2013 IEEE 13th International Conference on Data Mining

Discriminative Link Prediction using Local Links, Node Features and Community Structure

Abir De
IIT Kharagpur, India
abir.de@cse.iitkgp.ernet.in

Niloy Ganguly
IIT Kharagpur, India
niloy@cse.iitkgp.ernet.in

Soumen Chakrabarti
IIT Bombay, India
soumen@cse.iitb.ac.in

***Abstract*—A link prediction (LP) algorithm is given a graph, and has to rank, for each node, other nodes that are candidates for new linkage. LP is strongly motivated by social search and recommendation applications. LP techniques often focus on global properties (graph conductance, hitting or commute times, Katz score) or local properties (Adamic-Adar and many variations, or node feature vectors), but rarely combine these signals. Furthermore, neither of these extremes exploit link densities at the intermediate level of communities. In this paper we describe a discriminative LP algorithm that exploits two new signals. First, a co-clustering algorithm provides community level link density estimates, which are used to qualify observed links with a surprise value. Second, links in the immediate neighborhood of the link to be predicted are interpreted through a local model of node feature similarities. These signals are combined into a discriminative link predictor. We evaluate the new predictor using five diverse data sets that are standard in the literature. We report on significant accuracy boosts compared to standard LP methods (including Adamic-Adar and random walk). Apart from the new predictor, another contribution is a rigorous protocol for benchmarking and reporting LP algorithms, which reveals the regions of strengths and weaknesses of all the predictors studied here, and establishes the new proposal as the most robust.**

***Keywords*-Social network, Link prediction, Recommendation.**

1. Introduction

The link prediction (LP) problem [15] is to predict future relationships from a given snapshot of a social network. E.g., one may wish to predict that a user will like a movie or book, or that two researchers will coauthor a paper, a user will endorse another on LinkedIn, or 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 [17] provide a comprehensive survey. As we shall describe in Section 2, in trying to predict if nodes u, v in a social network are likely to be(come) related, LP approaches predominantly exploit three kinds of signals:

- When nodes have associated feature vectors (user demography, movie genre), node-to-node similarity may help predict linkage.
- Local linkage information, such as the existence of many common neighbors, may hint at linkage. The well-known Adamic-Adar (AA) [1] predictor and variants use such information.
- Non-local effects of links, such as effective conductance, hitting time, commute time [8], or their heuristic approximations are often used as predictors. The Katz score [13] is a prominent example. The random walk paradigm has also been combined [2] with edge features for enhanced accuracy.

Recently, stochastic block models [9], factor models and low-rank matrix factorization [14] have been used to "explain" a dyadic relation using a frugal generative model. These have rich connections to coding and compression. Co-clustering [7] and cross-association [4] are related approaches. Co-clustering exposes rich block structure in a dyadic[2] relation. E.g., in a user-movie matrix, it can reveal that some users like a wide variety of movies whereas others are more picky, or that some classic movies are liked by all clusters of people.

Although co-clustering provides a *regional* community density signal, it is arguably more meaningful than global, unbounded random walks. However, exploiting the signal from co-clustering is non-obvious. The generative model implicit in co-clustering is that edges in each dyadic block are sampled iid from a Bernoulli distribution with a parameter corresponding to the empirical edge density in the block. While the choices of blocks and their edge densities offer optimal global compression, they cannot predict presence or absence of *individual links* without incorporating node features and local linkage information.

Our key contribution (Section 4) is a new two-level learning algorithm for LP. At the lower level, we learn a local model for similarity of nodes across edges (and non-edges). This is combined, using a support vector machine, with an entirely new non-local signal: the output of co-clustering [7], suitably tuned into feature values.

To the best of our knowledge, this is the first work that brings all three sources of information (node features, immediate neighborhood, "regional" community structure) together in a principled way. Another contribution (Section 3) is a rigorous evaluation protocol for LP algorithms,

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

[2] Can be extended to larger arity.

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.68

using a new binning strategy for nodes based on their local link structure. The new protocol reveals the strengths and weaknesses of LP algorithms across a range of operating conditions.

On five diverse and public data sets (NetFlix, MovieLens, CiteSeer, Cora, WebKb) that are standard in the LP community, our algorithm offers (Section 5) substantial accuracy gains beyond strong baselines. Our main experimental observations are:

- The local similarity model on edges beats baselines in some regions of the problem space, but is, in the overall, not significantly better.
- The additional signal from communities, found through co-clustering, is strong and helpful.
- A global discriminative learning technique using the above signals convincingly beats baselines.

2. Related work

LP has been studied in different guises for many years, and formalized, e.g., by Liben-Nowell and Kleinberg [15]. Lu and Zhou [17] have written a comprehensive survey.

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

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

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

Random walks and conductance: Lu and Zhou [17] have shown that the best-performing definition were local [19] and cumulative (called "superposed" by Lu and Zhou) random walks (LRW and CRW). Although some of these approaches may feel ad-hoc, they work well in practice; Sarkar *et al.* [18] have given theoretical justification as to why this may be the case.

Probabilistic generative models: One of the two recent approaches that blend node features with linkage information is by Ho *et al.* [9], 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.

Supervised random walk (SRW): The other approach to blend node features with graph structure is supervised and discriminative [2], and based on personalized PageRank [11]. However, 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.

Unsupervised random walk: Before SRW, Lichtenwalter *et al.* [16] introduced an unsupervised prediction method called PropFlow. It is somewhat similar to rooted PageRank, but it is a more localized measure of propagation, and is insensitive to topological noise far from the source node.

Co-clustering: Similar documents share similar terms, and vice versa. In general, clustering one dimension of a dyadic relation (represented as a binary matrix A, say) is synergistic with clustering the other. Dhillon *et al.* [7] 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.

3. Evaluation protocol

An LP algorithm applied to the current snapshot of a live social network ranks node pairs that are predicted to materialize as edges soon. As time progresses, in principle, the quality of such predictions can be directly measured. In practice, multiple LP algorithms cannot be precisely compared on a live network. Furthermore, feeding back predictions to users may modify the future evolution trajectory. Finally, public data sets are mostly "fossilized" without any scope for time travel.

From the application perspective, the LP algorithm's output is best captured as a ranking of all current non-neighbors of every node q. In Facebook terms, every person gets a ranked recommendation for potential friends. If they agree to some of them and create a link, that is a positive judgment; if they ignore a high-ranking recommendation, that is a negative judgment. Therefore, instead of using a 0/1 per-edge loss, we need a ranking loss function involving something like AUC, MAP or NDCG [10] at each node, and then average these.

For evaluating algorithms within practical time on large graphs, we may not be able to afford evaluating a ranking loss at every node, Instead we may sample a subset of *query* nodes. 92% of all edges created on Facebook Iceland close a path of length two, i.e., a triangle [2]. In our experiments, only such nodes are sampled as query nodes Q. Even after this filter, query nodes are diverse with regard to the existing edge density in their immediate neighborhood. Two LP algorithms with comparable accuracy may perform quite differently at query nodes with many neighbors vs. few neighbors. As we shall see, bucketing overall MAP or NDCG performance by current query node degree can expose strengths and weaknesses of competing LP algorithms.

Once we sample Q, $|Q|(n-1)|$ edge slots remain, but this can also be too large. To sample edges in the absence of edge timestamps, we proceed as follows. Fix a query node 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. To avoid paring down the graph connectivity drastically, we use $\sigma \sim 0.8 \ldots 0.9$.

4. Proposed framework: CCLL

We, in this section, first clearly define the *link prediction* problem and then explain our scheme.

4.1. 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. LP systems often restrict the potential set of vs to ones that have a common neighbor c, i.e., (u, c) and (c, v) already exist in E.

4.2. 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 will aggregate several kinds of input signals, described throughout the rest of this section. Apart from subsuming existing signals from AA we will harness two new signals. First, in Section 4.3, we will describe a local learning process to determine effective similarity between two given nodes. Unlike AA and other node-pair signals, our new approach recognizes that propensity of linkage is not purely a function of node similarity; it changes with neighborhood. Second, in Section 4.4 we describe how to harness the output of a coclustering of the graph's adjacency matrix to derive yet more features. To our knowledge coclustering has never been used in this manner for LP.

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 confidence in the 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. (i) $f(u, v)[\text{AA}]$ is the block derived from the Adamic-Adar (AA) score (1). (ii) $f(u, v)[\text{LL}]$ is the block derived from local similarity learning (Section 4.3). (iii) $f(u, v)[\text{CC}]$ is the block derived from co-clustering (Section 4.4).

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 (ν) using a SVM and its possible variations. The details will be discussed in Section 4.5.

To exploit possible interaction between features, we can construct suitable kernels [3]. Given we have very few features, a quadratic (polynomial) kernel can be implemented by explicitly including all quadratic terms, i.e., we construct a new feature vector whose elements are (i) $f(u, v)[i]$ for all indices i, and (ii) $f(u, v)[i]\, f(u, v)[j]$ (ordinary scalar product) for all index pairs i, j. We can also choose an arbitrary informative subset of these. We will now describe the three blocks of features.

4.3. Learning local similarity

An *absolute* notion of similarity between u and v, based on node features θ_u, θ_v, is not strongly predictive of linkage; it also depends on the typical *distribution* of similarity values in the neighborhood [5]. 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 as 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

$$\Delta_w(u, v) = w_{uv} \cdot |\theta_u - \theta_v|, \tag{2}$$

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

4.3.1. 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, 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 (2) to the *set dissimilarity*

$$\Delta_w(u, A) = \frac{1}{|A|} \sum_{v \in A} \Delta_w(u, v). \tag{3}$$

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 $\Delta_w(u, v)$.

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. Subject to these constraints, we wish to choose w so as to minimize $\Delta_w(u, v)$. Details are in the full version of this paper [6].

4.3.2. Computation of LL features: The linear program outputs w^*_{uv}, from which we can compute

$$\delta_{uv} = w^*_{uv} \cdot |\theta_u - \theta_v|. \tag{4}$$

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

$$\begin{aligned}\bar{\Delta}_{w^*}(u,v) &= \sum_{i\in\Gamma(u)\cap\Gamma(v)} \frac{\Delta_{w^*}(i,u)+\Delta_{w^*}(i,v)}{|\Gamma(u)\cap\Gamma(v)|} \qquad (5)\\ &= \Delta_{w^*}(u,\Gamma(u)\cap\Gamma(v)) + \Delta_{w^*}(v,\Gamma(u)\cap\Gamma(v)).\end{aligned}$$

Finally, we return, $f(u,v)[LL] = \bar{\Delta}_{w^*}(u,v) - \delta_{uv}.$ (6)

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.

4.4. Co-clustering and “surprise”

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 $d(u,v)$, 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.

We turn the block density $d(u,v) \in [0,1]$ into features for discriminative learning by characterizing 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 $d(u,v)$, the surprise is inversely related to $d(u,v)$; in information theoretic terms, the surprise is $-\log d(u,v)$ bits. (So if $d(u,v) \to 0$, yet the edge exists, the surprise goes to $+\infty$.) Similarly, if the edge does not exist, the surprise is to $-\log(1-d(u,v))$ bits.

4.5. The discriminative learner for global model ν

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 $\hat{y}_{uv} = \mathrm{sign}(\nu \cdot f(u,v))$.

4.5.1. Loss function: ν can be learnt to minimize various loss functions. The simplest is 0/1 edge misclassification[3] loss $\sum_{q\in Q} \frac{1}{N-1} \sum_{v\neq q} [\![\hat{y}_{qv} \neq y_{qv}]\!]$, which is usually replaced by a convex upper bound, the hinge loss. As we have discussed in Section 3, for ranking losses, it is better to optimize the AUC, which is closely related [12] to the pairwise loss. which is again usually approximated by the hinge loss

$$\sum_{q\in Q} \frac{\sum_{g\in G(q), b\in B(q)} \max\{0, 1-\nu\cdot(f(q,g)-f(q,b))\}}{|G(q)|\,|B(q)|} \qquad (7)$$

Joachims [12] offers to directly optimize Λ for several ranking objectives; we choose area under the ROC curve (AUC) for training Λ, 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.

Dataset	N	E	$n(a)$	d_{avg}
Movielens	3952	5669	18	2.8689
CiteSeer	3312	4732	3703	2.7391
Cora	2708	5429	1433	3.89
WebKb	877	1608	1703	2.45
NetFlix	17770	20466	64	2.3034

Figure 1. 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.

4.5.2. Feature map: We now finish up the design of $f(u,v)[AA], f(u,v)[LL]$ and $f(u,v)[CC]$. The first two are straight-forward, we simply use the single scalar (1) for $f(u,v)[AA]$, and $f(u,v)[LL]$ is also a single scalar as defined in (6). The $f(u,v)[CC]$ case is slightly more involved, and has two scalar elements, one for each surprise value:

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

Accordingly, ν 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 [12], [20].

5. Experiments

We compare CCLL against several strong baselines such as Adamic-Adar (AA) [1], Cumulative Random Walk (CRW) [17], Supervised Random Walk (SRW) [2], and Generative Model (GM) [9] with four popular public data sets, summarized in Figure 1. Apart from using LL as features to CCLL, we run LL independently as a baseline.

5.1. Performance of CCLL compared to other algorithms

Figure 3 gives a comparative analysis of MAP (Mean Average Precision) values for all datasets and algorithms, and Figure 2 gives a more detailed view of precision vs. recall. We observe that, for all datasets, the overall performance of CCLL is substantially better than all other methods.

[3] $[\![B]\!]$ is 1 if B is true, 0 otherwise.

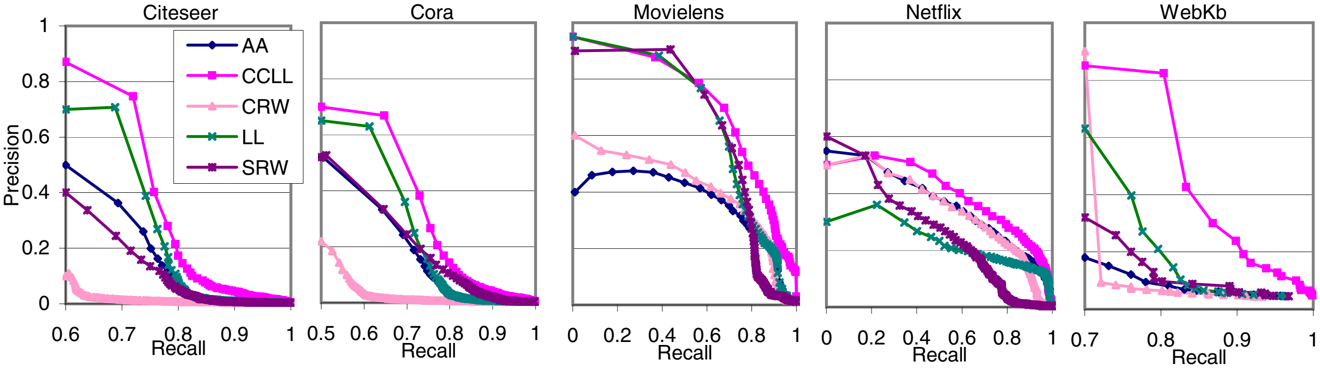

Figure 2. Precision vs. recall curves for all data sets and algorithms.

Performance of the probabilistic generative model is particularly poor. This was surprising to us, given stochastic block models (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 score of neighbors of any nodes 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.

AA, CRW all produce comparable performance. All these methods solely depend on link characteristics, for example AA depends on the number of triangles a node is part of, hence misses out the important node or edge feature information. Regarding CRW, as t goes to ∞, 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. We also found the inherent non-convexity of SRW to produce suboptimal optimizations. LL is better than SRW in all data sets except NetFlix. This is because the number of features in Netflix is small and the values assumed by each feature is very diverse, making the feature-based local prediction strategy ineffective.

A possible explanation of the superior performance of CCLL can be that the underlying predictive modules (LL, AA, CC) perform well in complementary zones which CCLL 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 5.3).

5.2. *Importance of various features*

Figure 4 dissects the various features involved in CCLL, fixing at 90% sampling rate henceforth. To understand the role of different features (local link structure, global community structure), we build our SVM model eliminating either LL or CC and calculate the ranking accuracy (MAP). The results show that the absence of each of the signals (LL and CC) significantly deteriorates the performance. However, closer scrutiny showed the interesting property that the deterioration occurs in different zones, that is, it is almost always true that the node whose MAP gets affected by elimination of LL does not face such problem when CC is removed.

5.3. *Workload Distribution*

In this section, we present some comparative analysis between CCLL and other four best benchmark algorithms on two representative datasets: Netflix and Movielens (Figure 5). The choice is motivated by the fact that Movielens gives best performance and Netflix gives worst performance (with CCLL) among all the five datasets. Netflix has few features while Movielens is feature-rich. Query nodes are bucketed based on the number of neighbors they have (changes from sparse to dense), and each bucket holds roughly one-sixth of the nodes. 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 neighbor increases). The two feature based algorithms, LL and SRW, perform well in the sparse zone, and the improvement in the dense (more social) zone is observed

Dataset	CCLL	LL	AA	CRW	GM	SRW
Netflix	0.602	0.475	0.538	0.543	0.127	0.456
Movielens	0.875	0.813	0.534	0.578	0.200	0.811
CiteSeer	0.772	0.739	0.665	0.531	0.145	0.628
Cora	0.723	0.681	0.614	0.477	0.058	0.627
WebKb	0.858	0.751	0.604	0.574	0.336	0.669

Figure 3. Mean average precision over all algorithms and datasets.

Dataset	CCLL	$SVM(LL, AA)$	$SVM(LL, CC)$
Netflix	0.6017	0.5761	0.5478
Movielens	0.8740	0.8436	0.8483
Citeseer	0.7719	0.7467	0.7677
Cora	0.7203	0.7044	0.7139
WebKb	0.8583	0.8106	0.8470

Figure 4. Feature ablation study.

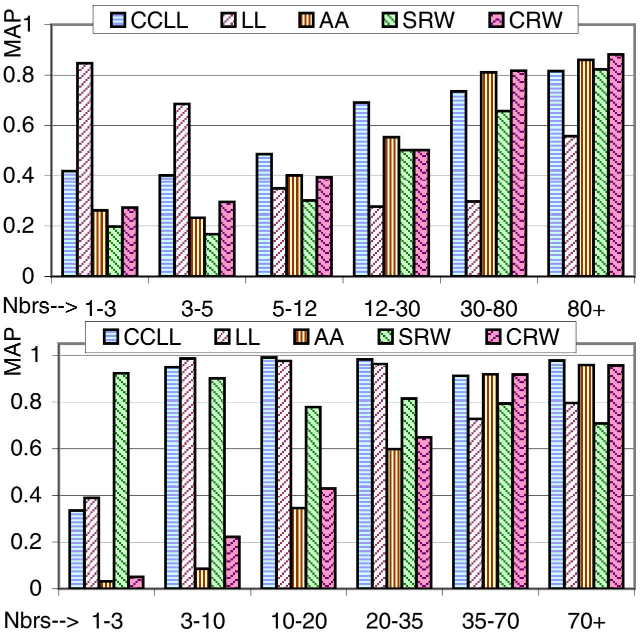

Figure 5. Workload distribution based on no. of neighbors on "Netflix" (above) and "Movielens" (below).

but not as significant as the two link-based algorithms. Clearly, in these two zones (sparse and dense), two different classes of algorithm work well. CCLL performs well in all the zones by appropriately learning signals in each zone. However, it even improves upon its two constituents in each and every zone. From Figure 5 we observe that CCLL performs best (substantially better than LL and AA), at intermediate density. There are two reasons behind it. Even though the graph is sparse, in these regions, $|G(q)|$ and $|B(q)|$ are not far apart, which helps CCLL to train better. Second, nodes in these zones are member of community coclustering structures with informative block densities and surprise feature. Factoring in the community signal helps to positively interpret the surprise.

6. Conclusion

We 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. On diverse standard public data sets, the resulting link predictor outperforms recent LP algorithms.

Acknowldegement: This work was supported by Google India under the Google India PhD Fellowship Award.

References

[1] L. A. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211 – 230, 2003.

[2] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM Conference*, pages 635–644, Hong Kong, 2011.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *SIGKDD Conference*, pages 79–88, Seattle, WA, USA, 2004. ACM.

[5] A. De, M. S. Desarkar, N. Ganguly, and P. Mitra. Local learning of item dissimilarity using content and link structure. In *RecSys*, pages 221–224, 2012. (Poster).

[6] A. De, N. Ganguly, and S. Chakrabarti. Discriminative link prediction using local links, node features and community structure. http://arxiv.org/abs/1310.4579, Oct. 2013.

[7] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. pages 89–98, Washington, D.C., 2003. ACM.

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

[9] Q. Ho, J. Eisenstein, and E. P. Xing. Document hierarchies from text and links. In *WWW Conference*, pages 739–748, Lyon, France, 2012. ACM.

[10] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR Conference*, pages 41–48, 2000.

[11] G. Jeh and J. Widom. Scaling personalized web search. In *WWW Conference*, pages 271–279, 2003.

[12] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, pages 377–384, 2005.

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

[14] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS Conference*, 2001.

[15] 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.

[16] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *SIGKDD Conference*, pages 243–252, Washington, DC, USA, 2010. ACM.

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

[18] P. Sarkar, D. Chakrabarti, and A. W. Moore. Theoretical justification of popular link prediction heuristics. In *IJCAI*, pages 2722–2727, Barcelona, Spain, 2011. AAAI Press.

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

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