFlix, MovieLens, CiteSeer, Cora, WebKb, Conflict and Protein-network). When comparing against the state of the art benchmark techniques (SRW, AA, CRW) using mean average precision (MAP), we see that the proposed method (LCG) and its precursor, CCLL, outperform the existing methods by a large margin. This, together with our detailed feature ablation studies, suggest that features at multiple granularities are responsible for the boost in accuracy. Moreover, in-depth exploration using feature ablation studies further confirm the complementarity of these features.

For several of these co-occurrence (say, movie-movie) networks we also have the data of the underlying bipartite (user-movie) network. We test the performance of the algorithms based upon the extent to which they can predict the links of a user in the bipartite network, which can be used in a recommendation application. Interestingly, the performance ranks of existing algorithms change under the new condition, however, LCG still performs best.

### 1.4 Contributions and roadmap

Section 2 describes related work. Section 3 defines the LP problem and describes the proposed learning framework. We first propose a local model from similarity between neighboring nodes. We demonstrate that such a local model captures the feature importance better than any global weight-based state-of-the-art counterpart. Second, we realize that community structure inherent in social networks can be exploited as a signal. Consequently, we turn co-clustering signals into suitable surprise values. Third, we combine these signals along with two local-link based signals and another global signal via a SVM to develop the improved LP algorithm. In Section 4 we describe various evaluation methodologies, including application-specific evaluation. In Section 5 we describe experimental results and present the evaluation of the proposed methods vis-a-vis existing methods. Through detailed analysis of each feature and extensive feature ablation study, we show that the four types of features: node, local, community and global are complementary and help in achieving significant improvement over state-of-the-art methodology. In section 6, we conclude with references. A preliminary version of parts of the work described here was reported in [10].

## 2 Related Work

LP has been studied in different guises for many years, and formalized, e.g., by Liben-Nowell and Kleinberg [26]. Lu and Zhou [31] have written a comprehensive survey. Lu *et al.* [30] attempts to address the fundamental question of predictability of links using structural consistency.

### 2.1 Local similarity

If each node $u$ is associated with a feature vector $\theta_u$, these can be used to define edge feature vectors $f(u,v) = f(\theta_u, \theta_v)$, which can then be involved in an edge prediction through logistic regression (i.e., $\Pr(\text{edge}|u,v) = \frac{1}{1+e^{-\nu \cdot f(u,v)}}$), or a SVM (predict an edge if $\nu \cdot f(u,v) > 0$). Obviously, this class of models misses existing neighborhood information.

To decide if nodes $u$ and $v$ may get linked, one strong signal is the number of common neighbors they already share. Adamic and Adar (AA) [1] refined this by a weighted

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



counting: common neighbors who have many other neighbors are dialed down in importance:

$$\mathrm{sim}^{\mathrm{AA}}_{i,j} = \sum_{k \in \Gamma(i) \cap \Gamma(j)} \frac{1}{\log d(k)}, \quad (1)$$

where $d(k)$ is the degree of common neighbor $k$.

The resource allocation (RA) predictor [46] is a slight variation which replaces $\log d(k)$ with $d(k)$ in (1). RA punishes the high-degree common neighbors more heavily than AA. Among ten local similarity measures, Lu and Zhou found [31, Table 1, page 9] RA to be the most competitive, and AA was close.

### 2.2 Random walks and conductance

AA, RA, etc. are specific examples of three general principles that determine large graph proximity between nodes $u$ and $v$: (a). Short path(s) from $u$ to $v$. (b). Many parallel paths from $u$ to $v$. (c). Few distractions (high degree nodes) on the path(s). Elegant formal definitions of proximity, that capture all of the above, can be defined by modeling the graph $(V, E)$ ($|V| = N, |E| = M$) as a resistive network and measuring effective conductance, or equivalently [12], modeling random walks [24], [36], [35] on the graph and measuring properties of the walk.

Many link predictors are based on such proximity estimates. The earliest, from 1953 [21] is Katz score (see section 3.6). Lu and Zhou [31] describe several other related variants. In their experiments, the best-performing definitions were local [36] and cumulative (called "superposed" by Lu and Zhou) random walks [28] (LRW and CRW), described next.

Suppose $q_u$ is the steady state visit probability of node $u$ (degree divided by twice the number of edges in case of undirected graphs). Let $\pi_u(0)$ be the impulse distribution at $u$, i.e., $\pi_u(0)[u] = 1$ and $\pi_u(0)[v] = 0$ for $v \neq u$. Define $\phi_u(t+1) = C^\top \pi_u(t)$, where $C$ is the $N \times N$ row-stochastic edge conductance (or transition probability) matrix. Then

$$\mathrm{sim}^{\mathrm{LRW}}_{uv}(t) = q_u \pi_u(t)[v] + q_v \pi_v(t)[u]. \quad (2)$$

For large $t$, the two rhs terms become equal, and LRW similarity is simply twice the "flow" of random surfers on the edge $\{u, v\}$. Lu and Zhou claimed [31, Table 3, page 16] that LRW is competitive, but the following cumulative random walk (CRW) is sometimes more accurate.

$$\mathrm{sim}^{\mathrm{CRW}}_{u,v}(t) = \sum_{\tau=1}^{t} \mathrm{sim}^{\mathrm{LRW}}_{u,v}(\tau). \quad (3)$$

Unlike Katz score,CRW does not converge with increasing $t$, so $t$ is chosen by validation against held-out data.

Many other variants of Katz score for measuring similarity between vertices have been proposed. Transferring similarity [34], is a decayed sum of weighted-distances between nodes. Here the distance is a correlation measure between the ratings of two nodes. Leicht-Holme-Newmann similarity [25] assigns high similarity between two nodes if their neighbors are similar. Yet another the measure is the matrix forest index [7] which uses laplacian matrix instead of adjacency matrix. Average commute time was proposed by Fouss *et al.*[14] as a measure of vertex similarity and shows a connection with graph laplacian which gives interpretability and helps easy computation of the similarity measure. All these measures yield similar performances as Katz measure [21], for link prediction.

Although some of these approaches may feel ad-hoc, they work well in practice; Sarkar *et al.* [33] have given theoretical justification as to why this may be the case.

Recently, Gong *et al.* [15] have proposed a framework called Social Attribute Network, where node attributes are considered, and extend the existing random walk based algorithms to this framework. However, they do not learn local functions of node attributes. Instead they use fixed rules to derive networks using node attributes.

### 2.3 Supervised random walk (SRW)

One of the first approaches to blend node features with graph structure is supervised and discriminative [3], and based on personalized PageRank [19]. Recall that $f(u, v)$ is an edge feature vector. The raw edge weight is defined as $a(u, v) = a(w \cdot f(u, v))$ for a suitable monotone function $a(\cdot) > 0$. The probability of a random surfer walking from $u$ to $v$ is set to

$$\Pr(u \to v) \stackrel{\text{def}}{=} C(u \to v) = C[v, u] = \frac{a(u, v)}{\sum_{v'} a(u, v')}.$$

where $C \in \mathbb{R}^{N \times N}$ is called the *edge conductance matrix*. If we are trying to predict out-neighbors of source node $s$, we set up a teleport vector $r_s$ defined as $r_s[u] = 1$ if $u = s$ and 0 otherwise, then find the personalized PageRank vector $\mathrm{PPV}_s$, defined by the recurrence

$$\mathrm{PPV}_s = \alpha C \ \mathrm{PPV}_s + (1 - \alpha) r_s.$$

During training, for source node $s$, we are given some actual neighbors $g$ and non-neighbors $b$, and we want to fit $w$ so that $\mathrm{PPV}_s[g] > \mathrm{PPV}_s[b]$.

SRW is elegant in how it fits together edge features with visible graph structure, but the latter is exploited in much the same way as Katz or LRW. Specifically, it does not receive as input regional graph community information. Thus, SRW and our proposal exploit different sources of information. Unifying SRW with our proposal is left for future work.

### 2.4 Matrix factorization

Another approach is to formulate the problem of link prediction as a regularized loss minimization problem [32], where the loss comprises of 3 parts: Latent node features ($u_i$), node features ($x_i$), and link features ($z_{ij}$), given in the first 3 terms in the final problem:

$$\min_{U,\Lambda,V,w,b} \frac{1}{\mathcal{O}} \quad \sum_{i=1, \substack{j \in \mathcal{O}^+ \\ k \in \mathcal{O}^-}}^{n} L(u_i^T \Lambda (u_j - u_k) + x_i^T V x_j + w^T z_{ij}) + (\text{bias+regularizer terms}) \quad (4)$$

Here $\mathcal{O}^-$ and $\mathcal{O}^+$ are sets of positive and negative edges ending with $i$. $\Lambda$ captures the asymmetry in the influence between $u_i$ and $u_j$. This problem is not convex, but can be solved using SGD. The main drawback of this model is its inability to utilize the node features $x_i$, since it tries to learn a global $V$.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



Also, recently, Ermis *et al.* [13] have formulated link prediction as a missing data completion problem from multiple heterogeneous sources, and came up with a generalized coupled tensor factorization approach. While they do not use a supervised learning framework, it will be interesting to explore use of signals from this approach in a learning framework as a future work.

### 2.5 Probabilistic generative models

**Stochastic block model (SBM)**: One of the two recent approaches that blend node features with linkage information is by Ho et al., although it is pitched not as a link predictor, but as an algorithm to cluster hyperlinked documents into a Wikipedia-like hierarchy. Documents directly correspond to social network nodes with local features. The algorithm seeks to cluster similar documents into the same or nearby topic nodes, and reward topic trees with dense linkages between documents belonging to small topic subtrees rather than span across far-away topic nodes. The model associates a parameter $\phi(t) \in (0,1)$ with each topic node $t$. If documents $u, v$ are attached to topic nodes $t(u), t(v)$, then the probability of a link between $u, v$ is estimated as $\phi(\text{LCA}(t(u), t(v)))$, where LCA is least common ancestor. These probabilities can then be used to rank proposed links. Guimera *et al.* [16] describe an interesting application of SBMs in detecting both missing and spurious links in experimentally generated biological networks.

**Community detection from Node attributes and Links (CESNA)**: The other recent approach, CESNA [43], gives a joint generative model for node attributes and community memberships which in turn is logistically combined to give edge probability. While this model elegantly combines graph structure with node attributes, we have observed that generatively modeling all node attributes using community signals results in unintended noise in the estimated parameters.

### 2.6 Co-clustering

Clustering one dimension of a dyadic relation (represented as a binary matrix $A$, say) is synergistic with clustering the other. For example, similar documents share similar terms, and vice versa. Dhillon *et al.* [11] proposed the first information-theoretic approach to group the rows and columns of $A$ into separate row and column *groups* (also called *blocks*, assuming rows and columns of $A$ have been suitably permuted to make groups occupy contiguous rows and columns) so as to best compress $A$ using one link density parameter per (row group, column group) pair. As we shall see, co-clustering and the group link densities can provide information of tremendous value to link prediction algorithms, of a form not available to AA, RA, LRW or CRW. However, the estimated block density is the result of a global optimization, and cannot directly predict one link. That requires combining the block density prior with local information (reviewed above). That is the subject of Section 3.

## 3 Proposed framework: LCG

We have reviewed in Section 2 several LP approaches. Some (AA, RA, CRW) involve no learning, others [17] propose generative probabilistic models that best "explain" the current graph snapshot (and then the model parameters of the "explanation" can be used to predict future links, although they did not study this application). In recent years, direct prediction of hidden variables through conditional probability [23] or discriminative [37] models have proved generally superior to modeling the joint distribution of observed and hidden variables [39]. As we shall see in Section 5, this is confirmed even among our comparisons of prior work, where supervised random walk [3] is superior to unsupervised approaches. However, before explaining our scheme, we clearly define the LP problem. For clarity we give the frequently used list of variables in Table 1 with their explanations.

| Variables | Explanation |
|---|---|
| $f(u,v)$ | Feature map vector for the node-pair $(u,v)$. |
| $\Delta_w(u,v)$ | Reference Triangulated dissimilarity (LL) between $(u,v)$. |
| $\delta_{uv}$ | $w^*_{uv} \cdot \lvert\theta_u - \theta_v\rvert$. Actual dissimilarity between $(u,v)$. |
| $Q$ | Set of query nodes. |
| $G(q)$ | Good nodes (neighbors) for *query* $q \in Q$. |
| $B(q)$ | Bad nodes (non-neighbors) for *query* $q \in Q$. |
| $\sigma$ | Fraction of neighbors (non-neighbors) sampled to train the discriminative model. |

TABLE 1: List of variables with their explanations.

### 3.1 Problem definition

We are given a snapshot of a social network, represented as a graph $(V, E)$ with $|V| = N$ and $|E| = M$. Let $u \in V$ be a node (say representing a person). Edges may represent "friendship", as in Facebook. Depending on the application or algorithm, the graph may be directed or undirected. The goal of LP is to recommend new friends to $u$, specifically, to rank all other nodes $v \in V \setminus u$ in order of likely friendship preference for $u$. One ranking device is to associate a *score* with each $v$, and sort them by decreasing score. LP algorithms vary in how they assign this score. We can also think about LP as associating a binary hidden variable with the potential edge $(u, v)$, then estimating the probability that this variable has value 1 (or true), based on observations about the currently revealed graph. The LP algorithm is considered high quality if the user accepts many proposed friends near the top of the ranked list.

### 3.2 Overview of two-level discriminative framework

LP can also be regarded as a classification problem: given a pair of nodes $u, v$, we have two class labels ("link" vs. "no link") and the task is to predict the correct label. To estimate a confidence in the (non) existence of a link, we aggregate several kinds of input signals to feed into a two-level discriminative model.

**In the first level** (section 3.3), we propose a local learning process to determine effective similarity between two given nodes. Unlike AA and other node-pair signals, our new approach incorporates the fact that propensity of linkage is not just a function of node similarity; it changes with neighborhood.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



**In the second level** (section 3.7), the local learning score is combined with other complementary signals (e.g. local-links, community and global scores) to learn the final discriminative link prediction model. In this upper level, we use AA, PA as local-link candidates and Katz measure (Eq. 12) to incorporate the global signals. Apart from these three signals, we harness the output of a co-clustering of the graph's adjacency matrix to derive yet more features. To the best of our knowledge, co-clustering has never been used in this manner for LP. Figure 1 gives an overview of the two-level learner.

For each proposed node pair $u, v$, these signals will be packed as features into a feature vector $f(u, v) \in \mathbb{R}^d$ for some suitable number of features $d$. We estimate a *global model* $\nu \in \mathbb{R}^d$, such that the score of existence of edge $(u, v)$ is directly related to $\nu \cdot f(u, v)$. $f(u, v)$ will consist of several blocks or sub-vectors, each with one or more elements.

- $f(u, v)$[LL] is the block derived from local similarity learning (Section 3.3).
- $f(u, v)$[LE] is the block derived from local edge scores and graph structure. We use the Adamic-Adar (AA) score (1), preferential-attachment (PA) score described in equation 11 below, or both.
- $f(u, v)$[CC] is the block derived from co-clustering (Section 3.5).
- $f(u, v)$[GG] is the block derived from global graph properties. We use the Katz measure defined in equation 12.

As we shall demonstrate in Section 5, these signals exploit different and complementary properties of the network. If $y(u, v) \in \{0, 1\}$ is the observed status of a training edge, we can find the weights ($\nu$) using a SVM and its possible variations. The details will be discussed in Section 3.7.

To exploit possible interaction between features, we can construct suitable kernels [5]. Given that we have very few features, a quadratic (polynomial) kernel can be implemented by explicitly including all quadratic terms. That is, we construct a new feature vector whose elements are

- $f(u, v)[i]$ for all indices $i$, and
- $f(u, v)[i]\, f(u, v)[j]$ (ordinary scalar product) for all index pairs $i, j$.

We can also choose an arbitrary informative subset of these. We will now describe the different blocks of features.

### 3.3 Learning local similarity

An *absolute* notion of similarity between $u$ and $v$, based on node features $\theta_u, \theta_v$, is not strongly predictive of linkage; it also depends on the typical *distribution* of similarity values in the neighborhood [9]. Also, the presence or absence of edge $(u, v)$ is rarely determined by nodes far from $u$ and $v$. Keeping these in mind, the first step of the algorithm learns the typical (dis)similarity between $u$ and $v$ and their common neighbors. We term this the *reference* dissimilarity. We then use this to predict the chance of link $(u, v)$ arriving.

Let $\Gamma(u)$ be the (immediate) neighbors of $u$. We will model the *edge dissimilarity* between $u$ and $v$ as

$$\Delta_w(u, v) = w_{uv} \cdot |\theta_u - \theta_v|, \tag{5}$$

where $\theta_u$ is a node feature vector associated with $u$, and $|\cdots|$ denotes the *element-wise* absolute value of a vector, e.g., $|(-2, 3)| = (2, 3)$, although other general combinations of $\theta_u$ and $\theta_v$ are also possible [3]. $w_{uv}$ is the weight vector fitted *locally* for $u, v$. (Contrast this with the global $\nu$ above, and the final proposal in SRW [3] that fits a single model over all node pairs.)

#### 3.3.1 Finding $w_{uv}$ and reference dissimilarity

Throughout this work, and consistent with much LP work, we assume that edges are associative or unipolar, i.e., there are no "dislike" or antagonistic links. Similar to AA and friends, when discussing node pair $u, v$, we restrict our discussion to the vicinity $N = \Gamma(u) \cup \Gamma(v)$.

For $A \subseteq V \setminus u$, $A \neq \varnothing$, we extend definition (5) to the *set dissimilarity*

$$\Delta_w(u, A) = \frac{1}{|A|} \sum_{v \in A} \Delta_w(u, v). \tag{6}$$

We define $\Delta_w(u, \varnothing) = 0$. $\Delta_w(u, A)$ is the average dissimilarity between $u$ and $A$. Note that $\Delta_w(u, \{v\})$ is simply $\Delta_w(u, v)$. Suppose we have to predict if the edge $(u, v)$ exists, given the training neighborhoods of $u$ and $v$, called $\Gamma(u)$ and $\Gamma(v)$; i.e., the subgraph we get to observe has nodes $\{u, v\} \cup \Gamma(u) \cup \Gamma(v)$ and already-materialized edges between them. Node $u$ has edges to nodes in $\Gamma(u) \cap \Gamma(v)$, but not to nodes in $\Gamma(v) \setminus \Gamma(u)$. We therefore assert the constraint that the average dissimilarity of $u$ to $\Gamma(u) \cap \Gamma(v)$ must be considerably smaller than that of $u$ to $\Gamma(v) \setminus \Gamma(u)$:

$$\Delta_w(u, \Gamma(v) \setminus \Gamma(u)) \geq \beta \Delta_w(u, \Gamma(u) \cap \Gamma(v))$$

and, by symmetry,

$$\Delta_w(v, \Gamma(u) \setminus \Gamma(v)) \geq \beta \Delta_w(v, \Gamma(u) \cap \Gamma(v))$$

Similarly, while edges from $u$ to $\Gamma(v) \setminus \Gamma(u)$ did not materialize, edges from $u$ to $\Gamma(u)$ did. This leads to the constraints:

$$\Delta_w(v, \Gamma(v) \setminus \Gamma(u)) \leq \alpha \Delta_w(v, \Gamma(u) \cap \Gamma(v))$$
$$\Delta_w(u, \Gamma(u) \setminus \Gamma(v)) \leq \alpha \Delta_w(u, \Gamma(u) \cap \Gamma(v))$$

The key idea here is that, if there is an edge $(u, v)$, we want to choose $w_{u,v}$ such that $\Delta_w(u, v)$ is low, *relative to* node pairs that are not neighbors. Conversely, if $(u, v)$ is not an edge, we want the dissimilarity to be large relative to nearby node pairs that are neighbors. We codify this through the following four constraints:

$$\begin{aligned}
\Delta_w(v, \Gamma(v) \setminus \Gamma(u)) &\leq \alpha \Delta_w(v, \Gamma(u) \cap \Gamma(v)) \\
\Delta_w(u, \Gamma(u) \setminus \Gamma(v)) &\leq \alpha \Delta_w(u, \Gamma(u) \cap \Gamma(v)) \\
\Delta_w(u, \Gamma(v) \setminus \Gamma(u)) &\geq \beta \Delta_w(u, \Gamma(u) \cap \Gamma(v)) \\
\Delta_w(v, \Gamma(u) \setminus \Gamma(v)) &\geq \beta \Delta_w(v, \Gamma(u) \cap \Gamma(v))
\end{aligned} \tag{7}$$

Figure 2 illustrates the constraints. Here $\alpha, \beta$ are suitable multiplicative margin parameters. Smaller (larger) value of $\alpha$ ($\beta$) allows a lower (higher) dissimilarity between the connected (disconnected) nodes. Here, we have experimentally selected $\alpha$ and $\beta$.

Subject to the above constraints (7), we wish to choose $w$ so as to minimize $\Delta_w(u, v)$. This is a standard linear program, which, given the typically modest size of $\Gamma(u) \cup \Gamma(v)$, runs quite fast.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



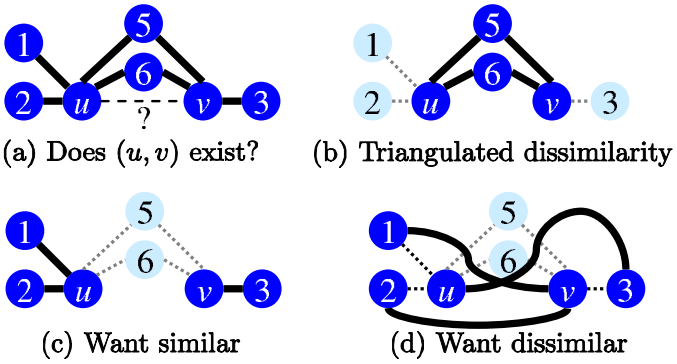


Figure 2: Local dissimilarity constraints. To find the score for pair $(u, v)$, $\Delta_w(u, \{5\} \cup \{6\}) + \Delta_w(v, \{5\} \cup \{6\})$ [panel (b)] is estimated under the imposition of the dissimilarity functions of existing edges: $\Delta_w(u, \{1\} \cup \{2\})$ and $\Delta_w(v, \{3\})$ are low [panel (c)] and those of non-existing edges $\Delta_w(v, \{1\} \cup \{2\})$ and $\Delta_w(u, \{3\})$ [panel (d)] are high.

#### 3.3.2 Computation of LL features

The linear program outputs $w^*_{uv}$, from which we can compute

$$\delta_{uv} = w^*_{uv} \cdot |\theta_u - \theta_v|. \tag{8}$$

But is $\delta_{uv}$ larger than "expected", or smaller? Plugging in the raw value of $\delta_{uv}$ into our outer classifier may make it difficult to learn a consistent model $\nu$ globally across the graph. Therefore, we also compute the *triangulated dissimilarity* between $u$ and $v$, using common neighbors $i$, as

$$\begin{aligned}\bar{\Delta}_{w^*}(u, v) &= \sum_{i \in \Gamma(u) \cap \Gamma(v)} \frac{\Delta_{w^*}(i, u) + \Delta_{w^*}(i, v)}{|\Gamma(u) \cap \Gamma(v)|} \\ &= \Delta_{w^*}(u, \Gamma(u) \cap \Gamma(v)) + \Delta_{w^*}(v, \Gamma(u) \cap \Gamma(v)).\end{aligned} \tag{9}$$

Finally, we return

$$f(u, v)[LL] = \bar{\Delta}_{w^*}(u, v) - \delta_{uv}. \tag{10}$$

If $f(u, v)[LL]$ is large and positive, it expresses confidence that link $(u, v)$ will appear; if it is large and negative, it expresses confidence that it will not. Around zero, the LL feature is non-committal; other features may then come to the rescue.

### 3.4 Local link scores

As mentioned, we use the Adamic-Adar (AA) (1) and preferential-attachment (PA) scores as local link scores. The preferential attachment score is motivated from the Barabasi-Albert preferential attachment model [4], which states that the probability of connection to a node $k$ is proportional to its degree $d(k)$. The score for an edge $(u, v)$ between nodes $u$ and $v$, is given by the product of degrees:

$$\text{sim}^{\text{PA}}(u, v) = d(u)d(v), \tag{11}$$

### 3.5 Co-clustering and "surprise"

Given a dyadic relation represented as a matrix, co-clustering [11] partitions rows and columns into row groups and column groups. We can think of the rows and columns of the input matrix being reordered so that groups are contiguous. The intersection of a row group and column group is called a *block*. The goal of co-clustering is to choose groups so that the blocks are *homogeneous*, i.e., edges within a block appear uniformly dense, with no finer structure (stripes) based on rows or columns. Co-clustering uses a data compression formalism to determine the optimal grouping.

Consider a query node pair $u, v$ where we are trying to predict whether edge $(u, v)$ exists. E.g., $u$ may be a person, $v$ may be a movie, and we want to predict if $u$ will enjoy watching $v$. In this person-likes-movie matrix, as a specific example, there may be row groups representing clusters of people that like most movies, and there may be column groups representing clusters of classic movies that most people like. In general, the block in which the matrix element $[u, v]$ is embedded, and in particular, its edge density $\rho(u, v)$, gives a strong prior belief about the existence (or otherwise) of edge $(u, v)$, and could be the feature $f(u, v)[CC]$ in and of itself.

Although block density $\rho(u, v) \in [0, 1]$, the penalty for deviating from it in the ultimate link decision is not symmetric (thanks again to graph sparsity). So a better formalism to capture a co-clustering based feature is the "surprise value" of an edge decision for node pair $u, v$. As an extreme case, if a non-edge (value 0) is present in a co-cluster block where all remaining elements are 1 (edges), it causes large surprise. The same is the case in the opposite direction.

There are various ways of expressing this quantitatively. One way of expressing it is that if an edge $(u, v)$ is claimed to exist, and belongs to a block with an edge density $\rho(u, v)$, the surprise is inversely related to $\rho(u, v)$; in information theoretic terms, the surprise is $-\log \rho(u, v)$ bits. (So if $\rho(u, v) \to 0$, yet the edge exists, the surprise goes to $+\infty$.) Similarly, if the edge does not exist, the surprise is $-\log(1 - \rho(u, v))$ bits.

### 3.6 Global graph property

Given a graph $G = (V, E)$ and two vertices $u, v \in V$, one of the common ways of measuring similarity between $u$ and $v$ is using global similarity indices [31]. For example, one can define the similarity to be the average number of steps taken by a random walker starting from $u$ to reach $v$ (average commute time), or the steady state probability of a random walk with restarts from vertex $u$, reaches vertex $v$, etc. In this paper we use the oldest global measure of vertex similarity called Katz score [21], which was also one of the best performers in the comparison [31].

Let $A$ be the adjacency matrix of graph $G = (V, E)$, and $u, v$ be two vertices. The Katz similarity score between $u$ and $v$ is given by:

$$\text{sim}_{\text{Katz}}(u, v) = \beta A_{uv} + \beta^2 (A^2)_{uv} + \cdots = (\mathbb{I} - \beta A)^{-1} - \mathbb{I}, \tag{12}$$

where $0 \leq \beta < 1/\lambda_1$, is a user provided constant, and $\lambda_1$ is the largest eigenvalue of $A$. This score is effectively a decayed count of the number of paths between $u$ and $v$, where the decay factor $\beta^k$ depends on the length of the path $k$. $\text{sim}_{\text{Katz}}(u, v)$ is used as a feature for the potential edge $(u, v)$ in the global model described below.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



### 3.7 Discriminative learner for global model

In order to obtain the best LP accuracy, the above signals need to be combined suitably. For each edge, there are two classes (present/absent). One possibility is to label these $+1, -1$, and fit the predictor $\hat{y}_{uv} = \text{sign}(\nu \cdot f(u, v))$.

#### 3.7.1 Loss function

$\nu$ can be learned to minimize various loss functions. The simplest is $0/1$ edge misclassification. However, as we have discussed in Section 4.1, for ranking losses, it is better to optimize the AUC, which is closely related [20] to the pairwise loss. Joachims [20] offers to directly optimize for several ranking objectives; we choose area under the ROC curve (AUC), although we evaluate the resulting predictions using MAP. During inference, given $q$, we find $\nu \cdot f(q, v)$ for all $v \neq q$ and sort them by decreasing score. Moreover, by encoding the loss as MAP, we obtain qualitatively similar performance.

#### 3.7.2 Feature map

We now finish up the design of $f(u, v)[LE]$, $f(u, v)[LL]$, $f(u, v)[CC]$ and $f(u, v)[GG]$. $f(u, v)[LE]$ is a vector of two components: $f(u, v)[AA]$ given by the single scalar (1) and $f(u, v)[PA]$ given by the scalar (11). $f(u, v)[LL]$ and $f(u, v)[GG]$ are single scalars as defined in (10) and (12), respectively. $f(u, v)[CC]$ has two scalar elements, one for each surprise value:

- $-\log \rho(u, v)$ for the "link exists" case, and
- $-\log(1 - \rho(u, v))$ for the "edge does not exist" case.

Accordingly, $\nu$ will have two model weights for the CC block, and these will be used to balance the surprise values from training data. The soundness of the above scheme follows from structured learning feature map conventions [20], [37]. Thus, $f(u, v)$ has a total of *six elements*.

## 4 Evaluation protocol

As described informally in Section 3.1, a LP algorithm applied to a graph snapshot is successful to the extent that *in future*, users accept high-ranking proposed links. In practice, this abstract view quickly gets murky, especially for graphs without edge creation timestamps. In this section we discuss the important issues guiding our evaluation protocol and measurements.

### 4.1 Labeling vs. ranking accuracy

Regarding the LP algorithm's output as a binary prediction (edge present/absent) for each node pair, comparing with the true state of the edge, and counting up the fraction of correct decisions, is a bad idea. This is because most potential edges are absent (label skew). The situation is similar to ranking in information retrieval (IR) [27], where, for each query, there are many fewer relevant documents than irrelevant ones. In LP, a separate ranking is produced for each node $q$ from a set of nodes $Q$, which are therefore called *query nodes*.

Fix a $q$ and consider the ranking of the other $N - 1$ nodes. Some of these are indeed neighbors (or will end up becoming neighbors). Henceforth, we will call $q$'s neighbors as *good* nodes $G(q)$ and non-neighbors as *bad* nodes $B(q)$. Ideally, each good node should rank ahead of all bad nodes. Because the LP algorithm is generally imperfect, there will be exceptions. The area under the ROC curve (AUC) is widely used in data mining as a accuracy measure somewhat immune to class imbalance. It is closely related to the fraction of the $|G(q)||B(q)|$ good-bad pairs that are in the correct order in LP's ranking. However, for the same reasons as in IR ranking [27], AUC tends to be large and undiscerning for almost any reasonable LP algorithm. Therefore, we adapt mean average precision (MAP), a standard ranking performance measure.

At each node, given a score from the LP algorithm on all other nodes as potential neighbors, and the "secret" knowledge of who is or isn't a neighbor, we compute the following performance metrics.

#### 4.1.1 Precision and recall

These are defined as

$$Precision(k) = \frac{1}{|Q|} \sum_{q \in Q} P_q(k) \tag{13}$$

$$\text{and} \quad Recall(k) = \frac{1}{|Q|} \Sigma_{q \in Q} R_q(k), \tag{14}$$

where $|Q|$ is the number of queries, $P_q(k)$ is Precision@$k$ for query $q$, and $R_q(k)$ is Recall@$k$ for query $q$. So $Precision(k)$ is the average of all Precision@$k$ values over the set of queries, and likewise with $Recall(k)$.

#### 4.1.2 Mean average precision (MAP)

First we define at query node $q$ the quantity

$$AvP(q) = \frac{1}{L} \sum_{k=1}^{N-1} P_q(k)\ r_q(k) \tag{15}$$

at each node, where $N-1$ is the number of nodes excluding the query node itself, $L$ is the number of retrieved relevant items and $r_i(k)$ is an indicator taking value 1 if the item at rank $k$ is a relevant item (actual neighbor) or zero otherwise (non-neighbor). Averaging further over query nodes, we obtain $MAP = \frac{1}{|Q|} \sum_q AvP(q)$.

In the remainder of this section, we explore these important issues:

- How should $Q$ be sampled from $V$? A related question is, how to present precision, recall, MAP, etc., with $Q$ suitably disaggregated to understand how different LP algorithms behave on different nodes $q \in Q$? (See Section 4.2.)
- Many graphs do not have link timestamps. For these, once $Q$ is set, how should we sample edges incident on $Q$ for training and testing? (See Section 4.3.)

### 4.2 Query node sampling protocol

In principle, the ranking prowess of an LP algorithm should be evaluated at *every* node. E.g., Facebook may recommend friends to all users, and there is a potential satisfaction and payoff from every user. In practice, such exhaustive evaluation is intractable. Therefore, nodes are sampled; usually $|Q| \ll N$. On what basis should query nodes be

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



sampled? In the absence of the social network counterpart to a commercial search engine's query log, there is no single or simple answer to this question. LP algorithms often target predicting links that close triangles if/when they appear. 92% of all edges created on Facebook Iceland close a path of length two, i.e., a triangle [3]. These nodes are sampled as query nodes $Q$.

Besides providing comparison of overall performance averaged over query nodes, in order to gain insight into the dynamics of different LP algorithms, we need to probe deeper into the structure of the network and check the strength/weakness of the algorithm vis-a-vis various structures. In our work, we bucket the selected query nodes based on

- the number of neighbors,
- the number of triangles on which they are incident.

### 4.3 Edge sampling protocol

If edges in the input graph have creation timestamps, we can present a snapshot to the LP algorithm and simulate further passage of time to accrue its rewards. Even this happier situation raises many troublesome questions, such as when the snapshot is taken, the horizon for collecting rewards, etc., apart from (the composition of) the query node sample.

To complicate matters further, many popular data sets (some used in Section 5) do not have edge timestamps. One extreme way to dodge this problem is the leave-one-out protocol: remove exactly one edge at a time, train the LP algorithm, and make it score that edge. But this is prohibitively expensive.

Rather than directly sample edges, we first sample query nodes $Q$ as mentioned in Section 4.2. This narrows our attention to $|Q| \cdot (N-1)$ potential edge slots incident on query nodes. Fix query $q$. In the fully-disclosed graph, $V \setminus q$ is partitioned into "good" neighbors $G(q)$ and "bad" non-neighbors $B(q)$. We set a train sampling fraction $\sigma \in (0,1)$. We sample $\lceil \sigma|G(q)| \rceil$ good and $\lceil \sigma|B(q)| \rceil$ bad nodes and present the resulting *training* graph to the LP algorithm. ($\sigma$ is typically 0.8 to 0.9, to avoid modifying the density and connectivity of the graph drastically and misleading LP algorithms.)

The good and bad training samples are now used to build our models as described in Section 3. The training graph, with the testing good neighbors removed, is used for co-clustering. This prevents information leakage from the training set. The remaining good neighbors and bad non-neighbors are used for testing. In case $|G(q)| = \lceil \sigma|G(q)| \rceil$ or $|B(q)| = \lceil \sigma|B(q)| \rceil$, we discard $q$, introducing a small bias after our sampling of $Q$. Effectively this is a "sufficient degree" bias, which is also found in prior art [3, Section 4: $K, \Delta$].

## 5 Experiments

We compare LCG against several strong baselines such as Adamic-Adar (AA) [1], Resource Allocation (RA) [31], Cumulative Random Walk (CRW) [31], Supervised Random Walk (SRW)[3], Matrix Factorization approach by Menon and Elkan (ME) [32], CESNA [43] and Stochastic Block Model (SBM) [17]. RA computes the score in a similar manner like AA, and therefore gives almost same performance. Therefore, we omit RA and only present those for AA. Apart from using LL as features to LCG and CCLL we run CCLL [10] and LL [9] independently as baselines.

| Dataset | N | E | $n(a)$ | $d_{avg}$ |
|---|---|---|---|---|
| NetFlix | 17770 | 20466 | 64 | 2.3034 |
| Movielens | 3952 | 5669 | 18 | 2.8689 |
| CiteSeer | 3312 | 4732 | 3703 | 2.7391 |
| Cora | 2708 | 5429 | 1433 | 3.89 |
| WebKb | 877 | 1608 | 1703 | 2.45 |
| Conflict | 230 | 320 | 3 | 2.5 |
| Protein | 2617 | 23710 | 76 | 9.1 |

TABLE 2: Summary of the datasets, where $N$ is the number of items, $E$ is the total number of links, $n(a)$ is the number of features and $d_{avg}$ is the average degree.

### 5.1 Dataset description

We used the following popular public data sets, also summarized in Table 2.

• **Netflix [18]:** The Netflix dataset consists of 2649429 users and 17770 movies. Each user has rated at least one movie. Each movie has 64 features obtained by SVD from many factors. From the raw data we constructed a "social network" of movies where two movies have an edge if they have at least a certain number of common viewers. By choosing the minimum number of common viewers to be 20, we obtain a network with 17770 nodes and 20466 edges.

• **Movielens [42]:** It has 6040 users and 3952 movies. Each user has rated at least one movie. Each movie has features which are a subset of a set of 18 nominal attributes (e.g. animation, drama etc.). From the raw data we constructed a "social network" between movies where two movies have an edge if they have at least a certain number of common viewers. By choosing the minimum number of common viewers to be 100, we obtain a network with 3952 nodes and 5669 edges.

• **CiteSeer [41]:** The CiteSeer dataset consists of 3312 scientific publications and the citation network consists of 4732 links. Each publication is tagged with a set of keywords. Total number of keywords is 3703.

• **Cora [41]:** The Cora dataset consists of 2708 scientific publications and the citation network consists of 5429 links. Here the total number of keywords is 1433.

• **WebKb [41]:** The WebKb dataset consists of 877 scientific publications and the citation network consists of 1608 links. Here the total number of keywords is 1703.

• **Conflict [32]:** This is a network of military disputes between countries, denoted Conflict. This graph has an edge between two countries if they have a conflict. Each node has 3 features, comprising of the country's population, GDP and polity.

• **Protein [38], [32]:** This network is a protein-protein interaction graph with 76 dimensional feature vector.

### 5.2 Performance comparison

Table 3 gives a comparative analysis of MAP (Mean Average Precision) values for all datasets and algorithms, and

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



| Dataset | LCG | CCLL | LL | ME | AA | CRW | SBM | SRW | CESNA |
|---|---|---|---|---|---|---|---|---|---|
| Netflix | **0.6776** (24.86%) | 0.6017 (10.85%) | 0.4750 (-12.49%) | 0.3431 | 0.5381 | *0.5428* | 0.1268 | 0.4564 | 0.4962 |
| Movielens | **0.8985** (10.73%) | 0.8747 (7.80%) | 0.8133 (0.23%) | 0.3681 | 0.5341 | 0.5784 | 0.20 | *0.8114* | 0.5020 |
| CiteSeer | **0.8866** (33.34%) | 0.7719 (16.09%) | 0.7393 (11.19%) | 0.2138 | *0.6649* | 0.5309 | 0.1452 | 0.6281 | 0.6445 |
| Cora | **0.8046** (28.24%) | 0.7234 (15.30%) | 0.6805 (8.46%) | 0.2022 | 0.6135 | 0.4726 | 0.0583 | *0.6274* | 0.4772 |
| WebKb | **0.9262** (38.38%) | 0.8583 (28.23%) | 0.7505 (12.13%) | 0.5468 | 0.6035 | 0.5736 | 0.3360 | *0.6693* | 0.3851 |
| Conflict | **0.8296** (13.02%) | 0.7858 (7.06%) | 0.7652 (4.25%) | 0.3345 | 0.6149 | 0.7093 | 0.1123 | *0.7340* | 0.3976 |
| Protein | **0.8968** (15.14%) | 0.8740 (12.21%) | 0.8213 (5.44%) | 0.1231 | 0.6424 | 0.7689 | 0.1954 | *0.7789* | 0.5129 |

TABLE 3: Mean average precision of proposed and baseline algorithms on all datasets when 90% of the links are used for training. Numbers in brackets denote percentage improvement over nearest baseline (shown in italics).

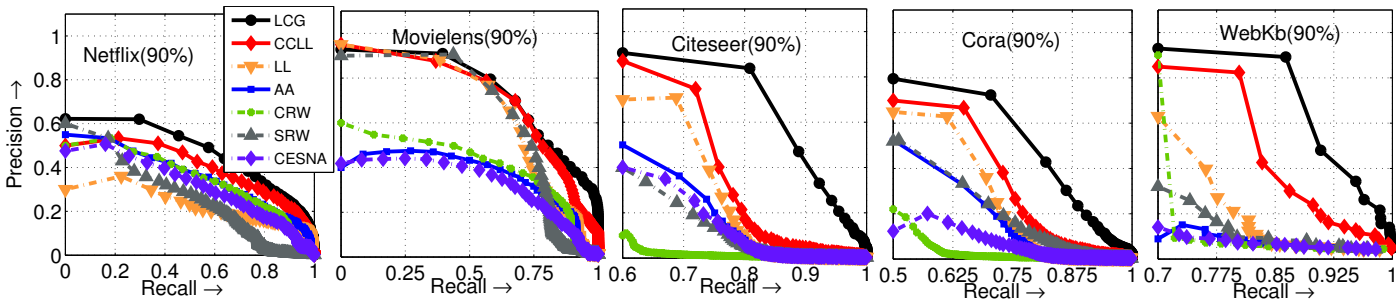


Figure 3: Precision vs. recall curves for five data sets and algorithms when 90% of the links are used for training.

Figure 3 gives a more detailed view in terms of precision vs. recall. We observe that, for all datasets, the overall performance of LCG is substantially better than all other methods compared here, including its old variant CCLL[10]. (For lack of space Figure 3 shows average precision-recall variation for five datasets, we observe qualitatively similar trends for two more datasets.)

Table 3 shows that performance of the stochastic block model (SBM) is particularly poor. This was surprising to us, given SBMs seem ideally suited for use in LP. Closer scrutiny showed model sparsity as a likely culprit, at least in case of Ho *et al.*'s formulation. They derive a tree-structured hierarchical clustering of the social network nodes, where the number of hierarchy nodes is much smaller than $N$, the number of social network nodes. Their model assigns a score to an edge $(u, v)$ that depends on the hierarchy paths to which $u$ and $v$ are assigned. Since the number of hierarchy nodes is usually much smaller than the number of social nodes, the scores of neighbors of any node have a lot of ties, which reduces ranking resolution. Therefore, MAP suffers. In contrast, the coarse information from co-clustering (CC) is only a feature into our top-level ranker.

Both AA and CRW show comparable performances. These methods only depend on link characteristics. AA depends on the number of triangles a node is part of, hence missing out the important node or edge feature information. Regarding CRW, as $t$ goes to $\infty$, it doesn't converge, and there is no consistent global $t$ for best MAP. SRW, which performs best among the baselines, uses node and link features (PageRank) but not community based (co-clustering) signal. Moreover, SRW learns only one global weight vector, unlike $w_{uv}$ in LL, a signal readily picked up by CCLL and LCG. We also found the inherent non-convexity of SRW to produce suboptimal optimizations. LL performs much better than SRW in all datasets except the movie recommendation datasets. The poor performance of LL on movie datasets can be attributed to lack of global signal.

The performance of ME is surprisingly poorer than AA. Although $u_i, x_i$ in Equation 4 are fitted for each node $i$, the important coupling model parameters $\Lambda, w, V$ are globally shared, which leads to underfitting. A serious drawback of ME is that, unlike SRW, it does not consider any structural property (neither local or global). As a result, the performance of ME is worse than AA and SRW, since SRW atleast captures strong global signals (although assign feature-weights globally) and AA is a reasonably good local signal. Moreover, even as the regularizer corresponding to $V$ (Eq: 4) is increased to a very high value, we did not observe a substantial decrease of predictive accuracy, which shows that $V$ as a single global parameter cannot make as much profitable use of $x_i$ as our local models can.

CESNA performs much better than ME, but still fails to beat AA in most cases. The broad idea of this predictor is that if nodes $u, v$ belong to many shared (latent) communities, the chance of existence of edge $(u, v)$ is higher. Moreover it captures the interaction between network structures and node attributes. Our scrutiny suggests that generatively modeling node attributes from latent communities is the likely culprit behind its poor performance. For example, in citation networks, phrase "large datasets" may be present in any paper on databases or data mining or Web search, which have diverse citation community membership. In our experience, modeling node attributes requires signals over and above community memberships in many networks.

We observe that global signals like Katz-measure significantly boost the performance particularly in citation graph (almost to 15% in Citeseer). In citation network, it is common that a paper from one domain cites a paper from other domain even though they are different in content (features), area/topic (community) and even in references (local-links). Such links may be explained by a global signal which measures the probability of reaching the destination node from the source node. In case of movie-movie networks, global signal often captures diversity in movies. For example, two highly famous movies that are entirely different

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



in all aspects can be watched by the same set of users. Global signals like Katz measure often capture presence of these links. We note that Netflix is a bigger dataset than Movielens; so the variability and diversity in its movies is quite high. Therefore LCG gives relatively higher boost in performance in Netflix (0.6017→ 0.6776) than in Movielens (0.8747→0.8985).

A possible explanation of the superior performance of LCG can be that the underlying predictive modules (LL, AA, CC, KM) perform well in complementary zones which LCG can aggregate effectively. In order to probe into this aspect we make a detailed study of the performance of the various algorithms with respect to various distribution of work load (elaborated in Section 5.5).

| Dataset | TS (%) | LCG | CCLL | CRW | SRW | CESNA |
|---|---|---|---|---|---|---|
| Netflix | 80 | 0.5855 | 0.5263 | 0.4493 | 0.3849 | 0.4308 |
| | 90 | 0.6776 | 0.6017 | 0.5428 | 0.5567 | 0.4962 |
| Movielens | 80 | 0.8827 | 0.8672 | 0.5018 | 0.7789 | 0.4641 |
| | 90 | 0.8925 | 0.8740 | 0.5784 | 0.8114 | 0.5020 |
| Citeseer | 80 | 0.8197 | 0.7000 | 0.3295 | 0.5567 | 0.5629 |
| | 90 | 0.8846 | 0.7719 | 0.5309 | 0.6281 | 0.6445 |
| Cora | 80 | 0.7453 | 0.6803 | 0.3400 | 0.5516 | 0.3881 |
| | 90 | 0.8046 | 0.7203 | 0.4726 | 0.6274 | 0.4772 |
| WebKb | 80 | 0.8726 | 0.8309 | 0.4924 | 0.3360 | 0.2887 |
| | 90 | 0.9249 | 0.8583 | 0.5468 | 0.5736 | 0.3851 |
| Conflict | 80 | 0.7908 | 0.7399 | 0.6578 | 0.7108 | 0.3779 |
| | 90 | 0.8296 | 0.7858 | 0.7093 | 0.7340 | 0.3976 |
| Protein | 80 | 0.8910 | 0.8521 | 0.7674 | 0.7295 | 0.4682 |
| | 90 | 0.8968 | 0.8740 | 0.7689 | 0.7789 | 0.5129 |

TABLE 4: Variation of Mean Average Precision with different training set sizes.

### 5.3 Stability to sampling

Among all the discussed methods, LCG, CCLL ME, CESNA and SRW use machine learning techniques. Hence in order to train the model, we randomly select a certain fraction of edges (say $TS\%$) and the same fraction of non-edges from the network. CRW is an unsupervised algorithm but since it performs a global random walk, it is affected by the sampling. The performance deteriorates when many edges are removed. Table 4 show the variation of performance with training sets of different sizes. We omit ME due to its relatively poor performance.

We conducted the experiment with 80% and 90% training samples. When we decrease the sample size from 90% to 80%, the performance deteriorates for all methods. But the deterioration is much smaller in CCLL, LCG compared to CRW, SRW and CESNA. This is because CCLL, LCG pick up strong signals from the AA and LL features. AA and LL work solely on local factors. Deletion of an additional 10% of edges hardly affects AA or LL score.

When we change training set size from 90% to 80%, the performance of LCG deteriorates a bit more than CCLL (with the exception of Citeseer). This is because LCG picks up a global signal; an additional 10% removal of edges affects this global (KM) score, which results in significant performance degradation.

### 5.4 Importance of various features

To understand the role of different features (local link structure, non-local effects), we build our SVM model with various sets of feature and investigate their importance in various datasets. We report the percentage change of these metrics upon adding different sorts (non-local/local) of features. More formally we define

$$\Delta_F(S) = \frac{\text{MAP}(S \cup F) - \text{MAP}(S)}{\text{MAP}(S)}$$

Here $\text{MAP}(X)$ is simply the value of the metric for the feature set $X$.

Tables 5 to 7 dissect the various features involved in LCG. For lack of space we present results for first five datasets. We observed similar trend for other datasets. While the model using all five features (LL, CC, KM, PA, AA) perform the best, removing PA from the set does not affect the MAP score much. An explanation is provided below.

*Non Local Effect (KM):* As already mentioned, the improvement of LCG over CCLL is due to addition of the non-local effect- the feature ablation study reiterates that. From Table 5, we observe that non-local effect provides substantial performance boost for all datasets. In case of Citeseer and Cora, the MAP values increase by more than 10%. It is due to the fact that in a citation network, a paper often refers to another paper that is at more than 1-hop distance (i.e. paper which is cited by related papers). So in case of citation networks, it is often natural that a paper cites another paper of different community. So the link emergence heavily depends on the non-local/distant dynamics in the graph. So Katz measure plays a major role in Link Prediction in citation graphs.

*Effect of Co-Clustering (CC):* Table 6 reflects that CC is a very strong signal for movie datasets. In these datasets, links between popular movies cannot be explained with LL or local link structures. Many users watch movies of thoroughly different dimensions. On the other hand, many users are more picky i.e. they only watch movies liked by most of the people. Although LL, global or link structures are unable to explain these connections, co-clustering picks up strong signals from such instances and turn into suitable scores. For Netflix, such signals are very dominant as the no. of diversified movies is quite high. Therefore LL fares poorly in Netflix but CC boosts its performance more significantly than in other datasets.

*Effect of Preferential Attachment (PA):* The networks in all the datasets follow power law. Thus it was reasonably intuitive that PA would be a key factor for link emergence. However it is surprising from Table 7 that the performance of PA is very poor. A careful investigation reveals that the sampling protocol is a likely culprit behind this. As we are performing uniform sampling, it is highly probable that the distribution of PA scores in the training and test sets become different. That is, given a query node $q$, the test set contains low degree neighbors while most of the high degree neighbors are present in the training set. Thus the SVM model favors PA to high degree neighbors- as a result it fares poorly in evaluation.

### 5.5 Workload distribution

In this section, we bucket the query nodes according to various categories and present some comparative analysis

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



| | 90% Sampling | | | | | 80% Sampling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Netflix | Movielens | Citeseer | Cora | WebKb | Netflix | Movielens | Citeseer | Cora | WebKb |
| $\Delta_{KM}$(LL,CC) | 7.1376 | 5.1043 | 15.2143 | 11.4442 | 7.8394 | 2.6494 | 1.6843 | 26.3476 | 12.9539 | 2.6592 |
| $\Delta_{KM}$(LL,CC,AA) | 9.6440 | 2.1836 | 10.8804 | 10.6436 | 5.7512 | 5.8332 | 1.6951 | 16.7000 | 9.0842 | 4.4891 |
| $\Delta_{KM}$(LL,CC,PA,AA) | 9.6440 | 2.1836 | 10.8804 | 10.6436 | 5.7512 | 5.8196 | 1.0764 | 17.7899 | 9.4420 | 1.6306 |

TABLE 5: Effect of Non-Local signal (Katz Measure) in %.

| | 90% Sampling | | | | | 80% Sampling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Netflix | Movielens | Citeseer | Cora | WebKb | Netflix | Movielens | Citeseer | Cora | WebKb |
| $\Delta_{CC}$(LL,PA) | 21.5095 | 8.5959 | 4.3955 | 2.9195 | 5.4229 | 32.4761 | 6.7105 | -2.3594 | 2.0064 | 2.7536 |
| $\Delta_{CC}$(LL,PA,KM) | 23.0013 | 11.1070 | 2.8043 | 1.6041 | 7.7321 | 46.0026 | 9.4305 | 3.6938 | -1.2640 | 3.3448 |
| $\Delta_{CC}$(LL,PA,KM,AA) | 12.5021 | 5.2724 | 2.0136 | 0.7387 | 7.5465 | 11.5026 | 4.6225 | 3.1199 | 0.6210 | 2.0108 |

TABLE 6: Effect of Co-Clustering (CC) in %.

| | 90% Sampling | | | | | 80% Sampling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Netflix | Movielens | Citeseer | Cora | WebKb | Netflix | Movielens | Citeseer | Cora | WebKb |
| $\Delta_{PA}$(LL,CC) | 3.6762 | -1.3135 | -4.8770 | -1.5916 | -2.3583 | 3.7422 | 3.2064 | -3.1653 | -0.2521 | 3.7426 |
| $\Delta_{PA}$(LL,CC,KM) | 0.2523 | 0.4865 | 0.4413 | 0.3512 | 2.1507 | 0.1363 | 0.4150 | -0.1131 | 1.1061 | 0.3722 |
| $\Delta_{PA}$(LL,CC,KM,AA) | 5.1167 | 0.0907 | 0.3428 | 0.4312 | 0.5068 | 5.3155 | -0.0445 | -0.2700 | 0.3242 | 0.8285 |

TABLE 7: Effect of Preferential Attachment (PA) in %.

between LCG, CCLL and four other best benchmark algorithms on two representative datasets: Netflix from movie networks, and WebKb from citation networks (Figures 4 and 5). Query nodes are bucketed based on:

- the number of neighbors they have (changes from sparse to dense), and
- the number of triangles formed.

Netflix is divided into six buckets of roughly equal number of nodes, while WebKb is divided into four buckets due to its smaller size.

#### 5.5.1 Workload distribution in movie networks

The workload distribution highlights the nature of each algorithm. The behavior of the algorithms is similar for the two workload distributions. It is seen that AA and CRW, which solely depend on link structure, improve as the graph becomes more dense (number of neighbors increases) or becomes more social (number of triangles increases). The two feature-based algorithms, LL and SRW, perform well in the sparse zone, and while the improvement in the dense (more social) zone is observed, it is not as significant as the two link-based algorithms. Clearly, in these two zones (sparse and dense), two different classes of algorithms work well.

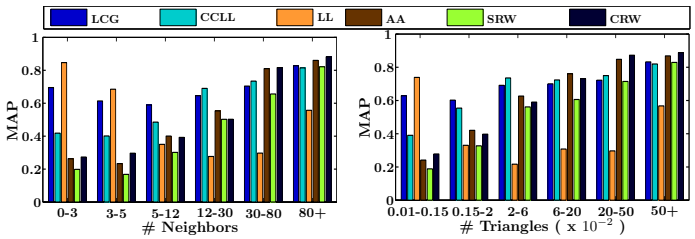


Figure 4: Workload distribution for **Netflix** dataset.

LCG and CCLL perform well in all the zones by appropriately learning signals in each zone. From Figure 4 we observe that LCG and CCLL perform quite well (substantially better than LL and AA), at intermediate density. There are two reasons behind it. Even though the graph is sparse, in these regions, number of edges and non-edges are close, which helps both LCG and CCLL to train better. Second, nodes in these zones are members of community co-clustering structures with informative block densities and surprise features. Factoring in the community signal helps to positively interpret the surprise.

CCLL is found to be faring poorly in very sparse regions of graphs. This is because there are not enough locality-based signals available in those sparse zones. Moreover, number of positive training samples is very low due to the absence of edges. Conversely, a non-local signal is expected to be more crucial in a sparse zone than in its dense counterpart. Since CCLL does not consider non-local effect, it fails to perform well in sparse regions. However, LCG picks up necessary non-local effects and performs substantially better than LL or CCLL in such regions.

*Dense vs Social:* We see a similar trend when the query nodes are bucketed based on the number of triangles formed. However, in the middle zone based on no. of neighbors (Figure 4, left), the network structure based algorithms e.g. AA or CRW perform worse than LCG and CCLL. This is not the case for bucketing based on social nature (Figure 4, right), where AA and CRW perform comparably. The main advantage of LCG and CCLL come from the fact that they are able to utilize the node signals for extremely unsocial nodes effectively.

#### 5.5.2 Workload distribution in citation networks

Figure 5 shows the workload distribution w.r.t both no. of neighbors and triangles. In contrast to the movie-movie datasets, WebKb shows almost monotonic decrease in performance for all the algorithms as the graph becomes dense from sparse. This can be attributed to two factors: relatively lower spread in degrees which leads to less variation in algorithms utilizing local and global signals, and relatively more informative node features, which is evident from the fact that LL performs better than only structure based methods e.g. AA or CRW. We note that LCG and CCLL

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



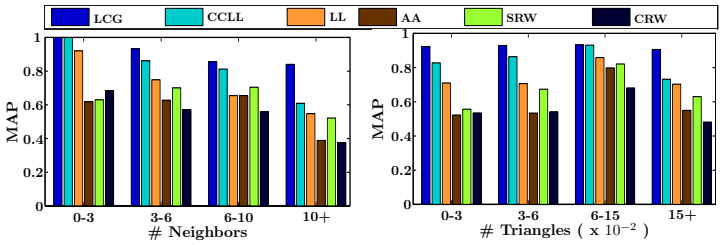


Figure 5: Workload distribution for **Webkb** dataset.

perform best as they use both node and graph structure information.

A surprising feature of graph structure based algorithms, e.g. AA or CRW, is that their accuracy decreases as the network becomes more dense. This can be attributed to the fact that in citation graphs, papers tends to cite a small representative set of papers in a similar area. So the existence of a frequent walk between papers $x$ and $y$ signifies that $x$ has already cited a similar paper lying as an intermediate node in the walk. Hence, if that paper covers the main idea/content of $y$, $x$ need not cite $y$.

### 5.6 Performance variation across datasets

From Table 3 we observe the wide variation of MAP over various datasets, across the most competitive algorithms. The variation may be due to various graph properties like density, clustering coefficient or due to the (un)structuredness of the constituent features. We organize the results of MAP with respect to the three parameters mentioned above, and present the same through Figure 7. Following the first sub-figure, we see Movielens is having the most well-structured feature space. It consists of genre and it is observed that people usually like similar movies. We find that it has the highest MAP. Feature richness is not a perfect predictor of high MAP. Netflix has rich features but performs relatively poorly. Like MovieLens, Netflix is also a movie-movie network, but its features are not as informative, because they are derived from many other factors (like year of the movie, actors, director etc.) apart from genre, and the distillation process seems noisy.

The other predictor of MAP is the average degree (Figure 7, middle), higher density means more signals to the classifier. Netflix which has very low average degree, performs poorly. The third factor which enhances accuracy is local connectedness - this is captured through clustering coefficient (Figure 7, right). The three paper repositories, WebKb, Cora and Citeseer, have very unstructured feature spaces (words) as the features are polluted by polysemy, synonymy, etc. However, the high clustering coefficient of WebKb balances this disadvantage and is a key reason for its good performance. For the other algorithms LL and SRW, the same ranking is maintained, although the performance gap between Movielens (first) and WebKb (second) increases. This is in line with the same observation mentioned before that LL and SRW cannot exploit link structure well. The rankings in CRW and AA are different and more in line with the ranking of the dataset with respect to density and clustering coefficient.

| Dataset | YALP | CCLL | LL | AA | CRW | SRW |
|---|---|---|---|---|---|---|
| Netflix | 0.3906 | 0.3512 | 0.3362 | 0.3450 | 0.3475 | 0.3543 |
| Movielens | 0.5814 | 0.5729 | 0.5430 | 0.5061 | 0.4849 | 0.5521 |

TABLE 8: MAP for collaborative recommendation-induced networks.

### 5.7 Evaluation on bipartite graphs

Collaborative filtering or recommendation can be regarded as a special case of link prediction, in which the graph is bipartite (people and movies, for example). Recommendation amounts to predicting missing edges in the bipartite graph, which has been gainfully modeled as matrix completion (people = rows, movies = columns) [29], [8], [22], [2]. The underlying assumption here is that the matrix is somehow *simple*, e.g., has low rank. Because of the specific nature of the problem, it has many particularly well-suited matrix completion algorithms. Although our goal is not to compare these with general link prediction algorithms, it is tempting to take advantage of many recommendation datasets to compare within and across link prediction algorithms. This section focuses on such a study.

Suppose we are given a user-movie network $G_u = (V_u, E_u)$. For each user $u_1$, we randomly sample 90% of his/her (liked) movie neighbors. Denote this movie-set as $M_{u_1}$. Now consider a movie $m_1 \notin M_{u_1}$. A reasonable assumption is that, if $m_1$ has large similarity to $M_{u_1}$, then $u_1$ is likely to watch/like $m_1$. Therefore, any suitable measure of similarity between $m_1$ and $M_{u_1}$ can be used as a user-movie score for the link $(u_1, m_1)$. We define (explained in Figure 6) the following user-movie score based on a given movie-movie similarity $P$:

$$s(u_1, m_1) = \frac{1}{|M_{u1}|} \sum_{j \in M_{u_1}} p_{m_1,j}, \tag{16}$$

This lets us convert Netflix and Movielens data into the format we need.

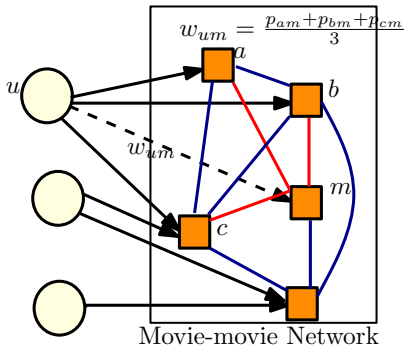


Figure 6: Recommendation as a quality-metric for Link Prediction.

For a given user $u_1$, we randomly sample $M_{u_1}$ which is used to obtain the scores $s(u_1, m_1)$ for all the movies $m_1 \in V_{m_1} - M_{u_1}$. Based on these scores, we obtain a ranked list of movies which ideally should place the neighbor-movies at the top. From the ranked lists for all users, we obtain precision, recall and Mean Average Precision (MAP). Table 8 shows the performance of various LP algorithms.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering



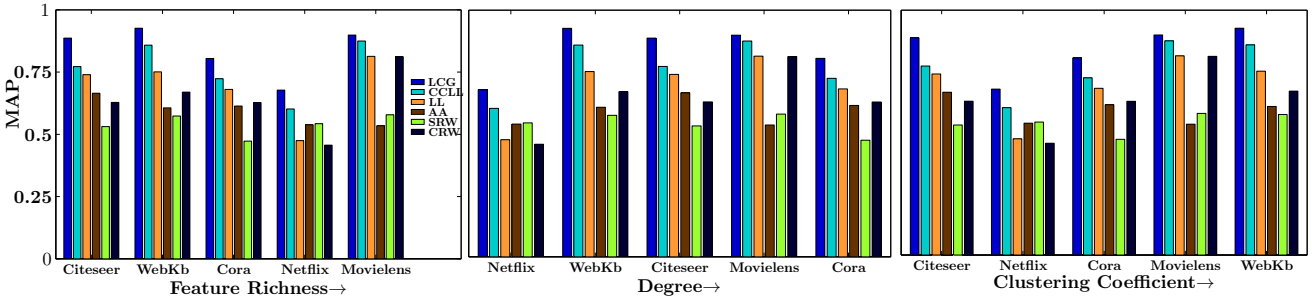


Figure 7: Variation of MAP over various datasets w.r.t. feature richness, average degree and clustering coefficient. (The same data is presented in three different orders).

We observe that, LCG performs best in both datasets. This is because it suitably combines different types of network-attributes. Another interesting observation is that the performance of LL is worse than SRW although it performs better when we consider a movie-movie network. This may be due to the presence of a class of users who watch movies of varied tastes which can be captured well when one considers long-range interactions in the movie-movie network; SRW specifically does that. Although the performances of LL and AA are poor, addition of co-clustering (CC) signal boosts the performance as reflected by a higher MAP of CCLL. This is because as CC captures community structures, so CCLL can infer the similarity and diversity between movies in different clusters. We re-emphasize that these are not realistic collaborative recommendation runs. There is no user profile, vital in any recommendation work, and also exploited by most LP systems. Also, the movie-movie edge weights capture similarities only *on average*, not per-user. The original network is the co-movie network, which is basically the one-mode projection of the user-movie network. Hence the diversity in choice of a movie among various user communities is not properly captured. (One can consider computing weights for a user-specific movie-movie network (two movies have an edge if the same user watches them), which is expected to improve performance. However, this is highly computation-intensive. These directions are outside the scope of the current work.) As expected, all these factors lead to substantially lower absolute MAP scores compared to "real" link prediction problems. Still, the exercise gives us confidence that LCG is a robust and versatile LP system.

## 6 Conclusion

The contributions in this paper include proposing use of local node based features and co-clustering (community) based features, in addition to the local and global graph features used by existing schemes. We also described a new two-level learning algorithm for link prediction. At the lower level, we learn a local similarity model across edges. At the upper level, we combine this with co-clustering signals using a SVM.

Another contribution of this paper is the extensive and systematic evaluation of different link-prediction algorithms. We also perform specific experiments to understand the areas of inefficiency of link prediction algorithms and consequently establish the importance of the proposed two-level learning scheme, and the new features proposed here. We show that our algorithm consistently outperforms four strong baselines when link information is neither too sparse nor too dense.

We also note that link prediction is a generic problem, with many potential applications e.g. recommendation, social search etc. Interestingly, we find that performance of link prediction algorithms on movie-movie networks and user-movie networks (used for recommendation), do not follow the same order. Hence, one must evaluate the link prediction algorithm with respect to the application in which the process is applied.

As future work, one can investigate link prediction in bipartite graphs or graphs with typed vertices. One can also investigate the use of new sources of information for link prediction. For example, Yuan et al. [44] use sentiment of the messages posted by users. A deeper question is to understand the connection between growth models such as preferential attachment, and use temporal graph data to train models that unify link prediction and network evolution [40], [45].

## References

[1] L. A. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211 – 230, 2003.
[2] O. Allali, C. Magnien, and M. Latapy. Link prediction in bipartite graphs using internal links and weighted projection. In *Netscicom*, 2011.
[3] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM Conference*, 2011.
[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
[6] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *SIGKDD Conference*, 2004.
[7] P. Y. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9 PART 2):1505–1514, 1997.
[8] F. C. T. Chua and E.-P. Lim. Modeling bipartite graphs using hierarchical structures. In *ASONAM*, 2011.
[9] A. De, M. S. Desarkar, N. Ganguly, and P. Mitra. Local learning of item dissimilarity using content and link structure. In *RecSys*, 2012. (Poster).
[10] A. De, N. Ganguly, and S. Chakrabarti. Discriminative link prediction using local links, node features and community structure. In *ICDM*, 2013.
[11] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *SIGKDD*, 2003.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2553665, IEEE Transactions on Knowledge and Data Engineering




[12] P. Doyle and L. Snell. Random walk and electric networks. In *Mathematical Association of America*, 1984.

[13] B. Ermi?, E. Acar, and A. Cemgil. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining and Knowledge Discovery*, 29(1):203–236, 2015.

[14] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE TKDE*, 19(3):355–369, 2007.

[15] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. R. Shi, and D. Song. Joint link prediction and attribute inference using a social-attribute network. *ACM Trans. Intell. Syst. Technol.*, 5(2):27:1–27:20, Apr. 2014.

[16] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *PNAS*, 106(52):22073–22078, 2009.

[17] Q. Ho, J. Eisenstein, and E. P. Xing. Document hierarchies from text and links. In *WWW Conference*, 2012.

[18] http://www.netflixprize.com.

[19] G. Jeh and J. Widom. Scaling personalized web search. In *WWW Conference*, 2003.

[20] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[21] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, Mar. 1953.

[22] J. Kunegis, E. W. De Luca, and S. Albayrak. The link prediction problem in bipartite networks. In *Computational intelligence for knowledge-based systems design*, pages 380–389. Springer, 2010.

[23] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[24] A. N. Langville and C. D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 1(3):335–380, 2004.

[25] E. Leicht, P. Holme, and M. E. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[26] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[27] T.-Y. Liu. Learning to rank for information retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, pages 225–331. Now Publishers, 2009.

[28] W. Liu and L. Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

[29] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[30] L. Lü, L. Pan, T. Zhou, Y.-C. Zhang, and H. E. Stanley. Toward link predictability of complex networks. *PNAS*, 112(8):2325–2330, 2015.

[31] L. Lu and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390:1150–1170, Mar. 2011.

[32] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[33] P. Sarkar, D. Chakrabarti, and A. W. Moore. Theoretical justification of popular link prediction heuristics. In *IJCAI*, 2011.

[34] D. Sun, T. Zhou, J.-G. Liu, R.-R. Liu, C.-X. Jia, and B.-H. Wang. Information filtering based on transferring similarity. *Physical Review E*, 80(1):017101, 2009.

[35] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *SIGKDD Conference*, 2007.

[36] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

[37] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(Sep):1453–1484, 2005.

[38] K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(1):326–333, 2004.

[39] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, 1998.

[40] W.-Q. Wang, Q.-M. Zhang, and T. Zhou. Evaluating network models: A likelihood analysis. *EPL (Europhysics Letters)*, 98(2):28004, 2012.

[41] www.cs.umd.edu/projects/linqs/projects/lbc.

[42] www.grouplens.org.

[43] J. Yang, J. J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, 2013.

[44] G. Yuan, P. K. Murukannaiah, Z. Zhang, and M. P. Singh. Exploiting sentiment homophily for link prediction. RecSys '14.

[45] J. Zhao, L. Miao, J. Yang, H. Fang, Q.-M. Zhang, M. Nie, P. Holme, and T. Zhou. Prediction of links and weights in networks by reliable routes. *Scientific reports*, 5, 2015.

[46] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.


PLACE PHOTO HERE

**Abir De** is a PhD student in the Department of Computer Science and Engineering in IIT Kharagpur, India. He received his BTech and MTech in Electrical Engineering from Electrical Engineering Department in 2011. His research interests are modeling and learning influence in social networks. He is the recipient of Google India PhD Fellowship in Social Computing in 2013. He was a visiting research scientist in MPI-Saarbrucken from Sep 2014-Dec 2014.

PLACE PHOTO HERE

**Sourangshu Bhattacharya** is an Assistant Professor in Dept. of Computer Science and Engineering, IIT Kharagpur. His areas of interest include modeling influence in social networks, distributed machine learning and optimization. He did his PhD in Computer Science, from Dept. of Computer Science & Automation , IISc Bangalore in 2008. He was visiting scholar at Helsinki University of Technology from Jan - May 2008 and Scientist at Yahoo! Labs from 2008 to 2013.

PLACE PHOTO HERE

**Sourav Sarkar** is a fourth year undergraduate student in the Computer Science and Engineering Department, IIT Kharagpur, India. He did his internship in Microsoft India Development Center where he developed an Automated Testing framework in C# for the middle-tier testing of Bing Sports Window 8 App. He secured 3rd position all over India in Indian National Mathematical Olympiad-2011.

PLACE PHOTO HERE

**Niloy Ganguly** is a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. He has received his B.Tech from IIT Kharagpur in 1992 and his PhD from BESU, Kolkata, India in 2004. He has spent two years as Post-Doctoral Fellow in Technical University, Dresden, Germany before joining IIT, Kharagpur in 2005. His research interests are in peer-to-peer networks, on-line social networks and wireless networks.

PLACE PHOTO HERE

**Soumen Chakrabarti** is Professor of Computer Science at IIT Bombay. He received his B.Tech in Computer Science from the IIT Kharagpur, in 1991 and his M.S. and Ph.D. in Computer Science from the University of California, Berkeley in 1992 and 1996. He was a Research Staff Member at IBM Almaden Research Center from 1996 to 1999, where he worked on the Clever Web search project and led the Focused Crawling project. In Spring 2004 he was Visiting Associate professor at Carnegie-Mellon University. During 2014–2016 he was Visiting Scientist at Google.

1041-4347 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.