Collective Graph Identification

GALILEO MARK NAMATA, University of Maryland, College Park
BEN LONDON, University of Maryland, College Park
LISE GETOOR, University of Maryland, College Park

Data describing networks—such as communication networks, transaction networks, disease transmission networks, collaboration networks, etc.—is becoming increasingly available. While observational data can be useful, it often only hints at the actual underlying process that governs interactions and attributes. For example, an email communication network provides insight into its users and their relationships, but is not the same as the "real" underlying social network. In this paper, we introduce the problem of *graph identification*, i.e., discovering the latent 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. This entails solving several canonical problems in network analysis: *entity resolution* (determining when two observations correspond to the same entity), *link prediction* (inferring the existence of links), and *node labeling* (inferring hidden attributes). While each of these subproblems has been well studied in isolation, here we consider them as a single, collective task. We present a simple, yet novel, approach to address all three subproblems simultaneously. Our approach, which we refer to as C^3, consists of a collection of Coupled Collective Classifiers that are applied iteratively to propagate inferred information among the subproblems. We consider variants of C^3 using different learning and inference techniques and empirically demonstrate that C^3 is superior, both in terms of predictive accuracy and running time, to state-of-the-art probabilistic approaches on four real problems.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms: Algorithm, Design, Experimentation, Performance

Additional Key Words and Phrases: entity resolution, link prediction, collective classification, semi-supervised learning

ACM Reference Format:
Namata, G., London, B., and Getoor, L. 2015. Collective Graph Identification. *ACM Trans. Knowl. Discov. Data.* V, N, Article XXXX (2015), 36 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In recent years, there has been a surge of interest in network analysis across a diverse set of domains, including social networks, technological networks, biological networks and more. In part, this interest is driven by the growing amount of available network data, such as e-mail, citation databases, epidemiological studies and social media. Network data contains a wealth of information (e.g., key individuals, communities and contagion trends) that, when uncovered, can bolster predictive models and elucidate general network dynamics. However, observed networks are typically noisy and, while they may provide useful signal for uncovering an underlying sociological or technological network, it is often very different from the true topology.

We define the process of discovering the latent structure that gives rise to the observed network as the problem of *graph identification*. Figure 1 illustrates an example of inferring an organizational network (b) from an email communication network (a). In this example, the nodes in the observed graph are the email addresses, the edges indicate a correspondence; both nodes and edges can be annotated with attributes (not shown), such as word counts, or properties of the sender or receiver. In the latent graph, the nodes are people (i.e., *entities*), the edges are manager-subordinate relationships, and a node label indicates a person's role in the organization. We refer to the observed network as the *input graph*, and the hidden network of interest as

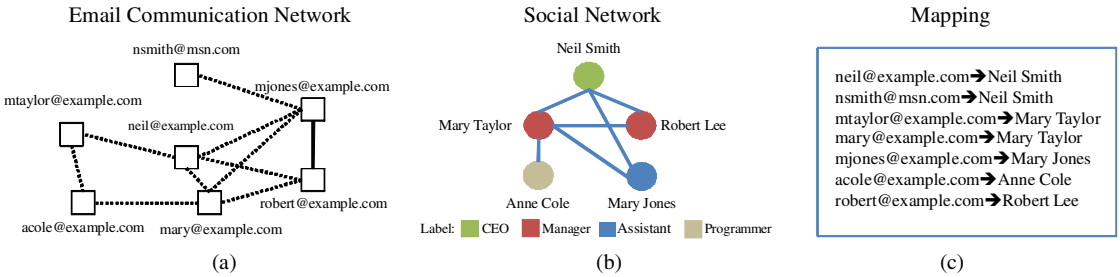

Fig. 1. Input and output of graph identification. (a) Input graph representing an email communication network in which nodes are email addresses and edges communications. (b) Output graph representing the organizational network discovered by graph identification. Nodes correspond to people and edges to manager-subordinate relationships. Nodes are colored according to their organizational roles. (c) Resolutions from input to output nodes.

the *output graph*. Graph identification uncovers the latent network by simultaneously solving three problems:

— **Entity resolution:** merging nodes in the input that refer to the same entity; e.g., "Do `nsmith@msn.com` and `neil@example.com` belong to the same person?"
— **Link prediction:** inferring links between nodes in the output graph, often based on links in the input graph; e.g., "Does a manager-subordinate relationship exist between Robert Lee and Mary Jones, given that they have corresponded?"
— **Node labeling:** labeling nodes in the output graph; e.g., "Is Neil Smith a CEO or assistant?"

The entity resolution step induces a mapping from the nodes in the input graph to those in the output graph, as illustrated in Figure 1(c).

Intuitively, each task can inform the others; this is the central hypothesis of graph identification. By solving these three subproblems collectively (i.e., jointly), reasoning can propagate among the subproblems, yielding a more accurate global solution. For example, in a bibliographic domain, predicting whether one paper cites another (i.e., link prediction) may allow us to infer that they have the same topic (i.e., node labeling), which in turn aids in determining whether they actually *are* the same paper (i.e., entity resolution); this last inference may in turn help to determine the citation links from these papers to other papers, closing the information propagation loop. While previous work [Neville and Jensen 2000; Liben-Nowell and Kleinberg 2003; Taskar et al. 2003; Carvalho and Cohen 2005; Singla and Domingos 2006; Bhattacharya and Getoor 2007; Sen et al. 2008] has addressed each of these tasks separately, or in subsets, to our knowledge, we present the first efficient, scalable solution to all subproblems simultaneously.

To address the problem of graph identification, we propose the C^3 (Coupled Collective Classifiers) algorithm. C^3 defines a global probabilistic model that captures the dependencies within and between the three subproblems. While it is conceivable that standard inference algorithms could jointly solve all three tasks, they do not work well in practice; due to the inherent complexity of the tasks, exact inference is prohibitively expensive, thus necessitating approximations. As we show in our experiments, these general-purpose approximations cannot match the performance of C^3, which is tailored to the problem of graph identification. C^3 uses the following iterative procedure: it begins with a round of *bootstrapping*, wherein a simple classifier is applied to each task independently, using only observed information from the input

graph. Then, it iteratively propagates the solutions among the three tasks, via relational features that model the various task interactions.

We consider multiple variants of C^3, based on different learning and inference paradigms, and empirically show that by propagating information, C^3 improves the predictive accuracy by as much as 114% on four real networks. In real data mining tasks, fully labeled training data is scarce and expensive. Recognizing this, we also present a *semi-supervised* variant of C^3, which is able to exploit training data that is only partially labeled. Finally, we highlight C^3's scalability and parallelizability on several large, synthetic network datasets.

The remainder of the paper is organized as follows. We begin with some motivating examples of the graph identification problem in various network domains in Section 2. We then discuss the subproblems that comprise graph identification, and review previous work in these problems in Section 3. Section 4 presents the necessary preliminaries and background, and Section 5 introduces the C^3 algorithm. We evaluate several variants of C^3 on a variety of real datasets, detailed in Section 6. We then report the results of these experiments in Section 7, and provide some discussion and suggestions for future work in Section 8.

2. MOTIVATING EXAMPLES

In this section, we discuss some motivating examples of graph identification problems. These were selected to illustrate the need and potential impact of C^3 across a diverse set of domains.

2.1. Organizational Networks

Recall our example from Section 1 of discovering an organizational network from an observed communications network. Specifically, the task is to identify the entities in the organization, determine the managerial relationships between these entities, and identify their roles. For large organizations, it may be difficult, if not impossible, to collect organizational data directly, yet the email communications may be available as a matter of public record [Klimt and Yang 2004]. These communications induce a network in which nodes represent email addresses and edges represent correspondences. Further, nodes and edges may be annotated with attributes, such as frequency of communication and linguistic content.

This observed network, however, may be a noisy proxy for the true network. To illustrate this, consider the small example networks shown in Figure 1. The nodes (email addresses) in the communication network do not accurately reflect the members of the organization. (For example, `mary@example.com` and `mtaylor@example.com` both belong to Mary Taylor). Moreover, the communication links are not the same as the managerial relationships between individuals. (Notice that `robert@example.com` corresponded with `mjones@example.com`, although Robert Lee and Mary Jones do not have a managerial relationship.) Finally, it may be that the employee roles are missing from the email data.

Although the observed network is noisy and incomplete, it can be used to infer the latent organizational network. This involves resolving email addresses to entities (these may be email addresses that have a similar writing style or communication pattern), infering managerial relationships (which may be strongly correlated with email correspondence), and identifying roles of individuals (reflected in the content of their communications and with whom they communicate).

2.2. Protein Networks

Protein-protein interaction (PPI) networks from high throughput experiments have become a widely studied source of data for understanding biological processes. These

networks have been used to explore various protein characteristics, such as their *essentiality* [Yu et al. 2007; Batada et al. 2007], *function* [Chua et al. 2006; Nabieva et al. 2005], and how they interact to perform higher level functions [Asthana et al. 2004; Bertin et al. 2007; Deng et al. 2002]. This line of research requires a complete protein network, in which the proteins are annotated with their function and mapped to their complexes, and all interactions are known. Unfortunately, these networks are notoriously noisy and incomplete. Comparisons of high-confidence networks involving well annotated species show as little as 9% overlap [Huang and Bader 2009]. Even well annotated species like yeast and worm are missing functional [Sharan et al. 2007] or complex annotations [Mewes et al. 2002]. Further, estimates for the presence of false and missing links in well studied organisms have been as high as 17% and and 51%, respectively [Huang and Bader 2009].

In spite of this, one can potentially infer a PPI network that is usable. For instance, the known annotations of one protein can be used to infer the annotations of another protein it interacts with [Nabieva et al. 2005; Chua et al. 2006]. Similarly, the autocorrelation between the function of interacting proteins, as well as attributes like cellular localization, can help determine the true set of interactions [Yu et al. 2006; Singh et al. 2006]. One can also use the topology of the PPI and the functional enrichment common in complexes to predict protein complexes [Asthana et al. 2004; Zhang et al. 2004].

2.3. Internet Topology

As Internet usage continues to grow, it is becoming essential that we understand the structure and design of the Internet, in order to understand its vulnerabilities and limitations [Spring et al. 2004b]. For example, we would like to have a map that shows all routers, information about these routers (such as model or geographic location), the administrative domain each router belongs to (known commonly as an autonomous system), and the existence and types of relationships between autonomous systems. The Internet, however, is not owned or managed by a single organization; instead, it is a collection of networks run by various Internet Server Providers (ISPs), who often do not publish the details of their networks. Generally, maps of the Internet are created using tools that provide only a partial view of the full network, and are notoriously error prone. For example, router level networks created using TTL-limited probes (i.e., traceroute) tend to inflate the true number of routers, and incorrectly record the existence of links between them [Sherwood et al. 2008].

As in the previous examples, while the available network map may be suspect, it can be used to infer a more reliable view. One can infer which IP addresses belong to the same router and, similarly, which routers belong to which autonomous system, using attributes (e.g., geographic location and DNS names) and observed or inferred connectivity. One can also infer the existence and types of relationships between autonomous systems by examining the attributes and connectivity of their constituent routers [Di Battista et al. 2007].

3. RELATED WORK

Graph identification consists of three subproblems, corresponding to the three primary components of the desired latent graph. First, nodes in the observed graph are resolved to nodes in the latent graph (i.e., entity resolution). Next, edges are inferred between nodes in the latent graph (i.e., link prediction). Finally, missing attributes or labels are predicted for nodes in the latent graph (node labeling). (Though we list these in a specific order, there is no fixed order of execution.) There is significant prior work on each of these tasks independently, which we review in this section. We also discuss approaches that solve some of these tasks jointly, by leveraging the interdependencies

between subproblems. To our knowledge, none of these approaches provide an efficient means of solving all three subproblems simultaneously.

3.1. Entity Resolution

Examples of entity resolution problems have been explored in many settings under a number of different names. Early work in entity resolution can be found in the field of medical record management under the name *record linkage* [Newcombe et al. 1959; Newcombe and Kennedy 1962; Fellegi and Sunter 1969; Winkler 1999]. In these early publications, the goal was to identify which medical records refer to the same individual or family. In computer vision, entity resolution was applied to identify which regions in the same image are part of the same object (referred to as the *correspondence problem*) [Ogale and Aloimonos 2005]. In natural language processing, there is interest in determining which noun phrases refer to the same underlying entity (*coreference resolution*) [Somasundaran et al. 2009; Mccallum and Wellner 2004]. The database tasks *deduplication* [Sarawagi and Bhamidipaty 2002] and *data integration* [Ullman 1997]—which determine when two tuples in or across databases refer to the same entity—are also instances of the general problem of entity resolution.

There are three general categories of approaches to entity resolution: local, relational, and collective. Local approaches are more traditional, relying solely on the attributes of the nodes. Given two nodes, attribute-based approaches generally make use of a similarity measure [Jaro 1995; Winkler 1999; Jaccard 1901] or a weighted combination of multiple similarity measures [Bilenko and Mooney 2003], over the attributes of the nodes. Pairs of nodes whose similarity is above some manually specified or learned threshold are predicted to resolve to the same entity. More recently, relational and collective approaches have been proposed which take the edges between these nodes into consideration. Relational approaches often use the attribute similarity of related nodes [Ananthakrishna et al. 2002; Kalashnikov et al. 2005], wherein related nodes with many similar attributes are more likely to be co-referent. Collective relational approaches take this one step further by taking into account that related nodes themselves may also need to be resolved, and that the entity resolution of a node needs to be applied jointly with the rest of the network [McCallum and Wellner 2003; Pasula et al. 2003; Dong et al. 2005; Singla and Domingos 2006; Bhattacharya and Getoor 2007].

A major issue in entity resolution is that it is a computationally difficult problem for large networks; a naïve algorithm is $O(|\mathbf{V}|^2)$ where $|\mathbf{V}|$ is the number of nodes in the network. For many networks, it is infeasible to compare all pairs of nodes for approaches which use expensive similarity measures. Similarly, for many probabilistic models, it is infeasible to explicitly represent all the variables required for inference. Thus, efficiencies have long been a focus of research in entity resolution. One technique, commonly known as *blocking* [Hernández and Stolfo 1995; Whang et al. 2009], adds a preprocessing step in which nodes are grouped into disjoint *blocks* using a cheap similarity or index-based computation; a computationally expensive entity resolution algorithm can then be run within each block, assuming the number of potential pairs in each block is sufficiently small. A related method, known as *canopies* [McCallum et al. 2000], places nodes into possibly overlapping subsets, rather than disjoint blocks. Potential co-referent pairs are then restricted to pairs of nodes that share at least one common canopy.

3.2. Link Prediction

Link prediction is a challenging problem that has been studied under various guises in different domains. For example, in social network analysis, there is work on predicting friendship links [Zheleva et al. 2008], event participation links (i.e., co-authorship

[O'Madadhain et al. 2005]), communication links (i.e., email [O'Madadhain et al. 2005]), links representing semantic relationships (i.e., advisor-of [Wang et al. 2010], and management relationships [Diehl et al. 2007]). In bioinformatics, there is interest in predicting the existence of edges representing physical protein-protein interactions [Szilagyi et al. 2005; Yu et al. 2006; Huang and Bader 2009], domain-domain interactions [Deng et al. 2002], and regulatory interactions [Réka et al. 2007]. Similarly, in computer network systems there is work on inferring unobserved connections between routers, as well as inferring relationships between autonomous systems and service providers [Spring et al. 2004b]. There is also work on using link prediction to improve recommender systems [Farrell et al. 2005; Huang et al. 2005], website navigation [Zhu 2003], surveillance [Huang and Lin 2008], and automatic document cross referencing [Milne and Witten 2008].

There are the two general categories of the current link prediction models: topology-based approaches and node attribute-based approaches. Topology-based approaches[Liben-Nowell and Kleinberg 2003; Yu et al. 2006; Clauset et al. 2008] typically rely on some notion of structural proximity, where nodes which are close are likely to share an edge (e.g., sharing common neighbors, nodes with a small shortest path distance between, etc.). Although topology has been shown useful in link prediction, topology-based approaches ignore an important source of information in networks: the attributes of nodes. Often there are correlations in the attributes of nodes which share an edge with each other. For example, individuals with common interests (e.g., sports, politics) are more likely to be friends than individuals with no interests in common. Also, in academic settings, an "advisor" edge can only exist between a student and a faculty node. Node attribute-based approaches [Getoor et al. 2003; Taskar et al. 2003; O'Madadhain et al. 2005; Richardson and Domingos 2006; Tang et al. 2011] use these correlations, often along with topology, in making its predictions.

A difficult challenge in link prediction is the large class skew between the number of edges which exist and the number of edges which do not. To illustrate, consider a directed graph denoted by $G(V, E)$. While the number of edges $|E|$ is often $O(|V|)$, the number of edges which do not exist is often $O(|V|^2)$ [Rattigan and Jensen 2005]. Consequently, the prior probability of edge existence is very small. This causes many supervised models, which naïvely optimize for accuracy, to learn a trivial model which always predicts that a link does not exist. A related problem in link prediction is the large number of edges whose existence must be considered. As with entity resolution, the number of potential pairs is $O(|\mathbf{V}|^2)$ where $|\mathbf{V}|$ is the number of nodes in the network. Applying complex inference models over such a large number of edges limits the size of the data sets which can be considered.

In practice, there are general approaches to addressing these issues either prior to or during the link prediction. With both large class skew and number of edges to contend with, the general approach is to make assumptions which reduce the number of edges to consider. One way to do this, similar to blocking in entity resolution, is to partition the nodes and only consider potential edges between nodes of the same partition; edges between partitions are not explicitly modeled and are assumed to not exist [Adafre and de Rijke 2005; Yu et al. 2006]. This is useful in domains where there is a natural partitioning criterion (e.g., geography in social networks, location of proteins in a cell) that makes edges across partitions unlikely. Another way is to define some simple, computationally inexpensive distance measure such that only edges whose nodes are within some distance are considered [Liben-Nowell and Kleinberg 2003; Diehl et al. 2007].

3.3. Node Labeling

A traditional problem in machine learning is to classify objects: e.g., given individuals in an organization classify each according to their roles; given proteins in an interaction network determine its biological function; given routers in a computer network determine its version; given a sentence, determine the part-of-speech for each word, etc. In networks, the problem of inferring labels has traditionally been applied to the nodes of the graph. Initial work in classification makes an independent and identically distributed (IID) assumption (i.e., the class labels are assumed conditional independent given object attributes). However, studies have shown that classifying nodes in a network can benefit from leveraging correlations between adjacent nodes [Chakrabarti et al. 1998; Neville and Jensen 2000; Getoor et al. 2001; Lafferty et al. 2001; Taskar et al. 2002; Lu and Getoor 2003; Macskassy and Provost 2007; Sen et al. 2008].

There are two main categories of collective classification algorithms, which vary based on their mathematical underpinnings, as well as how they exploit the relationships between the nodes. The first category, relational classifiers [Macskassy and Provost 2007], consider the *observed* attributes of related nodes. For instance, when classifying authors, we use the words present in their papers and the labels of the authors who they have co-authored with (if known) to arrive at the correct label. Although relational classifiers have been shown to perform well in some domains, overall, the results have been mixed. For instance, although there are reports that relational classification improves accuracy over traditional classification, in certain cases, relational classification can decrease accuracy [Chakrabarti et al. 1998]. The second category of algorithms go beyond the former by not only using the known attributes and labels of related nodes, but also using the predicted labels of other nodes whose labels are also unobserved [Chakrabarti et al. 1998; Neville and Jensen 2000; Getoor et al. 2001; Lafferty et al. 2001; Taskar et al. 2002; Lu and Getoor 2003].

3.4. Joint Inference

Most previous work explore these sub-problems of graph identification independently; yet, it is clear that inference from one task can inform another. Noting this, some attempts have been made to couple some of these tasks into a unified inference problem under various general statistical relational learning frameworks. One example is the work of Getoor et al. [2007] on Probabilistic Relational Models (PRMs). Their work explored using PRMs when there is both node label and link uncertainty. Similarly, there is work by Taskar et al. [2003] on Relational Markov Networks (RMNs). Taskar applied RMNs to the task of jointly inferring the labels and links between websites noting that certain relationships can only exists between nodes with a given label (e.g., an advisor relationship can only exist between a faculty and student node). More recently, Bhattacharya et al. [2008] proposed a generative model which jointly applies entity resolution and node labeling to movie data and Li et al. [2014] developed a "co-profiling" framework to predict node label and link type of various social networks.

There is also work in combining multiple inference problems in the computer vision and natural language processing literature. Roth et al. proposed frameworks for learning and applying multiple classifiers using a linear programming formulation [Roth and Yih 2004] and sequential learning [Roth et al. 2009]. Similarly, Heitz et al. [Heitz et al. 2008] proposed cascaded classification models for scene understanding. To our knowledge, previous work in joint models has not formulated the complex structured prediction problem in graph identification as interacting components that collectively infer the graph via a collection of probabilistic graph transformations.

Graph identification is related to the domain specific problems of information extraction in natural language processing [Poon and Domingos 2007; Sarawagi 2008;

Wick et al. 2008], network mapping in computer networks [Sherwood et al. 2008; Spring et al. 2004a,b], and biological network inference in bioinformatics [Martin et al. 2005]. While many of the underlying inferences are similar, the abstraction and tasks involved vary from graph identification. Information extraction traditionally infers structured output from unstructured data (e.g., newspaper articles, emails), while graph identification is specifically focused on inferring structured data (i.e., the cleaned graph) from other structured data (i.e., the noisy graph, perhaps produced from an information extraction process). Similarly, network mapping and biological network inference is mainly concerned with inferring network topology. More recently, there has been work on identifying latent relationship links between nodes, which can be used to improve collective classification performance [Tang and Liu 2009; Shi et al. 2011; McDowell and Aha 2013]. The links inferred here are not designed to have a semantic implementation, however, and are only inferred as an intermediate result toward the primary goal of improving collective classification. Consequently, work in these problems can be formulated as instantiations of the more general problem of graph identification.

This work is an extended version of our work in Namata et al. [2011] where we introduced the problem of graph identification and first proposed a collective approach to the problem. We incorporate this work into the current paper and further propose and evaluate four additional collective graph identification methods. We also provide an extensive evaluation on additional datasets and settings, exploring not only predictive performance, but also the convergence properties, applying graph construction procedures, and scalability results.

4. BACKGROUND

In this section, we introduce the notation and statistical tools needed to describe the C^3 model. Throughout the paper, we use uppercase symbols to denote random variables (e.g., Y) and a lowercase (e.g., y) to represent assignments. Bold indicates a vector or set of variables (e.g., $\mathbf{Y}$) or values (e.g., $\mathbf{y}$).

A *Markov random field* (also called a *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, where $\mathbf{Y}_c$ is the set of random variables in subset c. For each $c \in \mathcal{C}$, we have an associated *potential function* $\phi_c(\mathbf{Y}_c)$, which is a function mapping the domain of $\mathbf{Y}_c$ to a nonnegative real number. A Markov random field defines a 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 normalizing constant. The potential functions are often defined as a log-linear, weighted combinations of feature functions,

$$\phi_c(\mathbf{y}_c) = \exp\left(\sum_i w_i f_i(\mathbf{y}_c)\right) = \exp\left(\mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right). \tag{2}$$

As such, Equation 1 can be equivalently expressed as

$$P(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{c \in \mathcal{C}} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right). \tag{3}$$

For discriminative tasks, one is often interested in the modeling the conditional distribution, given an observed subset $\mathbf{X}$ (also called the *evidence* variables). We occasionally refer to the unconditioned variables $\mathbf{Y}$ as the *target* variables. A *conditional*

random field models the distribution,

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c), \tag{4}$$

where the partition function $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}'_c)$ now depends on $\mathbf{x}$.

Evaluating the likelihood necessitates computing $Z(\mathbf{x})$, which involves a sum over all possible assignments to $\mathbf{Y}$. Since this is exponential in the number of variables, $|\mathbf{Y}|$, this calculation is generally intractable. Because of this, it is common to approximate the likelihood using the pseudolikelihood [Besag 1975; Neville and Jensen 2007; Sutton and McCallum 2007],

$$\begin{aligned} P^*(\mathbf{y} \mid \mathbf{x}) &= \prod_i P(y_i | \mathbf{y}_{\setminus i}, \mathbf{x}) \\ &= \prod_i \frac{\exp\left(\sum_{c \in \mathcal{C}: y_i \in \mathbf{y}_c} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right)}{Z(\mathbf{y}_{c \setminus i}, \mathbf{x})} \end{aligned} \tag{5}$$

where $\mathbf{y}_{\setminus i} = \mathbf{y} \setminus y_i$, $\mathbf{y}_{c \setminus i} = \mathbf{y}_c \setminus y_i$ and

$$Z(\mathbf{y}_{c \setminus i}, \mathbf{x}) = \sum_{y'_i} \exp\left(\sum_{c \in \mathcal{C}: y_i \in \mathbf{y}_c} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{x}_c, \{\mathbf{y}_{c \setminus i} \cup y'_i\}) \right). \tag{6}$$

Note that $Z(\mathbf{y}_{c \setminus i}, \mathbf{x})$ only requires a sum over the possible values of Y_i; therefore, computing the pseudolikelihood is linear in $|\mathbf{Y}|$.

5. COUPLED COLLECTIVE CLASSIFIERS

Let $\mathbf{G}^{\mathcal{I}} = (\mathbf{V}, \mathbf{E})$ denote an input graph, where $\mathbf{V}$ and $\mathbf{E}$ are respectively a set of vertices and *directed* edges.[1] Each node $v \in \mathbf{V}$ represents a reference (i.e., mention) to an entity, and each edge $(v_i, v_j) \in \mathbf{E}$ represents an interaction between references v_i and v_j. Every v_i in the input graph has its associated attributes A_i; for example, if a node represents a reference to a publication, the attributes may describe its word distribution. Edges (v_i, v_j) may also have associated attributes, denoted A_{ij}; for instance, A_{ij} may represent the number of times v_i cited v_j. We use $\mathbf{A} = \{A_i\}_{v_i \in \mathbf{V}} \cup \{A_{ij}\}_{(v_i, v_j) \in \mathbf{E}}$ to denote the attributes of all input nodes and edges.

Recall that graph identification involves three tasks: entity resolution, link prediction and node labeling. To model entity resolution, we define binary random variables $\mathbf{R} = \{R_{ij}\}_{i,j=1}^{|\mathbf{V}|}$, where R_{ij} indicates whether references v_i and v_j should be resolved in the output graph. For link prediction, we define binary random variables $\mathbf{L} = \{L_{ij}\}_{i,j=1}^{|\mathbf{V}|}$, where L_{ij} indicates whether there is a link from v_i to v_j in the output graph.[2] For node labeling, we define a random variables $\mathbf{N} = \{N_i\}_{i=1}^{|\mathbf{V}|}$ representing the labels of each node, where k is the number of labels and N_i takes values in $\{1, 2, \ldots, k\}$.

We assume that certain subsets of $\mathbf{R}$, $\mathbf{L}$ and $\mathbf{N}$ are given as training data. We denote these observed (i.e., evidence) partitions by $\mathbf{R}_o \subset \mathbf{R}$, $\mathbf{L}_o \subset \mathbf{L}$ and $\mathbf{N}_o \subset \mathbf{N}$ respectively. The remaining variables, $\mathbf{R}_p = \mathbf{R} \setminus \mathbf{R}_o$, $\mathbf{L}_p = \mathbf{L} \setminus \mathbf{L}_o$ and $\mathbf{N}_p = \mathbf{N} \setminus \mathbf{N}_o$, are to be predicted

[1] For presentation, we have assumed that $\mathbf{G}$ is a simple graph, though C^3 easily extends to multimodal (hyper)graphs with more than one kind of edge.

[2] In practice, we do not instantiate all $|\mathbf{V}|^2$ variables in $\mathbf{R}$ and $\mathbf{L}$. Section 6 describes efficient filtering techniques to only instantiate variables for pairs that have are likely to be resolved/linked.

Table I. Example Features for a Publication Database

Task	Type	Feature Description
ER	Local	· Cosine similarity of *observed words* over nodes
	Intra-Rel.	· Jaccard similarity of the set of nodes adjacent via *observed edges*
		· Jaccard similarity of the set of nodes adjacent via *observed edges* to *observed* or *predicted co-referent nodes*
		· Indicator for whether or not a node exists that is *observed* or *predicted co-referent* to both 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 co-referent nodes*
		· Indicator for whether the *observed* or *predicted labels* of the nodes are the same
LP	Local	· Cosine 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.	· Indicator for whether the *observed* or *predicted labels* of the nodes are the same
		· Indicator for whether or not the nodes have *observed* or *predicted co-referent nodes* adjacent via *observed* or *predicted citation edges*
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 co-referent* with this *observed* and *predicted label*

(i.e., the target variables). Recall that all attributes $\mathbf{A}$ and edges $\mathbf{E}$ in the input graph are also observed.

Using this representation, we model graph identification using a conditional random field (Equation 4), wherein: $\mathbf{X} = \mathbf{R}_o \cup \mathbf{L}_o \cup \mathbf{N}_o \cup \mathbf{A} \cup \mathbf{E}$ denotes the observed resolutions, links, labels, attributes and edges; and $\mathbf{Y} = \mathbf{R}_p \cup \mathbf{L}_p \cup \mathbf{N}_p$ denotes the target resolutions, links and labels. Computing the normalizing constant for this model is roughly exponential in $|\mathbf{V}|^2$, making exact likelihood impractical. Thus, we use the conditional pseudolikelihood (Equation 5) instead, which factorizes neatly over the conditional distributions of the target variables as

$$\begin{aligned} P^*(\mathbf{y} \mid \mathbf{x}) &= P^*\left(\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p \mid \mathbf{r}_o, \mathbf{l}_o, \mathbf{n}_o, \mathbf{A}, \mathbf{E}\right) \\ &= \left(\prod_{r_p \in \mathbf{r}_p} P(r_p \mid \{\mathbf{y} \backslash r_p\}, \mathbf{x})\right) \left(\prod_{l_p \in \mathbf{l}_p} P(l_p \mid \{\mathbf{y} \backslash l_p\}, \mathbf{x})\right) \left(\prod_{n_p \in \mathbf{n}_p} P(n_p \mid \{\mathbf{y} \backslash n_p\}, \mathbf{x})\right). \quad (7) \end{aligned}$$

We have partitioned the product to highlight the conditional pseudolikelihood of each subproblem.

5.1. Features

C^3 makes use of two kinds of features: *local* and *relational*. Local features capture the dependencies between a single predicted variable and its associated evidence; for example, if v_i is a publication, a local feature $f(N_i, A_i)$ might model how its topic N_i depends on its word content A_i. Relational features capture interactions between multiple target variables, thus coupling the inferences on local predictions. We further differentiate between two kinds of relational features: *intra-relational* and *inter-relational*. Intra-relational features model interactions among variables of the same task, whereas inter-relational features propagate information be-

Table II. Example Features for an Organizational Network

Task	Type	Feature Description
ER	Local	· String similarity of *observed email addresses* · Cosine similarity of *observed word usage*
	Intra-Rel.	· Indicator for whether or not a node exists that is *observed* or *predicted co-referent* to both nodes · Jaccard similarity of the nodes adjacent via *observed communication edges* · Jaccard similarity of the nodes adjacent to *observed* and *predicted co-referent nodes* via *observed communication* edges
	Inter-Rel.	· Indicator for whether the *observed* or *predicted labels* of the nodes are the same · Jaccard similarity of the nodes adjacent via *observed* and *predicted managerial edges* · Jaccard similarity of the nodes adjacent to *observed* and *predicted co-referent nodes* via *observed* and *predicted managerial edges*
LP	Local	· Indicator variable of *observed words* in shared communications
	Intra-Rel.	· Indicator variable of *observed* and *predicted managerial edges* between nodes adjacent via *observed incoming and/or outgoing communication edges*
	Inter-Rel.	· Indicator for whether the *observed* or *predicted labels* of the nodes are the same · Indicator for whether or not the nodes have *observed* or *predicted co-referent nodes* adjacent via *observed* or *predicted managerial edges*
NL	Local	· *Observed words* in communications
	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 possible label value, the % of nodes which are *observed* and *predicted co-referent* with this *observed* and *predicted label*

Table III. Example Features for a Discourse Opinion Network

Task	Type	Feature Description
ER	Local	· Discourse and dialog continuity features defined in [Somasundaran et al. 2009]
	Intra-Rel.	· Indicator for whether or not a node exists that is *observed* or *predicted co-referent* to both nodes
	Inter-Rel.	· Indicator for whether or not the nodes are adjacent via *observed* and *predicted reinforcing edges* · Indicator for whether or not the nodes have *observed* or *predicted co-referent nodes* adjacent via *observed* and *predicted reinforcing edges* · Indicator for whether the *observed* or *predicted labels* of the nodes are the same
LP	Local	· Discourse and dialog continuity features defined in [Somasundaran et al. 2009]
	Intra-Rel.	· Indicator for whether or not a node exists that is *observed* or *predicted reinforcing* to both nodes
	Inter-Rel.	· Indicator for whether the *observed* or *predicted labels* of the nodes are the same · Indicator for whether or not the nodes are *observed* or *predicted coreferent* · Indicator for whether or not the nodes have *observed* or *predicted co-referent nodes* adjacent via *observed* and *predicted reinforcing edges*
NL	Local	· Opinion lexicon, dialog information, and unigram features defined in [Somasundaran et al. 2009]
	Intra-Rel.	· For each possible label value, the % of nodes adjacent via *observed co-occurence edges* with this *observed* and *predicted label*
	Inter-Rel.	· For each possible label value, the % of nodes adjacent via *observed* and *predicted reinforcing edges* with this *observed* and *predicted label* · For each possible label value, the % of nodes which are *observed* and *predicted co-referent* with this *observed* and *predicted label* · For each possible label value, the % of nodes which are *observed* and *predicted co-referent* and *reinforcing* with this *observed* and *predicted label*

tween variables of different tasks. For example, for node labeling, the intra-relational feature $f(N_i, \{N_j\}_{\forall j:(v_i,v_j)\in\mathbf{E}})$ might capture the intuition that the label of node N_i depends on the predicted label of its *observed* neighbors along edges $\mathbf{E}$ in the input graph; similarly, the inter-relational feature $f(N_i, \{N_j\}_{\forall j:L_{ij}=1})$ might indicate that the label of node N_i depends on the predicted label of its *inferred* neighbors along edges $\mathbf{L}$ in the output graph. 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 idea that nodes i and j are likely co-referent if they have a common neighbor k that they have each been resolved to; we may also have an inter-relational feature $f(R_{ij}, N_i, N_j)$ implying nodes i and j are likely co-referent if their inferred node labels N_i and N_j are the same.

A broad range of dependencies can be cast in terms of local and relational features. This is essential for graph identification because it allows us to exploit the diverse set of dependencies that have been proposed for each of the underlying subproblems. For example, previous work in entity resolution has proposed a variety of attribute similarity measures to detect co-referent pairs [Cohen et al. 2003]. Metrics have also been proposed to quantify the set similarity of node "neighborhoods," representing the intuition that co-referent nodes share common neighbors [Bhattacharya and Getoor 2007]. Common definitions of a node's neighborhood include adjacent nodes, all nodes within a given graph 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.

Prior work in link prediction also makes use of features based on attribute and neighborhood similarity. These features capture the assumption that many networks are *assortative*; 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 [Liben-Nowell and Kleinberg 2003]. In multi-relational networks, link prediction may rely on features derived from certain attributes of the various types of links; for example, the attributes of a communication edge between people may imply something about their social relationship.

Tables I, II and III provide more examples of local and relational features one can use for C^3 in a variety of domains. These represent the features used in our experimental evaluation in Section 6.1.

5.2. Inference

The goal of graph identification in the C^3 model is to maximize the likelihood of the output graph (target) variables, $\mathbf{Y}$, given the observed graph and attributes, $\mathbf{X}$. This form of inference is alternately called finding the *maximum a posteriori* (MAP) state, or *most probable explanation* (MPE). Recall that the true likelihood is intractable to compute; further, it is intractable to maximize. We therefore maximize the pseudolikelihood (Equation 7) instead, for which there are efficient algorithms.

C^3's inference procedure (given in Algorithm 1) is essentially block coordinate ascent on the pseudolikelihood. It begins by using a local classifier (i.e., one using only local features), parameterized by weights $\mathbf{w}^{\text{loc}}$, to infer the values of $\mathbf{R}_p$, $\mathbf{L}_p$, and $\mathbf{N}_p$. 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 uses a relational classifier (i.e., one containing local and relational features), with weights $\mathbf{w}$, to infer new variable values for the current iteration. The algorithm terminates when the variable values either converge, oscillate, or a user-specified maximum number of iterations is reached. Note that there is no guarantee that Algorithm 1 will

ALGORITHM 1: C^3 Inference

input: $\mathbf{Y}$, target variables
$\mathbf{x}$, evidence
$\mathbf{f}^{\text{loc}}$, local features
$\mathbf{f}$, all local and relational features
$\mathbf{w}^{\text{loc}}$, weights of features in $\mathbf{f}^{\text{loc}}$
$\mathbf{w}$, weights of features in $\mathbf{f}$
MAXITER, maximum number of iterations
output: $\mathbf{y}$, values of target variables
calls: INFER$(Y; \mathbf{x}, \mathbf{f}, \mathbf{w})$, which returns a value for variable Y given evidence $\mathbf{x}$, features $\mathbf{f}$ and weights $\mathbf{w}$.

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

converge, or if it does converge, that the solution is optimal. In practice, we find that the algorithm often converges quickly.

From the variational perspective of inference in probabilistic graphical models, Algorithm 1 is analogous to decoding the *mean field* approximation. Variational *marginal inference* (i.e., computing the marginal probabilities of cliques) is sometimes viewed as an optimization over the *marginal polytope* (i.e., set of realizable marginals)—which is convex, though requires an exponential number of constraints to describe. The mean field approximation replaces the marginal polytope with a tractable, though non-convex, inner bound, comprised of a polynomial number of local constraints. Decoding the mean field approximation finds a locally optimal assignment to the random variables, subject to the constraints. For more details about the mean field approximation, we refer the reader to Wainwright and Jordan [2008].

Algorithm 1 is also similar to the *iterative classification algorithm* (ICA) presented by Neville and Jensen [2000], or the link-based classification work by Lu and Getoor [2003]. These techniques were developed for the problem of *collective classification*, and typically used simple aggregations for relational features. C^3 can be considered a generalization of these approaches, since it couples multiple classifiers to perform multiple tasks simultaneously, using a richer set of relational features.

Given the predicted assignments to the target variables, we decode these to construct the output graph. First, an entity node is created for each collection of co-referent references. Because there may be more references (in the input graph) than entities (in the output graph), the link and label predictions made on the references must be aggregated for entities. Thus, edges are created between the entities based on whether the majority of their constituent link variables, $\mathbf{L}$ (defined over their constituent references) indicate that the entities are linked. It is possible that the labels assigned to references of an entity could be inconsistent; i.e., N_i may not equal N_j, despite the fact that references i and j are resolved. In these cases, we can define a procedure to resolve the inconsistencies prior to generating the output graph; one solution is to enforce transitivity over co-referent pairs, add edges between entities whose references have an edge, and use the majority value of $\mathbf{N}$ assigned to the constituent references.

ALGORITHM 2: C^3 Gibbs Sampling Inference

input: $\mathbf{Y}$, target variables
$\mathbf{x}$, evidence
$\mathbf{f}^{\text{loc}}$, local features
$\mathbf{f}$, all local and relational features
$\mathbf{w}^{\text{loc}}$, weights of features in $\mathbf{f}^{\text{loc}}$
$\mathbf{w}$, weights of features in $\mathbf{f}$
BURNIN, number of iterations for burn-in
NUMSAMPLES, number of iterations for sampling
output: $\mathbf{y}$, values of target variables
calls: SAMPLE$(Y; \mathbf{x}, \mathbf{f}, \mathbf{w})$, which samples a value for variable Y given evidence $\mathbf{x}$, features $\mathbf{f}$ and weights $\mathbf{w}$.

$i \leftarrow 0$
for each $Y \in \mathbf{Y}$
  $y^i \leftarrow$ SAMPLE$(Y; \mathbf{x}, \mathbf{f}^{\text{loc}}, \mathbf{w}^{\text{loc}})$
  Initialize sample counts, $c[Y, \cdot] = 0$
repeat
  $i \leftarrow i + 1$
  for each $Y \in \mathbf{Y}$
    $y^i \leftarrow$ SAMPLE$(Y; \mathbf{x} \cup \{\mathbf{y}^{i-1} \setminus y^{i-1}\}, \mathbf{f}, \mathbf{w})$
    if $i >$ BURNIN
      $c[Y, y^i] \leftarrow c[Y, y^i] + 1$
until $i =$ NUMSAMPLES $+$ BURNIN
for each $Y \in \mathbf{Y}$
  $y^i \leftarrow \arg\max_l \; c[Y, l]$
return $\mathbf{y}^i$

5.2.1. Using "Soft" Predictions. Thus far, we have assumed that the classifiers used in Algorithm 1 output a categorical class label. However, some multi-class predictors internally predict a real-valued score for each label, then output the label with the maximum score. We can alternately use the scores of the labels as "soft" predictions, instead of the "hard" decoding. In this subsection, we discuss several variants of Algorithm 1 based on this premise.

We will assume that the multi-class classifier outputs scores on the simplex (i.e., non-negative, summing to 1), which can be interpreted as a conditional distribution over labels [e.g., Wu et al. 2004]. Rather than using the highest scoring label (i.e., mode), we can use the following randomized algorithm: with probability $(1 - i/\text{MAXITER})$, where i is the current round of inference and MAXITER ≥ 1 is a predefined constant, we sample a label from the inferred label distribution; otherwise, we use the mode. In a variant we refer to as C^3-PS, we use this procedure for each call to INFER in Algorithm 1, and run for MAXITER iterations, without checking for convergence. By adjusting the sampling probability at each iteration, we are more likely to use the sampled value in early iterations (when predictions are more uncertain), and more likely to use the mode in later iterations.

Our second variant, C^3-GS (given in Algorithm 2), is inspired by *Gibbs sampling*. Gibbs sampling is a popular and well-studied method of approximating a distribution. In the interest of space, we refer the reader to Gilks et al. [1996] for an extensive review of Gibbs sampling theory, algorithms and practical concerns. It will suffice to say that Gibbs sampling is an iterative procedure that alternately samples from, then updates, its proposed distribution. In the case of C^3, sampling from the proposed distribution amounts to sampling a label for each node from its inferred label distribution, as output by a classifier conditioned on all other assignments; the distribution is updated when the sampled labels are "fed back" into the conditioned set. This process

requires a certain number of iterations of *burn-in* for the distribution to converge to a reasonably stable state. Therefore, we run the sampler for a predetermined number of iterations, discarding those from the burn-in phase; we then use the histogram of collected samples to estimate the label distribution of each target variable, and take the mode as the predicted label.

The label scores of a multi-class classifier can alternately be interpreted as *confidences*. Our third variant is based on the idea that collective performance is improved by gradually fixing the predictions based on their confidence. McDowell et al. [2009] showed that these "cautious" approaches can significantly improve performance over approaches that fix all of the predictions at once. In our cautious variant of C^3, denoted C^3-CI, we modify Algorithm 1 such that each iteration fixes the top $K = (|\mathbf{Y}_p|/\textsc{MaxIter})$ most confident predictions for each task. Clearly, after MAXITER iterations, all of the variables have been fixed to their predicted labels.

5.3. Weight Learning

In the previous subsection, we assumed that the parameters of the various classifiers are given. In this subsection, we propose an algorithm for learning the model parameters from training data. We consider two learning scenarios: *supervised* and *semi-supervised*. The latter scenario is more realistic for graph identification, but the former will be instructive for introducing the learning objective. We therefore begin with a discussion of supervised learning, then extend this to the semi-supervised setting.

5.3.1. Supervised Learning. For the following, we assume access to a set of labeled examples, wherein each example consists of a fully observed input and output graph. This is commonly referred to as *supervised* learning. Recall that Equation 7 decomposes into three terms; one for each subproblem, and its constituent variables, $\mathbf{R}$, $\mathbf{L}$ and $\mathbf{N}$. An inter-relational feature (i.e., defined across tasks) appears in more than one term with the same weight; e.g., $f(\mathbf{R}, \mathbf{L}, \mathbf{x})$ appears in both $\prod_{r \in \mathbf{r}} P(r \mid \{\mathbf{y} \setminus r\}, \mathbf{x})$ and $\prod_{l \in \mathbf{l}} P(l \mid \{\mathbf{y} \setminus l\}, \mathbf{x})$. We simplify Equation 7 further by copying these inter-relational features for each subproblem, thus allowing their associated weights to differ. This has the effect of decoupling the three learning subproblems, enabling efficient weight learning within each task. (Note that this does not decouple inference, since the copied features are deterministically dependent.)

To learn the weights for each subproblem, we propose a *max-margin* approach.[3] Recall that the predicted assignments to the target variables will be those that maximize the pseudolikelihood, $\prod_{v \in \mathbf{v}} P(v \mid \{\mathbf{y} \setminus v\}, \mathbf{x})$, for some $\mathbf{V} \in \{\mathbf{R}, \mathbf{L}, \mathbf{N}\}$. Therefore, to maximize predictive accuracy, we wish to find weights that maximize the pseudolikelihood ratio between the ground truth, $\hat{\mathbf{y}}$ (for the given task, $\hat{\mathbf{v}}$), and any other assignment; i.e., $\frac{P(\hat{v} | \{\hat{\mathbf{y}} \setminus \hat{v}\}, \mathbf{x})}{P(v | \{\hat{\mathbf{y}} \setminus \hat{v}\}, \mathbf{x})}$. Taking logarithm of this ratio, we see that we are equivalently maximizing the *margin*,

$$\Delta f_{(\mathbf{x}_f, \hat{\mathbf{y}}_f)}(\hat{v}, v) = \sum_{f \in \mathcal{F}: v \in \hat{\mathbf{y}}_f} w_f \cdot (f(\mathbf{x}_f, \{\hat{\mathbf{y}}_f \setminus \hat{v}\}, \hat{v}) - f(\mathbf{x}_f, \{\hat{\mathbf{y}}_f \setminus \hat{v}\}, v)), \tag{8}$$

[3]Though we present a max-margin formulation of C^3, we would like to emphasize that C^3 can easily be extended to other learning algorithms (e.g., logistic regression, naïve Bayes, etc.).

for each $\hat{v} \in \hat{\mathbf{v}}$ and $v \neq \hat{v}$. We can formally state this objective as

$$\begin{aligned} \max \quad & \gamma \\ \text{s.t.} \quad & \sum_{f \in \mathcal{F}} w_f^2 \leq 1, \\ & \forall \hat{v} \in \hat{\mathbf{v}},\, \forall v \neq \hat{v},\, \Delta f_{(\mathbf{x}_f, \hat{\mathbf{y}}_f)}(\hat{v}, v) \geq \gamma \end{aligned} \tag{9}$$

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:

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{f \in \mathcal{F}} w_f^2 \; + \; K \sum_{\hat{v} \in \hat{\mathbf{v}}} \xi_{\hat{v}} \\ \text{s.t.} \quad & \forall \hat{v} \in \hat{\mathbf{v}},\, \forall v \neq \hat{v},\, \Delta f_{(\mathbf{x}_f, \hat{\mathbf{y}}_f)}(\hat{v}, v) \geq 1 - \xi_{\hat{v}}, \end{aligned} \tag{10}$$

where $K \geq 0$ is a predefined constant. The above corresponds precisely to the optimization of a multi-class *support vector machine* (SVM) [Crammer et al. 2001].[4] Therefore, C^3 weight learning reduces to three SVM optimizations, one for each of the subproblems.

5.3.2. Semi-Supervised Learning. The training algorithm given in the previous subsection assumes access to a set of fully labeled training examples. However, for graph identification problems in practice, it is more likely that each example is only partially labeled, since obtaining ground truth is an expensive process. In fact, it is often the case that one must learn from a single, partially labeled network, then apply inference to the same network. This learning scenario is commonly referred to as *semi-supervised* (alternately, *transductive*, depending on one's assumption about how the data is generated). In this subsection, we propose learning algorithms for the semi-supervised setting. Our first algorithm is a simple, two-step approach; the second is an iterative algorithm that combines learning and inference.

The primary difficulty in learning from partially labeled graph identification data is initializing relational features of unlabeled target variables. Observe that we could train local (i.e., independent) classifiers for any subproblem using only the observed data. However, the relational features depend on both observed and inferred target variables. To solve the initialization problem, we start with a round of *bootstrapping*, wherein we train a local classifier for each task using only the observed local features (denoted $\mathbf{f}_{\text{loc}}$), then use the resulting classifier to predict the unobserved target variables. These predictions can then be used to compute the full set of features, which are needed to learn the weights of the full model. Though this approach is not entirely principled, it is justified by the mild assumption that the predictions are "close enough" to the ground truth, and provide some useful signal.

Algorithm 3 provides pseudocode for a two-step learning algorithm, based on the above method. In the first step, we bootstrap learning using only $\mathbf{f}_{\text{loc}}$, resulting in the local weights, $\mathbf{w}_{\text{loc}}$. These are used to predict the unobserved target variables, needed to compute the full feature set, $\mathbf{f}$. We then perform a full training step, using local, intra-relational and inter-relational features of both observed and predicted variables.

The second semi-supervised learning algorithm, C^3-EM (given in Algorithm 4), extends the first algorithm by performing alternating rounds of inference and weight optimization. This iteratively refines the predictions used to train the inter-relational features. At each iteration, the current weights are used to infer the unlabeled target

[4] We use multi-class SVM because the node-labeling problem is not binary.

ALGORITHM 3: C^3 Semi-supervised Weight Learning

input: $\mathbf{Y}_p$, target variables
$\mathbf{y}_o$, values of observed variables
$\mathbf{x}$, evidence variables
$\mathbf{f}_{\text{loc}}$, local features
$\mathbf{f}$, all local and relational features
output: $\mathbf{w}^{\text{loc}}$, weights of features in $\mathbf{f}^{\text{loc}}$
$\mathbf{w}$, weights of features in $\mathbf{f}$
calls: LEARN$(\mathbf{f}; \mathbf{y}, \mathbf{x})$, which returns weights of features $\mathbf{f}$ given labels $\mathbf{y}$ and evidence $\mathbf{x}$.
INFER$(Y; \mathbf{x}, \mathbf{f}, \mathbf{w})$, which returns a value for variable Y given evidence $\mathbf{x}$, features $\mathbf{f}$ and weights $\mathbf{w}$.

$\mathbf{w}^{\text{loc}} \leftarrow$ LEARN$(\mathbf{f}^{\text{loc}}; \mathbf{y}_o, \mathbf{x})$
for each $Y \in \mathbf{Y}_p$
  $y_p \leftarrow$ INFER$(Y; \mathbf{x}, \mathbf{f}^{\text{loc}}, \mathbf{w}^{\text{loc}})$
$\mathbf{f} \leftarrow \mathbf{f}^{\text{loc}} \cup \mathbf{f}^{\text{rel}}$
$\mathbf{w} \leftarrow$ LEARN$(\mathbf{f}; \mathbf{y}_o, \mathbf{x} \cup \mathbf{y}_p)$
return $(\mathbf{w}^{\text{loc}}, \mathbf{w})$

variables; then, the inferred values are used to re-optimize the weights. This is analogous to the *expectation-maximization* (EM) algorithm. The algorithm begins with the bootstrapping process defined above, to initialize the inter-relational features. It then iterates until the predictions converge, or a maximum number of iterations is reached. Since learning is coupled with inference, the output is the final set of weights and predictions.

ALGORITHM 4: C^3-EM Learning and Inference

input: $\mathbf{y}_o$, values of observed variables
$\mathbf{Y}_p$, target variables
$\mathbf{x}$, evidence variables
$\mathbf{f}^{\text{loc}}$, a set of local features
$\mathbf{f}^{\text{rel}}$, a set of relational features
MAXITER, maximum number of iterations
output: $\mathbf{w}$, feature weights
$\mathbf{y}$, values of target variables
calls: LEARN$(\mathbf{f}; \mathbf{y}, \mathbf{x})$, which returns weights of features $\mathbf{f}$ given labels $\mathbf{y}$ and evidence $\mathbf{x}$.
INFER$(Y; \mathbf{x}, \mathbf{f}, \mathbf{w})$, which returns a value for variable Y given evidence $\mathbf{x}$, features $\mathbf{f}$ and weights $\mathbf{w}$.

$\mathbf{w} \leftarrow$ LEARN$(\mathbf{f}^{\text{loc}}; \mathbf{y}_o, \mathbf{x})$
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  if $i = 1$, **then** $\mathbf{f} \leftarrow \mathbf{f}^{\text{loc}}$, **else** $\mathbf{f} \leftarrow \mathbf{f}^{\text{loc}} \cup \mathbf{f}^{\text{rel}}$
  for each $Y \in \mathbf{Y}_p$
    $y^i \leftarrow$ INFER$(Y; \mathbf{x} \cup \mathbf{y}^{i-1} \cup \mathbf{y}_o, \mathbf{f}, \mathbf{w})$
    $\mathbf{w} \leftarrow$ LEARN$(\mathbf{f}; \mathbf{y}_o, \mathbf{x} \cup \mathbf{y}^i \cup \mathbf{y}_o)$
until $i =$ MAXITER **or** $\mathbf{y}$ values converge
for each $Y \in \mathbf{Y}_p$
  $y \leftarrow$ INFER$(Y; \mathbf{x} \cup \mathbf{y}^i \cup \mathbf{y}_o, \mathbf{f}, \mathbf{w})$
return $\mathbf{w}$, y

Table IV. Overview of the Real World Datasets used in our experiments including the number of nodes and edges in the input and output graphs, as well as the number of labels in the output graph.

	Input Graph		Output Graph		
Dataset	Nodes	Edges	Nodes	Edges	Labels
CORA	5,915	34,466	2,708	5,428	7
CITESEER	5,664	29,100	3,312	4,732	6
ENRON	211	2,837	146	139	5
DISCOURSE	4,606	22,925	3,920	1,045	3

6. EXPERIMENTAL EVALUATION

In this section, we describe several experiments to evaluate the variants of C^3 given in Section 5. The results of these experiments are given in Section 7.

6.1. Datasets

Graph identification has a number of applications, including (but not limited to) publication databases, organizational networks and discourse opinion networks.[5] In this subsection, we describe several real datasets that fall into these categories (summarized in Table IV). We also describe a novel data generator to create synthetic networks, which we will use to evaluate the scalability of C^3.

6.1.1. Publication Database. A publication database maintains records for papers, which may include their content, topic and which papers they cite. The citations induce a graph, in which records are nodes, citations are (directed) edges and paper topic is a node label. We consider datasets from two such databases: CORA and CITESEER [Sen et al. 2008]. The CORA dataset contains 2,708 nodes and 5,428 edges. The CITESEER dataset contains 3,312 nodes and 4,732 edges. The nodes of both networks also contain, after pruning, 500 binary attributes representing the presence of a word in a paper. In CORA, each paper is labeled one of 7 categories; in CITESEER, there are 6 categories.

Because these datasets are carefully curated, the observed citation network closely matches the true network. However, the same cannot be said for the databases from which they came, since they are typically extracted from multiple noisy, possibly conflicting, sources. This motivates the need for graph identification. To simulate a noisy extraction process, we create noisy versions of CORA and CITESEER.[6]

We create an observed (i.e., input) network by adding a *reference* for each citation edge. For each reference, we copy the words from the corresponding paper entity, but introduce noise by replacing the observed word with a randomly chosen word that did not occur in that paper, with probability η_{attr}. Next, we create edges between references whose entities have a citation edge, and introduce noise by replacing a random η_{edge} proportion of the edges with random edges to previously unconnected references. In our experiments, we use noise rates 0.2 (low), 0.3 (medium), and 0.4 (high).

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, as discussed previously in Section 3.1 and Section 3.2, a filtering step is often applied to limit the potential pairs that are considered [McCallum et al. 2000; Taskar et al. 2003]. This is crucial for scalability, and

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

[6]While there are existing noisy versions of these datasets for entity resolution [e.g., Bhattacharya and Getoor 2007], link prediction [e.g., Liben-Nowell and Kleinberg 2003]) and node labeling [e.g., Sen et al. 2008] available, we are unable to use them directly, since they are over different subsets of the network.

has also been shown to improve the accuracy of the predictions. Any method that can quickly identify potential pairs while minimizing false negatives can be used. In our experiments, we use a blocking criterion for entity resolution that removes pairs that do not have at least two common neighbors in the input graph. For link prediction, the blocking criterion removes pairs that do not already have an edge between them in the input graph (meaning, we limit predicting new clean citation edges only between those with observed noisy edges). While this substantially reduces the number of potential pairs, in our experiments, this still left up to 120,000 pairs for entity resolution and 34,000 pairs for link prediction.

6.1.2. Organizational Network. The second type of network we consider is the corporate organizational and communication network detailed in Section 2.1. For this, we use the the ENRON dataset [Klimt and Yang 2004]. The communication (i.e., input) graph consists of 211 email address nodes and 2,837 directed communication edges; the organizational (i.e., output) graph contains 146 individuals, each with one of 5 job title labels, and 139 managerial relationships. For blocking entity resolution, candidate pairs of email addresses are limited those that are at most a graph distance 3 from each other in the communication network. For link prediction, candidate managerial relationships are limited to pairs that are connected in the communication network.

6.1.3. Discourse Opinion Network. We next consider discourse co-occurence and opinion reinforcement networks. For this, we use an annotated dataset from Somasundaran et al. [2009]. The input graph consists of a co-occurence network, wherein nodes are opinions in a discourse and edges exist between opinions that co-occur in the same portion of the discourse. Node attributes are engineered to capture discourse and dialogue continuity, opinion lexicons, dialog information and unigram features from the text. The output graph consists of opinion and object nodes. An object (e.g., a "remote control") is linked to the opinions that refer to it (e.g., the remote control may be linked with opinions such as "heavy" or "ergonomic"); opinions are linked to other opinions by reinforcement edges, indicating whether two opinions reinforce each other. Node labels indicate the polarity of each opinion as positive (e.g., "ergonomic"), negative (e.g., "heavy"), or neutral (e.g., "curved"). The full network consists of 4,606 opinion nodes, corresponding to 3,920 objects, with 22,925 co-occurence edges and 1,045 reinforcement edges. Candidate pairs for entity resolution and link prediction are limited to opinions that co-occur.

6.1.4. Synthetic Networks. To test the scalability of C^3, we developed a novel synthetic data generator that creates a noisy network, with ambiguous references that need to be merged to entities, missing edges which need to be predicted, and missing labels that need to be classified. The graphs and attributes created by this generator are modeled after the problem presented in Section 6.1.1, where the desired output graph is a clean citation network. Intuitively, the generator works by first creating a synthetic output graph that mimics the structure and attributes of real networks; then, it creates an observed graph by perturbing the output graph with various types of noise common to these networks.

The synthetic data generator begins by creating the structure of the output graph. A number of network generation models have been proposed for this; we implemented the widely used *forest fire* model [Leskovec et al. 2007], which models the heavy tailed degree distribution, "small world" phenomenon, and densification over time. We used a *forward burn probability* of 0.4 and a *backward burn probability* of 0.2. This creates the output graph nodes (paper nodes) and output graph edges (citation edges).

After generating the initial network structure, we add three sets of attributes to the nodes corresponding to the three types of inferences we will perform on the graph. The

Table V. C^3 Learning and Inference Variants

C^3 Variant	Weight Learning	Inference
C^3	Base Weight Learning (Algorithm 3)	Base Inference (Algorithm 1)
C^3-PS	Base Weight Learning (Algorithm 3)	Probabilistic Sampling Inference (Section 5.2.1)
C^3-CI	Base Weight Learning (Algorithm 3)	Cautious Inference (Section 5.2.1)
C^3-GS	Base Weight Learning (Algorithm 3)	Gibb Sampling (Algorithm 2)
C^3-EM	EM Learning (Algorithm 4)	EM Inference (Algorithm 4)

first set, used for node labeling, includes labels and attributes. We use the generator described in Rattigan et al. [2007] (5 labels, with 20% of the graph initially labeled randomly) to create the "topic" label, where "topic" has a high positive autocorrelation (i.e., papers which cite each other are likely to have the same topic). We then create 20 binary attributes based on those labels using the method described in Bilgic and Getoor [2008]. The second set of attributes, used for link prediction, consists of 20 attributes, generated according to the method described in Rattigan et al. [2007]. This method models the assortative hypothesis, that nodes with similar attributes are likely to share an edge. The last set of attributes, used for entity resolution, represent attributes that imply, non-uniquely, the entity it refers to (e.g., first author names non-uniquely imply a paper, since multiple papers may have the same first author name). To generate this attribute, we use the method described in Bhattacharya and Getoor [2007]. The resulting annotated network is the synthetic output graph.

We create an input graph from the output graph by removing (a subset of) the node labels and injecting noise, per the method described in Section 6.1.1. For the edge noise, we fix $\eta_{\text{edge}} = 1/2$ in all experiments.

6.2. Experimental Setup

The most natural setting for graph identification is semi-supervised, in which some of the true output graph is observed (that is, certain parts of the input graph can be trusted), and the rest must be inferred. We recreate this setting by revealing a certain proportion of the annotations for a noisy input graph. We experiment with varying percentages of annotations, depending on the dataset; for CORA, CITESEER and DISCOURSE, we use 25%, 50% and 75%; for ENRON, we use 20%, 30% and 40% (i.e.,). We construct five random annotation samples for each setting (and each noise level, for CORA and CITESEER) using stratified snowball sampling; the results are averaged over those five samples.

6.2.1. Variants of C^3 Tested. We apply the variants of C^3 described in Section 5 using the full set of features in Section 5.1 (i.e., Tables I, II and III). A summary of the variants, highlighting differences in their learning and inference procedures, is provided in Table V. In addition, we evaluate some specialized baseline variants to explore the impact of the inter and intra-dependencies. In the first variant, LOCAL, we use only features based on the observed node attributes (e.g., word content). These features are common in local (i.e., non-collective) entity resolution, link prediction, and node labeling [Chang and Lin 2001; Cohen et al. 2003], so LOCAL is a proxy for the standard classifier-based approach to these subproblems. The second variant, INTRA, uses only intra-relational features; i.e., features that capture dependencies within one task. This variant lets us study the relative impact of collective reasoning within each subproblem, yet treating each task independently [e.g., Neville and Jensen 2000; Singla and Domingos 2006; Bhattacharya and Getoor 2007].

6.2.2. Baselines. For comparison to existing methods, we also evaluate two baselines: PIPELINE and Markov logic networks (MLNs).

The PIPELINE method is the simplest, but arguably most common, approach used for problems that involve multiple tasks. In the PIPELINE approach, the graph identification tasks are performed sequentially, in a fixed order. At each stage of PIPELINE, collective inference is performed for a single task, using the same features as C^3, but using inter-relational features only from tasks that have already been completed. Consequently, PIPELINE propagates inference within each task, and from earlier stages to later stages, but not from later stages to earlier stages. Since PIPELINE is sensitive to ordering, we consider all six permutations of the three subproblems. To differentiate the results for each permutation, we postfix PIPELINE with an acronym corresponding to the order of tasks; e.g., a PIPELINE with ordering (entity resolution, link prediction, node labeling) is denoted PIPELINE-ELN. For succinctness, we will occasionally only present the performance of the best possible ordering, denoted PIPELINE*.

MLNs are a framework for probabilistic inference with first-order logic, proposed by Richardson and Domingos [2006]. In our experiments, we use an open source implementation of MLNs called Alchemy [Kok et al. 2006].[7] Because dependencies in MLN are defined in first-order logic, we define formulae to mimic features defined in Tables I–III. 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" (in terms of both running time and accuracy) performing combination.

To measure the accuracy of each subproblem, we use the average F1 score over the target variables. Due to blocking, there are a large number of pairs which none of the approaches explicitly predict and are implicitly inferred as not co-referent, or not linked. Thus, to isolate the differences between the various approaches, we compute F1 for entity resolution and link prediction only on the explicit predictions. We present scores for each subproblem and the average of all subproblems, for various levels of noise and annotation.

7. RESULTS AND DISCUSSIONS

In this section, we discuss the results of our experiments, examining accuracy and running time.

7.1. Prediction Quality

We begin by analyzing the predictive accuracy of the various graph identification algorithms. To separate inference from decoding, we evaluate accuracy before and after graph reconstruction (described at the end of Section 5.2). We identify trends in the overall prediction quality of all approaches, as well as for the individual tasks of entity resolution, link prediction, and node labeling. We then discuss the impact of applying the graph construction procedures, which may update some of the predicted values, in Section 7.1.4.

7.1.1. Accuracy. Table VI presents the overall F1 performance of all algorithms—reported as the average of the scores for the three subproblems—over multiple levels of noise, annotation. The best performance for each dataset is indicated in bold. We also present the relative improvement of each algorithm over the LOCAL baseline in Table VII. To verify statistical significance, we used a paired t-test with significance $> 95\%$. Table VIII summarizes the number of times one approach, shown in each row, significantly outperforms another approach, shown in the columns. We also present a representative subset of the individual subproblem performance in Table IX.

Looking at Table VIII, we find that C^3-EM has the best overall performance. Even when it is not the best, there are no instances in which it does significantly worse

[7]We had to modify Alchemy's code to improve the efficiency of *grounding* the Markov network.

than the best algorithm. Among the approaches that exploit varying subsets of the dependencies within and among tasks, LOCAL has the worst performance, followed by INTRA, and PIPELINE*. The trend in performance is directly correlated with the amount of intra- and inter-relational information used; more relational features leads to better the overall performance.

Comparing the performance of the two fully relational models, C^3 and MLN, we find that C^3 significantly outperforms MLN. Despite multiple attempts to optimize the performance of our MLN baseline, its performance remained relatively poor. One possible explanation is that there is insufficient training data for MLN weight learning given the number of dependencies and features involved. We may also need to look at extensions of the basic MLN model [Wang and Domingos 2008; Huynh and Mooney 2009]. Understanding the causes of the poor MLN performance and addressing these issues is part of our future work. Our experience with MLNs, however, highlights the challenge in efficiently and successfully modeling all of the complex dependencies involved in graph identification, and the relative advantages of using a simpler approach, like C^3.

7.1.2. Benefits of Exploiting Dependence. Comparing the INTRA and LOCAL approaches, we find that making use of the intra-dependencies can, by itself, significantly improve performance. This is consistent with previous work which explored these tasks in isolation. Similarly, comparing the relative performance INTRA to PIPELINE* and C^3, we find that further exploiting the inter-relational dependence yields a comparable, if not larger, improvement, with little impact on overall running time (exemplified in Table XIV). This illustrates the importance of these types of dependencies, which have not been widely studied.

We found the best performing PIPELINE* to be a surprisingly competitive baseline. However, comparing the per task performance (Table IX) and overall performance (Table VI) for the various PIPELINE orderings, there is a significant variance in the performance. Generally, the per-task performance of a task in PIPELINE is worst when occurring early in the ordering, where it is decoupled from the other tasks, and best when occurring later. Thus, while the best performing PIPELINE* may seem have competitive overall performance, it often does so at the expense of tasks earlier in the ordering. Furthermore, successful application of the PIPELINE* requires the nontrivial task of identifying which ordering is optimal for the overall performance. In our experiments, we use the brute-force approach wherein we simply evaluate all possible orderings; this may be infeasible in practice. On the other hand, the C^3 approach requires no ordering, and still significantly outperforms PIPELINE*.

7.1.3. Comparison of C^3 Variants. We now compare the C^3 variants. While the variants based on sampling, C^3-PS and C^3-GS occasionally show improvement over regular C^3, there is only one case in which the improvements are significant. In most cases, these variants actually result in significantly worse performance than C^3. The results are similar for the cautious variant, C^3-CI, which shows improvement in few cases, but generally insignificant. While this may be addressed by running more iterations, particularly for C^3-GS, this would be very costly, and these initial results do not indicate that it would yield significant improvement over standard C^3 inference.

On the other hand, in terms of overall F1 performance, C^3-EM generally outperforms C^3, significantly so in 12 cases. However, the improvement is not consistent among all datasets. While C^3-EM results in significant improvement over C^3 on over half of the cases for CORA and CITESEER, it provides only one case of significant improvement in ENRON and none in the DISCOURSE networks. While there are no cases where C^3-EM does significantly worse than C^3, the additional overhead of relearning

Table VI. Overall F1 performance—reported as the average of F1 scores for entity resolution, link prediction and node labeling—for various levels of annotation and noise. Bold indicates the highest value in a given column.

		Citeseer (Vary Noise Level)			Cora (Vary Noise Level)			Enron	Discourse
		Low	Medium	High	Low	Medium	High		
Low % Unknown	LOCAL	0.800	0.736	0.657	0.827	0.756	0.645	0.425	0.361
	INTRA	0.843	0.792	0.744	0.900	0.854	0.798	0.516	0.647
	PIPELINE*	0.872	0.834	0.793	0.937	0.911	0.878	**0.559**	0.706
	MLN	0.677	0.673	0.663	0.570	0.560	0.591	0.137	0.320
	C^3	0.882	0.853	0.819	0.950	0.928	0.898	0.550	0.729
	C^3-EM	0.882	**0.855**	**0.824**	**0.951**	0.927	**0.903**	0.549	**0.730**
	C^3-PS	0.881	0.851	0.817	0.947	0.923	0.891	0.536	0.679
	C^3-CI	**0.883**	0.853	0.820	**0.951**	**0.929**	0.899	0.550	0.729
	C^3-GS	0.881	0.852	0.817	0.877	0.830	0.784	0.459	0.624
Medium % Unknown	LOCAL	0.786	0.725	0.648	0.821	0.747	0.639	0.363	0.309
	INTRA	0.833	0.782	0.730	0.889	0.840	0.778	0.465	0.545
	PIPELINE*	0.853	0.816	0.767	0.921	0.888	0.850	0.509	0.604
	MLN	0.425	0.534	0.563	0.456	0.519	0.470	0.143	0.217
	C^3	0.861	0.828	0.782	0.933	0.899	0.862	**0.515**	0.657
	C^3-EM	**0.865**	**0.833**	**0.798**	**0.935**	**0.908**	**0.874**	0.509	**0.663**
	C^3-PS	0.860	0.827	0.782	0.931	0.896	0.856	0.497	0.477
	C^3-CI	0.861	0.826	0.783	0.934	0.900	0.862	0.514	0.659
	C^3-GS	0.858	0.823	0.777	0.845	0.799	0.745	0.383	0.314
High % Unknown	LOCAL	0.775	0.716	0.633	0.800	0.734	0.626	0.398	0.232
	INTRA	0.816	0.769	0.708	0.868	0.816	0.741	0.448	0.350
	PIPELINE*	0.831	0.794	0.743	0.895	0.861	0.811	0.479	0.419
	MLN	0.216	0.222	0.228	0.190	0.211	0.216	0.096	0.143
	C^3	0.836	0.800	0.750	0.903	0.869	0.819	0.479	0.483
	C^3-EM	**0.845**	**0.812**	**0.770**	**0.909**	**0.883**	**0.841**	**0.495**	**0.486**
	C^3-PS	0.836	0.801	0.750	0.902	0.868	0.820	0.437	0.312
	C^3-CI	0.836	0.801	0.750	0.903	0.871	0.819	0.479	0.484
	C^3-GS	0.833	0.797	0.744	0.811	0.762	0.692	0.322	0.176

classifiers at every iteration (discussed further in Section 7.2.1) should be taken into considering before applying this variant.

7.1.4. Applying Graph Construction Procedures. As discussed in Section 5.2, the predicted co-references, links, and labels may be inconsistent relative to some set of task and domain-specific hard constraints. Entity resolution, for example, may require transitivity in some cases (i.e., if pairs $\{A, B\}$ and $\{B, C\}$ are co-referent, then $\{A, C\}$ must also be co-referent). Similarly, for our DISCOURSE dataset, opinions which are predicted as having a reinforcing edge must, by definition, have the same label. These inconsistencies must be resolved prior to constructing the graph.

Inconsistencies can arise in all of the approaches we evaluated. In these situations, we can resolve the inconsistencies prior to generating the output graph. The procedure one uses to resolve inconsistencies can vary depending on the data. In our experiments, we use the following procedure for CORA, CITESEER, and ENRON: apply transitive closure over co-referent pairs, add edges between entities whose references have an edge, then use the majority label of the labels over its references. For DISCOURSE, we use a procedure that is specialized to its domain specific constraints: apply transitive closure over co-referent pairs, remove reinforcing edges between pairs not co-referent, then use the majority label over the labels of opinions transitively co-referent and reinforcing. We explore the impact of applying these procedures on the output of C^3, its variants, and our baselines prior to evaluation. The results are provided in Tables X–XIII.

Compared to not resolving inconsistencies, we see across-the-board improvement for all algorithms. This is especially true for LOCAL and INTRA, which see improvements of up to 81% and 51%, respectively. While LOCAL and INTRA do not capture

Table VII. Relative improvement over the overall F1 performance of LOCAL for various levels of annotation and noise. Bold indicates the highest value in a given column.

		Citeseer (Vary Noise Level)			Cora (Vary Noise Level)			Enron	Discourse
		Low	Medium	High	Low	Medium	High		
Low % Unknown	INTRA	5.37%	7.61%	13.24%	8.83%	12.96%	23.72%	21.41%	79.22%
	PIPELINE*	9.00%	13.32%	20.70%	13.30%	20.50%	36.12%	**31.53%**	95.57%
	MLN	-15.38%	-8.56%	0.91%	-31.08%	-25.93%	-8.37%	-67.76%	-11.36%
	C^3	10.25%	15.90%	24.66%	14.87%	22.75%	39.22%	29.41%	101.94%
	C^3-EM	10.25%	**16.17%**	**25.42%**	**14.99%**	22.62%	**40.00%**	29.18%	**102.22%**
	C^3-PS	10.13%	15.63%	24.35%	14.51%	22.09%	38.14%	26.12%	88.09%
	C^3-CI	**10.38%**	15.90%	24.81%	**14.99%**	**22.88%**	39.38%	29.41%	101.94%
	C^3-GS	10.13%	15.76%	24.35%	6.05%	9.79%	21.55%	8.00%	72.85%
Medium % Unknown	INTRA	5.98%	7.86%	12.65%	8.28%	12.45%	21.75%	28.10%	76.38%
	PIPELINE*	8.52%	12.55%	18.36%	12.18%	18.88%	33.02%	40.22%	95.47%
	MLN	-45.93%	-26.34%	-13.12%	-44.46%	-30.52%	-26.45%	-60.61%	-29.77%
	C^3	9.54%	14.21%	20.68%	13.64%	20.35%	34.90%	**41.87%**	112.62%
	C^3-EM	**10.05%**	**14.9%**	**23.15%**	**13.89%**	**21.55%**	**36.78%**	40.22%	**114.56%**
	C^3-PS	9.41%	14.07%	20.68%	13.40%	19.95%	33.96%	36.91%	54.37%
	C^3-CI	9.54%	13.93%	20.83%	13.76%	20.48%	34.90%	41.60%	113.27%
	C^3-GS	9.16%	13.52%	19.91%	2.92%	6.96%	16.59%	5.51%	1.62%
High % Unknown	INTRA	5.29%	7.40%	11.85%	8.50%	11.17%	18.37%	12.56%	50.86%
	PIPELINE*	7.23%	10.89%	17.38%	11.88%	17.30%	29.55%	20.35%	80.60%
	MLN	-72.13%	-68.99%	-63.98%	-76.25%	-71.25%	-65.50%	-75.88%	-38.36%
	C^3	7.87%	11.73%	18.48%	12.88%	18.39%	30.83%	20.35%	108.19%
	C^3-EM	**9.03%**	**13.41%**	**21.64%**	**13.63%**	**20.30%**	**34.35%**	**24.37%**	**109.48%**
	C^3-PS	7.87%	11.87%	18.48%	12.75%	18.26%	30.99%	9.80%	34.48%
	C^3-CI	7.87%	11.87%	18.48%	12.88%	18.66%	30.83%	20.35%	108.62%
	C^3-GS	7.48%	11.31%	17.54%	1.38%	3.81%	10.54%	-19.10%	-24.14%

all of the dependencies present in graph identification, the application of these procedures partially does. Even with the improvements, the trends from Tables VI – IX remain. Approaches that explicitly leverage intra- and inter-relational dependencies still significantly outperform those which do not. Furthermore, C^3-EM is still the overall best, for all levels of noise and annotations. While there is one case in which C^3-CI significant outperforms C^3-EM, the C^3-EM variant still significantly outperforms all other approaches. The improvements to C^3, and its variants, gained by the resolution procedures does suggest that there may be a benefit in explicitly supporting more consistency constraints.

7.2. Runtime Performance

7.2.1. Learning and Inference Time. In Table XIV, we list the average learning, inference, and overall runtimes for the CORA experiments. These were run on identical servers with dual quad-core Intel Xeon 2.66Ghz processors and 48GB of memory. All implementations are written in Java, except for Alchemy, which is written in C++. If we order the results based on the number of intra- and inter-relational features they capture, we see that there is a computational cost associated with capturing more dependencies. However, considering the significant improvement in predictive performance, the additional runtime required by C^3—which is still an order of magnitude faster than the slowest algorithm—does not seem like a bad trade-off.

Though C^3-EM is the best performing variant of C^3, it incurs significant overhead, due to the repeated calls to the SVM learning algorithm. This indicates that, in application, a two-step semi-supervised approach may be more practical.

7.2.2. Convergence Results. An important characteristic affecting running time of C^3 is the number of iterations required by inference. In our experiments, we allowed C^3 to run until the predictions either converged or began to oscillate, or until a maximum

Table VIII. Each entry indicates the number of times the approach in the given row significantly outperformed (in terms of average F1, over all levels of annotation and noise) the approach in the given column. We make 9 pairwise comparisons for the CORA and CITESEER datasets and 3 for ENRON and DISCOURSE.

	LOCAL	INTRA	PIPELINE*	MLN*	C^3	C^3-EM	C^3-PS	C^3-CI	C^3-GS
	CITESEER								
LOCAL	–	0	0	8	0	0	0	0	0
INTRA	9	–	0	9	0	0	0	0	0
PIPELINE*	9	9	–	9	0	0	0	0	0
MLN	0	0	0	–	0	0	0	0	0
C^3	9	9	9	9	–	0	0	0	6
C^3-EM	9	9	9	9	6	–	6	5	7
C^3-PS	9	9	9	9	1	0	–	0	5
C^3-CI	9	9	9	9	0	0	1	–	5
C^3-GS	9	9	6	9	0	0	0	0	–
	CORA								
LOCAL	–	0	0	9	0	0	0	0	0
INTRA	9	–	0	9	0	0	0	0	0
PIPELINE*	9	9	–	9	0	0	0	0	0
MLN	0	0	0	–	0	0	0	0	0
C^3	9	9	9	9	–	0	5	0	3
C^3-EM	9	9	9	9	5	–	8	5	6
C^3-PS	9	9	9	9	0	0	–	0	1
C^3-CI	9	9	9	9	2	0	6	–	6
C^3-GS	9	9	7	9	0	0	3	0	–
	ENRON								
LOCAL	–	0	0	3	0	0	0	0	0
INTRA	3	–	0	3	0	0	0	0	0
PIPELINE*	3	2	–	3	0	0	0	0	0
MLN	0	0	0	–	0	0	0	0	0
C^3	3	1	0	3	–	0	0	0	0
C^3-EM	3	2	1	3	1	–	1	1	0
C^3-PS	2	1	0	3	0	0	–	0	0
C^3-CI	3	1	0	3	0	0	0	–	0
C^3-GS	0	0	0	3	0	0	0	0	–
	DISCOURSE								
LOCAL	–	0	0	3	0	0	0	0	1
INTRA	3	–	0	3	0	0	2	0	2
PIPELINE*	3	3	–	3	0	0	2	0	3
MLN	0	0	0	–	0	0	0	0	0
C^3	3	3	3	3	–	0	3	0	3
C^3-EM	3	3	3	3	0	–	3	0	3
C^3-PS	3	1	0	3	0	0	–	0	3
C^3-CI	3	3	3	3	0	0	3	–	3
C^3-GS	1	0	0	3	0	0	0	0	–

number of iterations was reached. In Table XV, we list the number of times each stopping criterion was encountered and the average number of iterations required for all datasets.

The first thing we note is that C^3 never exhausted the maximum number of iterations; *all* of the experiments either converged or began oscillating. Of these two stopping criteria, oscillation was more common; convergence was detected only for the ENRON dataset. Examining the number of iterations required before reaching a stopping criterion, we find that C^3 typically requires few iterations. On average, C^3 ran for as few as 3.2 iterations, and at most 26.8. One area of future research is to rigorously prove that C^3 will converge, and identify the convergence rate. Since C^3 is related to the mean field approximation, some existing theory may apply. That said, all of the current empirical evidence indicates that fast convergence or oscillation is typical of

Table IX. Average F1 performance of entity resolution, link prediction and node labeling, individually. We also report the average of the three. Bold indicates the highest value in a given column.

	ER	LP	NL	Average	ER	LP	NL	Average
	CITESEER				CORA			
LOCAL	0.837	0.814	0.523	0.725	0.830	0.823	0.586	0.747
INTRA	0.901	0.860	0.586	0.782	0.892	0.842	0.787	0.840
PIPELINE-ELN	0.901	0.893	0.652	0.816	0.892	0.913	0.860	0.888
PIPELINE-ENL	0.901	0.908	0.616	0.808	0.892	0.916	0.829	0.879
PIPELINE-LEN	0.913	0.860	0.652	0.808	0.905	0.841	0.858	0.868
PIPELINE-LNE	0.916	0.860	0.620	0.799	0.909	0.841	0.827	0.859
PIPELINE-NEL	0.904	0.907	0.586	0.799	0.896	0.917	0.787	0.866
PIPELINE-NLE	0.916	0.890	0.586	0.797	0.911	0.898	0.787	0.865
MLN	0.596	0.743	0.264	0.534	0.403	0.786	0.369	0.519
C^3	**0.920**	0.910	0.654	0.828	**0.918**	0.918	0.861	0.899
C^3-EM	0.919	**0.913**	**0.667**	**0.833**	0.914	**0.935**	**0.875**	**0.908**
C^3-PS	0.917	0.911	0.654	0.827	0.908	0.920	0.861	0.896
C^3-CI	0.920	0.910	0.649	0.826	0.919	0.919	0.862	0.900
C^3-GS	0.917	0.910	0.643	0.823	0.908	0.922	0.864	0.898
	ENRON				DISCOURSE			
LOCAL	0.703	0.077	0.308	0.363	0.164	0.211	0.552	0.309
INTRA	0.891	0.100	0.405	0.465	0.552	0.530	0.553	0.545
PIPELINE-ELN	0.891	0.100	0.528	0.506	0.552	0.646	0.614	0.604
PIPELINE-ENL	0.891	**0.124**	0.513	0.509	0.552	0.655	0.601	0.603
PIPELINE-LEN	0.888	0.100	0.528	0.505	0.663	0.530	0.613	0.602
PIPELINE-LNE	0.894	0.100	0.404	0.466	0.663	0.530	0.596	0.597
PIPELINE-NEL	0.894	**0.124**	0.405	0.474	0.551	0.651	0.553	0.585
PIPELINE-NLE	0.894	**0.124**	0.405	0.474	0.665	0.533	0.553	0.584
MLN	0.187	0.007	0.235	0.143	0.151	0.195	0.306	0.217
C^3	0.894	**0.124**	0.528	**0.515**	**0.684**	0.671	0.617	0.657
C^3-EM	**0.896**	0.100	0.530	0.509	0.679	**0.689**	**0.622**	**0.663**
C^3-PS	0.826	**0.124**	**0.541**	0.497	0.431	0.404	0.594	0.477
C^3-CI	0.894	**0.124**	0.525	0.514	**0.684**	0.675	0.617	0.659
C^3-GS	0.527	**0.124**	0.499	0.383	0.201	0.194	0.546	0.314

algorithms based on pseudolikelihood [Besag 1975; Lu and Getoor 2003; Neville and Jensen 2000].

7.2.3. Parallelization Results. One benefit of C^3 learning and inference is that a significant portion of the algorithms can be parallelized. For C^3 semi-supervised weight learning (shown in Algorithm 3), the weights of the bootstrap classifiers can be learned in parallel; then, the unobserved target variables can be initialized in parallel. The second-stage classifiers can be similarly parallelized. Each iteration of inference can also be parallelized. We can exploit the fact that the features rely only on the values of target variables from the previous iteration; consequently, we can infer the values of the target variables within a particular iteration simultaneously without affecting the results.[8]

We examine the benefits of parallelizing C^3 by running a portion the CORA experiments a multithreaded implementation, using Java threads. We present the learning, inference, and overall run times, with respect to the numbers of available threads, in Figure 2. While the predicted values from these experiments are identical to those of the single-threaded implementations, there is a substantial improvement in the overall running time as we increase the number of threads. For example, going from 1 to 2 threads yields a x1.9 speed-up, which is almost linear. Comparing the improvements between the learning and inference times, we find that most of the improvement is due

[8]Although we focus our parallelization analysis here on C^3 for brevity, the learning and inference of all variants of C^3 can be parallelized using the same procedure to improve runtime.

Table X. Overall F1 performance—reported as the average of F1 scores for entity resolution, link prediction and node labeling—for various levels of annotation and noise, after applying constraints. Bold indicates the highest value in a given column.

		Citeseer (Vary Noise Level)			Cora (Vary Noise Level)			Enron	Discourse
		Low	Medium	High	Low	Medium	High		
Low % Unknown	LOCAL	0.813	0.766	0.667	0.881	0.826	0.793	0.713	0.523
	INTRA	0.862	0.812	0.760	0.933	0.902	0.861	0.743	0.637
	PIPELINE*	0.877	0.847	0.798	0.945	0.920	0.891	**0.764**	0.697
	MLN	0.798	0.788	0.723	0.826	0.796	0.783	0.402	0.470
	C^3	0.884	0.855	0.818	0.952	0.930	0.904	0.762	0.738
	C^3-EM	**0.886**	0.856	0.806	**0.953**	0.930	**0.905**	0.758	**0.744**
	C^3-PS	0.882	0.854	0.809	0.947	0.922	0.892	0.727	0.685
	C^3-CI	0.885	0.855	**0.819**	**0.953**	**0.931**	0.903	0.762	0.737
	C^3-GS	0.885	**0.857**	0.808	**0.953**	**0.931**	0.899	0.562	0.623
Medium % Unknown	LOCAL	0.780	0.695	0.622	0.853	0.794	0.752	0.655	0.429
	INTRA	0.844	0.788	0.711	0.913	0.876	0.829	0.701	0.521
	PIPELINE*	0.852	0.808	0.744	0.926	0.894	0.854	0.723	0.605
	MLN	0.495	0.602	0.565	0.488	0.463	0.439	0.306	0.334
	C^3	0.859	0.817	0.756	0.933	0.906	0.866	0.726	0.657
	C^3-EM	**0.863**	**0.819**	**0.766**	**0.936**	**0.909**	**0.874**	**0.727**	**0.668**
	C^3-PS	0.854	0.815	0.750	0.926	0.896	0.846	0.679	0.468
	C^3-CI	0.860	0.815	0.758	0.932	0.905	0.866	0.724	0.657
	C^3-GS	0.851	0.810	0.723	0.928	0.898	0.844	0.483	0.278
High % Unknown	LOCAL	0.749	0.681	0.592	0.817	0.746	0.678	0.602	0.290
	INTRA	0.818	0.758	0.675	0.885	0.842	0.777	0.642	0.326
	PIPELINE*	0.827	0.776	0.699	0.896	0.859	0.806	0.660	0.419
	MLN	0.363	0.348	0.342	0.310	0.319	0.316	0.211	0.190
	C^3	0.828	0.782	0.709	0.901	0.865	0.813	0.659	0.485
	C^3-EM	**0.840**	**0.791**	**0.724**	**0.910**	**0.877**	**0.832**	**0.666**	**0.487**
	C^3-PS	0.829	0.778	0.707	0.895	0.859	0.803	0.596	0.295
	C^3-CI	0.828	0.778	0.708	0.901	0.867	0.811	0.659	**0.487**
	C^3-GS	0.820	0.767	0.668	0.891	0.854	0.781	0.358	0.119

to faster inference. In learning, the number of parallel tasks is far less than the number of available threads; whereas, during inference, the number of target variables we can infer in parallel is typically much larger than the number of available threads. Learning does enjoy a slight benefit from parallelization during bootstrapping, but this effect is minimal. The implementation of SVM training we used does not itself support parallelization, though there are approaches for parallelizing weight learning within the classifiers themselves [Chang et al. 2008; Caruana et al. 2011].

7.2.4. Scalability Results. To work with real graph identification problems, it is important that a method be scalable. Accordingly, we investigated the scalability of C^3, shown in our experiments to have the best overall performance in terms of both accuracy and runtime, on large, synthetic datasets. For these experiments, we generated increasingly larger networks, as described in Section 6.1.4. We used the same features and settings used for the "medium" annotation levels in the CORA and CITESEER experiments. In Figure 3, we present the learning, inference, and overall runtime of C^3 on input networks ranging from 2,209 nodes and 15,636 edges to networks with up to 45,522 nodes and 353,228 edges.

Although our implementation is not specifically designed for large networks, we demonstrate that C^3 is able to scale well to such networks. Examining the individual learning and inference times, however, we find that the critical bottleneck is in learning. C^3's scalability is directly tied to the scalability of the underlying classifiers. In our experiments, we used the LibSVM implementation of SVMs [Chang and Lin 2001]. For this implementation, learning time is quadratic in the number of training instances. This may severely limit the scalability of our current implementation of C^3.

Table XI. Relative improvement over the overall F1 performance of LOCAL for various levels of annotation and noise, after applying constraints. Bold indicates the highest value in a given column.

		Citeseer (Vary Noise Level)			Cora (Vary Noise Level)			Enron	Discourse
		Low	Medium	High	Low	Medium	High		
Low % Unknown	INTRA	6.03%	6.01%	13.94%	5.90%	9.20%	8.58%	4.21%	21.80%
	PIPELINE*	7.87%	10.57%	19.64%	7.26%	11.38%	12.36%	**7.15%**	33.27%
	MLN	-1.85%	2.87%	8.40%	-6.24%	-3.63%	-1.26%	-43.62%	-10.13%
	C^3	8.73%	11.62%	22.64%	8.06%	12.59%	14.00%	6.87%	41.11%
	C^3-EM	**8.98%**	11.75%	20.84%	**8.17%**	12.59%	**14.12%**	6.31%	**42.26%**
	C^3-PS	8.49%	11.49%	21.29%	7.49%	11.62%	12.48%	1.96%	30.98%
	C^3-CI	8.86%	11.62%	**22.79%**	**8.17%**	**12.71%**	13.87%	6.87%	40.92%
	C^3-GS	8.86%	**11.88%**	21.14%	**8.17%**	**12.71%**	13.37%	-21.18%	19.12%
Medium % Unknown	INTRA	8.21%	13.38%	14.31%	7.03%	10.33%	10.24%	7.02%	21.45%
	PIPELINE*	9.23%	16.26%	19.61%	8.56%	12.59%	13.56%	10.38%	41.03%
	MLN	-36.54%	-13.38%	-9.16%	-42.79%	-41.69%	-41.62%	-53.28%	-22.14%
	C^3	10.13%	17.55%	21.54%	9.38%	14.11%	15.16%	10.84%	53.15%
	C^3-EM	**10.64%**	**17.84%**	**23.15%**	**9.73%**	**14.48%**	**16.22%**	**10.99%**	**55.71%**
	C^3-PS	9.49%	17.27%	20.58%	8.56%	12.85%	12.50%	3.66%	9.09%
	C^3-CI	10.26%	17.27%	21.86%	9.26%	13.98%	15.16%	10.53%	53.15%
	C^3-GS	9.10%	16.55%	16.24%	8.79%	13.10%	12.23%	-26.26%	-35.20%
High % Unknown	INTRA	9.21%	11.31%	14.02%	8.32%	12.87%	14.60%	6.64%	12.41%
	PIPELINE*	10.41%	13.95%	18.07%	9.67%	15.15%	18.88%	9.63%	44.48%
	MLN	-51.54%	-48.90%	-42.23%	-62.06%	-57.24%	-53.39%	-64.95%	-34.48%
	C^3	10.55%	14.83%	19.76%	10.28%	15.95%	19.91%	9.47%	67.24%
	C^3-EM	**12.15%**	**16.15%**	**22.30%**	**11.38%**	**17.56%**	**22.71%**	**10.63%**	**67.93%**
	C^3-PS	10.68%	14.24%	19.43%	9.55%	15.15%	18.44%	-1.00%	1.72%
	C^3-CI	10.55%	14.24%	19.59%	10.28%	16.22%	19.62%	9.47%	**67.93%**
	C^3-GS	9.48%	12.63%	12.84%	9.06%	14.48%	15.19%	-40.53%	-58.97%

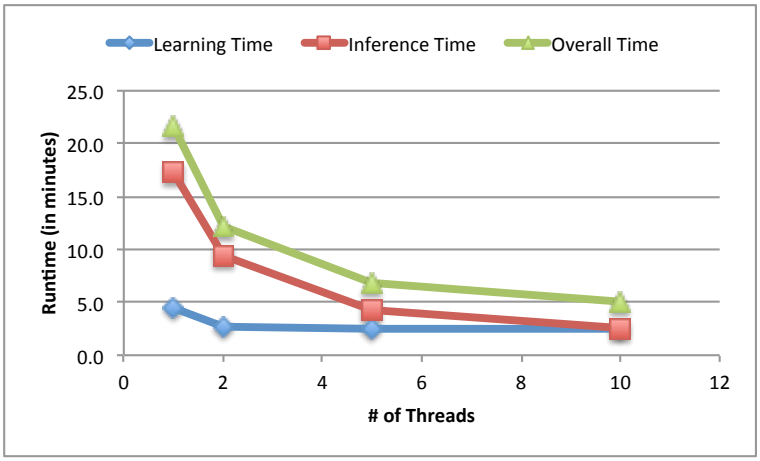

Fig. 2. Average running times (in minutes) for learning, inference and overall on the **Cora** dataset with multithreaded C^3, using varying numbers of threads.

There has recently been significant progress in the development of scalable SVM algorithms that may help with this. Beyond the parallelizations mentioned in Section 7.2.3, there are more efficient learning algorithms, such as stochastic subgradient methods [Shalev-Shwartz et al. 2007] and others [Cervantes et al. 2008].

8. CONCLUSION

Graph identification is an important emerging problem. As network data becomes increasingly available, the need to properly map from the noisy observations to the "true"

Table XII. Each entry indicates the number of times the approach in the given row significantly outperformed (in terms of average F1, over all levels of annotation and noise) the approach in the given column, after applying constraints. We make 9 pairwise comparisons for the CORA and CITESEER datasets and 3 for ENRON and DISCOURSE.

	LOCAL	INTRA	PIPELINE*	MLN*	C^3	C^3-EM	C^3-PS	C^3-CI	C^3-GS
	CITESEER								
LOCAL	–	0	0	8	0	0	0	0	0
INTRA	9	–	0	9	0	0	0	0	0
PIPELINE*	9	9	–	9	0	0	1	0	2
MLN	0	0	0	–	0	0	0	0	0
C^3	9	9	9	9	–	0	7	0	6
C^3-EM	9	9	9	9	4	–	9	4	7
C^3-PS	9	9	0	9	0	0	–	0	3
C^3-CI	9	9	8	9	1	0	7	–	6
C^3-GS	9	7	3	9	0	0	3	0	–
	CORA								
LOCAL	–	0	0	4	0	0	0	0	0
INTRA	9	–	0	8	0	0	0	0	0
PIPELINE*	9	8	–	9	0	0	0	0	0
MLN	2	0	0	–	0	0	0	0	0
C^3	9	9	7	9	–	0	0	0	5
C^3-EM	9	9	7	9	5	–	4	5	6
C^3-PS	9	8	3	9	0	0	–	0	3
C^3-CI	9	9	7	9	0	1	0	–	4
C^3-GS	9	6	3	9	1	0	0	0	–
	ENRON								
LOCAL	–	0	0	3	0	0	0	0	0
INTRA	1	–	0	3	0	0	0	0	0
PIPELINE*	1	1	–	3	0	0	1	0	0
MLN	0	0	0	–	0	0	0	0	0
C^3	1	2	0	3	–	0	1	0	0
C^3-EM	2	1	0	3	1	–	2	1	0
C^3-PS	0	0	0	3	0	0	–	0	0
C^3-CI	1	2	0	3	0	0	1	–	0
C^3-GS	0	0	0	0	0	0	0	0	–
	DISCOURSE								
LOCAL	–	0	0	3	0	0	0	0	2
INTRA	3	–	0	3	0	0	1	0	2
PIPELINE*	3	3	–	3	0	0	2	0	3
MLN	0	0	0	–	0	0	0	0	2
C^3	3	3	3	3	–	0	3	0	3
C^3-EM	3	3	3	3	1	–	3	1	3
C^3-PS	2	1	0	3	0	0	–	0	3
C^3-CI	3	3	3	3	0	0	3	–	3
C^3-GS	1	0	0	1	0	0	0	0	–

underlying network becomes more and more important, especially in social and biological sciences. Not only do the inferred networks prevent us from drawing erroneous conclusions, they expedite analysis by reducing the size of the data, possibly by orders of magnitude smaller than the observed data. The graph identification problem is extremely challenging, in terms of feature engineering, training and prediction.

In this work, we have formulated the problem as learning and inference in a probabilistic graphical model. This combines the problems of entity resolution, link prediction and node labeling in a single, coherent framework. To approximate the resulting inference problem, we developed C^3 and its variants, which capture the intra- and inter-relational dependencies, while remaining tractable. We empirically showed that C^3 can achieve significant gains in accuracy over existing approaches, over a range of application domains.

Table XIII. Average F1 performance of entity resolution, link prediction and node labeling, individually, after applying constraints. We also report the average of the three. Bold indicates the highest value in a given column.

	ER	LP	NL	Average	ER	LP	NL	Average
	CORA				CITESEER			
LOCAL	0.649	0.854	0.581	0.695	0.844	0.814	0.724	0.794
INTRA	0.825	0.913	0.625	0.788	0.872	0.919	0.838	0.876
PIPELINE-ELN	0.825	0.921	0.665	0.804	0.872	0.940	0.867	0.893
PIPELINE-ENL	0.825	0.932	0.627	0.795	0.872	0.943	0.837	0.884
PIPELINE-LEN	0.839	0.913	0.664	0.806	0.888	0.919	0.866	0.891
PIPELINE-LNE	0.852	0.914	0.657	0.808	0.896	0.919	0.866	0.894
PIPELINE-NEL	0.825	0.932	0.625	0.794	0.882	0.943	0.838	0.888
PIPELINE-NLE	**0.853**	0.916	0.625	0.798	0.896	0.932	0.838	0.889
MLN	0.568	0.854	0.384	0.602	0.174	0.742	0.474	0.463
C^3	0.852	**0.934**	0.666	0.817	0.902	**0.945**	0.871	0.906
C^3-EM	0.852	0.926	**0.679**	**0.819**	**0.903**	0.943	**0.881**	**0.909**
C^3-PS	0.846	**0.934**	0.666	0.815	0.876	0.944	0.869	0.896
C^3-CI	0.852	**0.934**	0.659	0.815	0.899	**0.945**	0.870	0.905
C^3-GS	0.832	0.933	0.667	0.810	0.876	0.944	0.875	0.898
	ENRON				DISCOURSE			
LOCAL	0.844	0.610	0.511	0.655	0.547	0.178	0.562	0.429
INTRA	0.909	0.631	0.563	0.701	0.556	0.443	0.565	0.521
PIPELINE-ELN	0.909	0.631	0.628	0.723	0.556	0.575	0.606	0.579
PIPELINE-ENL	0.909	**0.639**	0.610	0.719	0.556	0.584	0.606	0.582
PIPELINE-LEN	0.906	0.631	0.628	0.722	0.691	0.520	0.604	0.605
PIPELINE-LNE	0.911	0.631	0.563	0.702	0.692	0.520	0.592	0.601
PIPELINE-NEL	0.911	**0.639**	0.563	0.705	0.556	0.580	0.563	0.566
PIPELINE-NLE	0.911	**0.639**	0.563	0.705	0.692	0.523	0.564	0.593
MLN	0.195	0.415	0.308	0.306	0.430	0.247	0.324	0.334
C^3	0.911	**0.639**	0.628	0.726	0.699	0.665	0.608	0.657
C^3-EM	**0.914**	0.633	0.634	**0.727**	**0.709**	**0.680**	**0.614**	**0.668**
C^3-PS	0.807	0.593	**0.638**	0.679	0.429	0.409	0.564	0.468
C^3-CI	0.911	**0.639**	0.622	0.724	0.699	0.667	0.604	0.657
C^3-GS	0.510	0.419	0.520	0.483	0.192	0.196	0.445	0.278

Table XIV. Average running times (in minutes) for learning, inference and overall, for the CORA experiments.

	Learning Time	Inference Time	Overall Time
LOCAL	1.8	0.5	2.3
INTRA	4.6	7.5	12.1
PIPELINE-ELN	4.7	6.1	10.8
PIPELINE-ENL	4.7	6.1	10.7
PIPELINE-LEN	4.9	7.0	12.0
PIPELINE-LNE	5.1	7.1	12.3
PIPELINE-NEL	4.8	6.1	11.0
PIPELINE-NLE	5.3	6.9	12.3
MLN	761.6	183.6	945.1
C^3	5.2	20.3	25.5
C^3-EM	114.9	47.5	162.4
C^3-PS	5.2	36.9	42.1
C^3-CI	5.2	28.7	33.9
C^3-GS	5.2	728.1	733.3

Table XV. Number of times C^3 converged or began to oscillate, for all datasets. The number of trials for each experiment were: 45 for CORA; 15 for CITESEER; and 15 for ENRON and DISCOURSE. We also report the average number of iterations performed prior to reaching convergence or oscillation. Observe that all our C^3 experiments either converged or reached an oscillation point.

	# Converge	# Oscillate	Avg. # of Iterations
CITESEER	0	45	26.8
CORA	0	45	15.4
ENRON	14	1	3.2
DISCOURSE	0	15	8.9

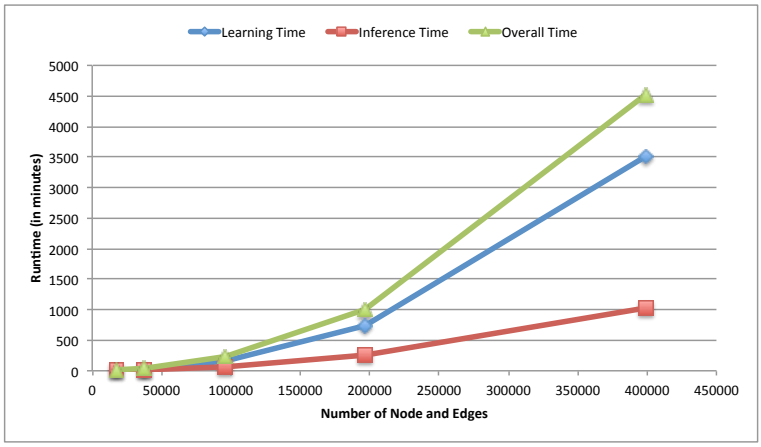

Fig. 3. Average running times (in minutes) for learning, inference and overall on a synthetic dataset as the number of nodes and edges in the input graph increase.

There is much room for further exploration. One research question is how one could apply graph identification to evolving networks. We have also identified convergence as an open theoretical problem. Lastly, one could explore other algorithms and models for graph identification, which may be tailored to specific application domains.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. 0746930 and IIS1218488.

REFERENCES

ADAFRE, S. F. AND DE RIJKE, M. 2005. Discovering missing links in wikipedia. In *Proc. of the SIGKDD Workshop on Link Discovery*.

ANANTHAKRISHNA, R., CHAUDHURI, S., AND GANTI, V. 2002. Eliminating fuzzy duplicates in data warehouses. In *Proc. of the International Conference on Very Large Databases*.

ASTHANA, S., KING, O. D., GIBBONS, F. D., AND ROTH, F. P. 2004. Predicting protein complex membership using probabilistic network reliability. *Genome Research 14,* 6, 1170–1175.

BATADA, N. N., REGULY, T., BREITKREUTZ, A., BOUCHER, L., BREITKREUTZ, B.-J., HURST, L. D., AND TYERS, M. 2007. Still stratus not altocumulus: Further evidence against the date/party hub distinction. *PLoS Biology 5,* 6, e154+.

BERTIN, N., SIMONIS, N., DUPUY, D., CUSICK, M. E., HAN, J.-D. J., FRASER, H. B.,

ROTH, F. P., AND VIDAL, M. 2007. Confirmation of organized modularity in the yeast interactome. *PLoS Biology 5,* 6, e153.

BESAG, J. 1975. Statistical analysis of non-lattice data. *The Statistician 24*, 179–195.

BHATTACHARYA, I. AND GETOOR, L. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data 1*, 1–36.

BHATTACHARYA, I., GODBOLE, S., AND JOSHI, S. 2008. Structured entity identification and document categorization: Two tasks with one joint models. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

BILENKO, M. AND MOONEY, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

BILGIC, M. AND GETOOR, L. 2008. Effective label acquisition for collective classification. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

CARUANA, G., LI, M., AND QI, M. 2011. A mapreduce based parallel svm for large scale spam filtering. In *Proc. of the International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 4. 2659–2662.

CARVALHO, V. R. AND COHEN, W. W. 2005. On the collective classification of email "speech acts". In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

CERVANTES, J., LI, X., YU, W., AND LI, K. 2008. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing 71,* 4–6, 611–619.

CHAKRABARTI, S., DOM, B., AND INDYK, P. 1998. Enhanced hypertext categorization using hyperlinks. In *Proc. of the SIGMOD International Conference on Management of Data*.

CHANG, C.-C. AND LIN, C.-J. 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

CHANG, E., ZHU, K., WANG, H., BAI, H., LI, J., QIU, Z., AND CUI, H. 2008. Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, Cambridge, MA, 257–264.

CHUA, H. N., SUNG, W.-K., AND WONG, L. 2006. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics 22,* 13, 1623–1630.

CLAUSET, A., MOORE, C., AND NEWMAN, M. E. J. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature 453*, 98.

COHEN, W. W., RAVIKUMAR, P., AND FIENBERG, S. E. 2003. A comparison of string distance metrics for name-matching tasks. In *Proc. of the IJCAI Workshop on Information Integration*.

CRAMMER, K., SINGER, Y., CRISTIANINI, N., SHAWE-TAYLOR, J., AND WILLIAMSON, B. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research 2*, 2001.

DENG, M., MEHTA, S., SUN, F., AND CHEN, T. 2002. Inferring domain-domain interactions from protein-protein interactions. *Genome Research 12,* 10, 1540–1548.

DI BATTISTA, G., ERLEBACH, T., HALL, A., PATRIGNANI, M., PIZZONIA, M., AND SCHANK, T. 2007. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking 15*, 267–280.

DIEHL, C., NAMATA, G. M., AND GETOOR, L. 2007. Relationship identification for social network discovery. In *Proc. of the AAAI Conference on Artificial Intelligence*.

DONG, X., HALEVY, A., AND MADHAVAN, J. 2005. Reference reconciliation in complex information spaces. In *Proc. of the SIGMOD International Conference on Manage-*

ment of Data.

FARRELL, S., CAMPBELL, C., AND MYAGMAR, S. 2005. Relescope: An experiment in accelerating relationships. In *Extended Abstracts on Human Factors in Computing Systems*.

FELLEGI, I. P. AND SUNTER, A. B. 1969. A theory for record linkage. *Journal of the American Statistical Association 64,* 328, 1183–1210.

GETOOR, L., FRIEDMAN, N., KOLLER, D., PFEFFER, A., AND TASKAR, B. 2007. Probabilistic relational models. In *An Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press.

GETOOR, L., FRIEDMAN, N., KOLLER, D., AND TASKAR, B. 2003. Learning probabilistic models of link structure. *Machine Learning 3*, 679–707.

GETOOR, L., SEGAL, E., TASKAR, B., AND KOLLER, D. 2001. Probabilistic models of text and link structure for hypertext classification. In *Proc. of the IJCAI Workshop on Text Learning: Beyond Supervision*.

GILKS, W. R., RICHARDSON, S., AND SPIEGELHALTER, D. 1996. *Markov chain Monte Carlo in Practice*. Chapman & Hall CRC.

HEITZ, G., GOULD, S., SAXENA, A., AND KOLLER, D. 2008. Cascaded classification models: Combining models for holistic scene understanding. In *Advances in Neural Information Processing Systems*.

HERNÁNDEZ, M. A. AND STOLFO, S. J. 1995. The merge/purge problem for large databases. In *Proc. of the SIGMOD International Conference on Management of Data*.

HUANG, H. AND BADER, J. S. 2009. Precision and recall estimates for two-hybrid screens. *Bioinformatics 25,* 3, 372–378.

HUANG, Z., LI, X., AND CHEN, H. 2005. Link prediction approach to collaborative filtering. In *Proc. of the ACM/IEEE-CS Joint Conference on Digital Libraries*.

HUANG, Z. AND LIN, D. K. J. 2008. The Time-Series Link Prediction Problem with Applications in Communication Surveillance. *Informs Journal On Computing 21*, 286–303.

HUYNH, T. AND MOONEY, R. 2009. Max-margin weight learning for markov logic networks. In *Machine Learning and Knowledge Discovery in Databases*, W. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor, Eds. Lecture Notes in Computer Science Series, vol. 5781. Springer Berlin / Heidelberg, 564–579.

JACCARD, P. 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles 37*, 547–579.

JARO, M. A. 1995. Probabilistic linkage of large public health data files. *Statistics in Medicine 14*, 491–498.

KALASHNIKOV, D. V., MEHROTRA, S., AND CHEN, Z. 2005. Exploiting relationships for domain-independent data cleaning. In *Proc. of the SIAM International Conference on Data Mining*.

KLIMT, B. AND YANG, Y. 2004. Introducing the enron corpus. In *Conference on Email and Anti-Spam*.

KOK, S., SUMNER, M., RICHARDSON, M., SINGLA, P., POON, H., AND DOMINGOS, P. 2006. The alchemy system for statistical relational ai. Tech. rep., Department of Computer Science and Engineering, University of Washington, Seattle, WA.

LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence datasets. In *Proc. of the International Conference on Machine Learning*.

LESKOVEC, J., KLEINBERG, J., AND FALOUTSOS, C. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from*

Data 1, 1, 2.

LI, R., WANG, C., AND CHANG, K. C.-C. 2014. User profiling in an ego network: Co-profiling attributes and relationships. In *Proc. of the International Conference on World Wide Web*.

LIBEN-NOWELL, D. AND KLEINBERG, J. 2003. The link prediction problem for social networks. In *Proc. of the ACM Conference on Information and Knowledge Management*.

LU, Q. AND GETOOR, L. 2003. Link-based classification using labeled and unlabeled data. In *Proc. of the ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.

MACSKASSY, S. A. AND PROVOST, F. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research 8*, 935–983.

MARTIN, S., ROE, D., AND FAULON, J.-L. 2005. Predicting protein-protein interactions using signature products. *Bioinformatics 21*, 218–226.

MCCALLUM, A., NIGAM, K., AND UNGAR, L. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

MCCALLUM, A. AND WELLNER, B. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proc. of the IJCAI Workshop on Information Integration on the Web*.

MCCALLUM, A. AND WELLNER, B. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems*.

MCDOWELL, L. K. AND AHA, D. W. 2013. Labels or attributes?: Rethinking the neighbors for collective classification in sparsely-labeled networks. In *Proc. of the ACM Conference on Information and Knowledge Management*.

MCDOWELL, L. K., GUPTA, K. M., AND AHA, D. W. 2009. Cautious collective classification. *Journal of Machine Learning Research 10*, 2777–2836.

MEWES, H. W., FRISHMAN, D., GÜLDENER, U., MANNHAUPT, G., MAYER, K., MOKREJS, M., MORGENSTERN, B., MÜNSTERKÖTTER, M., RUDD, S., AND WEIL, B. 2002. Mips: a database for genomes and protein sequences. *Nucleic acids research 30*, 31–34.

MILNE, D. AND WITTEN, I. H. 2008. Learning to link with wikipedia. In *Proc. of the ACM Conference on Information and Knowledge Management*.

NABIEVA, E., JIM, K., AGARWAL, A., CHAZELLE, B., AND SINGH, M. 2005. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics 21*, 302–310.

NAMATA, G. M., KOK, S., AND GETOOR, L. 2011. Collective graph identification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

NEVILLE, J. AND JENSEN, D. 2000. Iterative classification in relational data. In *Proc. of the AAAI Workshop on Learning Statistical Models from Relational Data*.

NEVILLE, J. AND JENSEN, D. 2007. Relational dependency networks. *Journal of Machine Learning Research 8*, 653–692.

NEWCOMBE, H. B. AND KENNEDY, J. M. 1962. Record linkage: making maximum use of the discriminating power of identifying information. *Communications ACM 5,* 11, 563–566.

NEWCOMBE, H. B., KENNEDY, J. M., AXFORD, S. J., AND JAMES, A. P. 1959. Automatic linkage of vital records. *Science 130*, 954–959.

OGALE, A. S. AND ALOIMONOS, Y. 2005. Shape and the stereo correspondence problem. *International Journal of Computer Vision 65,* 3, 147–162.

O'MADADHAIN, J., HUTCHINS, J., AND SMYTH, P. 2005. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations Newsletter 7,* 2, 23–30.

PASULA, H., MARTHI, B., MILCH, B., RUSSELL, S., AND SHPITSER, I. 2003. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems*.

POON, H. AND DOMINGOS, P. 2007. Joint inference in information extraction. In *Proc. of the AAAI Conference on Artificial Intelligence*.

RATTIGAN, M. J. AND JENSEN, D. 2005. The case for anomalous link discovery. *SIGKDD Explorations Newsletter 7*, 41–47.

RATTIGAN, M. J., MAIER, M., AND JENSEN, D. 2007. Exploiting network structure for active inference in collective classification. Tech. rep., University of Massachusetts Amherst.

RÉKA, J. A., DASGUPTA, B., DONDI, R., KACHALO, S., SONTAG, E., ZELIKOVSKY, A., AND WESTBROOK, K. 2007. A novel method for signal transduction network inference from indirect experimental evidence. *Journal of Computational Biology 14*, 407–419.

RICHARDSON, M. AND DOMINGOS, P. 2006. Markov logic networks. *Machine Learning 62*, 107–136.

ROTH, D., SMALL, K., AND TITOV, I. 2009. Sequential learning of classifiers for structured prediction problems. In *Proc. of the Conference on Artificial Intelligence and Statistics*.

ROTH, D. AND YIH, W. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Conference on Computational Natural Language*.

SARAWAGI, S. 2008. Information extraction. *Foundations and Trends in Databases 1,* 3, 261–377.

SARAWAGI, S. AND BHAMIDIPATY, A. 2002. Interactive deduplication using active learning. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*. New York, NY, USA, 269–278.

SEN, P., NAMATA, G. M., BILGIC, M., GETOOR, L., GALLAGHER, B., AND ELIASSI-RAD, T. 2008. Collective classification in network data. *AI Magazine 29,* 3, 93–106.

SHALEV-SHWARTZ, S., SINGER, Y., AND SREBRO, N. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proc. of the International Conference on Machine Learning*. 807–814.

SHARAN, R., ULITSKY, I., AND SHAMIR, R. 2007. Network-based prediction of protein function. *Molecular Systems Biology 3*, 88.

SHERWOOD, R., BENDER, A., AND SPRING, N. 2008. Discarte: a disjunctive internet cartographer. *SIGCOMM Computer Communication Review 38,* 4, 303–314.

SHI, X., LI, Y., AND YU, P. 2011. Collective prediction with latent graphs. In *Proc. of the ACM Conference on Information and Knowledge Management*. CIKM '11.

SINGH, R., XU, J., AND BERGER, B. 2006. Struct2net: Integrating structure into protein-protein interaction prediction. In *Pacific Symposium on Biocomputing*. 403–414.

SINGLA, P. AND DOMINGOS, P. 2006. Entity resolution with markov logic. *IEEE International Conference on Data Mining 21*, 572–582.

SOMASUNDARAN, S., NAMATA, G. M., GETOOR, L., AND WIEBE, J. 2009. Opinion graphs for polarity and discourse classification. In *TextGraphs-4: Graph-based Methods for Natural Language Processing*.

SPRING, N., MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. 2004a. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking 12,* 1, 2–16.

SPRING, N., WETHERALL, D., AND ANDERSON, T. 2004b. Reverse engineering the internet. *SIGCOMM Computer Communication Review 34,* 1, 3–8.

SUTTON, C. AND MCCALLUM, A. 2007. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proc. of the International Conference on Machine*

Learning.

SZILAGYI, A., GRIMM, V., ARAKAKI, A. K., AND SKOLNICK, J. 2005. Prediction of physical protein-protein interactions. *Physical Biology 2,* 2, S1–S16.

TANG, L. AND LIU, H. 2009. Relational learning via latent social dimensions. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

TANG, W., ZHUANG, H., AND TANG, J. 2011. Learning to infer social ties in large networks. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases*.

TASKAR, B., PIETER, A., AND KOLLER, D. 2002. Discriminative probabilistic models for relational data. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*.

TASKAR, B., WONG, M.-F., ABBEEL, P., AND KOLLER, D. 2003. Link prediction in relational data. In *Advances in Neural Information Processing Systems*.

ULLMAN, J. D. 1997. Information integration using logical views. In *Proc. of the International Conference on Database Theory*, F. N. Afrati and P. G. Kolaitis, Eds. Lecture Notes in Computer Science Series, vol. 1186. Springer, 19–40.

WAINWRIGHT, M. AND JORDAN, M. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc.

WANG, C., HAN, J., JIA, Y., TANG, J., ZHANG, D., YU, Y., AND GUO, J. 2010. Mining advisor-advisee relationships from research publication networks. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

WANG, J. AND DOMINGOS, P. 2008. Hybrid markov logic networks. In *Proc. of the AAAI Conference on Artificial Intelligence*. AAAI Press, 1106–1111.

WHANG, S. E., MENESTRINA, D., KOUTRIKA, G., THEOBALD, M., AND GARCIA-MOLINA, H. 2009. Entity resolution with iterative blocking. In *Proc. of the SIGMOD International Conference on Management of Data*. ACM, 219–232.

WICK, M. L., ROHANIMANESH, K., SCHULTZ, K., AND MCCALLUM, A. 2008. A unified approach for schema matching, coreference and canonicalization. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*.

WINKLER, W. E. 1999. The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau.

WU, T.-F., LIN, C.-J., AND WENG, R. C. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research 5*, 975–1005.

YU, H., KIM, P. M., SPRECHER, E., TRIFONOV, V., AND GERSTEIN, M. 2007. The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Computational Biology 3,* 4, e59+.

YU, H., PACCANARO, A., TRIFONOV, V., AND GERSTEIN, M. 2006. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics 22,* 7, 823–829.

ZHANG, L., WONG, S., KING, O., AND ROTH, F. 2004. Predicting co-complexed protein pairs using genomic and proteomic data integration. *BMC Bioinformatics 5*, 38.

ZHELEVA, E., GETOOR, L., GOLBECK, J., AND KUTER, U. 2008. Using friendship ties and family circles for link prediction. In *Proc. of the SIGKDD Workshop on Social Network Mining and Analysis*. Lecture Notes in Computer Science.

ZHU, J. 2003. Mining web site link structure for adaptive web site navigation and search. Ph.D. thesis, University of Ulster at Jordanstown, UK.