A Comparison of Statistical and Machine Learning Algorithms on the Task of Link Completion

Anna Goldenberg
Center for Automated Learning and Discovery
Carnegie Mellon University
Pittsburgh, PA 15213
anya@cs.cmu.edu

Jeremy Kubica
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
jkubica@cs.cmu.edu

Paul Komarek
Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213
komarek@andrew.cmu.edu

ABSTRACT
Link data, consisting of a collection of subsets of entities, can be an important source of information for a variety of fields including the social sciences, biology, criminology, and business intelligence. However, these links may be incomplete, containing one or more unknown members. We consider the problem of link completion, identifying which entities are the most likely missing members of a link given the previously observed links. We concentrate on the case of one missing entity. We compare a variety of recently developed along with standard machine learning and strawman algorithms adjusted to suit the task. The algorithms were tested extensively on a simulated and a range of real-world data sets.

Keywords
link analysis, link completion, evaluation, nearest neighbor, logistic regression, naive bayes, bayes nets

1. INTRODUCTION
Link data, in general described as a set of entities connected by either an observed or probabilistic relation, is an important source of information for a variety of fields including the social sciences, criminology, and data mining. Further link data is often a natural representation for data found in a vast range of fields, such as social network analysis (people are connected based on papers they have co-authored, meetings they have attended together, etc), internet analysis (pages are connected by references to and from them), market-basket analysis (items in a store are connected by shopping baskets where they have co-occured), etc.

The data, collected from the real life sources, is usually noisy and might contain gaps, i.e. links may be incomplete, containing one or more unknown members. Below we consider the problem of link completion, determining the missing member given a partial link. For example, if we know that Alice, Bob and a third person attended a meeting, we could ask which people are most likely to be that third person given what we know about people's previous co-occurrences. It is important to note that this question is similar to those found in the collaborative filtering domain [4]. Differences between the tasks as well as the adaptation of algorithms are discussed below.

The question of link completion is also important in a variety of other domains that seek to use information from links. For example, in the domain of market-basket analysis we may wish to know which was the item that did not scan correctly or was corrupted when transferring the information to the database, based on the other items that have appeared in the shopping basket (related to identity uncertainty). Link completion algorithms could also be used to predict the next item to be purchased given the items that a shopper is currently planning to buy, such information is tracked when buying products online (related to collaborative filtering). Here the transaction itself is the link to be completed and the entities are the items to buy.

The task of link completion serves another important role: providing a metric to quantitatively compare link analysis algorithms. Various link analysis techniques can be compared by how well they perform at the link completion problem. The intuition behind this comparison is that methods that are better at capturing the underlying relations among entities may perform better at completing noisy links.

We compare the performance of a variety of algorithms on this problem using real-world data sets. Specifically, we compare algorithms designed for link analysis (cGraph and EBS), statistical and machine learning algorithms (logistic regression, nearest neighbor, Bayesian Networks), and a variety of simple straw man comparisons (co-occurrence counting and the popular person algorithm). We provide a detailed analysis relating datasets, the task of link completion and the performance of the algorithms.

2. LINK DATA AND COMPLETIONS
2.1 Link Data
The data is assumed to consist of a set of links. Lets assume there are a total of N links in the dataset. Each link is a subset of entities that are linked together by some relation. Let M be the total number of entities. For example, in the domain of co-authorships, such as Citeseer dataset, N is the number of all publications in the dataset

and M is all authors that have occured in at least one publication. Authors in this case are linked by a common paper. In the domain of market basket analysis, N is the number of baskets and M is the number of items observed. The items may be linked by co-occurrence in the same transaction that puts them together in the same basket. Links can contain an arbitrary number of the entities and are not fixed size.

The word *link* should not be interpreted too narrowly. A link can be used to capture a range of different relations such as: direct interaction, co-occurrence, or even sharing a common attribute. For example, in the domain of food products the link `{flour, sugar, eggs}` may result from a cookie recipe and the link `{water, milk}` may result from the fact that both entities are liquids.

There is an abundance of ways to represent link data. The algorithms compared in this paper use one of the three following representations:

1. A sparse binary matrix $\mathfrak{M} = N \times M$, where
$$\mathfrak{M}_{\mathrm{ij}} = \begin{cases} 1 & \text{item j occured in link i} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$
This adjacency matrix representation is used by several algorithms discussed in this paper such as Bayes Nets (Section 4.6). It is also analogous to the user-item dataset representation in the domain of collaborative filtering, where the binary entries correspond to the presence or absence of the user's vote for a particular item [28].

2. An adjacency list of N entries where each entry (link) points to the entities that occured in the link. In this case we use the set notation to represent the list and use l_i to denote the ith such list of entities. Using the notation above we can state: $e_j \in l_i$ if and only if $\mathfrak{M}_{\mathrm{ij}} = 1$.

3. An adjacency list of M entries where each entry (entity) points to the set of links that it has occured in

All three representations are equivalent as they encode the same information. One or several of the data structures would be used by each of the presented algorithms interchangably for efficient storage and performance improvement. Unless otherwise specified, we will assume data to be represented as a collection of "links" where a *link* is a subset of entities that had been observed to co-occur.

2.2 Link Completion Task

Informally the link completion task is to identify the missing entity or entities in a link. Since one of the major reasons for selecting the task was to compare the link analysis algorithms, we concentrate specifically on the case when the arity of the link is known and we have to predict one missing entity in the link. This is similar to asking the question "Given that we know the following $k-1$ people were at a meeting, who is the kth person?" In this formulation, the task of link completion is similar to *AllBut1* metric used in the collaborative filtering domain, where the algorithms are compared based on how well they predict the voting of a particular user on one of the items given the user's voting information on all other items [4]. The two tasks differ slightly in that the *AllBut1* task would be equivalent to predicting whether or not the given entity was the missing member of the link, as opposed to finding the entity that was the most likely one to complete the link.

We compare our results based on a slightly richer metric. Instead of reporting a single entity for the link completion, we assign all entities a score such that a higher score implies a better match. This formulation of the problem allows us to also examine how often the true link completions have scores in the top ten or the top hundred. Further it allows us to ask: "How many entities would we have to include before we have a p percent chance of including the true entity?"

We define a *partial link* as a link with unobserved members and the *possible link completions* as those entities that did not occur in the partial link.

3. RELATED WORK

The field of link analysis includes a variety of techniques and spans a variety of domains including: criminal intelligence [30], large databases [11], marketing analysis [6], recommender systems [27, 29, 4, 28], analysis of co-citations [5, 8, 26] and analysis of the internet [5, 10, 15]. The task of link completion is a possible aspect of link analysis and is thus related to many other tasks in the field. For example, in the domain of market basket analysis the link completion task may take the form of predicting which additional items will be purchased given the current market basket. In the field of collaborative filtering [27, 4, 28, 2, 20], link completion is similar to predicting the missing vote from the set of user preferences [4].

A variety of techniques (some of which are included in the comparisons below) have been presented to solve various problems in link analysis, including: probabilistic models [5, 7, 9, 18], graph construction and analysis [15, 23, 1, 16, 24, 19], eigenvector methods [8, 10, 25], bayesian networks [4], nearest neighbor [4, 1, 28], etc. Many of these algorithms and techniques have either been tried on tasks similar to link completion such as bayes nets and clustering when applied to *AllBut1* task in the domain of collaborative filtering [4] or may prove effective on the task of link completion and provide an interesting domain for further investigation. Despite this, none of the above techniques have been applied to link analysis using the definition of the link completion as described in Section 2.2 and have been shown to work on such a variety of datasets as presented in this paper.

The algorithms used in the comparison, as well as the relevant references, are described below.

4. COMPETING ALGORITHMS

A variety of algorithms were compared based on the task of link completion. Below we describe the specific algorithms and their formulations that produce a ranked list of possible completions.

The following notation was used:

- $L_T = \{l_1, l_2, \ldots, l_N\}$ denotes a set of training links
- $\hat{L} = \{\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_{\hat{N}}\}$ refers to a set of partial links
- l_T represents a given training link
- $\hat{l}_p$ stands for a given partial link
- ξ^i denotes the ith entity in the given link
- $\Xi = \{\xi^1, \ldots, \xi^M\}$ is an unordered set of all available entities

4.1 Co-occurrence Counting

The *(Co-occurrence) Counting* algorithm is a simple strawman method. Every entity gets a score that is equal to the number of times it has co-occurred with each entity in the partial link. Entities that appeared in the partial link are assigned scores of zero. Formally the score for entity A as a link completion to a given partial link $\hat{l}_p$ is:

$$score(A) = \begin{cases} \sum_{i=1}^{M} \sum_{j=1}^{N} I(\xi_i \in l_j) * I(A \in l_j) & A \notin \hat{l}_p \\ 0 & A \in \hat{l}_p \end{cases} \tag{2}$$

where $I(S)$ is the indicator function (returns 1 if S is true and 0 otherwise).

4.2 Popular Algorithm

The *Popular (Person)* algorithm simply counts the number of links in which each entity has occurred and uses these counts as the scores for the possible link completions. Entities that occurred in the partial link are given a score of zero. Formally the score for entity A as a link completion to $\hat{l}_p$ is:

$$score(A) = \begin{cases} \sum_{j=1}^{N} I(A \in l_j) & A \notin \hat{l}_p \\ 0 & A \in \hat{l}_p \end{cases} \tag{3}$$

Thus more "popular"entities are given a higher score for completing the link.

4.3 Random Algorithm

The *Random* algorithm provides an approximate baseline performance. Each entity that did not appear in the partial link is assigned a random score in $[0,1]$. Entities that appeared in the partial link are assigned scores of zero.

4.4 Nearest Neighbor

The *Nearest Neighbor (NN)* algorithm assigns scores to the possible link completions based on which entities appear in links that are "closest" to the partial link. The *NN* algorithm has been applied extensively in the area of collaborative filtering using a variety of distance metrics [13, 29, 4]. There are arguments pro- and against different similarity metrics. For the purposes of comparison with the rest of the link analysis algorithms we chose to implement the classic Hamming distance metric as a measure of link similarity.

Formally, the distance measure is:

$$Dist(\hat{l}_p, l_T) = \begin{cases} |l_T| + |\hat{l}_p| - 2 * |l_T \cap \hat{l}_p| & |l_T \cap \hat{l}_p| < |l_T| \\ \infty & otherwise \end{cases} \tag{4}$$

We make a restriction that if the training link contains only members present in the partial link, $l_T \subseteq \hat{l}_p$, then we assign an infinite distance. In this case l_T does not contain any new information for predicting the missing link member.

For the task of link analysis, for a given partial link, we search over all links in the training set, and find the closest one(s). Formally, if $D* < \infty$ is the smallest distance found then the score for an entity A is:

$$score(A) = I(A \notin \hat{l}_p) * \sum_{L_T : Dist(l_T, \hat{l}_p) = D*} \frac{I(A \in l_T)}{|l_T| - |\hat{l}_p \cap l_T|} \tag{5}$$

where $I(S)$ is the indicator function (returns 1 if S is true and 0 otherwise).

4.5 cGraph

The *cGraph* algorithm, presented by [19], assumes that links are generated based on an unknown underlying graph structure that captures the pairwise relationships between entities. This structure is similar to a general social network, but is restricted so as to have a probabilistic interpretation.

The cGraph algorithm approximates this underlying graph using weighted counts of co-occurrences that are accumulated during a single scan of the data. The resulting edge weights are then calculated directly from these counts. The links are weighted by: the size of the link, a weighting function of the link type (*typical* weighting) and a weighting function on when the link occurred (*temporal* weighting). Formally the weight from A to B is approximated as:

$$W_{AB}(t) = \frac{\sum_{L:(A,B)\subset L} \left(\frac{U(L.type)T(L.time,t)}{|L|-1} \right)}{\sum_{L:A \in L} U(L.type)T(L.time,t)} \tag{6}$$

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.

The weighting functions determine how much a link of a given type that occurred at a given time is "counted" compared to other links. These functions can be set by hand or chosen by cross validation. Since the other algorithms did not make use of link type or temporal information, these weighting functions were chosen so as to ignore these factors. In the below experiments we set $T(L.time,t) = 1$ $\forall t, L.time$ and $U(L.type) = 1$ $\forall L.type$. Thus all algorithms used only the links themselves as input.

Scores for the possible link completions can then be determined from the learned graph. Specifically, we generate the scores using the random tree generation model presented in [19]. We make the simplifying assumption that the missing entity is the last entity to be added to the link. Thus the probability that A is the missing entity in the partial link $\hat{l}_p$ is:

$$P(A|\hat{l}_p) = \begin{cases} \frac{1}{|\hat{l}_p|} \sum_{B \in \hat{l}_p} \frac{P(A|B \wedge \hat{l}_p)}{\sum_{C \notin \hat{l}_p} P(C|B \wedge \hat{l}_p)} & A \notin \hat{l}_p \\ 0 & A \in \hat{l}_p \end{cases} \tag{7}$$

where $\hat{l}_p$ represents only the known entities. Each possible link completions' score is thus simply $P(A|\hat{l}_p)$.

4.6 Bayesian Networks

The *Bayesian Network (BN)* algorithm learns a network structure and parameters working with the matrix representation of the link data. The Bayesian network was learned using a version of the optimal reinsertion method presented in [22] that has been optimized for massive sparse data. The network thus forms a generative model of how links are created in which each node corresponds to an entity. Interestingly, the presence of an arc between two entities does not imply a working relationship between the entities: we have seen many cases in which the network modeled the presence of one or more parents of an entity in a link as decreasing the likelihood of that entity being in the link.

Bayesian Network models have been applied in the framework of collaborative filtering before, one of the tasks being to predict user's vote for the given item in the *AllBut1* scenario, which resembles the problem of link completion. The formulation of Bayes Nets as described in [4] is different then the one used in this comparative analysis. However, the function that both models optimize is the same: $argmax_D DagScore(D)$, where D is a directed acyclic graph (DAG) and *DagScore* is a complexity-penalized measure of how well the DAG explains the data.

During the link completion phase we wished to find the new entity that was the most probable completer of the link given the known members of the link, and under the assumption that there was *precisely* one completer. This computation can be performed easily by iterating over every possible link that contains one completer and querying the probability of that link. All queries thus involve known values of all variables, and so no expensive inference is required (which is important considering the hundreds of thousands of nodes in the network).

4.7 EBS

The *Empirical Bayes Screening (EBS)* algorithm originally presented by [6], adapted for link analysis as described by [12], is a statistical approach to ranking links according to their "interestingness". The algorithm gives preference to those links that cannot be described by the assumed independence assumption. For pairs, the assumption is complete independence: for items i and j: $e_{ij} = N * p_i * p_j$, where $p_i = n_i/N$ and N is the number of records in the dataset, where n_i is the number of occurences of entity i in the training dataset L_T and e_{ij} is the estimated number of occurences of links with two entities i and j in L_T. For links of size $k > 2$, the independence assumption is based on the all-2way interaction loglinear model [3].

EBS assumes that the number of occurences of links of particular size k are drawn from poisson distributions with parameters $(\lambda_{ij..k} e_{ij..k})$, where λs in turn come from a prior $\pi(\lambda|\theta)$ distribution:

$$\begin{aligned} n_{ij..k}|\lambda_{ij..k} &\sim Pois(\lambda_{ij..k} e_{ij..k}) \\ \lambda_{ij..k} &\sim Gamma(\alpha,\beta) \end{aligned} \tag{8}$$

One gamma prior is used here for reasons described in [12]. The parameter $\theta = \{\alpha,\beta\}$ is estimated from the data using Empirical Bayes. The product of unconditional distribution of ns is then maximized to obtain the maximum likelihood estimates for θ:

$$\begin{aligned} f(n) &= \int Pois(n|\lambda e)\pi(\lambda|\hat{\theta})d\lambda \\ \hat{\theta} &= argmax_\theta \prod_{\forall n} f(n) \end{aligned} \tag{9}$$

To complete the given link, the number of occurences of each possible completion is computed from the training set L_T. The algorithm then calculates the estimates of true counts based on the independence assumption. EBS ranks all possible completions described by (n_i, e_i) for each of the ith completion respectively, according to the posterior geometric mean $\Lambda = exp(\psi(\hat{\alpha} + n_i))/(\hat{\beta} + e_i)$ computed using $\hat{\theta} = \{\hat{\alpha},\hat{\beta}\}$ obtained from the counts and estimates of all links of size k in L_T.

4.8 Logistic Regression and Naive Bayes

Logistic regression (LR) fits a sigmoidal model in parameters β to binary data y, whose output is the expected value

$$\mathrm{E}(y_i|x_i) = \frac{\exp(\beta^T x_i)}{1 - \exp(\beta^T x_i)} \tag{10}$$

LR is often used as a binary classifier, requiring thresholding of $\mathrm{E}(y|x)$ to determine whether x_i belongs to the positive or negative class. A simple extension to multiclass problems is achieved by multiple experiments, each of which computes the expected value that x_i is in the j^{th} class. The final prediction is the class with greatest expected value. See [21] or [14] for details about LR.

To apply LR to the link completion task, we employ an adaptation similar to the multiclass extension described above.

For each entity $\xi^k \in \Xi$, execute the following steps:

1. Create a new training set $L^k = \{Ł_i^k \mid l_i^k = l_i \setminus \{\xi^k\}\}$.
2. Create a sparse binary matrix $X^k = [x_{ij}^k]$, where $x_{ij}^k = 1$ if $\xi^j \in l_i^k$ and $x_{ij}^k = 0$ otherwise, for $j \neq k$. Note that X^k has size $N \times (M-1)$.
3. Create an output vector $Y_k = [y_i^k]$ where $y_i^k = 1$ if $\xi^k \in l_i^k$ and $y_i^k = 0$ otherwise.
4. Learn sigmoidal model β^k for inputs X^k, output Y^k.
5. Create a new testing set $\widehat{L}^k$ and input matrix $\widehat{X}^k$.
6. Compute $E(\hat{y}_i^k|\hat{x}_i^k)$ by applying model β^k to $\hat{X}^k$. Note if p^k is in the incomplete testing link $\hat{l}_i$, then ξ^k cannot be the completion of $\hat{l}_i$. In this case, set $E(\hat{y}_i^k|\hat{x}_i^k) = 0$.

Upon completion, predict entity $\arg\max_k E(\hat{y}_i^k|\hat{x}_i^k)$ for the completion of testing row i. An obvious extension is ordering entities $\xi^k, k = 1,\ldots,M$ for testing row i according to $E(\hat{y}_i^k|\hat{x}_i^k)$, which is useful for some scoring metrics.

This technique requires M LR experiments on data of size $N \times (M-1)$. Both variables may be larger than 100,000 in real datasets, and hence the LR implementation must be very fast. We used the LR implementation described in [17]. Because this "leave-one-entity-out" technique can be used with any binary classifier which provides class probabilties, we include a Naive Bayes predictor for comparison.

5. DATA SETS

To test the algorithms, we used a variety of real world data sets. These data sets are described below and their sizes are summarized in Table 1. Upon publication of this paper we plan to make these data sets available over the web in the same form as used here.

Table 1: Data sets and their sizes

Data Set	Entities	Training Links	Test Links
Lab	114	69	19
Institute	456	1488	248
Drinks	136	4325	997
Manual	4088	4581	815
IMDB	100717	49298	910
Citeseer	104801	180395	905
SimLink	26	34500	10

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 publicly available data listed on Carnegie Mellon University Robotic Institute's webpages.

3. The *Drinks Data Set* consists of a series of popular bartending recipes found on various websites. Each link consisted of a list of all ingredients (entities) contained in that drink.

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

5. The *IMDB Data Set* is a collection of links extracted from movies (1900 to 1960) listed on the Internet Movie Database website. Each movie defines a single link containing all of its actors/actresses.

6. 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.

7. The *SimLink Data Set* is an artificial dataset containing complex interactions that would not be captured by simple strawman algorithms. It consists of 26 entities and 3450 links (each with exactly 5 entities). A portion of the links contain one or several of the 10 triples of entities that had mostly occurred together or solo but have very few pairwise occurrences. The test set consists of pairs selected randomly from each of those triples.

6. RESULTS

The algorithms were compared by examining their accuracy in assigning scores to the possible link completions. For each of the above data sets a predefined number of links was randomly chosen to form a test set. The number of links in the training and test sets are summarized in Table 1. For each of the links in the test set a single entity was randomly chosen to be the missing member and was removed from the link.

Tables 2 through 8 show the relative performance of the various algorithms on the above data sets. Note that in the below experiments we used a preliminary unoptimized version of the EBS algorithm. We are currently investigating optimizations. The results in the columns labeled 1, 10, and 100 indicate the percentage of test links for which the true entity had a score in the top 1, 10, and 100. Entries in these columns that are marked with a * are significantly better (using a likelihood ratio test with $\alpha = 0.05$) than those that are not marked by stars. If no entry in the column is marked then the performance of most algorithms on that data set were not significantly different. Finally, the results in the `Ave. Pos.` column indicate the average rank of the true entity's score among the other scores. The smaller number indicates better performance.

In addition Figures 1 through 7 also show the relative performance of the various algorithms on the above data sets. These figures indicate the percentage of test links (y value) for which the true entity has a score in the top Q entities (x value). This analysis is similar to asking "On average if we took the top Q scoring entities what is the probability that we would get the true missing entity?" The x values are plotted on a logarithmic scale.

7. DISCUSSION

The tests reflect the relative performance of the algorithms on the task of link completion. In addition they show some of the structure in the data sets and thus illustrate on which data sets a given algorithm may perform well or poorly.

7.1 Link Completion

In general, the goal of link analysis algorithms is to infer the underlying structure of the data sample available in the given domain. Evaluation of the algorithms that attempt to identify the "true" structure when it is in fact unknown in majority of real world datasets, is a very hard problem. A variety of evaluation methods are available in related fields such as collaborative filtering, market basket analysis, relational models. The choice of link completion has two purposes, first of all it is an interesting real life problem and it also is algorithm independent, allowing for comparison of statistical and heuristic algorithms. However, it might hinder the illustration of the true strength of some of the algorithms. It was chosen as a basis for comparing link analysis techniques, but there is no claim that it is the best testbed for link analysis algorithms in general. For example, as shown in [4] Bayes Nets had outperformed other techniques on the analogous task of *AllBut1* in the domain of collaborative filtering, whereas other tasks such as withholding more than one vote, had more variation in results and other algorithms had showed performance superior to Bayes Nets under different testing scenarios. In our study, no algorithm had outperformed others on all of the datasets, which is partially related to the difference in structures of the datasets, for example Co-occurence Counting algorithm has outperformed Logistic Regression only on the Institute and IMDB Datasets, whereas LR has performed better than Counting on all other datasets. It is evident from the data that in both datasets there is a group of very popular people, professors in the case of the institute and actors in case of the imdb datasets, that occur in a lot of publications/movies. Predicting those people most of the time lets such a simple heuristic method as counting win over much more complicated algorithms.

The task of link completion assumes that the arity of the link is known and the algorithms further assume that the entity that is missing comes from the pool of $M - |\hat{l}_p|$ entities encountered in the training set. The first assumption is in fact a viable realistic scenario, for example, a papparazzi could take a picture of a room where an important meeting was held and one person had been abstructed from the camera view. The second assumption is in fact a limitation of several of the learning algorithms. If the algorithm had learned probabilistic information from the data itself, it will not have confidence in probabilities of the entities that have not occured in the training set. The task of link completion however does reveal important properties of the algorithms, even if some of them could have shown better performance on other link analysis tasks.

7.2 Performance comparison

The co-occurrence counting and cGraph use a similar approach, weighted pairwise counting, and thus had similar performance in the above tests. Both algorithms take advantage of data with underlying pairwise structure. In other words, the algorithms benefit when each entity, for the most part, individually contributes evidence about other entities. Further, both algorithms only consider positive evidence an entity. For example, neither algorithm is able to capture the concept that seeing entity *A reduces* the chances of seeing entity *B*. Since both algorithms performed well on the Citeseer, IMDB, and Institute data sets, it is reasonable to believe that these data sets contained strong underlying positive pairwise structure.

In contrast other algorithms, such as Popular and Nearest Neighbor algorithms, did not even consider pairwise interactions and thus could not exploit any such underlying structure. Instead these algorithms utilized other forms of structure to improve performance. The nearest neighbor algorithm performed well on data sets with repetition in the link data. For example, it performed well on the institute and lab data sets, where we would expect subsets of people to appear together in multiple and similar links, such as collaborators co-authoring multiple papers together. The popular algorithm performed well in smaller data sets where a few entities frequently occured. One such data set is the Lab data set, where a few people appear in the majority of the links. In contrast the popular algorithm did not perform well on data sets, such as Institute or Citeseer, that were large and where there was no core group of frequent entities.

The EBS algorithm did not perform very well on real datasets, however it has shown significantly better performance on the SimLink data. Designed for capturing subtle, intricate interactions between people, EBS is perfectly suited for identifying the complex relations of the SimLink, where the performance of algorithms based on pairwise interactions was jeopardized. However, when the Count algorithm performs fairly well (in other words when we could expect the completion of the link to be the person who is a "friend" of a few other people that are present in the link) the EBS would show suboptimal performance since the described relation could be easily captured by a simple all-2way interaction model. It is possible to improve the performance of EBS on link completion task by optimizing the loss function, i.e. change the Λ ranking scheme to suit the task at hand better.

Logistic regression performed well on all experiments except SimLink. The fast LR implementation of [17] was virtually insensitive to the number of links or maximum link size, but the "leave-one-entity-out" approach was expensive for large numbers of entities. Until a better method of applying LR to link completion is found, LR is best suited to datasets with fewer entities and large numbers of links.

Overall most algorithms except for the simple strawman algorithms such as Random and Popular and EBS, that might have benefited from the change of the loss function, had performed similarly and relatively well. There has been a larger variance in algorithm performance on the Institute Dataset that could be easily explained by the dominance of the strong positive pairwise correlation of entities. There was no clear winner over all datasets, which suggests that the internal structure of the data might favor the selection of one algorithm over the others. For the future work, we would suggest an adaptive hybrid algorithm, that would start with testing for the simplest interactions of entities within the dataset by looking at the order of loglinear models or simply applying algorithms such as Popular and Count. Such testing would be inexpensive, but could give a better idea about the structure of the dataset. If the performance of the strawman methods were not satisfactory, but, for example, Count had performed significantly better than Popular, we could try cGraph that in essence is a more sophisticated version of Co-occurence Counting. If neither, Popular nor Count had any success with predicting missing entities or more generally, failed in revealing the structure of the dataset, we would suggest trying more sophisticated techniques such as Bayes Nets and EBS.

8. CONCLUSIONS

We have shown how to adapt a variety of algorithms including both link analysis specific and conventional statistical and machine learning algorithms to the task of link completion. We quantitatively compared these algorithms on a variety of real-world data sets on the task of link completion and used this comparison to explore both the strengths/weaknesses of some of the approaches as well as the domains to which they are well suited.

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.

9. ADDITIONAL AUTHORS

Additional authors: Andrew Moore and Jeff Schneider `School of Computer Science, e-mail: awm,schneide@cs.cmu.edu`

10. REFERENCES

[1] C. Aggarwal, J. Wolf, K. Wu, and P. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Knowledge Discovery and Data Mining*, pages 201–212, 1999.

[2] D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conference on Machine Learning*, pages 46–54, 1998.

[3] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, 1977.

[4] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *The Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[5] David Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*, 2001.

[6] B. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proc. 7th ACM SIGKDD Conference*, pages 67–76, 2001.

[7] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[8] E. Garfield, M. V. Malin, and H. Small. Citation data as science indicators. In Y. Elkana, J. Lederberg, R. K. Merton, A. Thackray, and H. Zuckerman, editors, *Toward a Metric of Science: The Advent of Science Indicators*. John Wiley and Sons, 1978.

[9] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI01 Workshop on Text Learning*, August 2001.

[10] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proc. 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[11] H. Goldberg and T. Senator. Restructuring databases for knowledge discovery by consolidation and link formation. In *1st Int'l Conference on Knowledge Discovery and Data Mining*, 1995.

[12] A. Goldenberg and A. Moore. Empirical bayes screening for link analysis. In *Proc. Text Mining and Link-Analysis Workshop at the 18th IJCAI Conference*, 2003.

[13] D. Greening. Building consumer trust with accurate product recommendations, 1997.

[14] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. Wiley-Interscience, 2 edition, 2000.

[15] H. Kautz, B. Selman, and M. Shah. The hidden web. *AI Magazine*, 1997.

[16] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[17] Paul Komarek and Andrew Moore. Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs. In *Artificial Intelligence and Statistics*, 2003.

[18] J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *18th National Conference on Artificial Intelligence*, pages 798–804. ACM Press, Jul 2002.

[19] Jeremy Kubica, Andrew Moore, David Cohn, and Jeff Schneider. Finding underlying connections: A fast graph-based method for link analysis and collaboration queries. In *Submitted*, Aug 2003.

[20] W. Lin, S. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.

[21] P. McCullagh and J. A. Nelder. *Generalized Linear Models*, volume 37 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, 2 edition, 1989.

[22] A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. Proc. 20th ICML Conference, 2003.

[23] M. Newman. Who is the best connected scientist? a study of scientific coauthorship networks. *Phys. Rev.*, 64(016131; 016132), 2001.

[24] M. Newman, D. Watts, and S. Strogatz. Random graph models of social networks. In *Proceedings of the National Academy of Science, to appear*, 2002.

[25] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors and stability. In *IJCAI*, pages 903–910, 2001.

[26] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS*, 2002.

[27] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *The 10th International World Wide Web Conference*, pages 285–295, 2001.

[29] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[30] M. Sparrow. The application of network analysis to crminal intelligence: an assessment of prospects. *Social Networks*, 13, 1991.

Tables and Figures

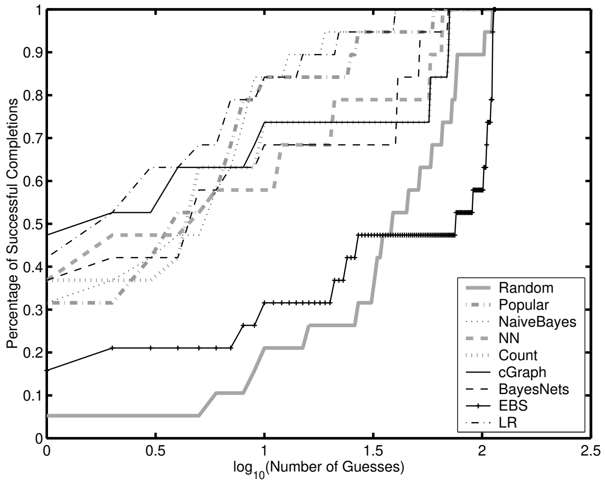

Figure 1: Result curves for the lab data set.

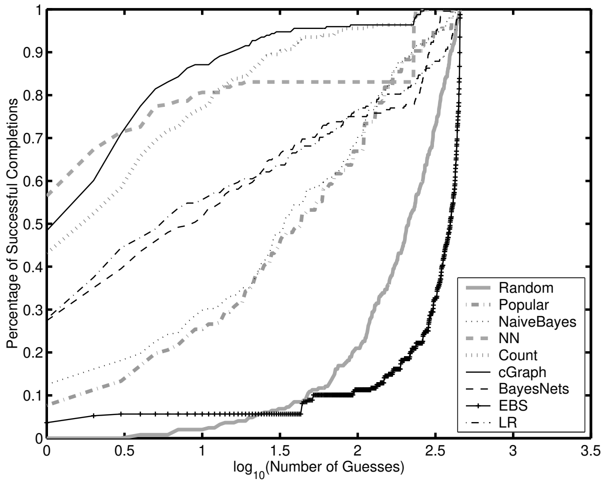

Figure 2: Result curves for the institute data set.

Table 2: Results on the lab data set

Algorithm	1	10	100	Ave. Pos.
Random	0.053	0.211	*0.895	45.89
Counting	0.368	0.737	*1.000	19.95
Popular	0.316	0.842	*1.000	9.37
Naive Bayes	0.368	0.842	*1.000	7.37
NN	0.474	0.579	*1.000	17.68
cGraph	0.526	0.737	*1.000	19.26
Bayes Nets	0.421	0.684	*1.000	17.79
EBS	0.211	0.316	0.579	59.68
LR	0.526	0.842	*1.000	6.53

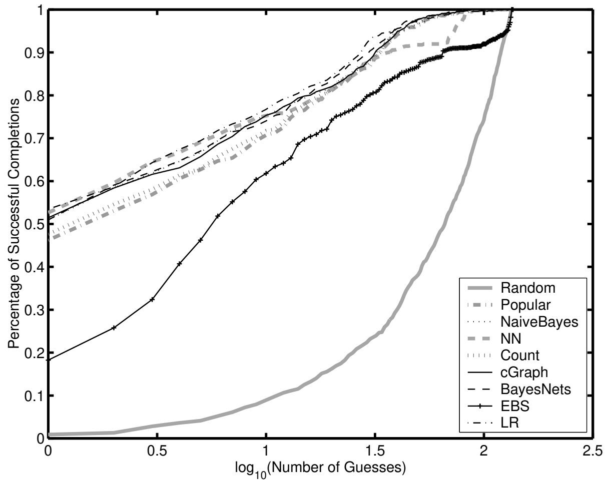

Figure 3: Result curves for the drinks data set.

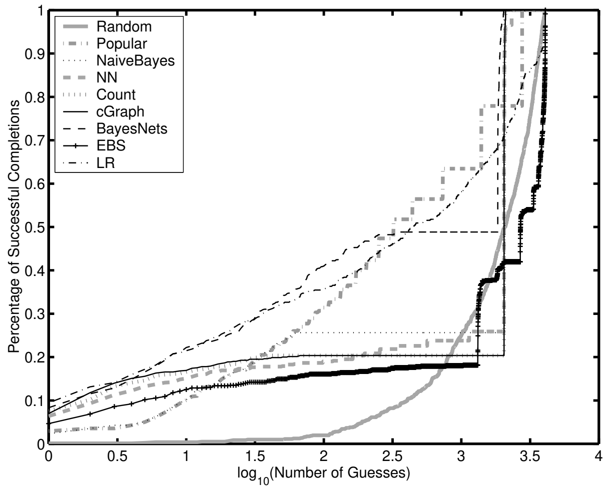

Figure 4: Result curves for the manual data set.

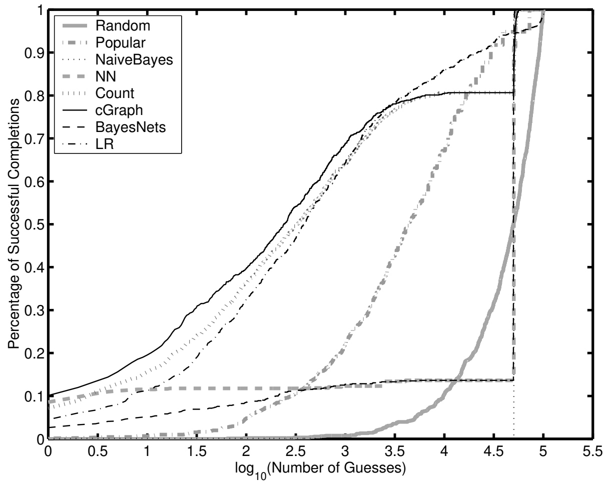

Figure 5: Result curves for the IMDB data set.

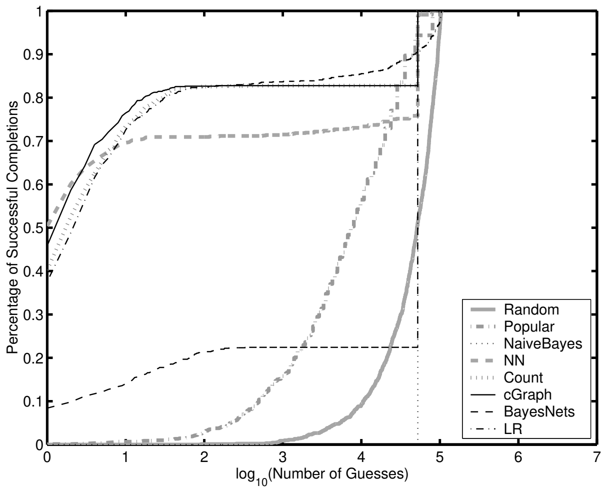

Figure 6: Result curves for the citeseer data set.

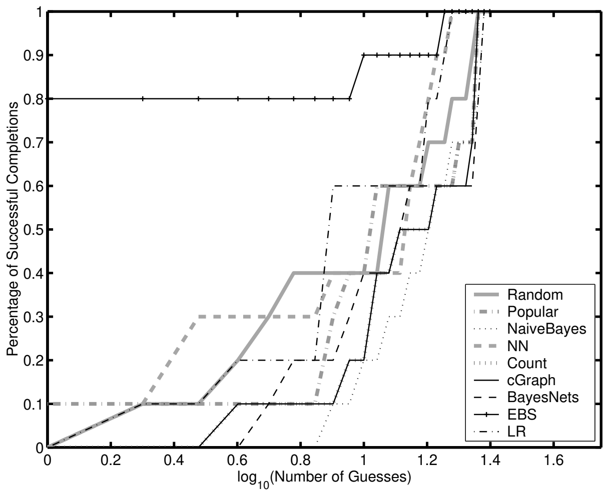

Figure 7: Result curves for the synthetic data set.

Table 3: Results on the institute data set

Algorithm	1	10	100	Ave. Pos.
Random	0.000	0.024	0.210	220.08
Counting	*0.524	*0.778	*0.956	16.81
Popular	0.113	0.266	0.669	85.94
Naive Bayes	0.161	0.302	0.706	81.17
NN	*0.673	*0.806	0.831	40.86
cGraph	*0.601	*0.871	*0.964	13.05
Bayes Nets	0.351	0.536	0.750	77.83
EBS	0.052	0.056	0.113	328.69
LR	0.375	0.565	0.766	83.61

Table 4: Results on the drinks data set

Algorithm	1	10	100	Ave. Pos.
Random	0.013	0.098	0.754	66.53
Counting	*0.546	*0.721	*1.000	10.04
Popular	*0.531	*0.713	*0.998	10.49
Naive Bayes	*0.585	*0.763	*0.999	9.51
NN	*0.596	*0.758	*1.000	11.76
cGraph	*0.584	*0.760	*1.000	9.31
Bayes Nets	*0.591	*0.753	*0.996	9.01
EBS	0.258	0.634	0.920	21.81
LR	*0.587	*0.781	*0.997	8.49

Table 5: Results on the manual data set

Algorithm	1	10	100	Ave. Pos.
Random	0.001	0.004	0.020	2068.49
Counting	*0.102	0.168	0.204	1634.91
Popular	0.036	0.109	0.314	939.02
Naive Bayes	0.036	0.112	0.256	1545.21
NN	*0.093	0.155	0.190	1569.54
cGraph	*0.118	0.174	0.204	1634.90
Bayes Nets	*0.115	*0.226	*0.411	982.56
EBS	*0.069	0.129	0.161	2459.59
LR	*0.131	*0.218	0.357	1216.75

Table 6: Results on the imdb data set

Algorithm	1	10	100	Ave. Pos.
Random	0.000	0.000	0.002	49900.88
Counting	*0.089	0.162	*0.363	10510.74
Popular	0.002	0.007	0.048	11393.75
Naive Bayes	0.001	0.001	0.001	50298.24
NN	0.098	*0.115	0.118	43715.08
cGraph	*0.120	*0.201	*0.399	10400.05
Bayes Nets	0.033	0.052	0.086	43203.38
EBS	n/a	n/a	n/a	n/a
LR	0.060	0.114	*0.326	7678.14

Table 7: Results on the citeseer data set

Algorithm	1	10	100	Ave. Pos.
Random	0.000	0.000	0.001	52158.88
Counting	0.526	*0.738	*0.825	9039.07
Popular	0.001	0.007	0.030	16304.57
Naive Bayes	0.000	0.000	0.000	52399.77
NN	*0.601	*0.697	0.709	13616.61
cGraph	*0.587	*0.769	*0.827	9037.53
Bayes Nets	0.098	0.146	0.214	40560.22
EBS	n/a	n/a	n/a	n/a
LR	0.488	*0.736	*0.824	9699.76

Table 8: Results on the synthetic data set

Algorithm	1	10	100	Ave. Pos.
Random	0.100	0.400	1.000	12.10
Counting	0.000	0.400	1.000	15.40
Popular	0.100	0.600	1.000	13.70
Naive Bayes	0.000	0.200	1.000	16.70
NN	0.100	0.400	1.000	11.10
cGraph	0.000	0.400	1.000	15.60
Bayes Nets	0.000	0.400	1.000	15.10
EBS	*0.800	0.900	1.000	3.60
LR	0.100	0.600	1.000	10.70