Finding Underlying Connections: A Fast Graph-Based Method for Link Analysis and Collaboration Queries

Jeremy Kubica JKUBICA@RI.CMU.EDU
Andrew Moore AWM@CS.CMU.EDU
David Cohn COHN+@CS.CMU.EDU
Jeff Schneider SCHNEIDE@CS.CMU.EDU
Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

Many techniques in the social sciences and graph theory deal with the problem of examining and analyzing patterns found in the underlying structure and associations of a group of entities. However, much of this work assumes that this underlying structure is known or can easily be inferred from data, which may often be an unrealistic assumption for many real-world problems. Below we consider the problem of learning and querying a graph-based model of this underlying structure. The model is learned from noisy observations linking sets of entities. We explicitly allow different types of links (representing different types of relations) and temporal information indicating when a link was observed. We quantitatively compare this representation and learning method against other algorithms on the task of predicting future links and new "friendships" in a variety of real world data sets.

1. Introduction

Many techniques in the social sciences, criminology, and graph theory deal with the problem of analyzing patterns found in the underlying structure and associations of a group of entities. A typical query may examine an entity's "importance" by looking at its relations to others and thus its position within the underlying structure (Wasserman & Faust, 1994). However, in many real world situations true structure may not be known. Instead the relevant information may be buried within large amounts of noisy data.

In this paper, we examine the problem of learning and querying a graph-based model of the underlying structure. The model is learned from link data, where each link is a single noisy input observation, specifically a set of entities observed to share some relation. As with most sources of real world data, we assume this data is noisy, possibly containing many irrelevant links, as well as links with spurious or missing members. Below, we present cGraph, a learning method that approximates the underlying structure. We account for different types of links, which may vary in their frequency and extent of noise. We also incorporate information about when a link occurred, allowing us to account for changes in the underlying structure over time.

We also introduce the "friends" problem, which examines how strongly two entities are connected and attempts to predict which entities will co-occur in future links. We use this query to test the effectiveness of the cGraph algorithm as compared to a variety of other algorithms.

The motivation for this research is the increasing number of analytical applications that are trying to exploit massive amounts of noisy transactional and co-occurrence data. This kind of data is important to insurance analysts, intelligence analysts, criminal investigators, epidemiologists, human resource professionals, and marketing analysts. Of the many questions such analysts might like to ask, one general question is "What are the underlying relationships among people in this organization?" and, at a more focused level: "Who are the other people most likely to be directly interacting with this person?" For example, if a key employee leaves a corporation, HR needs to know who is most likely to be impacted. Or if an individual is caught transporting illegal materials, law enforcement may need to rapidly deduce who are possible accomplices.

2. Collaborative Graph Model

Graphs, or social networks, are a common method for representing the relationships among a set of entities. Nodes represent the entities and edges capture the relations between them. For example, a simple graph representing the "friends" relation may consist of a graph with an undirected edge between any two entities that are friends.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

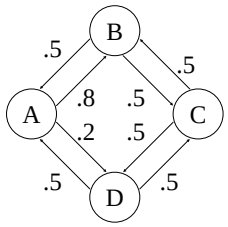

Figure 1. Example collaborative graph.

We use a directed weighted graph to capture the underlying relations between entities, but restrict the weights to form a multinomial distribution over the outgoing edges. In other words for W_{AB}, the weight of an edge from A to B, we have:

$$0 \leq W_{AB} \leq 1 \qquad \text{and} \qquad \sum_B W_{AB} = 1 \tag{1}$$

Thus W can be interpreted as a transition matrix. It should be noted that an **edge** is a weighted ordered pair representing a true underlying connection and is different from the input links. An example collaborative graph is shown in Figure 1. Such a graph-based structure is simple, capturing only pair wise relationships, but is fast to approximate and query and is easy to interpret.

3. Link Data and Generation Models

The model for the N_P entities is learned from a set of N_L observed links. A **link** is a single noisy input observation, an unordered set entities that are linked by some relation.[1] The observations are noisy - some relations that exist in truth may not be observed, and some observations may not be the result of actual relations. A given link may also have spurious or missing members. The word *link* should not be interpreted too narrowly. A link can be used to capture a range of different relations such as: direct communication, co-occurrence, or even sharing a common attribute.

3.1. Link Generation

Below we present several models for link generation. These models serve both to provide a model of link generation (for MLE graphs and creating artificial data) and also as motivation for the approximations presented below.

3.1.1. The Random Walk Model

The random walk model of the link generation considers a link as a random non-self-intersecting walk on the underlying graph. This walk continues for a set number of propagations or until it reaches a dead-end. Formally, given a link L_{j-1} that contains $j-1$ entities such that A_{last} was the last entity added to the link, the next entity for the link is chosen as:

$$P(A_i|A_{last} \wedge L_{j-1}) = \begin{cases} \frac{P(A_i|A_{last})}{\sum_{A_k \notin L_{j-1}} P(A_k|A_{last})} & A_i \notin L_{j-1} \\ 0 & A_i \in L_{j-1} \end{cases} \tag{2}$$

The link is terminated if there is no A_i with $P(A_i|A_{last} \wedge L_{j-1}) > 0$.

While this process can be viewed as a random walk, the transition probabilities are not independent. Since the walk must be non-self-intersecting, the probability of transitioning to the A_i depends not only on A_{last} but also on all of the other entities already in the link.

3.1.2. The Random Tree Model

The random tree model is a more realistic model of link generation, where the jth entity can be added to the link by any of the previous members of the link. A "seed entity" starts the link. Each new entity is then added to the link by a random entity that is already in the link. Formally, given a link L_{j-1} that contains $j-1$ entities and a randomly chosen "inviter" from the link, $A_{inv} \in L_{j-1}$, the next entity for the link is chosen as:

$$P(A_i|A_{inv} \wedge L_{j-1}) = \begin{cases} \frac{P(A_i|A_{inv})}{\sum_{A_k \notin L_{j-1}} P(A_k|A_{inv})} & A_i \notin L_{j-1} \\ 0 & A_i \in L_{j-1} \end{cases} \tag{3}$$

The link is terminated if there is no A_i and A_{inv} such that $P(A_i|A_{inv} \wedge L_{j-1}) > 0$.

3.2. Maximum Likelihood Estimate Graphs

It is possible to find the MLE of the underlying graph given the generative model and the set of observed links by just using a simple optimization scheme to search the graph weights. This optimization would include at least N_P^2 real valued parameters, making the search expensive for large numbers of entities. Unfortunately since the ordering of the links is unknown, the step of evaluating the likelihood of the data may also be computationally expensive. Assuming the random walk generative model, simply calculating the probability of a link with N entities requires examining up to $N!$ different orderings.

4. Approximating the Collaborative Graph from Link Data

The first and most important task for cGraph is to learn the underlying graph from the link information. If the learned graph does not reflect the true underlying structure, future queries to it may return incorrect results.

[1] For example, this paper links its authors by the *co-author* relation.

4.1. Collaborative Graph Approximation

The cGraph algorithm learns the underlying graph by using weighted counts of co-occurrences to approximate the edge weights. These counts can be accumulated during a single scan of the data. The resulting edge weights can then be computed directly from these counts.

We can motivate this approximation by interpreting W_{AB} as $P(B|A)$: the probability that if we randomly chose a link containing A and randomly chose a second entity in the link, this entity would be B. This simple interpretation leads to the approximation:

$$\begin{aligned} W_{AB} = \hat{P}(B|A) &= \textstyle\sum_L \hat{P}(B|A \wedge L)\hat{P}(L|A) \\ &= \frac{\sum_{L:(A,B)\subset L}\left(\frac{1}{|L|-1}\right)}{\sum_{L:A\in L} 1} \end{aligned} \quad (4)$$

where the second line follows from the assumption that both the link containing A and the second entity were chosen uniformly.

We can extend this sampling interpretation of edge weights further to account for noise. We assume that the links of different types have different and unknown probabilities of being relevant and noise free, denoted by $U(L.type)$. Similarly, links that occur at different times have different probabilities of being relevant to the current graph, denoted by $T(L.time,t)$. Thus the edge weights represent the probability of choosing B from randomly selected relevant (both in type and time) and noise free link containing A.

Given the above probabilistic interpretation, we can derive a closed form approximation of the edge weights as:

$$W_{AB}(t) = \frac{\sum_{L:(A,B)\subset L}\left(\frac{U(L.type)T(L.time,t)}{|L|-1}\right)}{\sum_{L:A\in L} U(L.type)T(L.time,t)} \quad (5)$$

where $|L|$ is the size of the link, $U(L.type)$ is the typical weighting of a link of type $L.type$, and $T(L.time,t)$ is the temporal weighting of a link at time t where the link occurred at time $L.time$. The choice of weighting functions is described in the following section. This approach is similar to the methods presented by Newman (2001) and Kautz *et. al.* (1997), but accounts for additional information contained in the links and is normalized to ensure a probabilistic interpretation.

The weighted counting approximation given in (4) can also be justified in the context of the above generative models. In both models the second entity is sampled directly as $P(B_2|A_1)$, where B_i indicates that entity B was the ith entity added to the link. Although we do not know the order that entities were added to the link, if we treat each possible link ordering as equally probable, we can approximate the weight of an edge from A to B as a percentage of possible link orderings in which A_1 and B_2. A combinatorial argument on either model leads to the same approximation:

$$\hat{P}(B_2|A_1) = \begin{cases} \frac{1}{(|L|-1)} & B \in L \\ 0 & B \notin L \end{cases} \quad (6)$$

We can then approximate the overall probability of $A_1 \rightarrow B_2$ as the average probability of $A_1 \rightarrow B_2$ in the links in which A appeared, which results in an approximation identical to that in (4). The difference between the approximate and exact model is that we are no longer accounting for the fact that no entity can be added to a link twice.

4.2. Temporal and Typical Weighting

The temporal weighting of a link, $T(L.time,t)$, determines the extent to which older links are counted as relevant. The intuition behind using a temporal weighting function is that we expect recent links to be more indicative of the current graph than links that occurred some time ago. Thus we choose $T(L,t)$ to be a exponential decay function:

$$T(L,t) = \begin{cases} e^{-\beta(t-L.time)} & t \geq L.time \\ 0 & t < L.time \end{cases} \quad (7)$$

where β is a parameter that determines the decay rate of the exponential. The choice of $\beta = 0$ makes all links that occurred on or before t equally likely.

The typical weighting of a link type determines how much a link of that type should be counted compared to links of other types. Thus we can restrict the weights to lie in the range of $[0,1]$, where a weight of 0 results in the cGraph algorithm effectively ignoring all links of that type.

4.3. Time Complexity and Scaling

To build the graph we only need to scan through the data one time and accumulate weighted counts of the number of occurrences and co-occurrences. If $|L_i|$ is the length of link i then there are $\sum_{i=1}^{N_L}|L_i|^2$ co-occurrences to count. Thus if $|L_{max}|$ is the length of the largest link, accumulating the counts has an upper bound of $O(N_L * |L_{max}|^2)$. In addition we need to normalize the outgoing edge weights. If we let B_{max} be the largest number of outgoing edges (branching factor) from a single node, the entire approximation requires time $O(N_L * |L_{max}|^2 + N_P * B_{Max})$.

Table 1 illustrates how this algorithm scales. The entry t_{build} is the time in milliseconds (or 0.0 if less than 0.1 ms) it takes to approximate the graph once. Figure 2 shows how the algorithm scales with the number of links (from a given data set) used to build the model. Even for large numbers of links and entities this approximation is reasonably fast.

5. Learning the Parameters

The approximation presented above contains several parameters, such as the link type weighting and the tempo-

Table 1. A summary of data sets: number of entities (N_P), number of links (N_L), maximum link length ($|L_{max}|$), average link length ($|L_{ave}|$), time (in milliseconds) to construct the graph (t_{build}), maximum branching factor of the resulting graph (B_{max}), average branching factor of the resulting graph (B_{ave}), and time (in milliseconds) to perform queries with depth 3 (t_Q).

DATA SET	ENTITIES	LINKS	$\|L_{max}\|$	$\|L_{ave}\|$	t_{build}	B_{max}	B_{ave}	t_Q
LAB	115	94	16	3.3	0.0	44	7.11	0.0
INSTITUTE	456	1737	49	2.9	40.0	159	26.4	62.3
DRINKS	136	5325	6	2.8	10.0	131	13.5	1.2
MANUAL	4272	5976	14	2.2	100.0	131	4.0	0.1
CITESEER (25%)	105357	45348	122	2.8	2234.2	227	2.6	33.3
CITESEER (50%)	105357	90697	157	2.8	3229.2	333	4.3	45.8
CITESEER (75%)	105357	136046	272	2.8	4357.5	475	6.1	95.3
CITESEER (100%)	105357	181395	385	2.8	5490.0	595	7.7	151.5

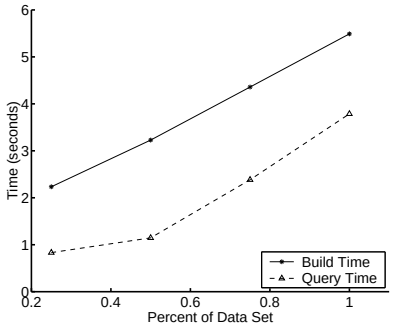

Figure 2. Average time to build the graph and query 25 entities as the number of links increases.

ral decay parameter. While it may often be possible to set these parameters a-priori using domain specific knowledge, a more appealing approach is to learn these parameters. If the parameters are unspecified, then the cGraph algorithm attempts to learn them using cross validation. The performance is measured by how closely the learned graph corresponds to the normalized counts from the cross validation set. Unless otherwise noted, the experiments below learn type weightings using cross validation and use $\beta = 0$.

6. Friends Identification

One of the primary goals of the cGraph algorithm is to answer such queries as: "List *A*'s top 10 current/future/new collaborators?" We call these queries the friend identification problem and consider two versions. The *All Friends* problem asks: "Given the data who are the *K* entities with whom *A* is most likely to appear future links?" The *New Friends* problem asks: "Given the data seen so far, who are the *K* entities with whom *A* has never appeared in a link and is most likely to appear with in future links?" These are predictive questions about co-occurrences for each entity as opposed to more standard web link analysis queries such as identifying authoritative pages (e.g. hubs and authorities (Gibson et al., 1998)).

Another way to look at the friendship problem is as a distance or proximity measure where two entities that are "close" are more likely to collaborate in the future. The simplest proximity function is just W_{AB}. Although this method is computationally cheap, it is unable to generalize past co-occurrences it has already seen.

We use a distance measure based on a nonself-intersecting walk that is able to capture the concept of "a friend of a friend." Specifically, we define the proximity between *A* and *B* to be the probability of a fixed length, random, nonself-intersecting walk from *A* will include *B*. Formally:

$$prox(A,B) = \sum_{v \in V_{(A,B,D)}} \left(\prod_{e_i \in v} P(e_i | A_j \, \forall j < i) \right) \qquad (8)$$

where $V_{(A,B,D)}$ is the set of all nonself-intersecting walks of length less than or equal to D from A to B and $e_i \in v$ is a directed edge in v. We approximate this metric by only considering paths of length at most three steps. All nodes which are not reachable from A in at most three steps are given an arbitrarily low proximity.

This measure uses a depth first search over the graph and thus may be computationally expensive. Formally we can place an upper bound on the running time of $O(min\{B_{max}^D, N_p\})$. As this expression shows, the friendship query is independent of the number of links and scales polynomially with the branching factor of the graph. Table 1 and Figure 2 again illustrate how this operation scales to different size data sets; t_Q indicates the average time of a friendship query on the learned graph.

7. Friends Identification Results

7.1. Data Sets

To test the cGraph algorithm we used a variety of real world data sets, described below and summarized in Ta-

ble 1. These data sets will be available (in the same form used here) at: `http://www.autonlab.org/`.

1. The *Lab Data* consists of co-publication links for members of our research lab and includes all people present in those links.

2. The *Institute Data* is a set of links of three different types (co-publication, common research interest, and advisor/advisee) that was collected from public data on Carnegie Mellon University Robotic Institute's webpages. The co-publication links were marked with the year of publication. This data set provides an example of multiple link types with different amounts of relevant information. We would expect an advisor/advisee link to be a better predictor of future collaboration than simply a common research interest.

3. The *Citeseer Data* is a collection of co-publication links from the Citeseer online library and index of computer science publications. Each publication served as a single link containing its authors. Since entities were represented by first initial and last name, a single name could correspond to multiple authors. In addition subsets consisting of 25%, 50%, and 75% of the Citeseer links were used to demonstrate scaling properties. These links were chosen randomly and the entries in Table 1 are averages over 12 runs.

4. The *Drinks Data Set* consists of popular bar tending recipes found on various websites. Each link consists of all ingredients (entities) contained in that drink.

5. The *Manual (Webpages) Data* is a set of links created by a human who manually read a set of public web pages and news stories related to terrorism and subjectively linked entities mentioned in the articles. These pages were often written/hosted by various governments and news organizations. Each link was created by hand from a single relation, such as `memberOf` or `funded`, found a web page and contained the entities for which this relation was mentioned on the page.

7.2. Competing Algorithms

The performance of the cGraph algorithm on the friends identification task was tested against a variety of other algorithms. Simple baseline and strawman comparisons were given by the *Random*, *(Co-occurrence) Counting*, and *Popular (Entity)* algorithms. The *Random* algorithm returns unique random entities as an entity's friends. The *(Co-occurrence) Counting* algorithm returns those entities that have most often co-occurred with an entity as its top friends. Finally, the *Popular (Entity)* algorithm counts up the number of links in which each entity has occurred and ranks the entities according to this count. The predicted friends of any entity are just the most "popular" entities (excluding the entity itself) according to this ranking.

Spectral Clustering algorithms include a broad range of techniques based on eigenvalue analysis of feature vectors that have been shown effective in the field of network analysis and bibliometrics (Garfield et al., 1978; Gibson et al., 1998; Ng et al., 2001). This analysis can be viewed as a form of factoring: it begins with a matrix A, where each row i corresponds to a link and each column j corresponds to an entity. The value of $A(i, j)$ indicates whether entity j participated in link i. A singular value decomposition (SVD) decomposes A into sets of left and right singular vectors (Press et al., 1993). The left singular vectors correspond to canonical sets of entities, which may be interpreted as latent groups, while the right singular vectors correspond to the "mixing proportions" in which these groups should be combined to best approximate the observed links. Spectral clustering consists of assigning each observed link to the canonical set that dominates its mixture.

The mixing proportions may also be used to estimate similarities between entities in one of a number of ways. In this paper, we estimate the "affinity" of two entities as the Euclidean distance between the vectors representing their mixing proportions. Friends are predicted as entities with the highest "affinity" to a given entity.

In this paper, we examined both traditional SVD-based spectral clustering (labeled as "Spectral-S") and a variation based on non-negative matrix factorization (NNMF) (Lee & Seung, 2001), labeled as "Spectral-N" below. NNMF produces a decomposition similar to the SVD, but adds the constraint that an entity's membership in the canonical sets must be strictly non-negative.

7.3. Cross Validation Tests

We used sequential cross validation on the above data sets to compare the performance of the algorithms. The performance of an algorithm is the percentage of guessed friends with which it actually co-occurred in the test set. The number of friends requested for each entity was equal to the true number of test set co-occurrences. It is important to appreciate that this test is very similar to a real world use of this algorithm. Given all data up to this time, we would like to make queries about possible connections in this time step.

Table 2 shows the performance of various algorithms on the above data sets. The performance is given as the mean success rate for 10 fold sequential cross validation. The runs were treated as independent and results marked with a * are significantly different (using $\alpha = 0.05$) from those that are not marked. If no entry in the column is marked then the performance of most algorithms on that test were not significantly different.

Table 2. The average 10 fold sequential cross validation results (percentage of correctly guessed test set co-occurrences).

	INSTITUTE		CITESEER		DRINKS		LAB		MANUAL	
ALGORITHM	ALL	NEW	ALL	NEW	ALL	NEW	ALL	NEW	ALL	NEW
CGRAPH	***38.65**	*1.54	*29.06	***0.200**	*46.10	**1.51**	**19.74**	1.86	**1.38**	0.21
POPULAR	10.98	***1.65**	0.39	0.069	***47.49**	0.84	14.50	**2.27**	1.37	**0.41**
COUNTING	34.56	0.18	***29.49**	0.001	*47.37	0.88	17.27	1.13	1.29	0.03
RANDOM	1.37	0.29	0.03	0.017	13.07	0.95	3.94	1.27	0.10	0.01
SPECTRAL-N	12.52	*0.72	2.90	0.016	18.15	0.74	13.85	2.02	0.44	0.09
SPECTRAL-S	13.16	*0.63	2.31	0.024	14.26	0.66	14.17	1.95	0.52	0.17

Table 3. The average 10 fold sequential cross validation results (percentage of correctly guessed test set co-occurrences) on the Institute data set for cGraph with different temporal decay rates.

β	ALL FRIENDS	NEW FRIENDS
0.0	35.86	0.66
0.5	40.99	0.92
2.0	41.25	0.79

Table 2 illustrates that there is no one algorithm that always outperforms the other algorithms. It is important to appreciate though that on all of the data sets and metrics cGraph is among the best performers. On the Institute, Citeseer, and Drinks data it is significantly better than many of the other algorithms. Further, even when it is not the single best performer, its performance is very often practically close to that of the best algorithm. Another interesting trend shown in the table is the good performance of the Popular algorithm. This success suggests a possible extension to the cGraph algorithm by incorporating priors for the entities.

7.4. Effect of Beta Term

To demonstrate temporal weighting's effect on performance sequential cross validation were run on the 1606 publication links in the Institute data set. These links were time stamped with the year of the publication. The goal was to predict collaborations in the *K*th subset given the previous subsets. Results for several values of β are shown in Table 3. For both metrics, the performance of cGraph with $\beta = 0.5$ was significantly better than with $\beta = 0.0$ with a significance level of $\alpha = 0.05$.

Initially increasing β can lead to increased performance at predicting future collaborations by placing more weight on recent collaborations. But as β continues to increase, the performance decreases as more recent information is discounted. In fact when β grows too large all information before the current time is effectively ignored.

8. Experiments with Known Graphs

Another important test of cGraph is whether it actually learns the "correct" graph or a good approximation. Queries to an incorrect graph are at best unreliable. As a test, we compared learned graphs to known graphs that generated the link data.

In addition to evaluating the performance of the cGraph algorithm, we compared its performance to that of several other learning algorithms. The first algorithm was a simplified version of the cGraph algorithm using fixed, equal link type weights. This algorithm counts links weighted solely by the size of the link and thus is similar to the approaches presented by Newman (2001) and Kautz *et. al.* (1997). The second algorithm was an unweighted link counting approach, which did not even take into account link size. The third algorithm used a hill-climbing search to find the MLE graph and weights as described in Section 3.2. The underlying generative model was assumed to be the random walk model on all of the tests. It is important to appreciate that we do not expect MLE to necessarily perform better, because the structure with the highest likelihood may not be the true underlying structure. The fourth algorithm also used a hill-climbing search to find the MLE graph for the random walk model, but assumed fixed, equal link type weights. Finally, to provide an approximate baseline, the fifth algorithm simply generated a random graph.

8.1. Data Generation for "Known Graph" Tests

At the start of each test, a ground truth graph was randomly generated. Edges added between two nodes were added in both directions, but often had different weights. A set of 2,000 links, including noise links, was then created using one of the generative models. A link was created by first choosing a link type and then choosing to make the link a completely random with probability $1 - \mathit{weight}(\mathit{linktype})$. All tests used three different link types with fixed weights of 0.9, 0.9, and 0.5. Thus while one link type contained significantly more noise links than the others, none of the types were completely noise free.

Table 4. Mean error (squared distance between the learned and known graphs) in reconstructing a graph.

Algorithm	20 Entities Walk	20 Entities Tree	50 Entities Walk	50 Entities Tree
cGraph (Learned)	1.47	1.52	*3.76	*3.78
cGraph (Fixed)	1.96	2.01	5.01	4.96
MLE (Learned)	*1.23	*1.35	10.05	10.12
MLE (Fixed)	1.34	1.46	12.46	12.40
Counting	2.40	2.44	6.04	5.94
Random Graph	11.74	11.34	29.83	30.33

8.2. Results on Known Graphs

Table 4 summarizes the resulting performance of the algorithms. Each column consists of the average error over 50 tests using the indicated generative model and number of entities. The error metric itself is the squared distance between the learned and known graphs:

$$error = \sum_A \sum_B (W_{AB}^{true} - W_{AB}^{learned})^2 \quad (9)$$

where W_{AB}^i is the weight of the edge from A to B in graph i. This metric provides a reasonable approximation of the distance between two graphs and is computationally efficient to calculate. The entries in Table 4 marked with a * are statistically significantly (with $\alpha = 0.05$) better than all unmarked entities in their column.

The results show a clear advantage to weighting the different link types and an advantage to using the cGraph algorithm over the MLE as the number of entities grows. Although the MLE algorithm outperformed the cGraph algorithm for a small number of entities, this difference is comparatively small compared to the difference between cGraph and the other algorithms. More significantly, the cGraph algorithm has a distinct advantage on data sets with more links and larger links. The link sizes above were limited to at most 5 entities. For more links or links of larger sizes, simply evaluating the probability for MLE may not be computationally feasible.

9. Related Work

As illustrated by several of our data sets and examples above, our work is similar to the analysis of co-citations, referral webs, and web structure. Within the areas of bibliometrics and web analysis eigenvector methods, such as spectral clustering, have been shown effective (Garfield et al., 1978; Gibson et al., 1998; Ng et al., 2001). These techniques presented a promising complement/alternative to our research and thus were considered in the above tests. It is important to appreciate that while our work is similar to the areas above, we are attempting to infer different information. Specifically, we wish to infer the underlying direct connections between entities. Co-citations and linked websites imply only relatively weak and indirect connections between the entities, such as common research topic.

The use of probabilistic models is another recent approach to the above problems (Cohn & Hofmann, 2001; Friedman et al., 1999; Getoor et al., 2001; Kubica et al., 2002). Cohn and Hofmann (2001) presented a model of document generation where they assume that documents are generated from a mixture of sources and that the contents of the document (citations and terms) are probabilistically determined from these sources. Similarly Kubica *et. al.* (2001) presented a model of link generation where links are generated from a single underlying group and then have noise added. These models differ significantly from ours in that we do not assume that a mixture of sources, but rather probabilistically determine an entity's membership in the link by which entities are already members of the link. Getoor *et. al.* (2001) present a technique that uses a probabilistic relational model to classify web pages. Here again there is an important difference in the type of information we are attempting to infer. We are not attempting to classify the entities or to create a full probabilistic model, but rather to predict the underlying direct connections between entities.

Our approach is similar to techniques presented by Newman (2001) and Kautz *et. al.* (1997). Both Newman and Kautz *et. al.* describe methods for approximating and querying a social network given link data. Although both present approaches based on a weighted count of co-occurrences, these differ from our own. Both works use heuristics to choose link weighting and neither work considers link reliability factors such as type or when the link occurred. Further, both works attempt answer questions different from the friendship problem. For example, Kautz *et. al.* attempt to make referrals by finding chains between two entities. While this is similar to our problem it lacks the important characteristic that multiple paths imply a stronger, although indirect, connection.

Many other techniques and fields, such as social network analysis, complement our research since they require that an underlying graph structure is known or can easily be inferred from data. For example p*, creates predictive models for relational variables (such as our friendship question) from a collection of explanatory variables extracted from demographics and a known social network (Anderson et al., 1999; Wasserman & Pattison, 1996). Another example is the work of Schwartz and Wood (1992) where a graph is used to find subgraphs of people with related interests. While Schwartz and Wood estimate the graph using link information, they limit this estimation to Boolean edges indicating the presence of any communication.

10. Conclusion

We examined the problem of quickly inferring an underlying graph structure to capture relationships between entities. This structure is learned from a collection of noisy link data, which for many problems may be easier to obtain than the true underlying graph. In conjunction, we introduced the "friends" problem, which examines how strongly two entities are connected even if they have never appeared in a link together. These problems have promise for applications in areas such as: sociology, marketing, security, and the study of artificial networks.

There are several natural extensions that we leave as future work. First is the incorporation of the concept of "popularity of entities" into the model. Motivated by both a natural extension to the generative models and the success of the Popular algorithm on the new friends problem, the incorporation of priors may boost performance. Second is an extension beyond positive pairwise relationships to include such relationships as *A*, *B* and *C* are all likely to appear together in pairs, but never as a triple.

Acknowledgements

Jeremy Kubica is supported by a grant from the Fannie and John Hertz Foundation. This research is supported by DARPA under award number F30602-01-2-0569. The authors would also like to thank Steve Lawrence for making the Citeseer data available.

References

Anderson, C. J., Wasserman, S., & Crouch, B. (1999). A p* primer: Logit models for social networks. *Social Networks*, *21*, 37–66.

Cohn, D., & Hofmann, T. (2001). The missing link - a probabilistic model of document content and hypertext connectivity. *Advances in Neural Information Processing Systems 13* (pp. 430–436). Cambridge, MA: MIT Press.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (pp. 1300–1309). San Francisco: Morgan Kaufmann Publishers.

Garfield, E., Malin, M. V., & Small, H. (1978). Citation data as science indicators. In Y. Elkana, J. Lederberg, R. K. Merton, A. Thackray and H. Zuckerman (Eds.), *Toward a metric of science: The advent of science indicators*. New York: John Wiley and Sons.

Getoor, L., Segal, E., Taskar, B., & Koller, D. (2001). Probabilistic models of text and link structure for hypertext classification. *IJCAI01 Workshop on Text Learning: Beyond Supervision*. Seattle, Washington.

Gibson, D., Kleinberg, J., & Raghavan, P. (1998). Inferring web communities from link topology. *Proceedings of the Nineth ACM Conference on Hypertext and Hypermedia*. New York: ACM Press.

Kautz, H. A., Selman, B., & Shah, M. A. (1997). The hidden web. *AI Magazine*, *18*, 27–36.

Kubica, J., Moore, A., Schneider, J., & Yang, Y. (2002). Stochastic link and group detection. *Proceedings of the Eighteenth National Conference on Artificial Intelligence* (pp. 798–804). New York: ACM Press.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems 13* (pp. 556–562). MIT Press.

Newman, M. E. J. (2001). Who is the best connected scientist? a study of scientific coauthorship networks. *Phys. Rev.*, *64*.

Ng, A. Y., Zheng, A. X., & Jordan, M. I. (2001). Link analysis, eigenvectors and stability. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (pp. 903–910). San Francisco: Morgan Kaufmann Publishers.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1993). *Numerical recipes in c : The art of scientific computing*. Cambridge, England: Cambridge University Press. Second edition.

Schwartz, M., & Wood, D. (1992). Discovering shared interests among people using graph analysis of global electronic mail traffic. *Communications of the ACM, 36*, 78–89.

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge, England: Cambridge University Press.

Wasserman, S., & Pattison, P. (1996). Logit models and logistic regression for social networks: I. an introduction to markov graphs and p*. *Psychometrika, 60*, 401–425.