Collective Graph Identification

Galileo Mark Namata
Dept. of Computer Science
University of Maryland
College Park, MD 20742, USA
namatag@cs.umd.edu

Stanley Kok
Dept. of Computer Science
University of Maryland
College Park, MD 20742, USA
skok@cs.umd.edu

Lise Getoor
Dept. of Computer Science
University of Maryland
College Park, MD 20742, USA
getoor@cs.umd.edu

ABSTRACT

Data describing networks (communication networks, transaction networks, disease transmission networks, collaboration networks, etc.) is becoming increasingly ubiquitous. While this observational data is useful, it often only hints at the actual underlying social or technological structures which give rise to the interactions. For example, an email communication network provides useful insight but is not the same as the "real" social network among individuals. In this paper, we introduce the problem of *graph identification*, i.e., the discovery of the true graph structure underlying an observed network. We cast the problem as a probabilistic inference task, in which we must infer the nodes, edges, and node labels of a hidden graph, based on evidence provided by the observed network. This in turn corresponds to the problems of performing *entity resolution*, *link prediction*, and *node labeling* to infer the hidden graph. While each of these problems have been studied separately, they have never been considered together as a coherent task. We present a simple yet novel approach to address all three problems simultaneously. Our approach, called C^3, consists of Coupled Collective Classifiers that are iteratively applied to propagate information among solutions to the problems. We empirically demonstrate that C^3 is superior, in terms of both predictive accuracy and runtime, to state-of-the-art probabilistic approaches on three real-world problems.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithm, Design, Experimentation, Performance

Keywords

entity resolution, link prediction, collective classification, semi-supervised learning

KDD'11, August 21–24, 2011, San Diego, California, USA.

1. INTRODUCTION

In recent years, there has been a surge of interest in network analysis applied to diverse domains including social networks, technological networks, biological networks and more. In part, this interest is driven by the burgeoning growth in the amount of digital information describing network data that is available including e-mail, citation collections, epidemiological data, and social media. Such data contain a wealth of information (e.g., key individuals, communities, and contagion trends) that, when uncovered, can help to create better predictive models and help elucidate general laws governing network evolution. However, the available network data is typically noisy, observational, and, while it provides useful signal for uncovering the underlying sociological or technological network, it is not the *same* thing.

We define the process of discovering the hidden structure which gives rise to observational network data as the problem of *graph identification*. Figure 1 illustrates an example of inferring a social network (Figure 1(b)) from an email communication network (Figure 1(a)). We refer to the observational network data as the *input graph*, and the hidden network of interest as the *output graph*. Graph identification uncovers the hidden network by simultaneously solving three problems:

Entity resolution: merging nodes in the input that refer to the same entity, e.g., "Do `Neil Smith` and `N. Smith` refer to the same person?".

Link prediction: inferring the links between nodes in the output graph, often based on links in the input graph, e.g., "Does an employer-employee relationship exist between Anne and Robert?".

Node labeling: determining the label of nodes in the output graph, e.g., "Is Neil a CEO, manager or assistant?".

In addition to constructing the hidden output graph, graph identification involves constructing the mapping from nodes in the input graph to nodes in the output graph (Figure 1(c)).

Each task informs the others, and by solving them simultaneously, we allow information to propagate among them to obtain better solutions. For example, in a bibliographic domain, predicting whether one paper cites another (link prediction) allows us to determine whether two papers cite common papers. Co-citation helps us to decide whether they have the same topic (node labeling), which in turn aids in ascertaining whether they are the same paper (entity resolution). This last information in turn helps to determine the citation links from the two papers to other papers, closing the information propagation loop. While previous work [19,

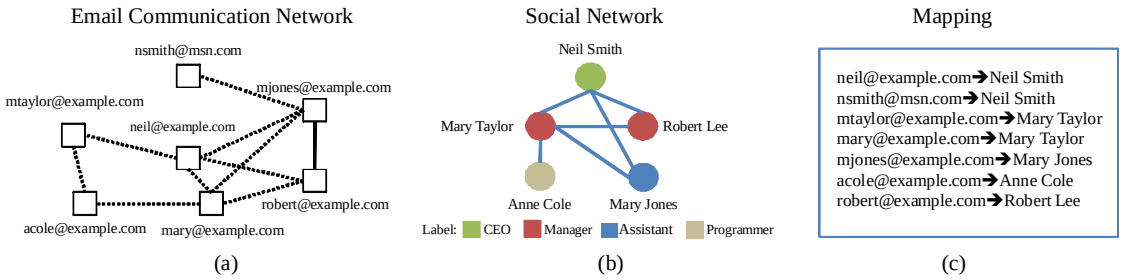

Figure 1: Input and output of graph identification. (a) Input graph representing a communication network where the nodes are email addresses and the edges are email communications. (b) Output graph representing the social network identified by graph identification. The nodes correspond to people and the edges to employee-manager relationships. The people are also labeled with their roles. (c) Mapping from input to output nodes.

13, 29, 3, 28, 2, 25] has addressed each of these tasks separately, to our knowledge, we are the first to efficiently address them simultaneously.

To address the problem of graph identification, we present the C^3 (Coupled Collective Classifiers) algorithm. C^3 defines a probabilistic model to capture the dependencies within each task as well as the relational interactions among all three. While it is conceptually possible for standard probabilistic inference algorithms to jointly solve all three tasks within the framework of our model, in practice they are too computationally expensive for large real-world datasets. C^3 uses an iterative procedure that simultaneously solves all three tasks. It begins by using a local classifier based solely on observed information in the input graph to solve each task independently. Then it iteratively propagates these solutions among the three tasks by means of relational features that capture the interactions within and among the tasks. Empirically, we observed that by propagating information, C^3 improved predictive accuracy by as much as 48%. To further tailor C^3 as a practical approach, we designed it to address the real-world scenario where it is costly to obtain fully labeled network data. C^3 adopts a *semi-supervised* learning algorithm that can exploit training data with only a small fraction of labeled examples.

The remainder of the paper is organized as follows. We begin with a background review in Section 2. Then we describe C^3 in detail (Section 3) and report our experiments (Section 4). Next we discuss related work (Section 5). Finally, we conclude with future work (Section 6).

2. BACKGROUND

Throughout the paper, we use an uppercase letter to represent a random variable (e.g., Y) and a lowercase letter (e.g., y) to represent its value. Bold letters represent a vector or set (e.g., $\mathbf{Y}$) and their values (e.g., $\mathbf{y}$).

A *Markov random field* (also known as *Markov network*) encodes a joint distribution over a set of random variables $\mathbf{Y}$. Let $\mathcal{C}$ denote a set of subsets (or cliques) of the random variables, and let $\mathbf{Y}_c$ denote the random variables in a subset c. For each $c \in \mathcal{C}$, we have an associated *potential* $\phi_c(\mathbf{Y}_c)$, which is a non-negative function defined over the joint domain of $\mathbf{Y}_c$. The Markov random field defines the following distribution:

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}_c) \tag{1}$$

where $Z = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}'_c)$ is a normalization constant. The potential functions are often represented more compactly as a log-linear combination over a set of features: $\phi_c(\mathbf{y}_c) = \exp\left(\sum_i w_i f_i(\mathbf{y}_c)\right) = \exp\left(\mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right)$. In this case, Equation 1 can be equivalently expressed as

$$P(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{c \in \mathcal{C}} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right). \tag{2}$$

In many applications, we are interested in conditional distributions where a subset of the variables $\mathbf{X}$ are provided as *evidence*, and we predict a set of *target* variables $\mathbf{Y}$. A *conditional Markov network* defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$, where the partition function $Z(\mathbf{x})$ now depends on $\mathbf{x}$: $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}'_c)$. Note that evaluating the above equation requires that we compute $Z(\mathbf{x})$, which in turn, requires that we sum over all possible assignments to $\mathbf{y}'$. Since this is exponential in $|\mathbf{Y}|$, computing $Z(\mathbf{x})$ and hence the equation are generally intractable.

A common approximation is the pseudolikelihood [1]:

$$\begin{aligned} P^*(\mathbf{y} \mid \mathbf{x}) &= \prod_i P(y_i | \mathbf{y}_{-i}, \mathbf{x}) \\ &= \prod_i \frac{\exp\left(\sum_{c \in \mathcal{C}: y_i \in \mathbf{y}_c} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right)}{Z(\mathbf{y}_c \setminus y_i, \mathbf{x})} \end{aligned} \tag{3}$$

where $\mathbf{y}_{-i} = y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m$ and $Z(\mathbf{y}_c \setminus y_i, \mathbf{x}) = \sum_{y_i} \exp\left(\sum_{c \in \mathcal{C}: y_i \in \mathbf{y}_c} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right)$. Note that we only sum over the possible values of y_i. Hence evaluating the normalization constants of all terms only requires time that is linear in $|\mathbf{Y}|$.

3. COUPLED COLLECTIVE CLASSIFIERS

C^3 takes a graph $(\mathbf{V}, \mathbf{E})$ as input where $\mathbf{V}$ and $\mathbf{E}$ are respectively a set of vertices and directed edges[1]. Each vertex

[1] C^3 extends straightforwardly to graphs with more than one

$v \in \mathbf{V}$ represents a reference to an entity, and each edge $(v_i, v_j) \in \mathbf{E}$ represents an interaction between references v_i and v_j. Each node v_i in the input graph has associated attributes A_i. For example, if a node represents a reference to a paper, the attributes may describe the words which appear in the paper. Edges (v_i, v_j) may also have associated attributes, denoted A_{ij}. An example of an edge attribute is the number of emails sent on a communication link from person v_i to v_j. We use $\mathbf{A} = \{A_i\} \cup \{A_{ij}\}$ where $i, j = 1, \ldots, |\mathbf{V}|$ to denote the attributes of all input nodes and edges.

C^3 jointly performs the three tasks of entity resolution, link prediction and node labeling. For entity **R**esolution, we define binary random variables $\mathbf{R} = \{R_{ij}\}$ where $i, j = 1, \ldots, |\mathbf{V}|$ and R_{ij} is an indicator variable denoting whether references V_i and V_j are co-referent. For **L**ink prediction, we define binary random variables $\mathbf{L} = \{L_{ij}\}$ where $i, j = 1, \ldots, |\mathbf{V}|$ and L_{ij} is an indicator variable denoting whether there is a link, or edge, from V_i to V_j, in the output graph[2]. For **N**ode labeling, we define random variables for each node representing its label $\mathbf{N} = \{N_i\}_{i=1}^{|\mathbf{V}|}$ and $N_i \in \{1, 2, \ldots, k\}$ where k is the number of possible label values.

We partition each set of variables into a set representing variables that are *observed* (i.e., evidence), and a set representing variables that are *predicted* (i.e., are target variables). We denote observed variables as $\mathbf{R}_o$, $\mathbf{L}_o$, and $\mathbf{N}_o$, and target variables as $\mathbf{R}_p$, $\mathbf{L}_p$, $\mathbf{N}_p$, where $\mathbf{R} = \mathbf{R}_o \cup \mathbf{R}_p$, $\mathbf{L} = \mathbf{L}_o \cup \mathbf{L}_p$ and $\mathbf{N} = \mathbf{N}_o \cup \mathbf{N}_p$. In addition, attributes $\mathbf{A}$ and edges $\mathbf{E}$ in the input graph are also assumed to be observed. Thus $\mathbf{R}_o$, $\mathbf{L}_o$, $\mathbf{N}_o$, $\mathbf{A}$ and $\mathbf{E}$ constitute evidence, i.e., $\mathbf{X} = \mathbf{R}_o \cup \mathbf{L}_o \cup \mathbf{N}_o \cup \mathbf{A} \cup \mathbf{E}$. The target variables $\mathbf{Y}$ are made up of the predicted variables, i.e., $\mathbf{Y} = \mathbf{R}_p \cup \mathbf{L}_p \cup \mathbf{N}_p$.

Given the above definitions and using Equation 3, we can represent the joint probability over the target variables $\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p$ given evidence $\mathbf{X}$ as follows:

$$P^*(\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p \mid \mathbf{x}) = \left(\prod_{r_p \in \mathbf{r}_p} P(r_p | \mathbf{y} \setminus r_p, \mathbf{x})\right)\left(\prod_{l_p \in \mathbf{l}_p} P(l_p | \mathbf{y} \setminus l_p, \mathbf{x})\right)\left(\prod_{n_p \in \mathbf{n}_p} P(n_p | \mathbf{y} \setminus n_p, \mathbf{x})\right). \quad (4)$$

3.1 Features

C^3 makes use of two kinds of features: *local* and *relational*. Local features capture the dependencies between a single predicted variable and evidence. For example, in a bibliographic domain, a local feature $f(N_i, A_i)$ represents how the topic N_i of a paper i depends on its content words A_i. Relational features capture the interaction between multiple predicted variables. We further differentiate between two kinds of relational features: *intra-relational* and *inter-relational*. Intra-relational features help in propagating information among variables of one task, whereas inter-relational features aid in disseminating information among variables of different tasks. For example, for node labeling the intra-relational feature $f(N_i, \{N_{ij}\}_{\forall j:(v_i,v_j)\in\mathbf{E}})$ represents the condition that the label of node N_i depends on the predicted label of its *observed* neighbors along edges $\mathbf{E}$ in the input graph, and the inter-relational feature $f(N_i, \{N_{ij}\}_{\forall j:L_{ij}=1})$ represents the condition that the label of node N_i depends on the predicted label of its *inferred* neighbors along edges L in the output graph. Similarly, for entity resolution, we may have an intra-relational feature $f(R_{ij}, \{R_{ik}, R_{jk}\}_{\forall k:R_{ik}=R_{jk}=1})$ representing the condition that nodes i and j are likely to be co-referent if they have a common neighbor k that they are predicted to be co-referent with. And we may have an inter-relational feature $f(R_{ij}, N_i, N_j)$ expressing the condition that nodes i and j are likely to be co-referent if their inferred node labels N_i and N_j are the same.

Note that a wide gamut of dependencies can be cast in terms of C^3's features. This is essential for graph identification because it allows us to exploit the diverse set of dependencies which have been proposed for each of the underlying tasks. Previous work in entity resolution, for example, has proposed using a variety of attribute similarity measures between potentially co-referent pairs of nodes [6]. Similarity measures have also been proposed to quantify the set similarity of "neighborhoods" of pairs of nodes [2]. Common definitions of a node's neighborhood include adjacent nodes, all nodes within a given shortest path distance, and all nodes which have an adjacent node in common (e.g., all papers which cite some common subset of papers). All of these definitions can be captured in our framework.

Work in link prediction also makes use of features based on attribute and neighborhood similarity. These features capture the assumption that many networks are homophilic, i.e., similar nodes are likely to share a link. Link prediction features also tend to rely on topology-based characteristics which capture the structural similarity (e.g., degree) or proximity (e.g., existence of paths) between two potentially adjacent nodes [13]. In multi-relational networks, link prediction may rely on features based on the attributes of links between the same pair of nodes (e.g., attributes of a communication edge between people imply something about their social relationship). For node labeling, features traditionally include the observed attributes of the given node, as well as observed and predicted values of nodes in its neighborhood. Table 1 contains more examples of local and relational features, and shows the diversity of features that we used in our experimental evaluation.

We use $\mathcal{F}$ to denote the set of features used by C^3, and $\mathbf{y}_f$ to denote the random variables used in the definition of features $f \in \mathcal{F}$. Then, from Equation 3 and Equation 4, we can represent the joint probability over the target variables $\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p$ as follows:

$$P^*(\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p \mid \mathbf{x}) = \prod_{y \in \mathbf{r}_p \cup \mathbf{l}_p \cup \mathbf{n}_p} \frac{\exp\left(\sum_{f \in \mathcal{F}: y \in \mathbf{y}_f} w_f \cdot f(\mathbf{x}_f, \mathbf{y}_f)\right)}{Z(\mathbf{y}_f \setminus y, \mathbf{x})}. \quad (5)$$

3.2 Weight Learning

Observe that Equation 4 decomposes into three terms, one for each of the $\mathbf{R}_p$, $\mathbf{L}_p$ and $\mathbf{N}_p$ target variables. A feature that is defined over more than one type of variable appears in more than one of the terms with the same weight (e.g.,$f(\mathbf{R}_p, \mathbf{L}_p, \mathbf{x})$ appears in both $\prod_{r_p \in \mathbf{r}_p} P(r_p | \mathbf{y} \setminus r_p, \mathbf{x})$ and $\prod_{l_p \in \mathbf{l}_p} P(l_p | \mathbf{y} \setminus l_p, \mathbf{x})$). We simplify the equation further by assuming that the appearances of such a feature in a term

kind of edge and hypergraphs. We focus on the case of a single edge type for simplicity of presentation.

[2]In practice, we do not instantiate all the $|\mathbf{V}|^2$ variables in $\mathbf{R}$ and $\mathbf{L}$. Section 4 describes how we use filtering techniques to only create variables for pairs that have some possibility of being co-referent/linked.

Table 1: Features used for entity resolution (ER), link prediction (LP), and node labeling (NL) for CORA, CITESEER, and ENRON. The evidence and inferred variables used in each features are in italics. We define two nodes V_i and V_j as *transitively co-referent* if there exists a co-reference path along *observed* and *predicted co-reference* edges between V_i and V_j.

Task	Type	Feature Description
		CORA/CITESEER Citation Network
ER	Local	· Jaccard similarity of *observed words* over nodes
		· Jaccard similarity of the set of nodes adjacent via *observed edges*
	Intra-Rel.	· Jaccard similarity of the set of nodes adjacent via *observed edges* to *observed* and *predicted transitively co-referent nodes*
		· Indicator for whether or not there is a co-reference path along *observed* and *predicted co-reference* edges between nodes
	Inter-Rel.	· Jaccard similarity of the set of nodes adjacent via *observed* and *predicted citation edges*
		· Jaccard similarity of the set of nodes adjacent via *observed* and *predicted citation edges* to *observed* and *predicted transitively co-referent nodes*
		· For each possible label value pair, an indicator variable for the *observed* or *predicted labels* of the nodes
LP	Local	· Jaccard similarity of *observed words* over nodes
		· Indicator variable of matches of *observed words* at both nodes
	Intra-Rel.	· Indicator variable for the existence of nodes adjacent to both nodes via *observed edges*
		· Indicator variable for the existence of nodes adjacent to both nodes via *observed* and *predicted citation edges*
	Inter-Rel.	· For each possible label value pair, an indicator variable for the *observed* or *predicted labels* of the nodes
		· Transitive existence of edge due to *observed* and *predicted citation edges* in *observed* and *predicted transitively co-referent nodes*
NL	Local	· *Observed words* of node
	Intra-Rel.	· For each possible label value, the % of nodes adjacent via *observed edges* with this *observed* and *predicted label*
	Inter-Rel.	· For each possible label value, the % of nodes adjacent via *observed* and *predicted citation edges* with this *observed* and *predicted label*
		· For each possible label value, the % of nodes which are *observed* and *predicted transitively co-referent* with this *observed* and *predicted label*
		ENRON Email Communication and Social Network
ER	Local	· String similarity of *observed email addresses*
		· Percent similarity of *observed word usage*
	Intra-Rel.	· Indicator for whether or not there is a co-reference path along *observed* and *predicted co-reference* edges between nodes
		· Jaccard similarity of the nodes adjacent via *observed communication edges*
		· Jaccard similarity of the nodes adjacent to *observed* and *predicted transitively co-referent nodes* via *observed communication* edges
	Inter-Rel.	· For each possible label value pair, an indicator variable for the *observed* or *predicted labels* of the nodes
		· Jaccard similarity of the nodes adjacent to *observed* and *predicted transitively co-referent nodes* via *observed* and *predicted managerial edges*
LP	Local	· Indicator variable of *observed words* in shared communications
		· Number of *observed communications* sent or received
	Intra-Rel.	· Indicator variable of *observed* and *predicted managerial edges* between nodes adjacent via *observed incoming and/or outgoing communication edges*
	Inter-Rel.	· For each possible label value pair, an indicator variable for the *observed* or *predicted labels* of the nodes
		· Transitive existence of edge due to *observed* and *predicted managerial edges* in *observed* and *predicted transitively co-referent nodes*
NL	Local	· *Observed words* in communications
		· Number of *observed communications* sent and/or received
	Intra-Rel.	· For each label, the % of nodes adjacent via *observed incoming and/or outgoing communication edges* with this *observed* and *predicted label*
		· For each label, the % of *observed communications* with nodes adjacent via *observed incoming and/or outgoing communication edges* with this *observed* and *predicted label*
	Inter-Rel.	· For each label, the % of nodes adjacent via *observed* and *predicted managerial edges* with this *observed* and *predicted label*
		· For each label, the % of *observed* and *predicted transitively co-referent nodes* with this *observed* and *predicted label*

are distinct from those in another term, thus allowing the weights of the feature to be different. This simplifies the weight learning algorithm by allowing it to find the optimal weights for each term separately.

In C^3, we are interested in inferring the most likely assignment of the variables (also known as the *maximum a posteriori* state). Hence, for each term $\prod_{v \in \mathbf{v}} P(v|\mathbf{y} \setminus v, \mathbf{x})$ ($\mathbf{v} \in \{\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p\}$), we want to find feature weights that maximize the ratio $\frac{P(v|\mathbf{y} \setminus v, \mathbf{x})}{P(v'|\mathbf{y} \setminus v', \mathbf{x})}$ between the conditional probability of each correct assignment v and every incorrect assign-

ment v'. Taking logs of the ratio, we see that we are equivalently maximizing the *margins* $\sum_{f \in \mathcal{F}: v \in \mathbf{y}_f} w_f \cdot (f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v) - f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v'))$ for each $v \in \mathbf{v}$ and $v' \neq v$, i.e.,

$$\begin{array}{c} \text{maximize } \gamma \quad \text{s.t. } \sum_{f \in \mathcal{F}} w_f^2 \leq 1 \text{ and} \\ \forall v \in \mathbf{v},\ \forall v' \neq v \ \ \Delta f_{(\mathbf{x}_f, \mathbf{y}_f)}(v, v') \geq \gamma \end{array}$$

where $\Delta f_{(\mathbf{x}_f, \mathbf{y}_f)}(v, v') = \sum_{f \in \mathcal{F}: v \in \mathbf{y}_f} w_f \cdot (f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v) - f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v'))$.

Applying a standard transformation to eliminate γ and introducing slack variables ξ_v to allow some constraints to be violated to accommodate non-linearly-separable data, we get:

$$\begin{array}{c} \text{minimize } \frac{1}{2} \sum_{f \in \mathcal{F}} w_f^2 \ \ + \ \ K \sum_{v \in \mathbf{v}} \xi_v \quad \text{s.t.} \\ \forall v \in \mathbf{v},\ \forall v' \neq v \ \ \Delta f_{(\mathbf{x}_f, \mathbf{y}_f)}(v, v') \geq 1 - \xi_v \end{array}$$

where K is a constant. The above is precisely the optimization that a multi-class support vector machine (SVM) [7] performs[3]. Hence we train three SVMs, one for each of the $\mathbf{R}_p$, $\mathbf{L}_p$, and $\mathbf{N}_p$ variables.

Thus far, we have assumed that training data is fully observed, i.e., we know the ground truth values of all $\mathbf{R}$, $\mathbf{L}$, and $\mathbf{N}$ variables. For large real-world networks, this is an impractical assumption because the ground truth values are seldom readily available and it is too costly to manually label them. Hence, we focus on the more realistic scenario of *semi-supervised* learning where only a small portion of the variables are observed.

One difficulty with partially observed data is that we cannot compute the values of relational features containing unlabeled variables. We solve this problem by first using the observed variables to train a new set of SVMs containing only local features, one SVM for each of the $\mathbf{R}_p$, $\mathbf{L}_p$, and $\mathbf{N}_p$ variables. Next these are used to infer the values of the target variables (recall that their values are not observed). With these inferred values, we can evaluate the relational features involving predicted variables, and hence learn feature weights that optimize the margins for the originally observed variables. Algorithm 1 contains the pseudocode for C^3's weight learning.

Even though we have derived the SVM optimization for C^3, we would like to emphasize that C^3 can easily be used with other classifiers (logistic regression, naive Bayes, etc.).

3.3 Inference

Algorithm 2 gives the pseudocode for C^3's inference procedure. Given a set of target variables $\mathbf{Y} = (\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p)$ and evidence $\mathbf{x}$, we begin by using a local SVM (i.e., one containing only local features) learned by Algorithm 1 to infer the values of each of the $\mathbf{R}_p, \mathbf{L}_p$, and $\mathbf{N}_p$ variables. At this point, the variable assignments are based solely on the evidence $\mathbf{x}$. The algorithm then proceeds to capture the dependencies between the variables. It iteratively evaluates the relational features using the variable values inferred in the previous iteration, and then infers new variable values for the current iteration. The algorithm terminates when the variable values converge or when a user-specified maximum number of iterations is reached.

Given an assignment of values to the predicted variables, we can construct an output graph. We create an entity node in the output graph for each collection of co-referent references. We also create edges between the entities based on whether the majority of their corresponding variables $\mathbf{L}$, defined over their references, indicate that the entities are linked. Finally, we can assign the label to an entity based on the values in $\mathbf{N}$ corresponding to its references. It is possible that assignments to these variables are inconsistent (i.e., N_i may not equal N_j even though references i and j are predicted co-referent). In these cases, we can define a procedure to resolve the inconsistencies prior to generating the output graph (e.g., enforce transitivity over co-referent pairs, add edges between entities whose references have an edge, and taking the mode label over the labels of its references). For the evaluation in this paper, we evaluate only over the predicted variables $\mathbf{Y}$; because we do not have appropriate ground truth to test against, we do not perform this additional set of steps.

[3]We use a multi-class SVM rather than a binary-class one because the node-labeling variables $\mathbf{N}$ can be assigned to one of more than two possible values.

Algorithm 1 C^3 Semi-supervised Weight Learning

input: $\mathbf{f}^{local}$, a set of local features
$\mathbf{f}^{relational}$, a set of relational features
$\mathbf{y}^{observed}$, values of observed variables
$\mathbf{Y}^{predicted}$, predicted variables
$\mathbf{x}$, evidence variables
output: $\mathbf{w}$, weights of $\mathbf{f}^{local} \cup \mathbf{f}^{relational}$
$\mathbf{w}^{local}$, weights of $\mathbf{f}^{local}$
calls: $LearnWeights(\mathbf{f}, \mathbf{y}, \mathbf{x}, C)$, which returns weights of features $\mathbf{f}$ given observed variables $\mathbf{y}$, evidence $\mathbf{x}$ and classifier C
$InferValue(Y, \mathbf{f}, \mathbf{w}, \mathbf{x}, C)$, which returns the MAP value of variable Y given features $\mathbf{f}$, their weights $\mathbf{w}$, evidence $\mathbf{x}$ and classifier C

$\mathbf{w}^{local} \leftarrow LearnWeights(\mathbf{f}^{local}, \mathbf{y}^{observed}, \mathbf{x}, \text{SVM})$
for each $Y \in \mathbf{Y}^{predicted}$
  $y^{predicted} \leftarrow InferValue(Y, \mathbf{f}^{local}, \mathbf{w}^{local}, \mathbf{x}, \text{SVM})$
$\mathbf{f} \leftarrow \mathbf{f}^{local} \cup \mathbf{f}^{relational}$
$\mathbf{w} \leftarrow LearnWeights(\mathbf{f}, \mathbf{y}^{observed}, \mathbf{x} \cup \mathbf{y}^{predicted}, \text{SVM})$
return $(\mathbf{w}, \mathbf{w}^{local})$

Algorithm 2 C^3 Inference

input: $\mathbf{Y}$, target variables
$\mathbf{x}$, evidence
$\mathbf{f}$, a set of local and relational features
$\mathbf{w}$, weights of features in $\mathbf{f}$
$\mathbf{f}^{local}$, a set of local features
$\mathbf{w}^{local}$, weights of features in $\mathbf{f}^{local}$
$maxIter$, maximum number of iterations
output: $\mathbf{y}$, values of target variables
calls: $InferValue(Y, \mathbf{f}, \mathbf{w}, \mathbf{x}, C)$, which returns the MAP value of variable Y given features , their weights $\mathbf{w}$, evidence $\mathbf{x}$, and classifier C

$i \leftarrow 0$
for each $Y \in \mathbf{Y}$
  $y^i \leftarrow InferValue(Y, \mathbf{f}^{local}, \mathbf{w}^{local}, \mathbf{x}, \text{ SVM})$
repeat
  $i \leftarrow i + 1$
  for each $Y \in \mathbf{Y}$
    $y^i \leftarrow InferValue(Y, \mathbf{f}, \mathbf{w}, \mathbf{x} \cup \{\mathbf{y}^{i-1} \setminus y^{i-1}\}, \text{ SVM})$
until $i = maxIter$ **or** $\mathbf{y}$ values converge
return $\mathbf{y}^i$

4. EXPERIMENTAL EVALUATION

We evaluate our approach using two types of networks: citations networks and email communication networks.[4]

Citation Networks We evaluate on two datasets, CORA and CITESEER [25]. In a citation network, nodes represent papers and directed edges represent citations. The CORA network contains 2708 nodes with 5428 edges. The CITESEER network contains 3312 nodes with 4732 edges. The nodes of both networks also contain, after pruning, 500 binary attributes representing the presence of a word in a paper, as well as a label indicating the topic of a paper (7 possible labels in CORA and 6 in CITESEER). Because noisy versions of these networks are not readily available[5], we create noisy versions of these graphs (i.e., input graphs) which attempt to mimic the types of noise likely encountered during the extraction of a network from multiple sources.

We create an input network by first adding a "reference" paper for a paper entity each time that paper is cited. For each reference, we copy the words from the corresponding entity, but introduce noise, with probability η_{attr}, by replacing the observed word with a randomly chosen word that did not occur in that paper. Next, for the citation links between the entity papers, we create a citation edge between each each reference, and introduce noise by replacing a percentage of the edges, η_{edge}, chosen randomly, with random edges between previously unconnected input nodes. These edges simulate the edges that may be encountered in a noisy extraction process. In our experiments, we used settings of η_{attr} and η_{edge} at 0.2, 0.3, and 0.4 (denoted Low, Medium, and High Noise, respectively).

For entity resolution and link prediction, because the inferences are made over pairs of nodes, there are important scalability issues. If done naively, both entity resolution and link prediction require $O(|V|^2)$ predictions. Clearly this will be intractable for all but the smallest of graphs. In both tasks, a filtering step is often applied to limit the potential pairs that are considered [17, 29]. This is crucial for making the algorithm scalable, and has been shown to improve the accuracy of the predictions. The filtering step is referred to as *blocking* [8] or *canopies* [17]. Any method that can quickly identify the potential pairs while minimizing the false negatives can be used. In our setting, the blocking criterion for entity resolution filters potential pairs as nodes which have at least two nodes, adjacent via edges in the input graph, in common. For link prediction, the blocking criterion filters potential pairs as nodes which have an extracted edge between them. Note that while this substantially reduces the number of potential pairs, in our experiments there remain up to $120,000$ pairs for entity resolution and $34,000$ pairs for link prediction.

Email Communication Network The second type of network we evaluate over is a corporate communication and social network, based on the ENRON dataset [10]. The input graph is an email communication network where the nodes correspond to email addresses, directed edges represent emails sent from one email address to another, and edge attributes indicate the words used and the number of communications between those email addresses. The output graph is a social network where the nodes represent people, edges indicate a managerial relationships, and the node labels indicate people's titles. We also have annotations on which email addresses belong to the same person. The full network consists of 211 email address nodes with 2837 directed communication edges corresponding to 146 individuals with 5 job title labels and 139 managerial relationships among them. Candidate pairs for entity resolution are limited to pairs of email addresses which are at most a distance three away in the communication network. Similarly, candidate managerial relationships are limited to pairs of nodes which share a communication edge.

The evaluation for these networks is semi-supervised; we train on the observed part of the network and predict the remaining parts of the network. We varied the percentage of missing annotations over the reference labels for node labeling and the potential pairs for entity resolution and link prediction, evaluating at 25%, 50%, and 75% for CORA and CITESEER and 20%, 30%, and 40% for the much smaller ENRON network (denoted Low, Medium, and High, respectively). We construct five random samples for each setting (and each noise level for CORA and CITESEER) using stratified snowball sampling.

4.1 Evaluation

We evaluated C^3 on the three networks using the features defined in Table 1. We used the LibSVM [4] implementation and set $maxIter = 10$ in Algorithm 2. We evaluated entity resolution, link prediction, and node labeling performance using the average F1 performance over the predictions for the target variables $\mathbf{Y}$, defined in Section 3. We also examined the average training, testing, and overall runtime for each of the approaches by defining variants of C^3 which use different subsets of full set of features. In the first variant, LOCAL, we use only features based on the observed attributes of the nodes (i.e., words, email address string). This is equivalent to commonly used approaches for entity resolution, link prediction, and node labeling which make predictions independently and base predictions on only observed attributes [4, 6]. The second variant, INTRA, performs C^3 using only the relational features which capture the intra-dependencies of the predictions (dependencies on predictions of the same type). This variant allows us to study the relative impact of capturing the collective propagation among target variables of the same type. The INTRA variant is also representative of approaches which perform collective entity resolution, collective link prediction, and collective node labeling as separate, unrelated tasks [19, 28, 2].

We also compare against two popular approaches to performing inference involving multiple tasks: PIPELINE and Markov Logic Networks (MLN) [22]. The PIPELINE approach performs tasks one at a time and in a fixed order. At each stage of the PIPELINE, we perform collective inference for a particular task only, using a similar learning and inference procedure to C^3 for comparability, but with the intra-relational features for that task and the inter-relational features from tasks which occurred earlier. Consequently, while the intra-dependencies are captured at each stage, the flow of information in PIPELINE does not allow earlier stages to use the predictions of later stages. This baseline is sen-

[4]Additional information about the datasets, features, and settings used for these experiments are available from http://www.cs.umd.edu/projects/linqs/c3.

[5]We note that while there are annotations for entity resolution (e.g., [2, 27]), link prediction (e.g., [13]), and node labeling (e.g., [25]) available, we are unable to use them directly since they are over different subsets of the network.

sitive to ordering so we consider all possible orderings (six in total for the three tasks in graph identification). Due to space, we only present the performance of the best possible ordering (denoted PIPELINE*). The other approach we compare to, MLN, is a state-of-the-art joint inference model proposed by Richardson and Domingos [22]. For this comparison, we use an open source implementation of MLN called Alchemy[11].[6] Because dependencies in MLN are represented using first order logic, we define first order logic formulae to mimic features defined in Table 1. We explored various data representations and parameters for Alchemy, including the option to perform MAP or marginal inference, and present the results for the best performing combination in terms of both runtime and performance.

4.2 Results

We present the overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performances) over the multiple levels of noise and annotation in Table 2. The best performance for each set is indicated in bold. We also perform statistical significance tests, using a paired-t test with significance $> 95\%$, over the F1 values for all pairs of approaches. The results are summarized in Table 3. The table indicates the number of times one approach, shown in each row, significantly outperforms another approach, shown in the columns. In Table 4, we list the average training, testing, and overall runtimes for a representative subset of the CORA experiments. These experiments were run on a single server with dual Intel Xeon 2.66Ghz processors and 48GB of memory. All implementations are in Java except for Alchemy which is in C++.

Comparing the performances of all the approaches, we see that C^3 is overall the best performing. C^3 is the best performing in all but one of the columns and in that single instance, the difference is not statistically significant. Looking at Table 3, we see that C^3 significantly outperforms all the other approaches in most cases while there are no instances where C^3 does significantly worse than any other algorithm. Next, looking at the approaches which exploit varying subsets of the dependencies within and among the different tasks, we see that LOCAL has the worst performance, followed by INTRA, and PIPELINE*. The trend in performance is directly correlated with the amount of intra- and inter-dependencies used by each approach; the more intra- and inter-dependencies are exploited, the better the overall performance. In addition, while increasing the number of dependencies used in the models increases runtime, we see that the difference in runtime is minimal compared to the performance improvement.

Relating the performance of the INTRA and LOCAL approaches, we see that making use of the intra-dependencies can, by itself, significantly improve performance. This is consistent with previous work which looked at these tasks in isolation and shows the importance of exploiting these types of dependencies. Similarly, comparing the relative performance of the INTRA to the PIPELINE* and C^3 approaches, we find that further making use of the inter-dependence yields a comparable, if not larger, improvement in performance with little impact on overall runtime. While using the inter-dependencies in these tasks has not been widely studied, the results show the importance of these types of dependencies. Relative to the PIPELINE* approach, we found the best performing PIPELINE* to be a competitive baseline in both performance and runtime. We note, however, that there is a significant variance in the performance of PIPELINE* based on the ordering of the tasks. Successful application of the PIPELINE* requires the non-trivial task of identifying which ordering is optimal which we accomplish by evaluating all possible orderings (generally requiring six times more runtime than C^3). Our C^3 approach, on the other hand, requires no ordering yet still significantly outperforms even the best performing PIPELINE* in all but three, statistically insignificant, cases.

Comparing the performance of the two joint models, we find that C^3 significantly outperforms MLN performance while being an order of magnitude faster. We found that despite multiple attempts to optimize the MLN, the performance of MLN in our experiments remained relatively poor. One possibility for this is that there is insufficient training data for the MLN weight learning given the number of dependencies and features involved. We may also need to look at extensions of the basic MLN model [30, 9]. Understanding the causes of the poor MLN performance and addressing those issues is part of our future work. Our experience with MLN, however, highlights the challenge in efficiently and successfully modeling all the dependencies to jointly infer the tasks involved in graph identification, and the advantages of using a simpler approach based on collections of coupled classifiers.

5. RELATED WORK

Graph identification is related to domain-specific problems such as information extraction in natural language processing [24], network mapping in computer networks [26], and biological network inference in bioinformatics [16]. While graph identification may provide a unifying paradigm for these problems and others, there are some important differences as well. Information extraction traditionally infers structured output from unstructured text (e.g., newspaper articles, emails), while graph identification is specifically focused on inferring structured data (i.e., the output graph) from other structured data (i.e., the input graph, perhaps produced from a noisy information extraction process). Network mapping and biological network inference are also related to graph identification, but they are mainly concerned with inferring only network topology.

There is significant prior work exploring the components of graph identification individually; representatives include work on collective classification (which we refer to as *node labeling*) [15, 18, 25], link prediction [13, 29, 5], and entity resolution [28, 2, 31]. More recently, there is work that looks at various ways these tasks are inter-dependent and can be modeled jointly [29, 23, 32, 21]. To our knowledge, however, previous work has not formulated the complex structured prediction problem as interacting components in order to collectively infer a graph.

Our iterative approach is similar in spirit to the iterative classification algorithm (ICA) presented by Neville and Jensen [19] and the link-based classification work by Lu and Getoor [14]. It is also related to work on relational dependency networks (RDN) [20] and stacked inference [12], which also use local conditional classifiers. In a RDN, a joint probability distribution is estimated using a variant of Gibbs sampling over the predicted local conditional distributions.

[6] We had to modify Alchemy to improve its efficiency when grounding its large underlying Markov network.

Table 2: Overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performance) on the different models. Bold indicates the highest value in a given column. C^3 is the best performing in all but one statistically insignificant case.

		CITESEER			CORA			ENRON
% Unknown		Low Noise	Medium Noise	High Noise	Low Noise	Medium Noise	High Noise	
Low	MLN	0.512	0.449	0.382	0.435	0.367	0.329	0.098
	LOCAL	0.648	0.526	0.409	0.682	0.557	0.390	0.352
	INTRA	0.678	0.574	0.468	0.747	0.631	0.507	0.426
	PIPELINE*	0.705	0.612	0.505	0.779	0.676	0.565	0.451
	C^3	**0.713**	**0.621**	**0.519**	**0.790**	**0.688**	**0.577**	**0.465**
Medium	MLN	0.373	0.399	0.385	0.352	0.373	0.310	0.089
	LOCAL	0.698	0.592	0.482	0.735	0.626	0.466	0.379
	INTRA	0.736	0.642	0.545	0.801	0.703	0.581	0.420
	PIPELINE*	0.756	0.672	0.579	0.830	0.745	0.641	0.440
	C^3	**0.763**	**0.679**	**0.588**	**0.837**	**0.753**	**0.649**	**0.457**
High	MLN	0.202	0.177	0.168	0.180	0.182	0.177	0.077
	LOCAL	0.713	0.619	0.507	0.744	0.648	0.502	0.388
	INTRA	0.748	0.668	0.568	0.805	0.716	0.592	0.441
	PIPELINE*	0.764	0.692	0.599	0.829	0.756	0.653	**0.458**
	C^3	**0.768**	**0.695**	**0.602**	**0.836**	**0.761**	**0.656**	0.455

Table 3: Each row indicates the number of times the approach, in each row, significantly outperforms the average overall performance of the approaches in each column, over all three levels of noise and three levels of sampling (a maximum of 9 pairwise comparisons for CORA and CITESEER and a maximum of 3 for ENRON).

	CITESEER					CORA					ENRON				
	MLN	LOCAL	INTRA	PIPELINE*	C^3	MLN	LOCAL	INTRA	PIPELINE*	C^3	MLN	LOCAL	INTRA	PIPELINE*	C^3
MLN	-	0	0	0	0	-	0	0	0	0	-	0	0	0	0
LOCAL	9	-	0	0	0	9	-	0	0	0	3	-	0	0	0
INTRA	9	9	-	0	0	9	9	-	0	0	3	2	-	0	0
PIPELINE*	9	9	9	-	0	9	9	9	-	0	3	3	0	-	0
C^3	9	9	9	8	-	9	9	9	9	-	3	3	2	2	-

Table 4: Average training, testing, and overall runtimes (in minutes) for each model over a subset of the experiments on CORA. PIPELINE* does not include the runtime of other orderings that are needed to identify the optimal one.

	Train Time	Test Time	Overall Time
MLN	535.0	45.3	580.3
LOCAL	3.6	0.3	3.9
INTRA	11.3	10.9	22.2
PIPELINE*	12.6	12.7	25.3
C^3	14.5	13.3	27.9

In stacked inference, classifiers retrained at every iteration are "stacked" such that the output of one classifier is used to augment the feature space and serves as input to another. We explored both approaches and found that in the semi-supervised setting described here they did not perform as well as C^3. Also, previous work in these methods have mainly looked only at the problem of node labeling, using simple aggregations for relational features. C^3 is a generalization of these approaches which use coupled classifiers to perform multiple tasks simultaneously, as well as using a richer set of relational features including aggregate, set similarity, and path-based features.

6. CONCLUSION

Graph identification is an important emerging problem. As more observational data describing networks becomes available, the need to properly map from the observational data to the "true" underlying social, technical or biologic network of scientific interest grows in importance. Correctly identifying these networks from noisy data before they are further analyzed is of huge importance. Not only do the inferred networks prevent us from drawing erroneous conclusions, they expedite our analysis as they are often orders of magnitude smaller than the observed ones. The problem is extremely challenging, in terms of propagating information correctly, training the models appropriately, and evaluating the results. In this work, we have formulated this problem as a probabilistic inference problem, and shown how to combine the results of entity resolution, link prediction, and node labeling in a coherent manner. We developed C^3 which can capture the intra- and inter-relational dependencies and showed that it can achieve significant performance gains over existing approaches. There is much room for further exploration; for example applying graph identification to evolving networks, studying convergence and complexity properties, exploring the use of other algorithms and models for graph identification, and applying the algorithm to other types of network data. In this paper, we have shown that a simple and intuitive coupled collective classification approach can be effective in this complex, highly inter-dependent, prediction problem.

7. ACKNOWLEDGMENTS

This work was supported by NSF Grant # IIS-0746930 and AFRL contract # FA8750-10-C-0191.

8. REFERENCES

[1] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

[2] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1:1–36, 2007.

[3] V. R. Carvalho and W. W. Cohen. On the collective classification of email "speech acts". In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98, 2008.

[6] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI Workshop on Information Integration*, 2003.

[7] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-taylor, and B. Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:2001, 2001.

[8] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[9] T. Huynh and R. Mooney. Max-margin weight learning for markov logic networks. In W. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5781 of *Lecture Notes in Computer Science*, pages 564–579. Springer Berlin / Heidelberg, 2009.

[10] B. Klimt and Y. Yang. Introducing the enron corpus. In *Conference on Email and Anti-Spam*, 2004.

[11] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, and P. Domingos. The alchemy system for statistical relational ai. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2006.

[12] Z. Kou and W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[13] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the ACM Conference on Information and Knowledge Management*, 2003.

[14] Q. Lu and L. Getoor. Link-based classification using labeled and unlabeled data. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

[15] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

[16] S. Martin, D. Roe, and J.-L. Faulon. Predicting protein-protein interactions using signature products. *Bioinformatics*, 21:218–226, 2005.

[17] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2000.

[18] L. McDowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[19] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of the AAAI Workshop on Learning Statistical Models from Relational Data*, 2000.

[20] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

[21] H. Poon and P. Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[22] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[23] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Conference on Computational Natural Language*, 2004.

[24] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[25] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[26] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. *SIGCOMM Computer Communication Review*, 38(4):303–314, 2008.

[27] P. Singla and P. Domingos. Multi-relational record linkage. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2004.

[28] P. Singla and P. Domingos. Entity resolution with markov logic. *IEEE International Conference on Data Mining*, 21:572–582, 2006.

[29] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of the Conference on Neural Information Processing Systems*, 2003.

[30] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1106–1111, 2008.

[31] M. Wick, A. Culotta, K. Rohanimanesh, and A. McCallum. An entity-based model for coreference resolution. In *Proceedings of the SIAM International Conference on Data Mining*, 2009.

[32] M. L. Wick, K. Rohanimanesh, K. Schultz, and A. McCallum. A unified approach for schema matching, coreference and canonicalization. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2008.