ABSTRACT

Title of dissertation: IDENTIFYING GRAPHS FROM NOISY OBSERVATIONAL DATA

Galileo Mark S. Namata Jr.
Doctor of Philosophy, 2012

Dissertation directed by: Professor Lise Getoor
Department of Computer Science

There is a growing amount of data describing networks – examples include social networks, communication networks, and biological networks. As the amount of available data increases, so does our interest in analyzing the properties and characteristics of these networks. However, in most cases the data is noisy, incomplete, and the result of passively acquired observational data; naively analyzing these networks without taking these errors into account can result in inaccurate and misleading conclusions. In my dissertation, I study the tasks of *entity resolution*, *link prediction*, and *collective classification* to address these deficiencies. I describe these tasks in detail and discuss my own work on each of these tasks. For entity resolution, I develop a method for resolving the identities of name mentions in email communications. For link prediction, I develop a method for inferring subordinate-manager relationships between individuals in an email communication network. For collective classification, I propose an adaptive active surveying method to address node labeling in a query-driven setting on network data. In many real-world settings, however, these deficiencies are not found in isolation and all need to be addressed to infer the desired complete and accurate network. Furthermore, because of the dependencies typically found in these tasks, the tasks are inherently inter-related and must be performed jointly. I define the general problem of *graph identification* which simultaneously performs these tasks; removing the noise and missing values in the observed input network and inferring the complete and accurate output network. I present a novel approach to graph identification using a collection of Coupled Collective Classifiers, C^3, which, in addition to capturing the variety of features typically used for each task, can capture the intra- and inter-dependencies required to correctly infer nodes, edges, and labels in the output network. I discuss variants of C^3 using different learning and inference paradigms and show the superior performance of C^3, in terms of both prediction quality and runtime performance, over various previous approaches. I then conclude by presenting the Graph Alignment, Identification, and Analysis (GAIA) open-source software library which

not only provides an implementation of C^3 but also algorithms for various tasks in network data such as entity resolution, link prediction, collective classification, clustering, active learning, data generation, and analysis.

IDENTIFYING GRAPHS FROM NOISY OBSERVATIONAL DATA

by

Galileo Mark S. Namata Jr.

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Lise Getoor, Chair/Advisor
Professor Amol Deshpande
Professor Carl Kingsford
Professor William Rand
Professor Najib M. El-Sayed

Foreword

Portions of this dissertation are derived from research and publications co-authored by the candidate and published elsewhere. Chapter 2 is based on the book chapter *A Survey of Link Mining Tasks for Analyzing Noisy and Incomplete Networks* [121] and magazine article *Collective Classification in Network Data* [149]. The entity resolution work in Chapter 3 is based on the paper *Name Reference Resolution in Organizational Email Archives* [47]. The link prediction work in the paper Chapter 4 is based on *Relationship Identification for Social Network Discovery* [48]. Work in Chapter 5 is based on an active submission *Active Surveying for Query-Driven Collective Classification.* Finally, Chapter 6 is an extension of the work *Collective Graph Identification* [120].

Dedication

To my loving and supportive parents.

Acknowledgments

I am forever grateful to everyone whose encouragement and support allowed me to complete my Ph.D. studies and dissertation. Due to my name sake, Galileo Galilei, performing scientific research has been a lifelong dream but one I never really expected to ever attain.

First and foremost, I thank my advisor, Lise Getoor, for her patience, support, and guidance from the very beginning. She taught me so much through her guidance and example – from writing good papers and presentation skills to being passionate about my research and always striving for my best. I cannot express my extreme gratitude for all she has done and cannot imagine a more perfect advisor.

Next, I would like to thank my other committee members, Carl Kingsford, Amol Deshpande, William Rand, and Najib El-Sayed, for taking time from their busy schedules to review my dissertation, participate in my defense, and providing insightful suggestions. I would like to give a special thanks to Amol for his guidance and help through our collaborations together and to Carl for allowing me to learn about biological networks from his always energetic group meetings and for introducing me to the cluster resources which allowed me to take my research to the next level.

Graduate school takes many years to complete but my friendships here have made that time fly. I thank the LINQS members – Indrajit Bhattacharya, Rezarta Islamaj, Prithviraj Sen, Louis Licamele, Mustafa Bilgic, Elena Zheleva, Hossam Sharara, Walaa Moustafa, Lilyana Mihalkova, Stanley Kok, Stephen Bach, Ben Lon-

don, Theodoros Rekatsinas, Bert Huang, and Angelika Kimmig, for their mentorship, our stimulating discussions, our memorable lunch outings, the wonderful work atmosphere, and their friendships. To my research collaborators, I have learned so much from each and every one of you and it has been a tremendous pleasure working with all of you. I give special thanks Chris Diehl whose guidance, advice, and friendship have been invaluable not only for my time at graduate school but also in defining my long term goals.

I would like to thank the department staff for all their hard work and always being so helpful with dealing with the logistics of being a graduate student. I especially thank Fatima Bangura and Felicia Chelliah whose positive attitude and beautiful smiles were always something to look forward to and whose active community service has inspired me to do more in my own community.

Last, but certainly not least, I would like to thank my family. To my extended family, especially my many cousins, thank you for always one of my biggest sources of support, encouragement, and laughter. To my siblings, Anna Lisa Namata Licud and Jonathan James Namata, thank your for a lifetime of always being there and being my oldest and closest confidants. To my future wife, Ivy Pimentel, thank you for all your patience, your heart, and being all that you are. I look forward to our life together. The final acknowledgement of this dissertation goes specifically to Galileo B. Namata and Melba S. Namata, my parents. Thank you for the unconditional love, endless encouragement, and for all the many sacrifices you made to get me where I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is a growing wealth of data describing networks of various types including social networks, communication networks, transportation networks, and biological networks. At the same time, there is growing interest in analyzing these networks in order to uncover (1) general laws that govern their structure and evolution, and (2) patterns and predictive models to develop better policies and practices. However, a fundamental challenge in dealing with this newly available observational data describing networks is that the data is often of dubious quality—it is noisy and incomplete—and before any analysis method can be applied, the data must be cleaned, missing information inferred, and mistakes corrected. Skipping this cleaning step can lead to flawed conclusions for measures as basic as the label and degree distribution; for more complex analytic queries, the results are even more likely to be inaccurate and misleading. In this dissertation, we identify and develop approaches to the inference tasks involved in addressing common deficiencies in network data.

Deficiencies in network data can be caused by errors in the set of nodes, edges, and attribute values. Determining the nodes is often challenging because the nodes are often constructed from data in which the identifiers are ambiguous. Social networks, for example, can be generated using name mentions from the text of email communications. Name references, however, are typically ambiguous, relying on a

shared context between the individuals communicating that may not be immediately available. For example, in the message "How's John doing today?" there is a shared context that a common friend named "John" is ill. We need this shared context to uniquely identify the person to which this name mention refers. Due to this ambiguity when creating nodes for people from name mentions, multiple nodes which refer to the same underlying individual may be incorrectly instantiated. To resolve this deficiency, these duplicate nodes must be merged together, a task referred to as *entity resolution*.

Next, the set of edges between nodes are often spurious and incomplete. In using high throughput experiments for generating protein networks, for example, the presence of spurious and missing links have been shown to be as high as 17% and 51%, respectively, even for well studied organisms [76]. Similarly, while we may have an accurate set of edges observed, the observed edges may not semantically be what we are interested in analyzing. For example, we maybe interested in studying the managerial relationships within a company social network but the only available relationship maybe observed communication relationships. Directly using the communication relationships to represent managerial relationships is incorrect given that communications reflect multiple types of relationships (e.g., friendships, adversarial, family). To acquire the network structure desired, spurious edges must be removed and missing edges must be inferred, a task referred to as *link prediction*.

Beyond the structure of a network, an important and widely studied deficiency in network data is that the node attribute values of interest are often only partially observed. For example, while the text of papers in a citation network maybe avail-

able, the topics of those papers may not be provided. Typically, there is a cost (e.g., time, money, resources) associated with acquiring attribute values and often the available budget does not allow for acquiring the values of all the nodes. Consequently, the missing attribute values must be inferred, a task referred to as *collective classification*.

There has been a significant amount of work in the individual tasks of entity resolution, link prediction, and collective classification. In practice, however, these problems do not occur in isolation. Networks with duplicate nodes are also very likely to have missing and spurious edges and missing attribute values. In such cases, entity resolution is needed to infer the correct set of nodes over which the edges of link prediction are defined and the labels of collective classification are assigned. Similarly, link prediction edges can be used to guide which nodes should be merged in entity resolution, as well as to guide in the prediction of labels in collective classification. The attribute values provided by collective classification may also guide the entity resolution and link prediction. As this highlights, these tasks are inherently inter-related tasks that must be performed simultaneously to clean and complete the network. We define this joint application of entity resolution, link prediction, and collective classification as the problem of *graph identification*.

In the next section, we discuss various examples of entity resolution, link prediction, and collective classification, with an emphasis on highlighting how they fit in the problem of graph identification. The examples were chosen to demonstrate the applicability and impact progress work in these tasks, particularly under graph identification, can achieve in multiple domains. The examples also show the diver-

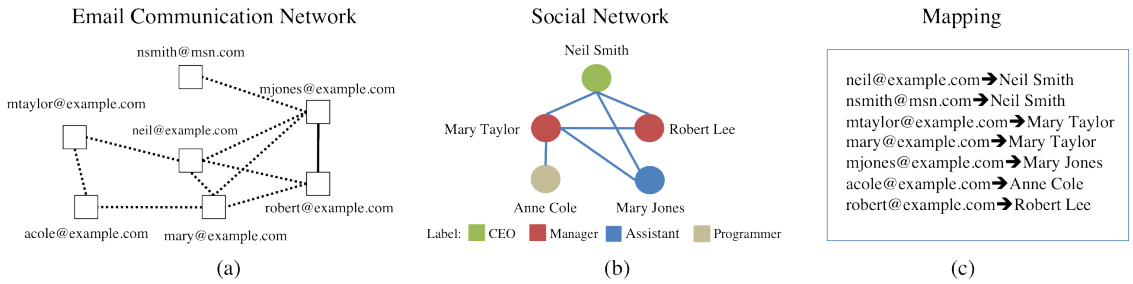


Figure 1.1: An illustration of graph identification. a) shows the input graph, which represents a communication network, where the nodes are email addresses, the edges are email communications, and the edges have attributes describing the communication content b) shows the output graph, which represents the social network, which is identified, or inferred, from the input graph. The nodes in the output graph represent entities (people), and the edges represent social relationships, in this case employee-manager relationship. In addition, nodes in the output graph are labeled with their functional role in the company.

sity in the features and types of inferences used within and across tasks in graph identification.

1.1 Motivating Examples

1.1.1 Social Network Analysis

Suppose we wish to understand and analyze the social network of a large organization. Specifically, we wish to analyze the organizational structure, including the managerial relationships and roles of the individuals. For certain organizations,

it may be very difficult, if possible, to gather such a network directly. What may instead be available for such an organization are archived email communications [90]. These communications define a network where nodes represent email addresses, edges represent communication between the email addresses, and attributes for these nodes and edges may include traffic statistics (e.g., frequency of communications) and content (e.g., presence of a word or phrase in an email). This available network, however, is inherently inappropriate for our analysis. To illustrate, consider the small example networks shown in Figure 1.1. The nodes in the communication network do not accurately reflect the individuals in the organization. If we perform analysis, substituting email address nodes for people nodes, even a basic statistic, like the number of individuals, would be inflated by the fact that people have multiple email addresses (i.e., *mary@example.com* and *mtaylor@example.com* both belong to Mary Taylor). Moreover, the communication network links are not the same as the desired social relationships between individuals (i.e., email communications exist between *robert@example.com* and *mjones@example.com* although their users, Robert Lee and Mary Jones, do not share a managerial relationship). The attributes for our analysis are also not given (i.e., the email addresses are not explicitly annotated with roles). Although the communication network is not directly appropriate for our task, we can use the information in the communication network to infer the appropriate social network. This requires identifying the people and the correspondence of email addresses to people (these may be email addresses which have similar writing and communication patterns), managerial relationships (who are likely to email each other regarding work events), and their roles (reflected

in the content of communications and with whom they communicate).

1.1.2 Protein Networks

In recent years, protein-protein interaction (PPI) networks from high throughput experiments have become a widely studied source of data for understanding biological processes in organisms. These networks have been used to explore various characteristic about proteins from protein essentiality [186, 12] and function annotation [34, 119], to patterns of how these proteins interact to perform higher level functions [10, 14, 45]. Ideally, an accurate and complete protein network, where the proteins are annotated with their function, all edges are accounted for, and proteins are mapped to their complexes, is available. However, a fundamental challenge of studying these networks is that these networks are notoriously noisy and incomplete. Comparisons of high confidence networks of well annotated species show as little as 9% overlap [76]. Estimates for the presence of false links and missing links have been shown to be as high as 17% and 51%, respectively, for well-studied organisms [76]. Moreover, most proteins, even for comparatively well-annotated species like yeast and worm, have most proteins without any functional [153] or complex annotations [117]. Although this available network may not be ideal for our analysis, we can use it to infer a network that is more suitable. We can use the known annotations of some proteins to infer the missing annotations of proteins related by observed or predicted interactions [119, 34]. Similarly, we can use auto-correlation between the function of interacting proteins, as well as attributes like cellular localization, to

infer the true set of interactions [187, 158]. We can also use the topology of the PPI and the functional enrichment common in complexes to predict protein complexes [10, 190].

1.1.3 Internet Topology

As Internet usage continues to grow, it is becoming increasingly essential that we understand the structure and design of the Internet in order to understand its vulnerabilities and limitations [165]. Ideally, for example, we would like to have a map of the Internet which shows all routers, information about these routers (such as type or geographic location), the administrative domain of the routers (known commonly as autonomous systems), and the existence and types of relationships between those autonomous systems. The Internet, however, is not owned or managed by a single organization. Instead, the Internet is a collection of networks run by different Internet Server Providers (ISPs) who do not publish many of the details regarding their networks. Generally, the available maps of the Internet are created using tools which provide only a partial view of the full network and are error prone. For example, router level networks created using TTL-limited probes (i.e., traceroute) tend to over inflate the true number of routers, as well as incorrectly record the existence of links between those routers [156]. As in the previous examples, while the available networks may be ill-suited for our analysis, we can use them to infer a network which is appropriate for further analysis. We can infer which IP addresses belong to the same router and, similarly, which routers belong

to which autonomous system by using their attributes (e.g., geographic location and DNS names) and their observed and inferred connectivity. We can also infer the existence and types of relationships between the autonomous systems by looking at the attributes and connectivity of its routers [46].

1.2 Outline and Contributions

In this dissertation, we present our research on entity resolution, link prediction, and collective classification first as independent tasks and then as tasks within the problem of graph identification. We begin by providing some additional background for the tasks of entity resolution, link prediction, and collective classification in Chapter 2. We provide formal definitions, describe the characteristics and challenges, and survey previous approaches to these problems. We also describe previous work which looks at jointly performing pairs of these tasks, including domain-specific problems, related to the problem of graph identification.

Next, we discuss our work on entity resolution in Chapter 3. Specifically, we look at entity resolution applied to name references in email communications. Name references are the various forms of an individual's given name, along with their nicknames, that may appear in the body of an email communication. Name references are an important element in understanding the social network. Before we can process the email content in an archive and associate activities and other attributes with individuals, we need to understand who is participating in and is mentioned in the email communications. For example, consider an email containing

the body:

"Will Bob be joining us later? See you soon!"

In this example, the word "Bob" is a name mention to a specific person whose identity, given the topic of conversation along with the name reference, is clearly known by both the email sender and recipient. Yet to someone without knowledge of the context, the reference is meaningless and consequently it is unclear if the message is in regard to a social occasion (e.g., lunch) or business event (e.g., meeting). In this work, we test the hypothesis that communications around the name mention can provide this missing context. We developed unsupervised algorithms which leverages communication traffic (quantity, time, and direction) to resolve the given name mentions in a communication. We evaluate our approach on manually annotated name mentions over corporate emails and demonstrate how effective traffic information alone can accurately disambiguate name references.

In Chapter 4, we present our work on link prediction as applied to predicting social relationships. As with the work described in Chapter 3, our work in link prediction is applied in the context of email communication networks. In this work, we consider the scenario from domains such as intelligence analysis and litigation support where an analyst is attempting to reconstruct a representation of the social network from the data with minimal context. For this setting, the link discovery process of identifying relationships is inherently a collaborative process between human and machine. Consequently, the goal of the link prediction task here is not

to simply restore the missing edges, but rather to focus the analyst's attention on both the (1) relevant communications relationships that express the given social relationship of interest (2) and the relevant message traffic that supports this association. Our hypothesis in this work is that the words used in the communications, as well as the amount and direction of traffic between individuals, are indicative of the relationships individuals share. We propose a supervised ranking approach to test this hypothesis and evaluate the approach on predicting subordinate-manager relationships in a major corporation. We show that not only does our approach highlight the most likely subordinate-manager relationships for a given individual, it also provides a natural way of highlighting the relevant communication and text which supports that decision.

Next, we present our work in collective classification in Chapter 5. While traditional collective classification aims to learn a predictor that accurately labels all available data, in this work we consider the setting in which one is primarily interested in labeling a particular subset of nodes which we refer to as the *query set*. For example, when labeling a social network, we may only be interested in the labels of key high-ranking or influential individuals. Accurate classification of the rest of the social network may only be useful to help collectively classify the targeted nodes. Furthermore, in many practical scenarios, labels and network structure may not be immediately available for all nodes, and certainly are not available for the nodes in the query set. Instead, there is a cost for acquiring this information. We therefore define the problem of query-driven collective classification in an active surveying setting. The goal in this task is to identify the labels and structural

information to acquire, given some budget constraints, to maximize the collective classification performance over the query set. The underlying hypothesis in this work is that active learning approaches which specifically address the query-driven nature of the problem will yield better performance compared to traditional active learning approaches. Leveraging common assumptions on feature and structural smoothness, We propose a novel adaptive algorithm, ASQ2C, and empirically show its superiority over standard active learning approaches on four real-world datasets.

In many real-world settings, the deficiencies addressed by entity resolution, link prediction, and collective classification are not found in isolation and all need to be addressed to infer the desired complete and accurate network. Furthermore, because of the dependencies typically used in these tasks, the tasks are inherently inter-related and must be performed jointly. In Chapter 6 we formally define the general problem of graph identification which simultaneously performs these tasks; removing the noise and missing values in the observed input network and inferring the complete and accurate output network. The main hypothesis in this work is that jointly performing these tasks can yield better overall performance. We present a novel approach to graph identification using a collection of Coupled Collective Classifiers, C^3, which can not only capture the variety of local features typically used for each task, but also intra- and inter-dependency required in order to correctly infer nodes, edges, and labels in the output network. We discuss variants of C^3 using different learning and inference paradigms and show the superior performance of C^3, in terms of both prediction quality and runtime performance, over approaches which look at each task individually and over previous joint approaches.

A major obstacle in studying network data is the lack of a software system which has support for representing and applying various tasks (such as those defined in this chapter) on these networks. While various implementations are available for specific tasks like visualization, clustering, and collective classification, most systems are ad-hoc and are developed solely for the task at hand. Moreover, the different systems are developed with various architectures, programming languages, and rarely share even a common input and output format. Consequently, it is very difficult to directly use these systems, particularly for more complex tasks like graph identification. As part of our dissertation work, we developed the Graph Alignment, Identification, and Analysis (GAIA) software library which provides a common, reusable interface for various problems involving relational data. We provide an overview of GAIA in Chapter 7 describing its goals and high level architecture. We also provide an overview of the tasks it currently supports including entity resolution, link prediction, collective classification, graph identification, clustering, active learning, data generation, sampling, and analysis.

To summarize, the rest of the thesis is organized as follows. We first discuss related work in Chapter 2. We then discuss our work in entity resolution, link prediction, and collective classification in Chapters 3, 4, and 5 respectively. We present our work in graph identification, which jointly performs the preceding tasks, in Chapter 6. We then introduce the GAIA software library to perform these and a variety of other tasks on network data in Chapter 7. Finally, we summarize our contributions, discuss future directions, and conclude in Chapter 8.

Chapter 2

Related Work

In this chapter, we begin by discussing work in entity resolution, link prediction, and collective classification. We then discuss work related to graph identification problem including work on joint inference models and similar domain-specific problems. To illustrate these different problems, we use a simple author collaboration network (shown in Figure 2.1 – Figure 2.4). In the collaboration network figures, the nodes represent authors and the edges between the authors indicate that the authors have co-authored at least one paper together. The shading of the nodes indicates the research area of the authors; to make it simple, here we assume there are just two areas, shown either in white (i.e., theory) or gray (i.e., systems), if observed, and shown as a '?' if it is unobserved.

2.1 Entity Resolution

Many networks have uncertain and imprecise references to real-world entities. The absence of identifiers for the underlying entities often results in noisy networks which contain multiple references to the same underlying entity. In this section, we look at the problem of resolving which references refer to the same entity, a problem known as *entity resolution*.

Examples of entity resolution problems can be found in many domains, often

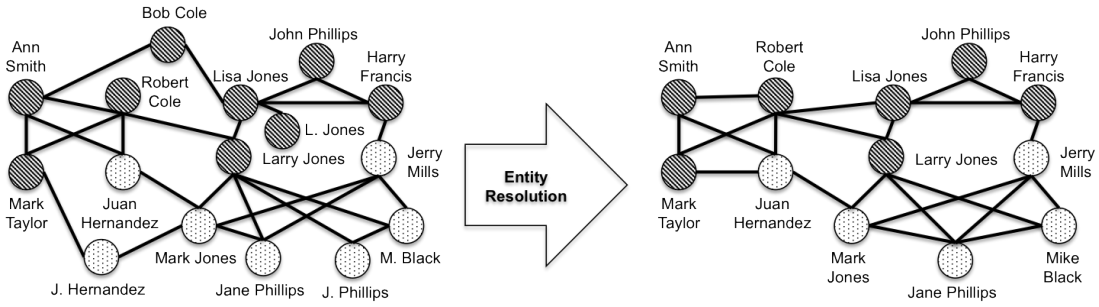


Figure 2.1: Example of a entity resolution problem. In this example, the nodes on the left are ambiguous due to variations in the spelling of their names. While attributes may suffice to resolve the entities in some cases (e.g., Juan Hernandez and J. Hernandez are likely the same person due to the similarity in their names), some cases (e.g., J. Phillips can refer to either Jane or John Phillips) it may not. However, if we use the edges (i.e., both Jane Phillips and J. Phillips have collaborated with Larry Jones), we may be able to improve our predictions.

under different names. The earliest applications of entity resolution is on medical data [126, 125, 59, 183]. In this work, in a problem referred to as record linkage, the goal was to identify which medical records refer to the same individual or family. Next, in computer vision, entity resolution was applied in identifying which regions in the same image are part of the same object (the correspondence problem) [129]. Also, in natural language processing, there is interest in determining which noun phrases refer to the same underlying entity (coreference resolution) [113]. The problems of deduplication [146] and data integration [176], determining when two tuples in or across databases refer to the same entity, can also be seen as entity resolution.

There are three general categories of approaches to entity resolution: attribute-based, naive relational, and collective relational. Attribute-based approaches are the traditional approaches to entity resolution which rely solely on the attributes of the reference nodes. Given two reference nodes, $r_i, r_j \in R$, the attribute-based approaches generally make use of a similarity measure [80, 81, 183], $sim_A(r_i, r_j)$, or a weighted combination of multiple similarity measures, over the attributes of the reference nodes. More recently, naive and collective relational approaches have been proposed which take the edges between these nodes into consideration. The naive relational approaches consider the attribute similarity of related reference nodes [7, 85]. The collective relational approaches, on the other hand, use the relationships to make decisions jointly [51, 16, 112, 133, 160].

A major issue in entity resolution is that it is a known hard problem computationally for large networks; a naive algorithm is $O(N^2)$ where N is the number of references in the network. For many networks, it is infeasible to compare all pairs of references for approaches which use expensive similarity measures. Similarly, for many probabilistic models, it is infeasible to explicitly represent all the variables required for the inference. Thus, efficiencies have long been a focus for research in entity resolution. One mechanism for doing this involves computing the matches efficiently and employing techniques commonly called 'blocking' to place nodes into disjoint 'blocks' using cheap and index-based similarity computations [74, 180]. The number of potential pairs is greatly reduced by assuming that only pairs of nodes in the same block can be co-referent pairs. Another mechanism, proposed by McCallum et al. [110], relaxes the use of disjoint blocks and places nodes into possibly

overlapping subsets called 'canopies.' Potential co-referent pairs are then restricted only to pairs of nodes which share at least one common canopy.

Another issue in entity resolution is referred to as "canonicalization" [42, 182]. Once the reference nodes are resolved to their corresponding entities, there is the problem of constructing a standard representation of the entity from the attributes of those references. In particular, canonicalization resolves the inconsistencies in the attributes among the reference nodes. Simple heuristics for determining the appropriate values for the attributes and edges of an entity based on the attributes of the references are possible; often these amount to choosing the longest string, or the most recently updated value. Such approaches, however, are not robust to noisy and incomplete attributes. Another approach is, instead of returning a single value for an attribute, keeping all the values, returning a ranked list of the possible values and edges [170, 8]. When there are a large number of references, however, the ranked list may be too long. Culotta et al. [42] addresses this by using adaptive similarity measures to select values in order to create a standard representation most similar to each of the different records. A unified approach was also proposed by Wick et al. [182] which performs entity resolution and canonicalization jointly using a discriminatively-trained model. We note that the problem of canonicalization is related to the problem of performing node labeling after entity resolution. In the case of a noisy network, our node labeling problem can be cast as the canonicalization of the predicted entity. In general, however, canonicalization resolves inconsistencies in the observed attributes of the merged references while node labeling may also include inferring the value of some previously unobserved attribute.

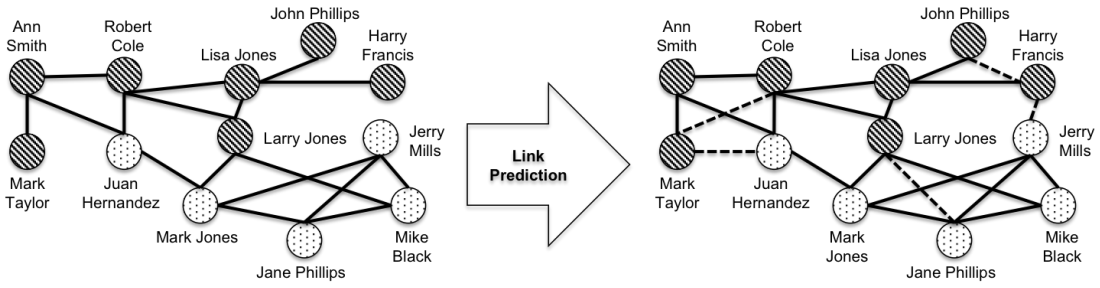


Figure 2.2: Example of a link prediction problem. The graph on the left represent a collaboration network at time t, and the graph on the right represent the predicted collaboration network at time $t + 1$. Predicted collaboration edges are highlighted using a dashed line.

2.2 Link Prediction

Link prediction is a challenging problem that has been studied in various guises in different domains. For example, in social network analysis, there is work on predicting friendship links [191], event participation links (i.e., co-authorship [131]), communication links (i.e., email [131]), links representing semantic relationships (i.e., advisor-of [169], and subordinate-manager [48]). In bioinformatics, there is interest in predicting the existence of edges representing physical protein-protein interactions [166, 187, 76], domain-domain interactions [45], and regulatory interactions [5]. Similarly, in computer network systems there is work in inferring unobserved connections between routers, as well as inferring relationships between autonomous systems and service providers [165]. There is also work on using link prediction to improve recommender systems [57, 77], website navigation [193], surveillance [78],

and automatic document cross referencing [118].

There are the two general categories of the current link prediction models: topology-based approaches and node attribute-based approaches. Topology-based approaches[99, 187, 35] 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 [169, 135, 131, 140, 62] 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 $O(|V|^2)$ [137]. Consequently, the prior probability of edge existence is very small. This causes many supervised models, which naively 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 $O(|V|^2)$. 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 common way to do this is to partition the set of nodes where we only consider potential edges between nodes of the same partition; edges between partitions are not explicitly modeled and are assumed not to exist [3, 187]. This is useful in many domains where there is some sort of natural partition among the nodes available (e.g., geography in social networks, location of proteins in a cell) which make 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 [99, 48].

2.3 Collective Classification

A traditional problem in machine learning is to classify objects: e.g., given a corpus of documents classify each according to its topic label; given a collection of email communications determine which are not spam; given individuals in a collaboration network determine a characteristic of that individual; 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).

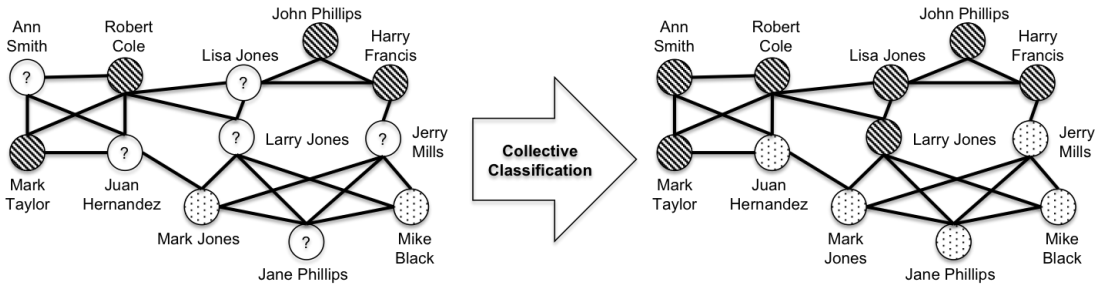


Figure 2.3: Example of a collective classification problem. Nodes with a question mark are nodes whose labels are unknown. Collective classification uses the attributes and labels of neighboring nodes. Ann Smith, for example, is likely to have the same research area as her co-authors, Robert Cole and Mark Taylor.

In graphs, however, studies have shown that predicting the labels of nodes can benefit by using correlations between the node label and the labels of related nodes. For example, in the collaboration network in Figure 2.3, nodes with a question mark represent authors whose research areas are unknown. While we can use attributes of the author (e.g., titles of their papers) to predict the label, we can also use the research areas of the other authors they share a co-authorship edge with. The author, Ann Smith, is likely to work in theory given she has only co-authored with individuals in the theory field.

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 [104], 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 class label. Although relational classifiers have been shown to perform well in some domains, overall the results have been mixed. For instance, although there have been reports of classification accuracy gains using such techniques over traditional classification, in certain cases, these techniques can harm classification accuracy [30]. The second category of algorithms go beyond that by not only using the known attributes and labels of related nodes, but to also use the predicted labels of other nodes whose labels are unobserved [30, 123, 101, 65, 95, 168]. For instance, going back to the classification example in Figure 2.3, authors which share a co-authorship edge to other authors *predicted* to have a certain research area, are likely to work in the same area.

2.4 Joint Inference

We define graph identification as a probabilistic joint inference task in which we must infer the nodes, edges, and node labels of a hidden graph based on evidence provided by the observed network. This in turn corresponds to the problems of performing entity resolution, link prediction, and collective classification to infer the hidden graph. While we are the first to define and jointly solve all the inference tasks involved, there is related work in the joint inference of subsets of these tasks and other tasks. In this section, we provide an overview of this work.

Most previous work explore these components of graph identification independently. Although they exploit the intra-dependence of the predictions in each component, there is little work in exploiting the observation that the components

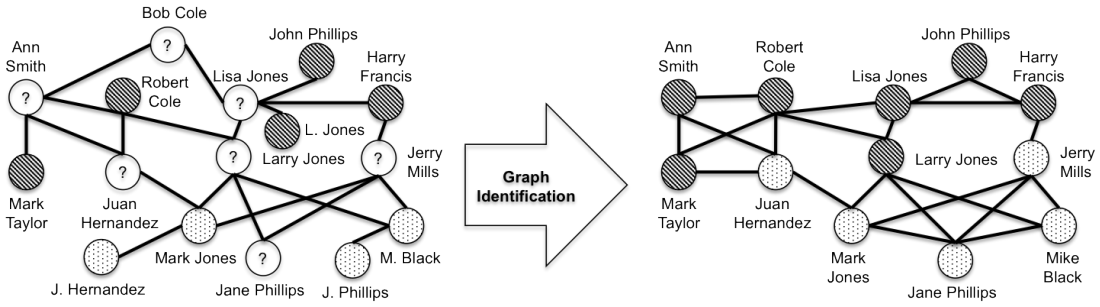


Figure 2.4: Example of a graph identification problem. In this example, the nodes on the left are ambiguous due to variations in the spelling of their names, there are unobserved edges between the ambiguous nodes, and nodes with a question mark are nodes whose labels are unknown. Performing graph identification requires applying jointly performing the entity resolution, link prediction, and collective classification tasks.

are inter-dependent. The few that explore this inter-dependence mainly come from the statistical relational learning area where various general frameworks have been proposed which model the dependencies between predictions. One example is the work of Getoor et al. [64] on Probabilistic Relational Models (PRM). Their work explored using PRMs when there is both attribute and structural uncertainty. Similarly, there is work by Taskar et al. [169] on Relational Markov Networks (RMN). Taskar applied RMNs to the task of jointly inferring the labels and the existence of edges 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. [17] proposed a gen-

erative model which jointly applies entity resolution and node labeling to movie data.

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 [142] and sequential learning [141]. Similarly, Heitz et al. [72] proposed cascaded classification models (CCM) for applying a set of models involved in the task of holistic scene understanding. To our knowledge, previous work in joint models has not formulated the complex structured prediction problem in graph identification as interacting components which 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 [134, 145, 182], network mapping in computer networks [156, 164, 165], and biological network inference in bioinformatics [107]. 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. Consequently, work in these two problems can be formulated as instantiations of the more general problem of graph identification.

Another related line of research is the problem of modeling network evolution. A number of global properties of graphs have been found in real networks [55, 11, 98, 97, 128]. Properties include scale-free degree distributions [2, 56, 11], the small-world phenomenon [179, 11], and densification and shrinking diameters of dynamic networks over time [98]. An important aspect of these models is modeling how to randomly generate edges between the nodes of the graph to capture these properties. The preferential attachment model [11], for example, creates edges based on the degree of nodes (i.e., higher degree nodes are more likely to be incident to more edges). The Forest Fire model [98], on the other hand, generate edges for nodes in an epidemic fashion, growing outward from some initial set of neighboring nodes. As before, however, the focus of this work is only the topology while graph identification is also interested in the attributes of the graph. Also, work in this area are mainly interested in generating random graphs which exhibit some global property while the transformations in graph identification are interested in inferring a particular graph given some noisy or incomplete input.

Chapter 3

Entity Resolution of Name References in Email Archives

In this chapter, we discuss our work in entity resolution for email communications. We look at name mentions in email communications and develop unsupervised models for ranking individuals by the likelihood of being the target of a given name mention. In this work, we focus on how to leverage temporal traffic information (i.e., time, frequency, and number of communications between individuals around the time of the name mention) to infer the target of the name mention.

3.1 Introduction

Within the networked world, email has become a ubiquitous form of global communication. Whether communicating with friends or colleagues in a local area or halfway around the world, email allows us to maintain or develop relationships with others at any distance. Given email traffic is a reflection of the relationships in an underlying social network, email archives present a potentially rich collection of evidence that can be used to infer the structure, attributes and dynamics of the social network. The challenge is to infer these properties from email data that is often ambiguous, incomplete, and context-dependent.

Email collections contain both structured and unstructured data. The structured data or metadata indicates which parties communicated and when the commu-

nication occurred. By focusing solely on the metadata, we can identify communication patterns, but we cannot easily ascribe meaning to the underlying relationships. The unstructured data in the body of the email can clarify the roles of individuals and their relationships with others. Yet without the appropriate context, an outside observer may find a message provides little insight.

When communicating with others, people constantly rely on shared context to simplify communication. Shared context is common knowledge among individuals that allows them to use ambiguous references which are clear within the shared context. A common example of this occurs when two people refer to a mutual friend by a first name or a nickname in conversation.

"How's John doing today? Is he feeling better?"

Given the topic of conversation along with the name reference, it is clear to both parties who John is. Yet to someone without knowledge of the context, the reference is meaningless.

Consider the problem of exploiting name references in the email body. Name references are an important element in understanding the social network. Before we can process the email content in the archive and associate activities and other attributes with individuals, we need to infer the number and identities of the individuals generating the observed traffic. Each individual has two classes of references: *network references* and *name references*. Network references in the context of email are simply the individual's email addresses. Note that this is potentially a many to

many mapping: individuals may have multiple email addresses and a single email address may serve more than one individual. There is also a temporal component; an individual may have one email address for the time they are in one position in the company, but when they change roles within the company, perhaps moving to another division, their email address may change. Name references are the various forms of an individual's given name along with their nicknames that may appear in the email body. In order to define an individual's identity and draw broader connections across emails in the archive, we need to be able to map both name references and network references to the individual.

In this chapter, we focus on the problem of resolving ambiguous name references, specifically first name references, to network references. The core challenge in this problem is identifying ways to exploit context from the email archive to effectively resolve the ambiguity. We describe this in the next section. Next we formally define the general problem of name reference entity resolution. Then we discuss the types of context available that can potentially be exploited. We investigate several different approaches, which vary in the context features and temporal models used, and introduce a methodology for evaluating their performance. Finally we present results from our algorithm evaluation on the Enron email archive and conclude with thoughts on future work.

3.2 Exploiting Context

When reading email, what types of context do we exploit to resolve ambiguous name references? In addition, what context does an email collection offer when analyzing relationships retrospectively? Below is a list of some of the contextual cues available to us for understanding name references:

- The participants in the conversation
- The larger group of people known by the participants in the conversation and the types of relationships among them
- The individuals that the participants in the conversation have recently communicated with, either before or after the email was sent
- The topic of conversation in the email
- Recent topics of conversation among the participants and others outside the current conversation, either before or after the email was sent
- Cues contained within other emails in the thread
- Related name references within the current email
- Prior knowledge linking individuals to topics of conversation

This list of contextual cues is by no means exhaustive. Yet it reminds us of the two broad classes of context that email captures: *social context* (who's talking?) and *topical context* (what are they talking about?).

Our long term goal is to exploit both to characterize the underlying social network, as each form of context can help clarify ambiguities in the other. Yet the challenge of capturing and exploiting dynamic topical context is a significant research thrust on its own, as evidenced by the work in the topic detection and tracking community [6].

Our focus in this chapter will be to investigate the discriminative power of dynamic social context. We want to first understand the performance of algorithms that leverage the patterns of communication among network references to estimate the mapping between name and network references.

3.3 Problem Definition

Let $\mathcal{E} = \{e_i\}$ be a set of email addresses observed in the email collection and let $\mathcal{N} = \{n_j\}$ be a set of observed name references in the email bodies. The set $\mathcal{E}$ may be extracted from the email metadata, or the set may come from another source such as an employee directory, which lists individuals together with their emails. The set $\mathcal{N}$ is the result of an entity extraction process that identifies name references within the email bodies.

The objective of name reference entity resolution is to construct a mapping from a set of observed name references $\mathcal{N} = \{n_j\}$ in the email collection to either ranked subsets of network references, $\mathcal{E}_j$, where $\mathcal{E}_j \subseteq \mathcal{E}$ or the null network reference ϕ, if no network reference is sufficiently probable. The null network reference ϕ serves two purposes. First, it is not a given that there exists a corresponding net-

work reference for each name reference. An email collection may not contain email exchanges between all individuals referenced within the email bodies. Second, the entity extraction process will incorrectly declare some terms in the email collection to be name references, for which there is no network reference. In both cases, the appropriate response is to map the given name reference to ϕ.

For each name reference n_j, the corresponding candidate set $\mathcal{E}_j$ is ranked based on the context of the name reference. A scoring function g is used to compute the strength of association $g(e_c, n_j|C_j)$ between each candidate $e_c \in \mathcal{E}_j$ and n_j, given the context C_j associated with n_j. Once all of the candidates have been scored, they are ranked in descending order and only those candidates with scores $g(e_c, n_j|C_j) > \lambda$ are retained. The most likely network reference $\tilde{e}(n_j)$ is either the candidate with the maximum score greater than the threshold or ϕ otherwise.

In this chapter, we explore the use of the email traffic context for ranking the candidate set. We define the *email traffic network* for a set of email messages $\mathcal{M} = \{m_i\}$ as follows: we have a directed hypergraph $\mathcal{G}_{\mathcal{M}}$ with the set of vertices $\mathcal{E}$ and hyperedges $\mathcal{H} = \{(e_{s_i}, \mathcal{E}_{r_i}, t_i)\}$. For each email message m_i, there is a hyperedge from the sender network reference e_{s_i}, $e_{s_i} \in \mathcal{E}$, to the set of recipients of the message, $\mathcal{E}_{r_i} \subseteq \mathcal{E}$. The attribute t_i is the time at which the email was sent.

3.4 Name Reference Entity Resolution Process

The general name reference entity resolution process is composed of three phases: *candidate set generation*, *candidate ranking* and *candidate rejection* illus-

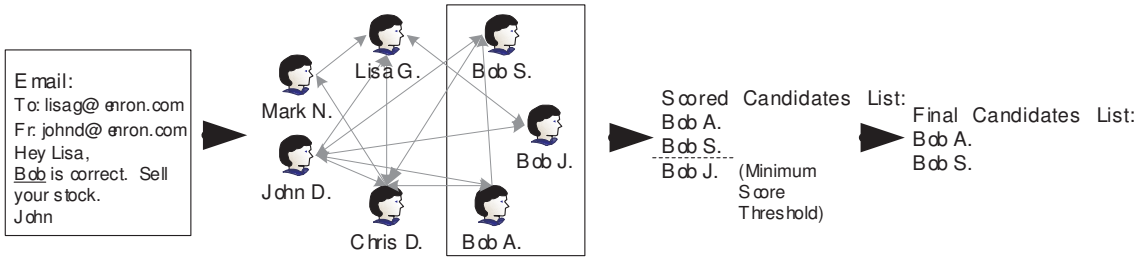


Figure 3.1: The name reference resolution process

trated in figure 3.1. Given we envision a data analyst reviewing the candidate associations in rank order to identify the true referent, our overall goal is to minimize the number of candidates the user must evaluate while identifying as many true network references as possible. When the true referent is a member of the candidate set, we want the algorithm to rank the true network reference as high as possible. Given the true referent may not be part of the candidate set at all, we also want to reject as many candidates as possible without severely impacting recall.

3.4.1 Candidate Set Generation

The role of the candidate set is to restrict our attention to a small number of likely candidates prior to scoring the candidates. In our initial approach, we use two levels of screening. We begin with the strong assumption that if any communication has occurred between the true referent and the email participants, the sender was involved. Therefore we initially restrict the candidate set to those network references where at least one email communication has been observed with the sender.

Although we expect this assumption will be true in many cases, there are

clearly instances where it will break down. For example, not all name references correspond to individuals that the email sender knows personally. Within the context of an organization, references may be made to individuals many levels removed in the management hierarchy. It is also not a given that an active relationship will be observable through email communication. The parties involved may be in close physical proximity allowing direct communication or may use other means of communication. A third possibility is that the email communications are simply not available in the email collection for one of a variety of reasons. Regardless of these factors, as we show in the results section, our approach able to achieve surprisingly high recall.

Our second level of screening relies on available name information for the network references. We assume that some name information is initially available either from the name tags attached to email addresses or from the email addresses themselves. In our initial experiments, we examine name references that match exactly at least the first or last name associated with the candidate network reference. Clearly this constraint can be relaxed by employing a string comparison function to look for close name matches.

3.4.2 Candidate Scoring

As mentioned earlier, our main interest is in defining and evaluating candidate scoring functions that leverage dynamic social context. If we begin with the presumption that name reference usage is often connected to events occurring around

the time of the reference, the question is what fraction of the name references can we resolve by ranking candidates based on the level of email traffic around the time of the reference? To explore this, we introduce a class of scoring functions and explore the sensitivity of their performance along four general dimensions.

- The relationships examined
- The time scale at which the email traffic is viewed
- The summary statistic used to characterize relationship activity during a given time interval
- The degree of traffic history considered

We consider each of these dimensions next and then describe two different temporal models which make use of features defined according to these dimensions.

3.4.2.1 Relationships

Given our assumption of direct communication between at least the email sender and the true referent, our objective is to characterize the degree of communication between the email participants, the sender and recipients $\mathcal{E}_p = e_s \cup \mathcal{E}_{r_i}$, and the candidate network reference e_c.

Specifically we consider models that exploit either solely the traffic between the sender e_s and the candidate e_c (denoted **sender-only**) or models that exploit the pairwise traffic between all the email participants, sender and recipients, and the candidate e_c (denoted **sender+recipients**). When integrating traffic from the

Table 3.1: Summary statistic definitions. $m(e_1, e_2, T_k, \mathcal{G}_\mathcal{M})$ is the number of messages sent from network reference e_1 to network reference e_2 over the time interval T_k. $\mathcal{I}(\cdot)$ is the indicator function.

Name	*Definition*
Binary, Sender-Only, Bidirectional	$\mathcal{I}(m(e_s, e_c, T_k, \mathcal{G}_\mathcal{M}) + m(e_c, e_s, T_k, \mathcal{G}_\mathcal{M})$
Count, Sender-Only, Bidirectional	$m(e_s, e_c, T_k, \mathcal{G}_\mathcal{M}) + m(e_c, e_s, T_k, \mathcal{G}_\mathcal{M})$
Count, Sender-Only, Unidirectional	$m(e_s, e_c, T_k, \mathcal{G}_\mathcal{M})$
Count, Sender+Recipients, Bidirectional(β)	$(1-\beta)(m(e_s, e_c, T_k, \mathcal{G}_\mathcal{M}) + m(e_c, e_s, T_k, \mathcal{G}_\mathcal{M}))+$ $\frac{\beta}{\lvert\mathcal{E}_{r_i}\rvert}\sum_{e_{r_i}\in\mathcal{E}_{r_i}}(m(e_{r_i}, e_c, T_k, \mathcal{G}_\mathcal{M}) + m(e_c, e_{r_i}, T_k, \mathcal{G}_\mathcal{M}))$

sender and recipients' pairwise interactions with the candidate, we want to understand the relative discrimination power offered by each and identify summary statistics that effectively leverage the relationships for candidate scoring.

3.4.2.2 Time Scale

To examine the email traffic at a given time scale, we first partition the time axis into regular intervals of duration Δt. The phase of the partition is fixed by first selecting a reference time t_0 such that $t_0 \leq t_i < t_0 + \Delta t$ where t_i is the time of the email containing the name reference. The time intervals $\{T_k\}$ are defined as $T_k = \{t' : t_0 + k\Delta t \leq t' < t_0 + (k+1)\Delta t, k \in \mathbb{Z}\}$ so that the time interval T_0 includes the time t_i of the email [1]. In our experiments, we investigate daily and weekly time intervals (denoted **daily** and **weekly**). The weekly time intervals are phased such that they begin on Sunday.

3.4.2.3 Summary Statistics

Once the time axis is partitioned into regular intervals, our next step is to compute a summary statistic or feature $s(\mathcal{E}_p, e_c, T_k, \mathcal{G}_{\mathcal{M}})$ for each interval T_k that provides an indication of relationship activity among some or all of the email participants $\mathcal{E}_p$ and the candidate e_c. We consider the following variations on computing the statistic:

Binary versus Count. For any pair of network references, for the given interval, we may either have a 0/1 indicator which denotes whether or not there has been an email exchange between the pair (denoted **binary**) or we may want to use the frequency information and keep track of the number of messages exchanged (denoted **count**).

Unidirectional versus Bidirectional. For any network reference, we may be interested in only the messages sent from the network reference to the candidate reference (denoted **unidirectional**) or we may be interested in bidirectional exchanges where the candidate and network references can take on either the sender or recipient roles (denoted **bidirectional**)

As mentioned earlier, we can distinguish models which make use of the sender-only traffic information versus the sender+recipients traffic information. In the latter case, we introduce the parameter β to weight the sender versus recipient contributions. Table 3.1 summarizes the statistics used in the experiments.

[1] Although the definition of T_0 is dependent on the email of interest, we will not explicitly indicate this dependence to avoid additional complexity in the notation.

3.4.2.4 Integrating Traffic History

The final step in computing the candidate score $g(e_c, n|C)$ given the context $C = \{\mathcal{E}_p, T_0, \mathcal{G}_\mathcal{M}\}$ involves integrating the summary statistics for time intervals around the time of the email containing the name reference. We compute the local time average of the summary statistics using either a non-causal autoregressive (denoted **AR**) or moving average filter (denoted **MA**) that incorporates both future and past traffic patterns around the time of the name reference. The autoregressive filter is defined as

$$
\begin{aligned}
& g_{\text{AR}}(e_c, n|\mathcal{E}_p, T_k, \mathcal{G}_\mathcal{M}) \\
&= \frac{(1-\alpha)}{2} g_{\text{AR}}(e_c, n|\mathcal{E}_p, T_{k-1}, \mathcal{G}_\mathcal{M}) + \frac{(1-\alpha)}{2} g_{\text{AR}}(e_c, n|\mathcal{E}_p, T_{k+1}, \mathcal{G}_\mathcal{M}) + \alpha s(\mathcal{E}_p, e_c, T_k, \mathcal{G}_\mathcal{M}) \\
&= \frac{\alpha}{2} \sum_{i=0}^{\infty} (1-\alpha)^i (s(\mathcal{E}_p, e_c, T_{k-i}, \mathcal{G}_\mathcal{M}) + s(\mathcal{E}_p, e_c, T_{k+i}, \mathcal{G}_\mathcal{M}))
\end{aligned}
$$

while the moving average filter is defined as

$$g_{\text{MA}}(e_c, n|\mathcal{E}_p, T_k, \mathcal{G}_\mathcal{M}) = \frac{1}{\text{2M+1}} \sum_{i=-M}^{M} s(\mathcal{E}_p, e_c, T_{k-i}, \mathcal{G}_\mathcal{M}).$$

In practice, when evaluating the AR filter, we terminate the summation once a convergence criterion is met. The degree of traffic history incorporated into the candidate score $g(e_c, n|\mathcal{E}_p, T_0, \mathcal{G}_\mathcal{M})$ is controlled by the parameters α for the AR model and M for the MA model.

3.4.3 Candidate Rejection

Once the scores have been computed for all network references in the candidate set, the candidates with a score below the specified threshold λ are removed from

the candidate set. The objective of candidate rejection is to remove candidates that are deemed unlikely to correspond to name references without rejecting a significant fraction of true referents. The degree of performance achieved is dependent on the ability of the scoring function to separate the true referents from the other candidates.

Within the context of the models proposed above, performance is clearly depends on the following two factors. First, it is dependent on the legitimacy of the general assumption that a high degree of communication activity around the time of the name reference is indicative of a potential correspondence between a name and network reference. Second, performance is also dependent on the model's characterization of what qualifies as a high degree of traffic. All relationships are clearly not equivalent. Yet our baseline models do not attempt to capture external factors influencing the relationship activity. We will revisit these issues in later discussion.

3.5 Experiment Design

With a set of models defined, the next major task at hand is evaluating their performance on a representative dataset. The bulk of our efforts to date have focused on data preparation, ground truth generation and definition of evaluation protocols. A number of subtle but important issues arise as one considers the various elements of the overall experiment. We review all aspects of the approach in the following sections.

3.5.1 Dataset Preparation

3.5.1.1 The Data: Enron Email Corpus

With the recent release of the Enron email dataset [157], researchers have been given a unique opportunity to glimpse inside a large corporation and observe a subset of email traffic among the employees. The Enron email dataset is the collection of email from the folders of 151 Enron employees. The data is available in several forms. CMU first released the original email data. Since then USC/ISI and more recently UC Berkeley have released normalized forms of the data in a MySQL database. Our results are based on the USC/ISI version of the dataset. There are over 250000 email messages in the dataset with the majority of the traffic occurring in the 2000-2002 time frame.

We initially chose to resolve name references in only those emails exchanged between the core 151 employees. This was done primarily to reduce confounding effects of observability in our experiments. Given we can only observe pairwise relationships where at least one of the participants is a member of the set of 151 employees, constraining the set of emails in this way guarantees that all relationships we will consider in the resolution process are observable in the email collection, assuming emails haven't been lost or deleted.

There are 7644 emails in the ISI database that were exchanged among the 151 employees. A non-trivial number of duplicate emails exist that need to be removed to avoid skewing the results of the analysis. After deduplication of this set, 6550 emails remain. This is the set of emails from which the name references will be

extracted.

3.5.1.2 Extracting Enron Employee Names

To support named entity extraction and candidate set generation, we constructed a network reference set $\mathcal{E}$ of 7864 Enron email addresses and a corresponding list of employee names by parsing the email addresses. In total, there are 29176 *enron.com* email addresses in the collection. This includes employee email addresses along with group mailing lists. Given the most common email address format often corresponding to employees is *<name1>* . *<name2>* @*enron.com*, we parsed these addresses and saved only those where either *name1* or *name2* matched a first or last name in the employeelist table in the ISI database. This reduced the list to 7713 email addresses that are distinct from the email addresses listed for the 151 employees in the ISI database.

As others have noted, some employees have multiple email addresses in the collection. We believe that in most cases this is due to an employee moving within the company. Therefore each email address and its associated relationship structure characterizes the employee's role over a certain time period in the company. We chose not to de-duplicate the email addresses in order to preserve this context.

3.5.1.3 Constructing the Email Traffic Network

The hypergraph $\mathcal{G}_{\mathcal{M}}$ representing the email traffic network captures the observed traffic exchanged between the 7864 Enron email addresses in $\mathcal{E}$. Since the

Enron email collection is the union of email folders corresponding to the given 151 Enron email addresses, $\mathcal{G}_{\mathcal{M}}$ only captures the traffic exchanged between those 151 email addresses and the remaining 7713 email addresses in $\mathcal{E}$. There are 64449 emails in the ISI database that were exchanged among the 7864 email addresses. After deduplication of this set, 55395 emails remain. Therefore $\mathcal{G}_{\mathcal{M}}$ is composed of 7864 nodes and 55395 hyperedges.

3.5.1.4 Detecting Name References

To detect name references in the email bodies, we initially scan through the emails searching for words that match exactly one or both first and last names of an employee on the list of 7864 Enron employee names. We also merge adjacent partial name matches, assuming in most cases this results in a full name not listed on the employee list.

For our initial experiments, we chose to focus on resolving first name references to others outside of the email conversation. Therefore to filter out name references not of interest, we saved only partial name detections that matched one of the 151 Enron employee first names. Then we filtered out first name references at the beginning or end of the email text composed by the sender, assuming those are references to either the sender or recipients.

3.5.2 Ground Truth Generation

To evaluate algorithm performance, we manually identified the true network references associated with a set of first name references. In some cases, the true referent was obvious from other name references in the sender's message or the attached message. In others, we needed to search through the traffic to find other emails in the thread or previous conversations to clarify the reference. When multiple email addresses appear to correspond to the referenced individual, the email address in use around the time of the name reference is chosen as the true referent.

After this processing, we have 84 labelled first name references with candidate sets of size 2 or greater. Of these, 54 have candidate sets that contain the true referent. A number of first name references with no obvious context in the message could not be resolved after further searching of the email collection.

3.5.3 Performance Evaluation

When evaluating the performance of a given scoring function, we have two objectives. First, we want to understand how well the scoring function ranks the true referent relative to other candidates on average in a candidate set. We refer to this as the *relative ranking performance* of the scoring function. Second, we want to characterize the ability of the scoring function to rank true referents higher than other candidates in general across candidate sets. We refer to this as the *absolute ranking performance* of the scoring function. We consider each evaluation task in the following sections.

3.5.3.1 Relative Ranking Performance

To provide insights into relative ranking performance, three performance metrics are evaluated for each scoring function. First, we compute the *rank 1 rate (R1R)* which is the fraction of candidate sets containing true referents over which the true referent is the top ranked candidate. This is expressed as

$$R1R = \frac{1}{|\mathcal{N}_t|} \sum_{n \in \mathcal{N}_t} \mathcal{I}\left(\tilde{e}(n) = e_{true}(n)\right)$$

where $\mathcal{I}(\cdot)$ is the indicator function, $e_{true}(n)$ is the true network reference associated with the name reference n and $\mathcal{N}_t = \{n : n \in \mathcal{N}, e_{true}(n) \in \mathcal{E}(n)\}$ is the set of name references with the true referent in the corresponding candidate sets. Note the R1R is computed assuming no candidate rejection.

The rank 1 rate provides an intuitive summary of performance, but can be misleading in this context given the variable sized candidate sets. Therefore to establish a relative baseline, we compute the expected value of the *random rank 1 rate (RR1R)* achieved by random selection of the top ranked candidate from each candidate set. This is expressed as

$$RR1R = \frac{1}{|\mathcal{N}_t|} \sum_{n \in \mathcal{N}_t} \frac{1}{|\mathcal{E}(n)|}.$$

Since the rank 1 rate gives no indication of how severe the failure is when the true referent is not rank 1, we also compute a metric we refer to as the *average true referent rank (ATRR)*. The ATRR is the average of the ratio of the true referent rank and the candidate set size. This is expressed as

$$ATRR = \frac{1}{|\mathcal{N}_t|} \sum_{n \in \mathcal{N}_t} \frac{1}{|\mathcal{E}(n)|} \sum_{k=1}^{|\mathcal{E}(n)|} k\mathcal{I}\left(e_{(k)}(n) = e_{true}(n)\right)$$

where $e_{(k)}(n)$ is the network reference with rank k in the candidate set $\mathcal{E}(n)$. Each true referent rank is normalized by the corresponding candidate set size to account for the variation in the number of candidates and reduce the sensitivity of the measure to large candidate sets.

3.5.3.2 Absolute Ranking Performance

Assessing the absolute ranking performance involves evaluating the scoring function's ability to rank true referents higher than other candidates across all candidates nominated for a given set of name references. Our interest in characterizing ranking performance from this perspective stems from our desire to reject as many candidates as possible without a significant loss of true referents. If the scoring function is able to separate the two classes of candidates with reasonable success, we will achieve our aim.

A natural measure of ranking performance advocated in the literature [4, 38, 39, 60] is the *area under the receiver operating characteristic (ROC) curve*. The ROC curve is a standard depiction of a detector's performance from classical signal detection theory, showing the detector's true positive rate versus false positive rate [173]. The area under the ROC curve (AUC) provides a measure of the separability achieved by the detector between the two classes. More specifically, the empirical AUC is an estimate of the probability that the detector will rank a randomly selected positive example higher than a randomly selected negative example, assuming all ties are broken uniformly at random [4]. When the AUC=1.0, perfect separability

is achieved. When the AUC=0.5, the detector performs no better than random chance.

If one defines the true referents to be the positive class and the other candidates to be the negative class, the AUC of the scoring function is the area under the empirical ROC curve generated by sweeping the threshold over the range of scores and computing the (false positive rate,true positive rate) operating points on the curve. This empirical AUC can be directly expressed in the following manner

$$AUC = \frac{1}{N_{TR}N_{OC}} \sum_{n_1 \in \mathcal{N}_t} \sum_{n_2 \in \mathcal{N}} \sum_{e_{oc} \in \mathcal{E}(n_2)/e_{true}(n_2)} \mathcal{I}\left(g(e_{true}(n_1), n_1|C_1) > g(e_{oc}, n_2|C_2)\right) + \frac{1}{2}\mathcal{I}\left(g(e_{true}(n_1), n_1|C_1) = g(e_{oc}, n_2|C_2)\right)$$

where $N_{TR} = |\mathcal{N}_t|$ is the number of true referents and $N_{OC} = \sum_{n \in \mathcal{N}} |\mathcal{E}(n)/e_{true}(n)|$ is the number of other candidates overall [4].

3.6 Discussion

We now examine the performance of the various scoring functions on the labelled name reference data. Figures 3.2-3.4 present a series of summary plots showing the rank 1 rates, average true referent ranks and AUCs of the scoring functions as a function of the amount of traffic history considered.

Consider first the R1R and ATRR metrics measuring relative ranking performance. For all models, as the filter duration is increased [2], incorporating more

[2]The duration of the MA filter is simply the number of time intervals over which the filter averages the summary statistic. We have defined the duration of the AR filter to be twice the

traffic history into the scoring process, the relative ranking performance generally increases and approaches a maximal level of performance. In terms of rank 1 rate, the performance of these simple models approaches 0.8 in most cases with sufficient history and significantly outperforms the random selection baseline. The finer level distinctions among the models can not yet be made; if one assumes the name reference resolutions are independent, there is no statistically significant different in performance among the models considered.

It is important to note that the success of these models is not based on the motivating assumption we made at the beginning of this investigation; namely, successful relative ranking is not based on observing increased communications activity over a short time interval around the time of the name reference. In contrast, the models exploit long term communication patterns that occur over 6 months or more to achieve peak performance.

Now let us consider the AUC metric measuring absolute ranking performance. In contrast to the relative ranking results, we see a significant distinction between the sender-only models and the sender+recipients models. As the influence of the relationships between the recipients and the candidate is increased, the AUC curve continues to shift lower indicating that separability between the true referents and the other candidates is decreasing across all filter durations.

At first glance, it may seem that the trends for the relative and absolute ranking performance measures are inconsistent. Why should the relative ranking

number of time intervals required for the impulse response of the filter to decay to 10% of its peak response.

performance be fairly insensitive to the influence of the recipients while the absolute ranking performance is much more so? This result suggests that while the relative rankings are not changing significantly, the variances of the true referent and other candidate score distributions are increasing, causing the decrease in separability. Adding the pairwise relationship statistics for the recipients could be inducing this result. Further investigation is needed to verify if this is indeed occurring.

To summarize, we demonstrate that our simple temporal traffic models produce a significant improvement in relative ranking performance over a baseline model which does not exploit traffic information. Furthermore, evaluations based on absolute ranking performance show that for a range of models, sender specific models outperform sender+recipient models.

3.7 Related Work

This chapter uses a social network generated from the email traffic of the Enron data set as a tool for name reference resolution. In this section, we describe some of the relevant related work on social networks, the Enron data set and entity resolution.

3.7.1 Social Networks

There has been a great deal of recent work in social network generation, analysis and mining. Using semantic associations from email communication, for example, McArthur and Bruza [108] propose methods of generating a social network using

implicit and explicit connections between people. Liben-Nowell and Kleinberg [99] use co-authorship to create social networks to predict future interactions among members of a given social network. Studies have also been done on creating and mining social networks to identify possible collaborators for a given problem [130, 87] and clustering people of similar interests [148]. Schwartz and Wood generate a social network using the *to* and *from* fields of email messages to discover users of a particular interest and field.

3.7.2 Enron

The release of the Enron data set in 2003 provided an unprecedented collection of emails from a major organization for use in research. Klimt and Yang [90] provides an overview of this corpus including the number of employees, the number of emails and a representative social network derived from the email traffic. Moreover, they used the Enron data to explore methods of email classification [89]. Corrada-Emmanuel [37] created MD5 hashes of the Enron emails and contact information to identify and deduplicate emails. Using the structure of the emails, Keila and Skillicorn [88] found a relationship in the word use pattern with message length as well as relationships among individuals. Skillicorn [161] further demonstrated methods to detect unusual and deceptive email communications. Diesner and Carley [49] used analysis of the email social network patterns over time to explore crisis detection in email. Moreover, a number of useful tools have also been developed in order to navigate and view email archives [70].

3.7.3 Entity Resolution in Email

There has been limited work in named entity resolution in email systems. Abadi [1] uses emails from an online retailer for anaphora resolution within email orders. Abadi's research, however, is designed for the resolution of pronouns referring to product orders rather than individuals and relies mainly on NLP for resolution. Holzer, Malin and Sweeney [75] on the other hand use social networks created from online resources like personal websites to resolve email aliases. Their approach of using social networks derived from relations from other sources, including proximity of references in a given web site, is particularly effective in controlled environments such as the university used in their evaluation. Of note, is Malin's evaluation of methods of disambiguation in relational environments [105]. Although Malin's work used actor collaborations in the Internet Movie Database rather than email, Malin did find that methods which leverage community, in contrast to exact similarity provide more robust disambiguation capability, supporting our approach to the problem.

3.8 Conclusion

In this chapter, we have examined ways in which email traffic can be used to resolve ambiguous name references within the body of the email messages. Our contributions include 1) a formal statement of the problem, 2) the definition of the resolution process in terms of candidate generation, candidate scoring and candidate rejection, and 3) the development of a suite of models for candidate scoring, which

exploit both role and temporal information. We have validated our methods on name resolution within a real-world corporate email archive, the Enron collection. An additional contribution is our evaluation methodology; we have proposed an evaluation based on both absolute rank and relative rank. Our overall goal is to develop robust ways of exploiting context information during the resolution process. The email traffic network is just one element of the context information and we explore additional context information in later publications [54]. As a first step, here, we have shown how simple email traffic models can achieve impressive resolution performance.

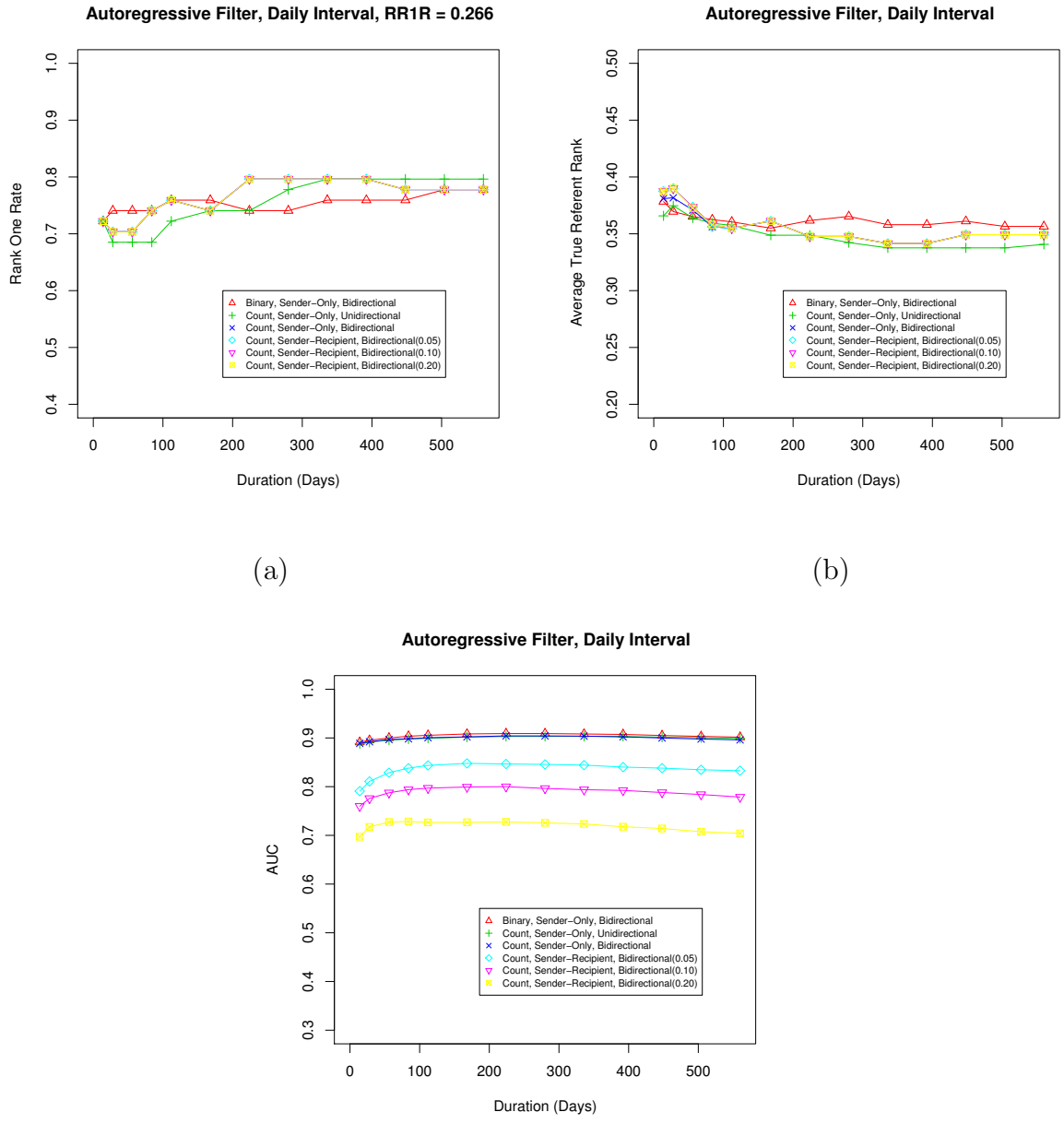
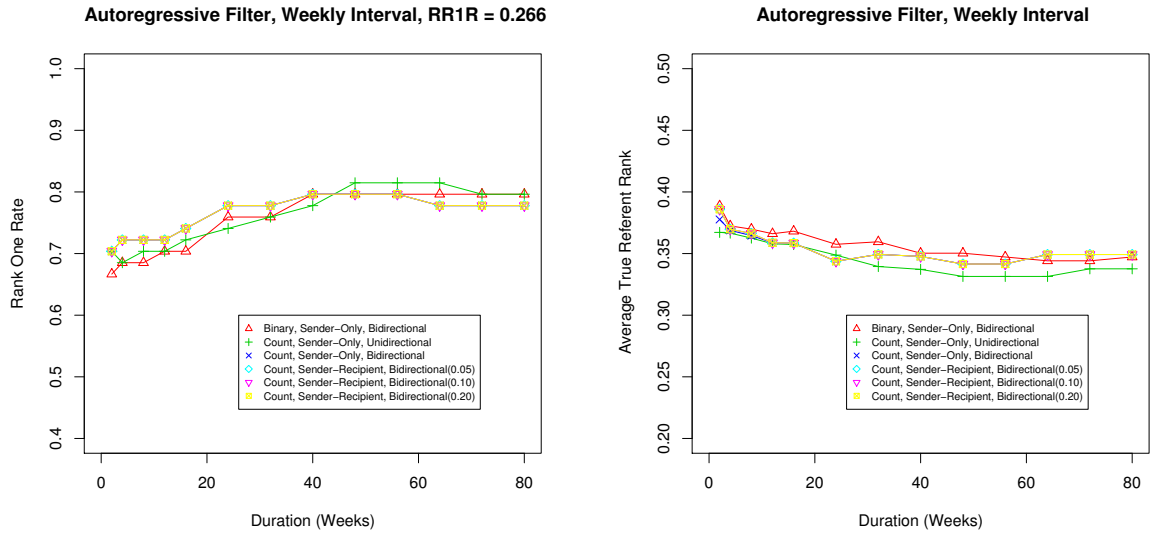
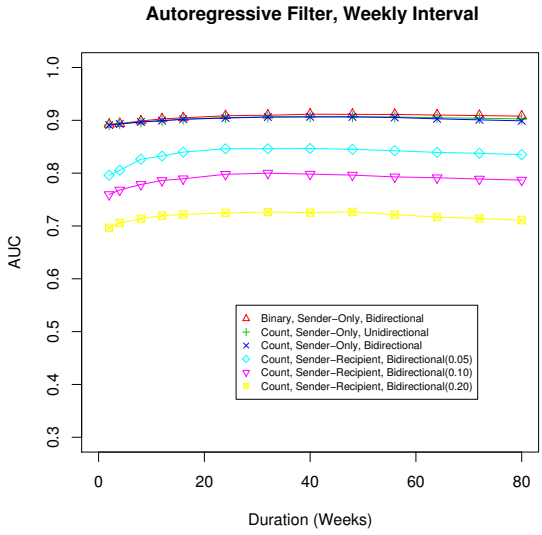


Figure 3.2: Autoregressive Filter Performance: (a) Daily Interval Rank 1 Rates, (b) Average True Referent Ranks and (c) Areas Under the ROC Curves



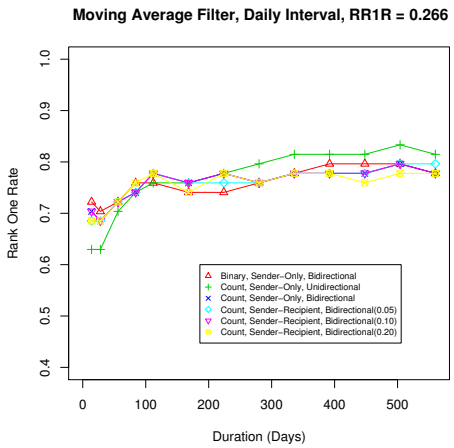
(a)

(b)

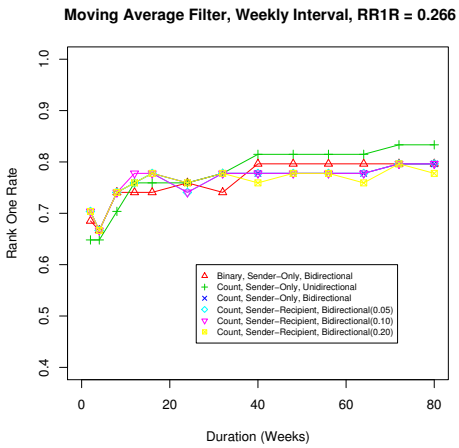


(c)

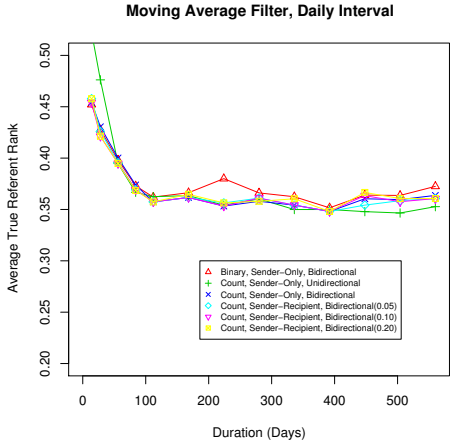
Figure 3.3: Autoregressive Filter Performance: (a) Weekly Interval Rank 1 Rates, (b) Average True Referent Ranks and (c) Areas Under the ROC Curves



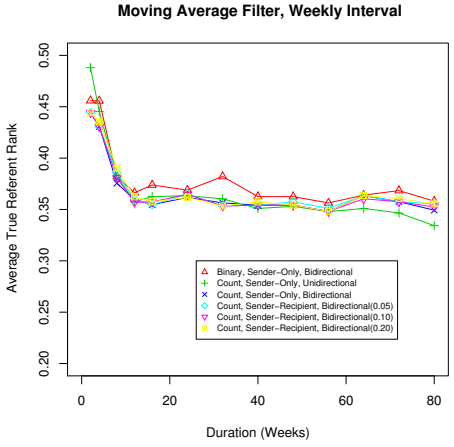
(a)



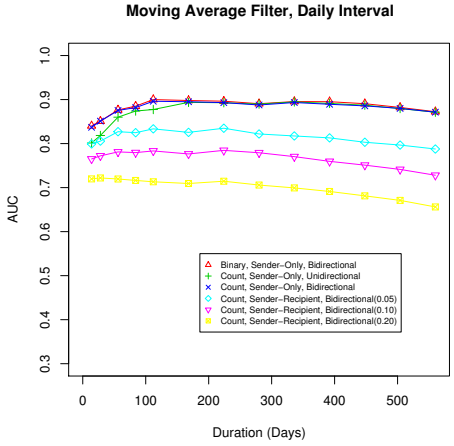
(d)



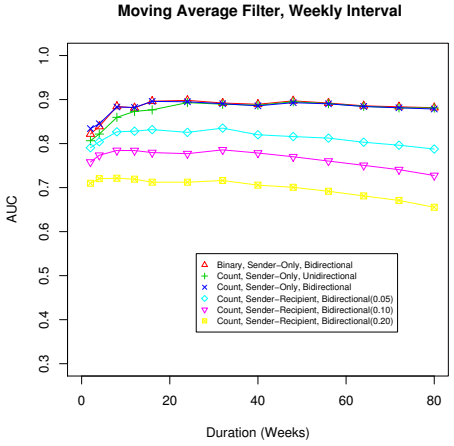
(b)



(e)



(c)



(f)

Figure 3.4: Moving Average Filter Performance: (a) Daily Interval Rank 1 Rates, (b) Average True Referent Ranks and (c) Areas Under the ROC Curves (d) Weekly

Chapter 4

Link Prediction for Social Network Discovery

In this chapter, we discuss our work in the link prediction of social relationships. We specifically address the challenge of identifying relevant communications that substantiate a given social relationship type. We propose a supervised ranking approach to the problem and assess its performance on predicting manager-subordinate relationships in an email archive.

4.1 Introduction

The Internet provides an increasing number of avenues for communication and collaboration. From instant messaging and email to wikis and blogs, millions of individuals are generating content daily that reflects their relationships with others in the world, both online and offline. Now that storage has become vast and inexpensive, much of this data will be archived for years to come. This provides new opportunities and new challenges. As networked groups and organizations increasingly leverage online means of communication and collaboration, there is an opportunity to observe the formation and evolution of roles and relationships from the communications archives. Such data provides a rich collection of evidence from which to infer the structure, attributes and dynamics of the underlying social network. Yet numerous challenges emerge as one contends with data that is often

ambiguous, incomplete and context-dependent.

If we wish to analyze the underlying social network that is at least partially represented by a collection of informal, online communications, it is important to think carefully about the data transformations required prior to conducting any type of analysis. At the highest level, we are fundamentally interested in discovering entities and the types of relationships they share. This implies that we must do more than simply adopt the communications (hyper)graph as a surrogate for the social network. Entities can and often do use more than one account online and not all communications relationships are equivalent. In fact, the social network can be thought of as a collection of networks with different relationship types (e.g. friendship, trust, advice, management). Human relations are multi-faceted and context-dependent. Therefore it is important to tease the communications apart and understand what types of relationships are being expressed among the entities.

We consider the scenario from domains such as intelligence analysis and litigation support where an analyst is attempting to reconstruct a representation of the social network from the data with minimal context. In this setting, the network discovery process of predicting relationship links is inherently a collaborative process between human and machine. While our problem is an instance of the more general link prediction tasks, our goal is not just to predict the existence of relationships, but to focus the analyst's attention on relevant communications relationships that express a given social relationship along with relevant message traffic that supports this association.

In this chapter, we propose a supervised ranking approach to address the re-

lationship link prediction problem. We begin the discussion in the following section with a formal definition of the problem. We discuss our approach to learning a relationship ranker from traffic statistics and message content and present an evaluation of these methods on a manager-subordinate link prediction task in email. We then review related work and conclude with thoughts on future directions.

4.2 Problem Definition

Informal, online communications such as instant messaging, text messaging and email are composed of structured and unstructured data. At the most basic level, this includes the network references corresponding to the sender and one or more recipients, the date and time of the communication and the message content. We will define a *communications archive* $\mathcal{C}$ as a set of observed messages exchanged among a set of network references N:

$$\mathcal{C} = \{m_k = (n_k^s, N_k^r, d_k, b_k) : n_k^s \in N, N_k^r \subseteq N\}. \tag{4.1}$$

For each message m_k, n_k^s is the sender's network reference, N_k^r is the set of recipient network references, d_k is the date and time and b_k is the body of the message. Every archive has a corresponding *communications graph* $\mathcal{C}_g = \{N, L\}$ that represents the message data as a set of dyadic communication relationships

$$L = \left\{l_{ij} = (n_i^s, n_j^r, M_{ij}) : n_i^s, n_j^r \in N, M_{ij} \subseteq \mathcal{C}\right\}. \tag{4.2}$$

among the network references N. For each directed relationship l_{ij}, n_i^s is the sender's network reference, n_j^r is the recipient's network reference and M_{ij} is the set of messages sent by n_i^s that include n_j^r as one of the recipients.

The link prediction task here involves identifying a mapping from the dyadic communications relationships L to one or more social relationships from a predefined set S. To emphasize the collaborative nature of our approach to the task, it is not our intention to develop an algorithm that automatically maps communications relationships to social relationships without intervention. A validated social network is one that the analyst believes is supported by evidence in the data. Therefore the machine's role in a collaborative approach to the task is to focus the analyst's attention on potentially relevant relationships along with supporting evidence in the message traffic.

We envision the analyst navigating the communications graph by following paths and incrementally investigating relationships in the ego networks corresponding to network references along the path. The *ego network* for a given entity in a network is generally defined as the subgraph that represents all of the direct relationships between the selected entity (the ego) and others (the alters). Formally in the case of the communications graph, the ego network $\mathcal{E}(n_i)$ for a given network reference $n_i \in N$ can be defined as

$$\mathcal{E}(n_i) = \mathcal{E}_o(n_i) \cup \mathcal{E}_i(n_i) \tag{4.3}$$

where

$$\mathcal{E}_o(n_i) = \{l_{ij} = (n_i, n_j, M_{ij}) \in L\}. \tag{4.4}$$

is the set of directed communications relationships from the ego to the alters and

$$\mathcal{E}_i(n_i) = \{l_{ji} = (n_j, n_i, M_{ji}) \in L\}. \tag{4.5}$$

is the set of directed communications relationships from the alters to the ego. For the purposes of ranking communications relationships within an ego network, we will initially restrict our attention to the set $\mathcal{E}_o(n_i)$ to avoid training and testing on the same message traffic.

Relationships in a given ego network $\mathcal{E}_o(n_i)$ will be ranked with a learned scoring function h that assigns a real-valued score to the relationship indicating its relative likelihood of expressing the social relationship of interest. If multiple social relationships are defined in the set S, there will be a corresponding scoring function for each social relationship. The task therefore is to learn a scoring function from a set of known relationships that successfully ranks relevant communications relationships higher than irrelevant relationships.

4.3 Learning to Rank Relationships

4.3.1 Objective

From initial exploration of the data or external sources of information, we assume a set of ego networks in the communications graph have been labeled, indicating whether or not the communications relationships exhibit the social relationship of interest. Initially we will approach the problem of learning multiple scoring functions independently. Therefore in each learning exercise, our goal is to learn a single scoring function for the given social relationship.

For a subset $N_t \subseteq N$ of network references in the collection, we assume the

corresponding set of ego networks

$$\bar{\mathcal{E}} = \{\bar{\mathcal{E}}(n_i) : n_i \in N_t\} \tag{4.6}$$

are fully labeled

$$\bar{\mathcal{E}}(n_i) = \{(l_{ij}, s_{ij}) : l_{ij} \in L, s_{ij} \in \{0, 1\}\} \tag{4.7}$$

where s_{ij} indicates whether the communications relationship exhibits the given social relationship. Given a feature extraction process $f(l) \in \mathbb{R}^p$ that maps a specified communications relationship r to a p-dimensional feature vector, we can reexpress the labeled training data as

$$\bar{\mathcal{F}} = \{\bar{\mathcal{F}}(n_i) : n_i \in N_t\} \tag{4.8}$$

where

$$\bar{\mathcal{F}}(n_i) = \{(f_{ij}, s_{ij}) : l_{ij} \in L, f_{ij} = f(l_{ij}), s_{ij} \in \{0, 1\}\}. \tag{4.9}$$

The goal is to estimate a scoring function h that yields good generalization performance in terms of the *mean reciprocal rank* of relevant relationships on unseen ego networks. The rank of a relevant relationship is defined with respect to the irrelevant relationships within the corresponding ego network. For the ego network $\bar{\mathcal{E}}(n_i)$,

$$\mathcal{F}_r(n_i) = \{f_{ij} : (f_{ij}, s_{ij}) \in \bar{\mathcal{F}}(n_i), s_{ij} = 1\} \tag{4.10}$$

is the set of feature vectors corresponding to the relevant communications relationships and

$$\mathcal{F}_o(n_i) = \{f_{ij} : (f_{ij}, s_{ij}) \in \bar{\mathcal{F}}(n_i), s_{ij} = 0\} \tag{4.11}$$

is the set of feature vectors for the irrelevant communications relationships. The *rank* $r(f_r, n_i)$ of a relevant relationship $f_r \in \mathcal{F}_r(n_i)$ is therefore defined as

$$r(f_r, n_i) = 1 + |\{f_o : h(f_o) \geq h(f_r), f_o \in \mathcal{F}_o(n_i)\}| \tag{4.12}$$

where $h(f) \in \mathbb{R}$. The mean reciprocal rank $MRR(\bar{\mathcal{F}})$ for the scoring function on the labeled ego networks is then

$$MRR(\bar{\mathcal{F}}) = \frac{1}{R} \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \frac{1}{r(f_r, n)} \tag{4.13}$$

where $R = |\cup_{n \in N_t} \mathcal{F}_r(n)|$.

4.3.2 Approach

Given the complexity of learning a scoring function that directly optimizes the mean reciprocal rank, we will indirectly optimize a bound on this criteria by minimizing the number of *rank violations* committed by the scoring function. The ranking performance of the scoring function can be assessed by considering how well the function satisfies a series of pairwise ranking constraints. For every possible pairing of relevant and irrelevant relationships in an ego network, we desire a scoring function that scores the relevant relationships higher than the irrelevant relationships so that

$$h(f_r) - h(f_o) > 0$$

$$\forall f_r \in \mathcal{F}_r(n), f_o \in \mathcal{F}_o(n), n \in N_t. \tag{4.14}$$

A violation of one of these constraints is what we will refer to as a rank violation. Clearly the number of rank violations maps directly to the rank as implied by equation 4.12. Section 4.3.3 clarifies the connection between the number of rank violations and the mean reciprocal rank. The important observation is that the minimization of rank violations leads to maximization of a lower bound on mean reciprocal rank.

We pursue a large-margin approach to learning the scoring function following in the spirit of prior large-margin ranking work [73, 83, 185]. We define the *rank margin* as

$$m(f_r, f_o) = h(f_r) - h(f_o) \tag{4.15}$$

for a pair of relevant and irrelevant relationships (f_r, f_o). A positive rank margin implies the rank constraint for the pair is satisfied. The magnitude of the rank margin gives a measure of the degree of satisfaction.

We will assume the scoring function h takes a generalized linear form

$$h(f) = w \cdot \Phi(f) : \mathbb{R}^p \rightarrow \mathbb{R} \tag{4.16}$$

where Φ is an arbitrary nonlinear mapping. We will estimate the scoring function through minimization of the following regularized objective function

$$C(w) = \frac{1}{2}||w||^2 + \lambda \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} g\left(m(f_r, f_o)\right) \tag{4.17}$$

where g is a convex margin loss function. At the optimum of this objective function,

$$w^* = \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} \alpha(f_r, f_o) \left(\Phi(f_r) - \Phi(f_o)\right) \tag{4.18}$$

where $\alpha(f_r, f_o) = -\lambda g'(m^*(f_r, f_o))$ and $m^*(f_r, f_o)$ are the rank margins at the optimum. Substituting into equation 4.16, we find the optimum scoring function takes the form

$$h(f) = \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} \alpha(f_r, f_o) \left(\Phi(f_r) - \Phi(f_o)\right) \cdot \Phi(f). \tag{4.19}$$

Given the transformed feature vectors enter the expansion solely as dot product terms, we can employ kernel functions $K(x, y) = \Phi(x) \cdot \Phi(y)$ satisfying Mercer's Theorem which provides a range of functional forms. This ultimately yields the general scoring function

$$h(f) = \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} \alpha(f_r, f_o) \left(K(f_r, f) - K(f_o, f)\right). \tag{4.20}$$

The corresponding dual objective function for the general nonlinear case is obtained by substituting equations 4.18 and 4.20 into equation 4.17 yielding

$$\begin{aligned} C(\alpha) = \frac{1}{2} & \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} \\ & \sum_{n' \in N_t} \sum_{f'_r \in \mathcal{F}_r(n')} \sum_{f'_o \in \mathcal{F}_o(n')} \\ & \alpha(f_r, f_o)\alpha(f'_r, f'_o)(K(f_r, f'_r) \\ & -K(f_o, f'_r) - K(f_r, f'_o) + K(f_o, f'_o)) \\ & +\lambda \sum_{n \in N_t} \sum_{f_r \in \mathcal{F}_r(n)} \sum_{f_o \in \mathcal{F}_o(n)} g\left(m(f_r, f_o)\right). \end{aligned} \tag{4.21}$$

4.3.3 Lower Bound on the Mean Reciprocal Rank

In order to show that minimizing the number of rank violations is a reasonable proxy for maximizing the mean reciprocal rank (MRR), we need to understand how

these quantities are related. For a fixed number of rank violations, the resulting MRR varies depending on how the rank violations are distributed across the relevant relationships. If the rank violations are concentrated, so that a small number of relevant relationships are low ranked, the MRR will be higher than the case where the same number of rank violations are distributed across a larger number of relevant relationships. It is this line of thought that leads to bounds on the MRR for a given number of rank violations.

Let us assume that there are M relevant relationships and that the maximum possible rank for the ith relationship is $N_r^i + 1$. This implies that the maximum number of rank violations that can be associated with the ith relevant relationship is N_r^i.

A useful analogy for this discussion is to imagine we have M bags and N balls. Each bag has one ball prior to assigning any of the N balls. The ith bag can hold $N_r^i + 1$ balls. In this scenario, the mean reciprocal rank is the mean reciprocal number of balls in a bag. To lower bound the MRR, we need to determine the assignment of N balls to the bags that minimizes the mean reciprocal number of balls in a bag.

To minimize the MRR for N balls, consider a process whereby the balls are incrementally assigned to the bags so that at each step the MRR is minimized. This implies that we want to assign the next ball to the bag that maximizes the incremental reduction in MRR. If there are b balls in a bag already, the incremental

reduction in MRR from an additional ball is

$$\frac{1}{M}\left(\frac{1}{b}-\frac{1}{b+1}\right)=\frac{1}{Mb(b+1)}. \tag{4.22}$$

Therefore, at each step, we should add the next ball to a bag with the least number of balls that can accept an additional ball. By uniformly adding balls to bags that can accept them, we will maintain a minimum MRR throughout the process.

Let $S_k(k \geq 2)$ be the number of bags that can hold k or more balls. We will make p passes down the line of bags adding one ball to each bag that can accept one until all N balls are placed. On the ith pass,

$$B_i = \min\left(S_{i+1}, N-\sum_{j=0}^{i-1} B_j\right) \tag{4.23}$$

balls are placed. $B_0 = 0$ by definition. The lower bound on MRR is therefore

$$\begin{aligned} MRR_{\min} &= \frac{M-B_1}{M}+\frac{1}{M}\left(\frac{1}{p+1}B_p+\sum_{i=1}^{p-1}\frac{1}{i+1}(B_i-B_{i+1})\right) \\ &= 1-\frac{1}{M}\sum_{i=1}^{p}\frac{1}{i(i+1)}B_i. \end{aligned} \tag{4.24}$$

The key observation here is that all of the B_i for $i < p$ remain constant and B_p decreases as N decreases. Therefore the lower bound on MRR is strictly monotonically increasing with a decreasing number of rank violations.

4.4 Message Ranking

After ranking communications relationships with the scoring function, a natural question to ask is how does each message contribute to the overall score for a

From	*Recipients Include*	*From*	*Recipients Include*
n_a	n_b	n_b	n_a
n_a	n_c and not n_b	n_b	n_c and not n_a
n_c	n_a and not n_b	n_c	n_b and not n_a
n_c	n_a and n_b		

Table 4.1: List of possible communications events corresponding to a dyadic relationship (n_a, n_b). N_c is the common set of network references with whom both n_a and n_b communicate. n_c is a generic reference to any network reference in N_c.

given relationship? If we define a scoring function with the form

$$h(f) = w \cdot \Phi(f) = w \cdot \sum_{m_i \in M} \Phi'(f_m) = \sum_{m_i \in M} h_m(f_{m_i}) \tag{4.25}$$

where the relationship score can be expressed as a linear combination of message scores $h_m(f_{m_i})$, we can immediately assess the relative contributions and sort the messages based on the message scores. We will employ a feature space and kernel function for content-based relationship ranking that admits this decomposition.

4.5 Manager-Subordinate Relationship Link Prediction

To evaluate the utility of the proposed approach, we consider the problem of manager-subordinate relationship link prediction within an email archive. For this task, the goal is to identify relationships within each ego network where the alter is the ego's manager. In the following, we present two relationship summarization methods for exploiting relationship traffic statistics and message content.

4.5.1 Traffic-Based Relationship Ranking

In a hierarchical organization, it seems reasonable to believe that traffic patterns alone can provide significant indicators of organizational structure, assuming

that issues of observability do not unduly complicate matters. Within the literature, there is evidence that group structure evident in email communications corresponds well to organizational constructs [174]. Similarly, we investigate whether management behavior is evident in the traffic statistics.

For a given dyadic relationship (n_a, n_b), we compute a number of traffic-based features between the network references n_a, n_b and the set of network references N_c with whom both n_a and n_b communicate. n_c is a generic reference to any network reference in N_c. The common associates are included to allow the ranker to key on potential differences in communication patterns with fellow colleagues and the manager. For each type of communication event listed in Table 4.1, from the specified network reference that includes/excludes the specified recipients, we compute the number of messages of this type and the quartiles for the distribution of the number of recipients observed across those messages. Including summary statistics for the number of recipients is potentially important for capturing differences in information distribution behavior and indications of group communications/directives.

4.5.2 Content-Based Relationship Ranking

Although traffic statistics alone may be sufficient for ranking relationships, they do not provide insight to the analyst that enables her to make a judgement about the type of social relationship expressed. Ultimately message content must be identified that substantiates the social relationship. Therefore we will assess the performance of a ranker that directly exploits the message content and allows us to

rank the messages within the communications relationship.

Prior to computing feature vectors for individual relationships, we perform filtering steps on the message content to remove spurious characters and eliminate text from previous messages in the thread. Then we construct a master term list for the communications archive to define the feature space. For each communications relationship, we summarize the traffic by simply counting the term frequencies across the set of messages corresponding to the communications relationship. No stop word removal or term weighting was applied prior to learning the ranker.

4.6 Results

To assess the performance of our approach, we utilized the Enron email dataset along with organizational ground truth derived from an internal Enron document. This dataset is the collection of email from the folders of 151 Enron employees released as part of the government investigation into Enron's financial practices. Our results are based on the UC Berkeley version of the collection containing approximately 250000 unique email messages mainly occurring in the 2000-2002 time frame.

Using an internal Enron document specifying the direct reports for Enron employees over 2000-2001, we identified 43 individuals in the collection with observable manager-subordinate relationships and nontrivial ego networks. We constructed the ego networks corresponding to each employee over this time frame and retained only those relationships where a minimum of 5 emails were sent in each direction. The

Approach	*MRR*
Content-Based with Attribute Selection	0.719
Content-Based	0.660
Traffic-Based (From n_a including n_c and not n_b)	0.613
Traffic-Based	0.518
Random	0.211
Worst-Case	0.141

Table 4.2: Mean reciprocal rank for the various approaches. The MRR reported for the learned rankers results from the best performing regularization parameter.

resulting ego networks range in size from 2 to 107 relationships.

For both traffic and content-based relationship ranking, we use a linear kernel function and evaluate generalization performance using leave-one-ego-network-out cross-validation. We report the mean reciprocal rank (MRR) for the best performing regularization parameter. We also provide results for the worst case, where all the rank constraints are violated, and the average case for random selection. The results are provided in Table 4.2.

4.6.1 Traffic-Based Relationship Ranking

The linear ranker trained on all of the traffic statistics performs well relative to the baselines. By reducing the feature space to a single dimension, we achieve a significant additional improvement. Ranking relationships solely based on the number of emails sent from the ego to the common network references and not to the alter yields the best performance. After some reflection on group dynamics, this result is intuitively appealing. First the feature emphasizes relationships where there is a large set of common network references. For a manager and subordinate, these will likely correspond to fellow members of the group that the manager leads. At the same time, the feature deemphasizes relationships where more emails are

sent from the ego to the common set and the alter. When both ego and alter are colleagues, these events are more likely than when the alter is the manager.

4.6.2 Content-Based Relationship Ranking

We explored two content-based ranking approaches. In the first approach, a linear ranker was trained on the relationship term frequencies for all 19067 terms. Examining the absolute value of the resulting weight vector, we determined the 1000 most discriminative terms. Then we trained another linear ranker only on the term frequencies for the selected terms.

We found that content-based ranking consistently outperforms traffic-based ranking. We also found that attribute selection provides a significant additional performance improvement. As shown in Table 4.2, the content-based ranker trained in the constrained term space yields the highest MRR of 0.719. Examining the top ranked terms in the weight vector, we find terms indicative of the relationship of interest. Some notable words appearing in the top 20 include "please", "report", "project", "termination", and "executed".

We note that there are some ego networks in which content-based ranking performs worse than traffic-based ranking. The messages in these relationships suggest that the problem may be caused by more complex relationships. For example, in one ego network where content-based ranking performs significantly worse, the ego is a senior legal analyst. Although this individual had only one assigned manager, she performed tasks for other individuals, such as writing and analyzing legal

documents, similar to those performed for her direct manager.

4.6.3 Content-Based Message Ranking

To qualitatively evaluate message ranking, we examined the highly ranked messages identified by the top performing content-based ranker. In cases where the manager-subordinate relationship achieved rank 1, we found that definitive evidence was usually contained within the top 10 messages. Definitive evidence for this type of social relationship includes emails with weekly reports, vacation requests, and project assignments. For example:

> *From: Cheryl Nelson [cheryl.nelson@enron.com]*
>
> *To: Mark E Taylor [.taylor@enron.com]*
>
> *Subject: Holiday Vacation*
>
> *Hi Mark,*
>
> *I would like to take Wednesday, December 27th as a vacation day because I could not get a flight on the 26th. Since I do not plan to leave town until December 24, I could catch up with my work by working on sat. December 23rd. Let me know if this is okay with you.*

Although the analyst may have some preconceived notions about the nature of the relationship that are accurate, there are other aspects that may be specific to the domain or organization and therefore difficult to anticipate. For example, message ranking revealed "workload updates" requested by one manager from subordinates. Workload updates are weekly reports. This process also identified emails

that provide evidence for the social relationship in ways one would not expect. For example:

> *From: Christian Yoder [christian.yoder@enron.com]*
>
> *To: Elizabeth Sager [elizabeth.sager@enron.com],*
>
> *Genia Fitzgerald [genia.fitzgerald@enron.com]*
>
> *Subject: Happiness*
>
> *Happiness is looking at the new legal org chart (which Jan just now dropped on my desk). I always approach these dry documents as though they were trigrams resulting from throwing the coins and consulting the I-Ching. At the top of the trigram which I find myself listed in I see a single name: Elizabeth Sager, and at the bottom I see the name Genia FitzGerald. ... cgy*

As this example hopefully illustrates, message ranking may help the analyst gain additional insights and move beyond evidence that can be discovered through simple keyword queries.

4.7 Related Work

In the scenario we are considering, where an analyst is examining a collection of online communications with minimal context a priori, it will be important to have a number of tools to examine the data from varying perspectives. By focusing solely on the communications events through analysis of the communications graph, we can identify groups/communities and key individuals that are influential based on

their position in the graph. Yet in general, we can conclude little about the nature of the relationships without exploiting the corresponding content.

Within the context of email exploitation, McCallum et al. [109] took the first step toward a richer model of email relationships by proposing a generative model that captures the dependencies between topics of conversation and relationships. Since then, several other generative models have been proposed [178, 163, 192, 189] that support joint relationship-topic clustering or group-topic clustering. These algorithms provide utility when initially exploring the data. Yet as the analyst discovers various relationship types of interest, these approaches do not provide a mechanism to capture and exploit the analyst's relationship labels so that additional relevant content tailored to her information needs can be identified. Our approach therefore provides a complimentary capability by leveraging the context provided by the analyst.

Other related approaches in the literature have focused primarily on processing the communications events to understand the structure of the social network. Eckmann et al. [52] develop an information-theoretic approach to email exchange that allows for separating static and dynamic structure which appears to correspond to formal and ad-hoc organizational structure. Tyler et al. [174] present a group detection algorithm that segments the communications graph by eliminating edges with low betweenness centrality. The validity of the groups detected within HP Labs was verified through interviews. Diesner and Carley [50] analyze global properties of the Enron communications graph and rank network references using various centrality measures from social network analysis to identify influential individuals.

O'Madadhain and Smyth [132] propose an approach for ranking vertices in graphs representing event data and demonstrate a weak correlation between network reference rank and position in the organizational hierarchy using a corporate archive of email events.

In recent years, there has been increasing interest in defining learning methods that address ranking tasks [73, 83, 60, 25]. Our approach is inspired by earlier work on large-margin methods for ranking [73, 83, 185] that learn a scoring function through minimization of the number of rank violations on the training data. Similar to [185], our general objective is to learn a ranker that successfully ranks relevant objects higher than irrelevant objects across a set of object sets. In the case of [185], the object sets are collections of retrieved documents corresponding to various queries. In our scenario, the object sets are the communications relationships in the labeled ego networks. We chose to minimize the number of rank violations in order to indirectly maximize the mean reciprocal rank. As we have established in Section 4.3.3, minimization of rank violations maximizes a lower bound on the mean reciprocal rank (MRR). Recent work [84, 26] has examined the problem of directly optimizing multivariate performance measures similar to MRR that more accurately represent ranking performance across object sets of varying size. Additional work is needed to define suitable methods for direct optimization of the MRR.

4.8 Conclusions and Future Work

In this chapter, we presented a formal definition of the relationship link prediction task and proposed a supervised ranking approach to the problem. We showed that through minimization of rank violations, we can indirectly learn a relationship ranker that maximizes a lower bound on the mean reciprocal rank. Through experimentation on the Enron email dataset, we demonstrated the utility of this approach on a manager-subordinate relationship link prediction task. Using traffic and content-based features, the ranking method is able to routinely cue the analyst to relevant communications relationships. Message ranking using the content-based ranker provided additional guidance by illuminating compelling evidence within the message traffic substantiating the social relationship.

Cueing the analyst to relevant relationships and message content is an important first step; yet it is only half of the collaborative cycle we envision. As the user navigates the communications graph, she will make judgements about relationships and message content. These judgements can be exploited to incrementally refine the scoring function as her exploration proceeds. The goal is to enable continuous learning behind the scenes that supports her in the discovery process. To realize this capability, a number of challenges must be addressed such as automated model selection (feature selection and hyperparameter tuning) and learning from multiple types of rank constraints indicating what relationships and message content are relevant. Other questions emerge about how to most effectively leverage unlabeled relationships in the communications graph and direct labeling efforts to rapidly accelerate

the learning. These are some of the issues we will focus on in future research.

Chapter 5

Active Surveying for Query-driven Collective Classification

In this chapter, we discuss our work in collective classification of node labels. For this work, we describe a common, but previously unexplored problem setting we define as query-driven collective classification. We look at query-driven collective classification in an active surveying setting where the labels and most of the network structure is initially unknown but can be acquired, with some cost and subject to budget constraints, for learning a semi-supervised collective classification model. Leveraging common assumptions on feature and structural smoothness, we propose a novel adaptive algorithm and empirically show the superiority of our approach over standard active learning approaches on four real-world datasets.

5.1 Introduction

Collective classification, the task of labeling nodes in a network, is an important problem in many domains, such as analysis of social networks, biological networks, and citation databases [104, 149]. While traditional learning aims to learn a predictor that accurately labels all available data, we consider the case in which one is primarily interested in labeling a particular subset of nodes, which we refer to as the *query set*. For example, when labeling a social network, we may only be interested in the labels of key high-ranking or influential individuals. Accurate

classification of the rest of the social network may only be useful to help collectively classify the targeted nodes. We refer to this problem as *query-driven collective classification*.

In many practical scenarios, labels and network structure may not be immediately available for all nodes, and certainly are not available for the nodes in the query set. Instead, there is a cost for acquiring this information. We therefore explore the problem of query-driven collective classification in an active learning setting. In traditional active learning, the learner controls the sequence of training examples received. Unlike previous work [21, 94, 102, 150], we do not restrict the training examples to simple instance-label pairs; we instead explicitly consider other information that is inherent to relational domains. This leads to a more general view of information acquisition, which we refer to as *active surveying*. Whereas prior work in this area [154] was geared specifically to the problem of identifying opinion leaders, here we present a more general view. In our setting, a survey returns not only the label(s) of a node, but also any missing attributes and links. In social network analysis, the neighbor information returned by a survey is often referred to as the node's *ego network*. The relational information is particularly valuable in network domains, since one can exploit the potential correlations between connected data points.

We require that the learning algorithm can never directly survey a query node. For various reasons in practice, surveying a query node may incur a prohibitive cost. For instance, in a viral marketing campaign, the surveying action may reveal the product to targeted influentials, when the goal of the campaign is to limit exposure of

the product to only those predicted to promote it. Thus, the challenge is to identify the optimal subset of non-query nodes to survey, subject to budget constraints, that will enable us to correctly predict the labels of the query nodes.

We analyze the surveying problem using a distributional "smoothness" assumption. We define a query-driven problem to be smooth if the distribution of labels, conditioned on some measurable distance function, changes proportionally to the distance. This distance function can be computed using features or network structure, depending on the problem domain. If the smoothness property holds for a given dataset and metric, then surveying nodes based on their proximity to the query nodes should minimize the deviation between the query and survey node distributions. Therefore, the smoothness assumption theoretically implies that minimizing this distance minimizes the average loss over the query nodes. Based on this analysis, we develop several active surveying strategies: one that leverages feature smoothness; one the leverages structural smoothness; and Adaptive Surveying for Query-driven Collective Classification (ASQ2C), a novel adaptive algorithm that automatically chooses between the two, based on an empirical estimate of the so-called *assortativity* in the current observed graph. We evaluate these strategies on several real-world networks using an iterative classification algorithm to perform collective classification.

We begin by motivating our research by providing examples of real-world problems in Section 5.2. We then review some background and related work in Section 5.3. In Section 5.4, we introduce the problem of active surveying for query-driven collective classification and define the smoothness property. We then discuss some

relevant metrics to use for surveying under various smoothness assumptions and present the ASQ2C algorithm. We evaluate our query-driven surveying strategies in Section 5.5.

5.2 Motivating Examples

In this section, we present three real-world examples of active surveying for query-driven collective classification.

5.2.1 Intelligence Gathering

The query-driven active setting is particularly apt for intelligence gathering, specifically for analyzing organized crime and terrorist networks. In this scenario, we may be interested in ascertaining the affiliation, disposition, or role (i.e., label) of key individuals (i.e., query nodes) in a population. For context specific reasons, these individuals may be inaccessible, making it difficult, if not impossible, to ascertain their affiliations or dispositions directly. Moreover, the full network may be largely unobserved. Through surveillance, we can acquire information about the network, including the labels of less important people, who may be more accessible. This proxy knowledge can then be used to model the interaction of characteristics, connections, and labels; furthermore, we may uncover relationships between the observed and unobserved portion of the network, which can be used to infer correlation of labels. Surveillance or investigation, however, are expensive in terms of both time and resources, and so we aim to identify the optimal set of people to investigate,

given a budget.

5.2.2 Disease Transmission

Consider the task of monitoring the spread of an infectious disease in a social network. In this context, the goal is to determine the infection status of individuals in a population. While the population may be known beforehand, we may have little or no information about the relevant characteristics or relationships between people. Further, the set of individuals we are most interested in—those "at-risk", who are likely to become infected—may comprise only a small portion of the overall network. The at-risk population may not have access to healthcare, or may be reluctant to get tested, so this portion of the network may be unobservable. Yet we can survey the observable network to identify contributing factors to infection, such as an demographics, genetics and medical history, which people in the query set may exhibit. Moreover, since there is an undeniable causal link between infection and one's proximity to, and interaction with, those infected, identifying the infection status of related or connected individuals may offer insight about the query set. Since some diseases must be handled with discretion (such as sexually-transmitted diseases) and certain people may be less cooperative than others when it comes to testing and observation, there is a natural cost structure associated with data acquisition. Thus, as before, we are only able to test and observe a subset of the overall population, so identifying the optimal subset to survey is important.

5.2.3 Viral Marketing

Finally, we consider the context of marketing. Suppose we are introducing a new product and are interested in creating awareness of this product through viral marketing. Given the recent proliferation of online social networks, there are various means of identifying key opinion leaders and information hubs (i.e., the query set), who comprise the optimal entry points into a market. Yet before advertising to them, we must predict whether these individuals are likely to adopt and promote our product. Receiving positive reviews would be beneficial, but having opinion leaders disseminate negative feedback would be especially detrimental to sales. As before, we can survey a less influential test market to model the behavior of the target market without risk of negative publicity. We can also look at how people that are connected to the opinion leaders react to the product, with the assumption that they likely share similar opinions. Using their estimated reactions to the product, we can target our marketing to the subset of opinion leaders likely to give positive reviews, while minimizing the overall cost of the marketing.

5.3 Background

For the following, let $\mathcal{X} \subseteq \mathbb{R}^d$ denote a d-dimensional instance space, $\mathcal{Y}$ a finite set of labels, and $\mathcal{Z} \triangleq \mathcal{X} \times \mathcal{Y}$ their cross-product. We are given a relational graph $G = (\mathcal{V}, \mathcal{E})$, in which the nodes $\mathcal{V}$ represent individuals and the edges $\mathcal{E}$ represent relationships between them. We assume that $\mathcal{V}$ is fully-specified, although $\mathcal{E}$ is presumably incomplete. Each node is associated with a vector of attributes

$v.X \in \mathcal{X}$ and a label $v.Y \in \mathcal{Y}$, although the latter is, for the most part, assumed to be hidden.

We define a *relational* learning algorithm $\mathcal{A}$ as a function mapping an input graph G to a hypothesis space $\mathcal{F}$. Let f_G denote a hypothesis returned by running $\mathcal{A}$ on G, and note that f_G can leverage any information revealed during training to perform collective inference. Accordingly, we denote the prediction of a single instance $v \in \mathcal{V}$ by $f_G(v)$. If f_G is *real-valued* (confidence-rated or probabilistic), we will use $f_G(v; y)$ to denote the predicted confidence (or probability) that $v.Y = y$. (If f_G outputs a probability distribution, then we require that $\sum_{y \in \mathcal{Y}} f_G(v; y) = 1$.) Accordingly, we use $h_G(v)$ to denote the *maximum a posteriori* (MAP) assignment $h_G(v) \triangleq \text{argmax}_{y \in \mathcal{Y}} f_G(v; y)$.

We measure the error (or *loss*) of f_G by a function $\ell : \mathcal{F} \times \mathcal{V} \to \mathbb{R}$, that returns a real-valued measure of the discrepancy between $f_G(v)$ and $v.Y$. Denote by $\mathcal{L}(\mathcal{U}) \triangleq \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \ell(f_G(u))$ the average loss over a subset of nodes $\mathcal{U} \subseteq \mathcal{V}$. This can be equivalently denoted by $\mathbb{E}_{u \in \mathcal{U}}[\ell(f_G, u)]$.

5.3.1 Collective Classification

The task of inferring node labels of network data using local and global structural information is generally known as *collective classification*. The underlying assumption of collective classification models is that the relationships between nodes can be used to supplement local information (attributes) used in prediction. For instance, a node's label might be positively or negatively correlated with that of

its neighbors. Some collective methods rely solely on this structural information to propagate labels [104]. A number of collective classification models have been proposed [149] and shown to outperform their non-relational counterparts in relational domains. This is especially true in semi-supervised settings like ours, in which labeled and unlabeled instances are connected in the same network [21].

The aforementioned approaches operate under the assumption that all nodes are equally important. To our knowledge, the query-driven approach to collective classification has received little attention. In a non-relational setting, Fawcett and Provost [58] present an approach to instance-varying cost sensitive classification, in which the cost of misclassifying an instance varies depending on its characteristics (e.g., the cost of misclassifying a fraudulent ATM transaction is a function of the amount involved in the transaction). There has also been work in query-specific belief propagation methods for graphical models [33].

5.3.2 Active Learning and Inference

While most prior work in collective classification has focused on the "passive" setting, in which labeled data is drawn randomly from an unknown distribution, we consider the "active" setting, in which the learning algorithm (or predictor) can determine the sequence of examples. The learner is given an initial set of annotations with which to bootstrap learning (or inference), after which it is allowed to request additional examples (subject to some budget constraint) to improve performance. In active learning, the benefit is two-fold: by selecting the most informative exam-

ples, the learner can refine the model for problematic or ambiguous instances, while potentially reducing the sample complexity of the learning algorithm [21, 102, 195]. In the transductive setting, where the labeled and unlabeled instances belong to the same network, additional labeled instances can inform the predictions of related nodes, in a process commonly referred to as *active inference* [20, 139].

Prior work in active learning [21, 94, 102] for relational data has focused on acquiring only label information, with the assumption that the network and all other attributes are observed. Here, we make no such assumptions; instead, we explicitly assume that the available network is largely incomplete. We therefore allow the learner (or classifier) to obtain a richer form of feedback, including (but not limited to) labels, attributes, and network structure. In the context considered herein, we begin with a partially labeled network, with partially specified neighborhoods; surveying any node returns its label, along with any edges connected to it. There may also be contexts in which a survey returns the ground truth for missing or noisy attribute values. Because this form of data acquisition is more general than traditional active learning, we refer to it as "active surveying", acknowledging Sharara et al. [154], who coined the term for the task of identifying key opinion leaders in a social network.

5.3.3 Active Strategies

The effectiveness of active methods is largely predicated on the strategy for acquiring new information. The goal is to select a sequence of surveys that max-

imizes the quality of the learned model, while minimizing the amount, or cost, of the acquired information. Since determining an optimal solution is often intractable [20, 143], active methods typically rely on intuitive heuristics. Popular strategies for active learning and inference are *uncertainty sampling* and *structure-based sampling*, respectively. The following sections discuss these techniques (in the non-query-driven setting).

5.3.3.1 Uncertainty Sampling for Active Learning

Reasoning that instance ambiguity leads to error, uncertainty sampling focuses attention on those instances that the current model finds most difficult to classify. In classification, this requires either confidence-rated prediction or an ensemble of classifiers. There are numerous measures of uncertainty; arguably, the most common of which is the information-theoretic *entropy*, due to Shannon [152], defined as $H(X) \triangleq \sum_x \Pr[X = x] \log \Pr[X = x]$. Since this value is negative, it is common to use the negative entropy as a measure of uncertainty. The objective is thus to minimize the cumulative entropy of the predictions over all unlabeled nodes, $\sum_{u \in \mathcal{U}} -H(f(u))$. It is straightforward to show that this quantity is minimized by obtaining the labels of the most uncertain instances (assuming the learning algorithm is able to exploit the new information). Since deterministically selecting the most uncertain instances can sometimes result in exploring outlier regions of the instance space [144], uncertainty-based methods typically perform random sampling, weighted by uncertainty, which increases robustness to outliers.

Note that uncertainty sampling optimizes for the *entire* unlabeled space. In the query-driven setting, however, this is not efficient, since the distribution we are interested in may differ from the global distribution.

5.3.3.2 Structure-based Sampling for Active Inference

Another broad category of strategies leverages the structure of the network [21, 139]. These approaches rely on the assumption that, during inference, the true labels of nodes with certain structural properties are likely to propagate and positively impact the inference of the most nodes. One heuristic, for example, is to survey the nodes with highest degree, with the intuition that these nodes have the greatest influence over the connected nodes. In other words, the labels of high degree nodes are likely to correlate with those of their neighbors [139]. Other common heuristics include various centrality measures such as closeness and betweenness centrality [102] with the assumption that nodes most central to a given connected component are most likely to provide the most influence over nodes in that connected component.

Note that these structure-based strategies have only been applied in settings where the network structure is *fully* observed. In an active surveying setting, where few, if any, edges are observed, these strategies do not have enough information to function effectively.

5.4 Query-driven Active Surveying

In this section, we define the problem of query-driven collective classification with active surveying. We motivate the discussion of surveying strategies by introducing the notion of smoothness. We then leverage the smoothness assumption to derive several active surveying strategies.

5.4.1 Problem Definition

The learning problem is defined as follows. In query-driven applications, we are given a specified (proper) subset of the full vertex set, $\mathcal{Q} \subset \mathcal{V}$. We refer to this set as the *query set*. Let $\mathbb{Q}$ denote the distribution over this subset and note that it is assumed to be different from the global distribution $\mathbb{P}$. The labels of the query set are hidden and assumed to be unobtainable; thus *our primary objective is to predict the labels of this subset.* To do so, we will train a transductive model, leveraging the label and structural information from the rest of the network.

The query-driven objective may seem counterintuitive at first; after all, most learning algorithms strive for generalization with respect to the global distribution. In a sense, query-driven learning seems tantamount to overfitting. The key distinction is that we optimize for the query set, not the training set; and since the labels of the query set are unobservable, there is no way to overfit that distribution. Like related work on transduction [61], the potential benefits of the query-driven approach are increased accuracy (with respect to the query set) and decreased sample complexity. In contrast to the transductive setting, we are not interested in labeling

all of the unlabeled nodes; only the query set.

We obtain training data via a sequence of *surveys*. Each survey returns the label of, as well as all edges adjacent to, a specified node. Let Ψ denote the survey operator. Thus, surveying a node completely reveals all information about the node; until a node is surveyed, one cannot assume that its adjacent edge set is completely specified. Let $\mathcal{S}$ denote the set of nodes that have been surveyed and $\mathcal{U}$ denote the nodes that have yet to be surveyed. When considering which nodes to survey, we may refer to a subset $\mathcal{U}^c \subseteq \mathcal{U}$ as the *survey candidates*.

Acquiring complete information is considered expensive; we therefore assume some cost structure associated with surveying. Let $\varphi : \mathcal{V} \to \mathbb{R}^+$ denote a real-valued cost function. For the nodes in the query set, the cost is infinite[1]; for all other nodes, the cost is a positive real number. For the purposes of this research, since our study focuses on the efficacy of our survey strategies, we will assume that the cost of a survey is uniform for all non-query nodes.

Our learning objective can be stated as the cost of the queries and the expected loss over the query set:

$$\operatorname{argmin}_{\mathcal{S}} \mathbb{E}_{q \in \mathcal{Q}} \left[\ell(f_G, q) \,|\, G \leftarrow G \cup \mathcal{S} \right] + \sum_{s \in \mathcal{S}} \varphi(s).$$

Determining the optimal set of surveys is obviously hard, since we cannot measure the expected error term. Even if we could measure the objective, the problem is equivalent to exactly solving a *knapsack problem*, which is NP-hard. As such, we consider an iterative greedy approach, in which we survey a fixed number of nodes

[1]While in certain settings query nodes may trivially be surveyed directly, we focus on the more challenging setting where nodes in the query set cannot be surveyed.

at each time step. Without loss of generality, assume for the moment that we survey one node at a time; at each iteration, the objective is

$$\text{argmin}_{u \in \mathcal{U}} \, \mathbb{E}_{q \in \mathcal{Q}} \left[\ell(f_G, q) \,|\, G \leftarrow G \cup \Psi(u) \right] + \varphi(u).$$

Still, we cannot measure this objective. We discuss heuristics to approximate it in the following section, and address surveying strategies based on these heuristics in 5.4.3.

5.4.2 The Smoothness Assumption

To motivate the discussion of survey strategies, we examine the following scenario. Recall that $\mathcal{Q}$ is the set of query nodes and $\mathcal{S}$ the surveyed nodes, and let $\mathbb{Q}$ and $\mathbb{S}$ denote their respective empirical distributions. That is, for a random variable Z taking values in $\mathcal{Z}$, $\mathbb{Q}(Z) = \Pr[Z \in \mathcal{Q}]$, and similarly for $\mathbb{S}$. If the loss is bounded by M for any $z \in \mathcal{Z}$, then by the triangle inequality, we have that

$$\begin{aligned}
&\mathbb{E}_{q \in \mathcal{Q}} \left[\ell(f_G, q) \,|\, G \right] \\
&\quad = \sum_{z \in \mathcal{Z}} \ell(f_G, z) \, \left(\mathbb{Q}(z \,|\, G) - \mathbb{S}(z \,|\, G) + \mathbb{S}(z \,|\, G) \right) \\
&\quad \leq \mathbb{E}_{s \in \mathcal{S}} \left[\ell(f_G, s) \,|\, G \right] + M \sum_{z \in \mathcal{Z}} | \, \mathbb{Q}(z \,|\, G) - \mathbb{S}(z \,|\, G) \, | \\
&\quad = \mathbb{E}_{s \in \mathbb{S}} \left[\ell(f_G, s) \,|\, G \right] + M \, || \, \mathbb{Q}(Z \,|\, G) - \mathbb{S}(Z \,|\, G) \, ||_{\text{TV}} \,, \qquad (5.1)
\end{aligned}$$

where $|| \cdot ||_{\text{TV}}$ is the *total variation norm*. We interpret 5.1 to mean that the difference between the average errors over $\mathcal{Q}$ and $\mathcal{S}$ is a function of the *statistical distance* between their respective distributions. Furthermore, note that $\mathbb{E}_{s \in \mathcal{S}} \left[\ell(f_G, s) \,|\, G \right]$ is an empirically measurable quantity, which is (typically) minimized by the learning

algorithm. Thus, in order to minimize the error over $\mathcal{Q}$, we must not only minimize the empirical error over $\mathcal{S}$, but also survey nodes such that the $\mathbb{S}$ becomes "close" to $\mathbb{Q}$.

Since the labels of $\mathcal{Q}$ and the unsurveyed set $\mathcal{U}$ are hidden, deciding which subset $\mathcal{S}$ will minimize the distance between $\mathbb{Q}$ and $\mathbb{S}$ is hard. Fortunately, intuition offers a solution in the form of *distributional smoothness*. A common assumption in semi-supervised learning is that the distribution over the instance space is "smooth"—that is, high density areas are likely to exhibit the same labels. This assumption has been used to explain the effectiveness of instance-based methods, such as k-nearest neighbors [40] and various semi-supervised approaches [194]. We can adapt this reasoning to the query-driven setting. Let P be some property associated with each node, taking values in a space $\mathcal{P}$. For instance, a specific feature value, or perhaps its encoded location in the network. We say that a query-driven problem is *smooth with respect to a distance function* d if there exists a constant $\beta \geq 0$ such that, for any $p, p' \in \mathcal{P}$,

$$\left|\left| \Pr_{v \in \mathcal{V}}[v \mid v.P = p] - \Pr_{v \in \mathcal{V}}[v \mid v.P = p'] \right|\right|_{\text{TV}} \leq \beta \; d(p, p'). \tag{5.2}$$

In other words, the statistical distance[2] between the conditional distributions of a node with property p versus a node with property p' should be bounded by a constant multiplier of the distance between p and p'. 5.2 suggests a strategy for minimizing the distance between $\mathbb{Q}$ and $\mathbb{S}$ without having access to the labels: *if*

[2]One could define smoothness using an alternate notion of statistical distance. In this case, the total variation norm fit nicely with the preceding analysis.

the smoothness property holds for a given distance function, then survey nodes in $\mathcal{U}$ *that have minimal distance to nodes in* $\mathcal{Q}$.

Identifying a distance function for which the smoothness assumption holds is a fundamental challenge in the query-driven setting. There are a number of metrics to choose from, and the appropriateness of any given one depends on the data. We emphasize the fact that smoothness is an *assumption* that we make about a particular problem. Indeed, in certain applications, this assumption may not hold for *any* metric. Yet it is reasonable to assume that it does hold in certain cases, given insight into the problem domain.

5.4.2.1 Feature Smoothness

A common assumption in data analysis is that the distribution exhibits smoothness with respect to a similarity or distance function in feature space. In the query-driven setting, we can assume that nodes that are similar (or close) in feature space will exhibit similar label distributions; in other words, the problem is smooth with respect to attribute similarity (or distance).

The exact nature of the similarity or distance function is context-specific. One popular similarity measure for arbitrary vectors is *cosine similarity*, with Euclidean distance as the associated distance function. This has been shown particularly effective with text data represented as TF/IDF-weighted word frequencies [106]. If the data contains string values, one may also desire string similarity measures, such as the string edit distance (which commonly refers to the *Levenshtein distance*).

5.4.2.2 Structural Smoothness

A fundamental assumption at the heart of relational domains is that the labels of related (i.e., connected) nodes are correlated. Collective methods have been shown to outperform traditional local models because they can exploit these correlations (e.g., [149]). Consequently, a natural similarity criterion for network data is adjacency.

Since the structure of the network may be only partially observed, there may be few direct adjacencies to the query set. One can address this problem by also applying a *link predictor* to the graph. Much work has been done on this topic, resulting in learning algorithms to infer the existence of missing edges. If these methods are too expensive, one can use a simpler, path-based link predictor instead. One such method [99] is the *Katz score*, defined as

$$d_{\text{Katz}}(q,u) \triangleq \sum_{t=1}^{\infty} \beta^t \left| A^t(q,u) \right|,$$

where $\beta \in [0,1]$ is an attenuating constant and $A^t(q,u)$ is the set of all length-t paths between q and u. (For efficiency, one can approximate this score by giving an upper bound to the maximum length considered, since longer paths will have little impact on the score.) Note that this is a purely structural measure, whose effectiveness cannot be explained by attribute similarity. Furthermore, since it will tend to assign higher scores to directly adjacent nodes, it provides an easy way to integrate observed edges; one can therefore use the Katz score as a single indicator of both observed and inferred adjacency.

5.4.3 Survey Strategies

We now discuss exactly *how* we determine which nodes to survey subject to a budget of k surveys. Under the smoothness assumption, we expect high utility from nodes that are close (with respect to a metric d) to the query nodes $\mathcal{Q}$. This invokes two questions: (1) how to compute utility for each unsurveyed node; (2) how to sample within the budget.

To address the first question, we could compute an aggregate utility value for each $u \in \mathcal{U}$ by summing $d(q, u)$ over all $q \in \mathcal{Q}$. However, since $\mathcal{Q}$ may exhibit high variance, the aggregated utility may yield little overall benefit. For example, suppose that $\mathcal{Q}$ lies on the surface of a multidimensional sphere (in feature space); applying an aggregate feature similarity will result in selecting nodes at the middle of the sphere, which, while equidistant to all query nodes, may not be as informative as those closer to the perimeter. As such, instead of computing an aggregate utility, we could sample from the full cross-product of $\mathcal{Q} \times \mathcal{U}$ according to which u is the best proxy for each q. For each $q \in \mathcal{Q}$, we compute the utility of every $u \in \mathcal{U}$ with respect to q, then add the highest scoring u to a pool of survey candidates $\mathcal{U}^c$. The usefulness of each survey candidate is thus conditioned on a particular query node, instead of over all query nodes. Interpreted differently, the utility measures the amount of proxy information for a *specific* query node.

Given $\mathcal{U}^c$ and a budget constraint of k surveys, we must determine how to sample from this set. Assuming the utility function is perfect, we could just select the top-k nodes. Yet since the utility is predicated on an *assumption* about the data,

a deterministic selection might yield suboptimal results. For this reason, we propose introducing stochasticity by performing a weighted random sampling according to utility.

To summarize, for each query node, we select its proxy from the pool of unsurveyed nodes, based on the given utility (i.e., distance) function, and flag it as a survey candidate. From the pool of survey candidates, we then perform a weighted sampling, proportional to the utility. The following section introduces an adaptive surveying strategy to combine feature- and structure-based criteria.

5.4.4 An Adaptive Survey Strategy

Any smoothness assumption—be it feature-based, structural, or otherwise—is only an assumption, and is wholly data-dependent. There is no single utility function that will always work. That said, given a set of potentially useful metrics, one can adaptively select the best one for the given problem and current information.

We develop the Adaptive Surveying for Query-driven Collective Classification (ASQ2C) algorithm to adaptively choose between feature-based and structural metrics. This algorithm uses a novel mechanism for determining when to trust structural measures by using the *assortativity* [127] of the currently observed graph. Let e_y be the fraction of edges in the network that connect two nodes of class y. Let s_y be the fraction of edges with source nodes that are in class y. Similarly, let t_y be the fraction of destination nodes in class y. The assortativity of a graph is defined as

$$\text{assortativity}(G) = \frac{\sum_{y \in \mathcal{Y}} e_y - \sum_{y \in \mathcal{Y}} s_y t_y}{1 - \sum_{y \in \mathcal{Y}} s_y t_y}.$$

Algorithm 1 ASQ2C Algorithm

Input: Initial network $G = (\mathcal{V}, \mathcal{E})$; set of query nodes $\mathcal{Q}$; cost function φ; feature similarity d_{fs}; structural similarity d_{ss}; survey budget B; survey batch size k.
Output: the surveyed network G.

$\mathcal{S} \leftarrow \emptyset, \quad \mathcal{U} \leftarrow \mathcal{V}$
while $B > 0$ **do**
    $\alpha \leftarrow$ Estimate assortativity of G
    With probability $p = |\alpha|$, $d \leftarrow d_{\text{ss}}$; else $d \leftarrow d_{\text{fs}}$
    $\mathcal{U}^c \leftarrow \emptyset$
    for $q \in \mathcal{Q}$ **do**
        $u_q \leftarrow \text{argmax}_{u \in \mathcal{U} \setminus (\mathcal{U}^c \cup \mathcal{Q})} d(q, u)$
        Add u_q to $\mathcal{U}^c$ with weight $d(q, u_q)$
    end for
    $\mathcal{U}^s \leftarrow$ Weighted sampling of k nodes from $\mathcal{U}^c$
    for $u \in \mathcal{U}^s$ **do**
        $G \leftarrow G \cup \Psi(u)$
        $\mathcal{S} \leftarrow \mathcal{S} \cup u, \quad \mathcal{U} \leftarrow \mathcal{U} \setminus u$
        $B \leftarrow B - \varphi(u)$
    end for
    $f_G \leftarrow \mathcal{A}(G)$
end while

Informally, assortativity is a measure of how correlated the nodes in a network are. We use this as an indicator of when there is sufficient correlation to use the structural similarity as the utility function. More specifically, with probability equal to the absolute value[3] of the assortativity, we decide to exploit the structural smoothness; otherwise, we use the feature smoothness. Note that because the labels of most nodes and edges are initially unobserved, we cannot compute assortativity of the fully observed graph exactly. We instead estimate the assortativity of the currently observed graph using the observed edges and both the observed and predicted labels.

The rest of the algorithm follows the strategy outlined in 5.4.3. The details of the ASQ2C algorithm are shown in Algorithm 1.

[3]The assortativity ranges from -1 to 1: positive scores indicate correlation, and negative scores indicate anticorrelation. In either case, the magnitude is the quantity we are interested in, as it indicates how much signal can be obtained from network structure.

5.5 Empirical Evaluation

We evaluate our approach using several benchmark collective classification datasets. We begin by describing the characteristics of these networks, and our general experimental setup. We evaluate our active surveying strategies on these networks and compare the performance to active learning approaches.

5.5.1 Experimental Setup

In these experiments, we use four real-world networks: CORA, CITESEER, WIKIPEDIA, and PUBMED[4]. The first two, CORA and CITESEER, are networks of computer science publications. In these publication networks, each node represents a publication and each edge a citation. Each node is annotated with a vector of binary word indicators (i.e., whether it contains each word) and a label indicating the paper topic. The WIKIPEDIA network consists of Wikipedia articles, wherein each node represents an article and each edge a hyperlink between articles. Each node is annotated by a vector of TF/IDF-weighted word frequencies and a label specifying the general category. Finally, the PUBMED citation network is a set of articles related to diabetes from the PubMed database. Node attributes are TF/IDF-weighted word frequencies and the labels specify the type of diabetes addressed in the publication.

For each dataset, we limit our experiments to the largest connected component. For the purposes of collective classification, we ignore the directionality of hyperlinks and citations. To prepare the word attribute data, we use stemming, stop-word

[4]Datasets available from: http://www.cs.umd.edu/projects/linqs/projects/lbc.

Table 5.1: Statistics on the four real-world networks used in the evaluation

Network	# Nodes	# Edges	# Labels	Avg. Degree
CORA	2485	5209	7	4.2
CITESEER	2110	3705	6	3.5
WIKIPEDIA	2776	30574	12	22
PUBMED	19717	44338	3	4.5

removal, and filter for the highest TF/IDF-weighted words to reduce the size of the dictionary to 500. Statistics for the resulting networks are given in Table 5.1.

In all of our experiments, the learning algorithm receives a partially observed network where the node labels are hidden, but the node features, a random 10% of the edges, and attributes are observed. Whenever a node is surveyed, the learner acquires the node's label and its incident edges.

5.5.2 Methodology

We compare our adaptive query-driven approach (ASQ2C) to two commonly used active learning baseline strategies: uniform random sampling (RAND) from the unsurveyed nodes $\mathcal{U}$, and weighted uncertainty sampling (UNC) over the $\mathcal{U}$ based on entropy [144]. We also compare to variants of ASQ2C which only exploit one of the smoothness types each: QD_{FS} for feature smoothness and QD_{SS} for structural smoothness. As mentioned in Section 5.4.2, we use cosine similarity for feature smoothness and the approximate *Katz score* for structural smoothness. In all experiments, we set the survey batch size $k = 10$ and allow the algorithm to run for 30 iterations (yielding an effective budget of 300 surveys).

Our algorithm is largely agnostic to the underlying collective classification

model. For our experiments, we use a semi-supervised variant of the Iterative Classification Algorithm (ICA) [21] to perform the collective classification. In ICA, each node is annotated with a vector of its attribute values (i.e., words), its label, and the label distribution of its neighbors. ICA learns two base classifiers: a local classifier and a relational classifier. The local classifier, trained on the observed labels using only the attribute values, is used to bootstrap the unobserved labels prior to learning the relational classifier. The local classifier is also used to bootstrap the unobserved labels prior to applying the relational classifier during inference. The relational classifier, trained on the observed labels using the attribute values and neighbor label distribution, is then iteratively applied during inference to propagate the labels. We use linear support vector machines [31] for both classifiers.

To evaluate our approaches under different conditions, we explore various query set generating processes. We evaluate both on query sets that are generated by uniform random sampling and query sets generated by targeting a particular structural or feature characteristics, described in greater detail below.

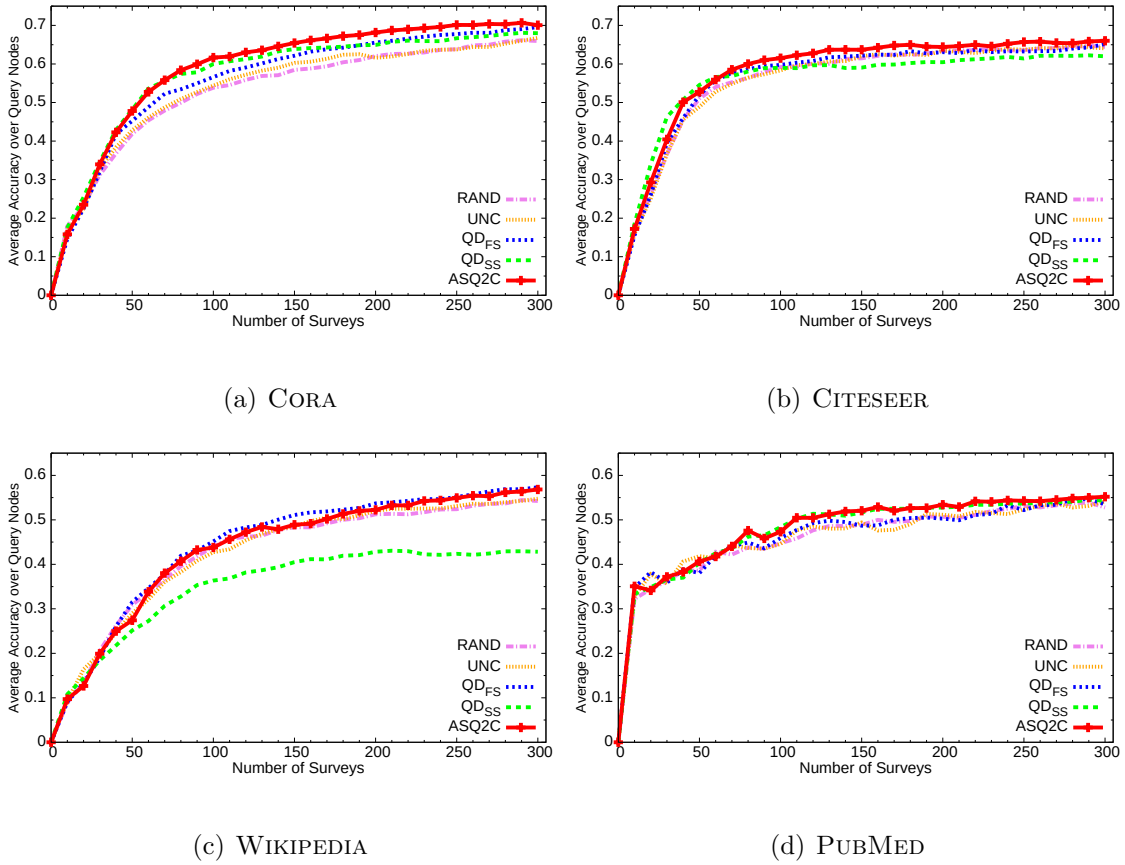
5.5.3 Sampled Query Sets

For our first set of experiments, we create query sets by randomly sampling (uniformly and without replacement) 5% of the nodes. Figure 5.1 plots the average classification accuracy (each point averaged over 40 runs) as additional surveys are performed. Table 5.2 lists the number of iterations that ASQ2C outperforms each other method on average, and lists in parentheses the number of times the improve-

ment by ASQ2C is statistically significant via a paired t-test. We find that in all cases, at least one of our query-driven approaches outperforms both RAND and UNC. ASQ2C performs best for over a majority of the budgets considered, with most of these gains deemed statistically significant. Specifically, ASQ2C achieves performance improvements of up to 17% over RAND and UNC. It is important to note that neither QD_{FS} nor QD_{SS} performs uniformly well on *all* datasets, thus motivating the adaptive strategy of ASQ2C. We find that the structural distance criterion works well for CORA, CITESEER, and PUBMED; this is likely due to the fact that paper topic is typically correlated across citations. In these datasets, attribute similarity is not as strong an indicator, and so QD_{FS} does not perform as well. However, in the WIKIPEDIA dataset, we find that QD_{FS} performs very well, while QD_{SS} performs the worst; this is likely due to the fact that WIKIPEDIA articles often link to a large number of unrelated articles, whereas their word frequencies are better indicators of topic. Analyzing the true assortativity of these datasets supports this claim. We find that CORA, CITESEER, and PUBMED have high assortativities with respective values of 0.79, 0.67 and 0.69; meanwhile, WIKIPEDIA has a low assortativity of 0.36.

Focusing on the query-driven strategies, we find that ASQ2C generally outperforms QD_{FS} and QD_{SS} on all citation networks, by as much as 12% and 8% respectively. Only on the WIKIPEDIA dataset did a non-adaptive strategy generally outperform our adaptive approach, typically in the early iterations (i.e., low survey budgets); and even in this case, ASQ2C is still competitive. We note, however, that the non-adaptive strategies are only useful if we know *a priori* which metric to use

in advance, which is rarely the case in practice.



(a) Cora (b) Citeseer

(c) Wikipedia (d) PubMed

Figure 5.1: Accuracy per iteration (i.e., survey budget) of active surveying averaged over 40 runs each of the Cora, Citeseer, Wikipedia, and PubMed networks. Each point indicates the average accuracy after surveying some number of nodes.

5.5.4 Targeted Query Sets

In practice, query sets are selected for some context-specific reason, and thus may have certain targeted characteristics. For example, in the disease transmission example of Section 5.2.2, where physical contact is a significant factor, query nodes may tend to be highly interconnected. Similarly, in the viral marketing example

Table 5.2: Number of iterations (out of 30) where ASQ2C scores higher on average (wins) or lower (losses) than each other method. Of those, the number of significant wins and losses, using paired t-tests with 90% significance, are listed in parentheses.

		# of Wins		# of Losses	
CORA	RAND	29	(28)	1	(0)
	UNC	29	(27)	1	(0)
	QD_{FS}	30	(25)	0	(0)
	QD_{SS}	24	(21)	6	(0)
CITESEER	RAND	30	(20)	0	(0)
	UNC	30	(23)	0	(0)
	QD_{FS}	30	(18)	0	(0)
	QD_{SS}	24	(23)	6	(2)
WIKIPEDIA	RAND	24	(9)	6	(1)
	UNC	26	(11)	4	(1)
	QD_{FS}	4	(0)	26	(7)
	QD_{SS}	28	(26)	2	(0)
PUBMED	RAND	27	(17)	3	(0)
	UNC	26	(18)	4	(1)
	QD_{FS}	26	(14)	4	(1)
	QD_{SS}	22	(1)	8	(0)

of Section 5.2.3, query nodes may share a common characteristic such as being popular or prolific. To study the impact of more targeted generating processes, we next generate query sets with two types of targeted sampling: a structure-based context and a feature-based context.

To generate a structure-based query-generating process, we select a query set using *snowball sampling*. In snowball sampling, we initialize the query set using a *seed* node. We then proceed to sample each of its neighbors with probability p_{neigh}; if we do not sample a neighbor (which occurs with probability $1-p_{\text{neigh}}$), then we select a random node from the remaining unsampled network. We repeat this process for each node currently in the query set, until the number of query nodes reaches 5% of the overall network. We perform this procedure for $p_{\text{neigh}} = 0.1, 0.5, 0.9$. Note that, for higher values of p_{neigh}, the query set tends to be a connected component.

Conversely, for lower values of p_{neigh}, the query set tends to be randomly distributed throughout the network. We test this structure-based setup using the CITESEER network (Figure 5.2), repeating the experiment for 40 runs by sampling query sets using different random seeds.

To recreate a targeted sample based on feature, we first identify a set of words such that the probability of occurrence is low (below 5%) and which a domain expert may find interesting. We then generate the query set from all documents that contain the word. For this set of experiments, we focus on the PUBMED network. We used domain knowledge to select words such as "death", "hypoglycemia", and "suppress" as the criteria for adding a paper to the query set. Figure 5.3 shows the results of these experiments.

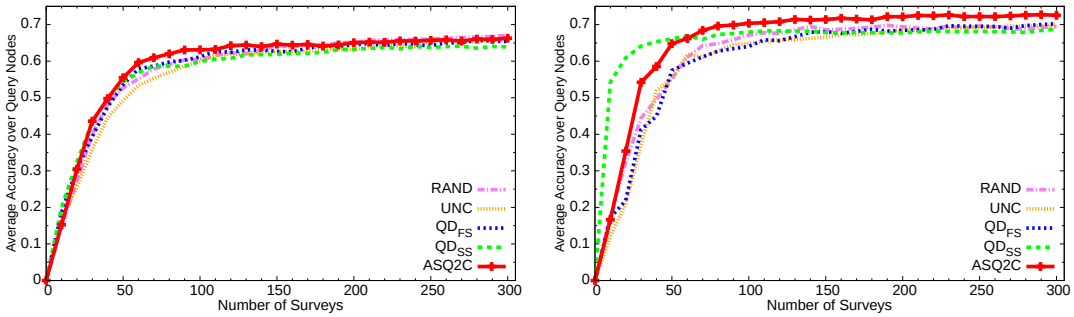
Examining the results, we see similar trends as before, with ASQ2C showing even greater improvement over the baselines. Two important observations when comparing the targeted query-set setting with the random query-set setting. First, while ASQ2C is still overall the best performing, there are cases where either QD_{FS} or QD_{SS} outperform ASQ2C on targeted query sets. We see this change when comparing low and high values of p_{neigh} and when comparing the results between the randomly generated and attribute-based query sets. The effectiveness of the non-adaptive smoothness heuristics is especially noticeable when the number of surveyed nodes is particularly small (i.e., the learner's budget is small). This effect implies that when budget is particularly low for query sets that exhibit clear biases, and there is domain knowledge that can identify in advance whether feature or structure smoothness is more likely, using either QD_{FS} or QD_{SS} alone can potentially yield

better results. For most greater budgets, however, and in the absence of prior knowledge about the general characteristics of the data, ASQ2C generally yields the best performance.

Next, we observe a general upward trend when comparing the results from the randomly generated query sets to targeted query sets. In both cases, the stronger the bias for the query set sampling, the greater the improvement over the non-query-driven strategies. For example, while the percent improvements of ASQ2C over RAND and UNC reach up to 12% and 17% for uniformly random query sets, we find improvement as great as 22% and 68% accuracy for high values of p_{neigh}. Similarly, in the PubMed experiments, where we reach up to 10% and 11% improvement over RAND and UNC on a uniformly random query set, using ASQ2C on query sets defined by the word attributes improves accuracy by up to 28% and 44%. Consequently, while ASQ2C already yields significant improvements in the uniformly random query-set setting from the previous section, the results from tests in this section indicate that the more realistic setting where the query nodes are selected based on some measurable criteria will benefit even more.

5.6 Conclusion

Query-driven collective classification is an important but understudied problem, applicable to a variety of domains. The query-driven setting, when coupled with active surveying for partially observed networks, is natural in practice. It provides an opportunity to develop high impact algorithms for maximizing predictive



(a) CITESEER, $p_{\text{neigh}} = .10$ (b) CITESEER, $p_{\text{neigh}} = .90$

Figure 5.2: Accuracy per iteration averaged over 40 runs on the CITESEER dataset where the query set is selected using snowball sampling.

performance, over a range of annotation budgets. We identify two forms of data smoothness, feature-based and structure-based, and demonstrate how to exploit them for query-driven active surveying. We then develop the ASQ2C algorithm to automatically determine the optimal smoothness assumption, given the observed information. We evaluate these survey strategies on real network data and show that our query-driven methods exhibit significant advantages over traditional (non-query-driven) active learning heuristics. There is much room for further exploration: for example, query-driven active surveying in which surveys may return incomplete or noisy information; exploring non-uniform cost structures; and application in dynamic networks. Nevertheless, our work identifies this important and challenging problem setting, and represents a major first step in addressing it.

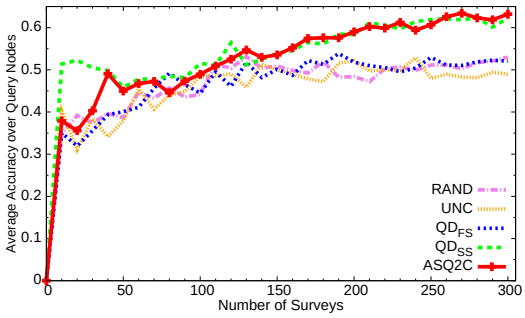


Figure 5.3: Accuracy per iteration averaged over 40 runs on the PubMed dataset where the query set is selected by filtering on keywords (e.g., death, hypoglycemia, stress).

Chapter 6

Collective Graph Identification

In previous chapters we discussed the problems of entity resolution, link prediction, and collective classification as individual tasks. While each of these problems have been studied separately, they have never been considered together as a single coherent task. In this chapter, we discuss how these tasks are inherently inter-related in the the problem we define as *graph identification*. We develop an approach using coupled collective classifiers and empirically show the importance of jointly performing these tasks. We also show the superiority of our proposed approach over previously proposed joint approaches in both prediction quality and runtime on a variety of real-world datasets.

6.1 Introduction

In recent years, there has been a surge of interest in network analysis applied to diverse domains including social networks, technological networks, biological networks and more. In part, this interest is driven by the burgeoning growth in the amount of digital information describing network data that is available including e-mail, citation collections, epidemiological data, and social media. Such data contain a wealth of information (e.g., key individuals, communities, and contagion trends) that, when uncovered, can help to create better predictive models and help elucidate

general laws governing network evolution. However, the available network data is typically noisy, observational, and, while it provides useful signal for uncovering the underlying sociological or technological network, it is not the *same* thing.

We define the process of discovering the hidden structure which gives rise to observational network data as the problem of *graph identification*. Figure 6.1 illustrates an example of inferring a social network (Figure 6.1(b)) from an email communication network (Figure 6.1(a)). We refer to the observational network data as the *input graph*, and the hidden network of interest as the *output graph*. Graph identification uncovers the hidden network by simultaneously solving three problems:

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

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

Node labeling:[1] determining the label of nodes in the output graph, e.g., "Is Neil a CEO, manager or assistant?".

In addition to constructing the hidden output graph, graph identification involves constructing the mapping from nodes in the input graph to nodes in the output graph (Figure 6.1(c)).

[1]Node labeling is also known as *collective classification*. In this chapter, we use the term node labeling to emphasize that this problem predicts the label attributes of nodes.

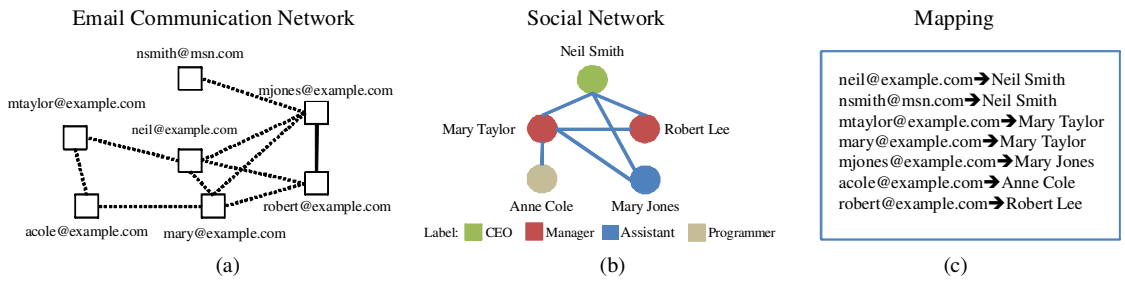


Figure 6.1: Input and output of graph identification. (a) Input graph representing a communication network where the nodes are email addresses and the edges are email communications. (b) Output graph representing the social network identified by graph identification. The nodes correspond to people and the edges to employee-manager relationships. The people are also labeled with their roles. (c) Mapping from input to output nodes.

Each task informs the others, and by solving them simultaneously, we allow information to propagate among them to obtain better solutions. For example, in a bibliographic domain, predicting whether one paper cites another (link prediction) allows us to determine whether two papers cite common papers. Co-citation helps us to decide whether they have the same topic (node labeling), which in turn aids in ascertaining whether they are the same paper (entity resolution). This last information in turn helps to determine the citation links from the two papers to other papers, closing the information propagation loop. While previous work [123, 99, 169, 28, 160, 16, 149] has addressed each of these tasks separately, to our knowledge, we are the first to efficiently address them simultaneously.

To address the problem of graph identification, we present the C^3 (Coupled

Collective Classifiers) algorithm. C^3 defines a probabilistic model to capture the dependencies within each task as well as the relational interactions among all three. While it is conceptually possible for standard probabilistic inference algorithms to jointly solve all three tasks within the framework of our model, in practice they are too computationally expensive for large real-world datasets. C^3 uses an iterative procedure that simultaneously solves all three tasks. It begins by using a local classifier based solely on observed information in the input graph to solve each task independently. Then it iteratively propagates these solutions among the three tasks by means of relational features that capture the interactions within and among the tasks. To further tailor C^3 as a practical approach, we designed it to address the real-world scenario where it is costly to obtain fully labeled network data. C^3 adopts a *semi-supervised* learning algorithm that can exploit training data with only a small fraction of labeled examples. We consider multiple variants of C^3 based on different learning and inference paradigms and empirically show that by propagating information, C^3 significantly improved predictive accuracy on four real-world networks. We also provide scalability results for C^3 by showing its runtime performance over large synthetic network datasets, as well as its runtime performance when using multiple threads as we exploit the natural parallelizability of its computation.

The remainder of the paper is organized as follows. We discuss the problems involved in graph identification, previous work in those problems, and early attempts to jointly perform subsets of those problems in Section 6.2. We then provide a background review in Section 6.3. We describe C^3 in detail (Section 6.4) and report our experiments (Section 6.5). We conclude with future work (Section 6.6).

6.2 Graph Identification

Graph identification is the problem of discovering the hidden structure which gives rise to observational network data. The problem consists of three tasks, corresponding to the three major components of a graph. First, we merge nodes in the observational network data to the nodes they refer to in the hidden network (entity resolution). Next, we infer the edges between the nodes of our hidden network (link prediction). Finally, we infer the attribute values of our hidden network (node labeling). There is significant prior work exploring the tasks within graph identification individually. In this section, we discuss the previous work in each of these tasks. We also discuss the limited related work at various ways these tasks are inter-dependent, including various joint approaches that may also be used to perform the tasks jointly. We note that to our knowledge, none of these approaches have ever been used for the complex structured prediction problem of collectively inferring a full graph.

6.2.1 Independent Models

The three tasks within graph identification are individually well studied. For all three tasks, previous work can be naturally separated into two broad categories, local approaches and approaches which exploit the dependence of predictions within each task (intra-dependence). The local approaches, consisting the early work in each task, focused on using the node attributes to perform the inference. For entity resolution, various attribute similarity measures have been proposed such that nodes

whose attributes are similar above a defined threshold are predicted co-referent (e.g., name mentions with similar spelling likely refer to the same individual) [18, 36]. Early work in link prediction also used similarity with the observation that many networks are homophilic (e.g., individuals with similar characteristics are likely to be friends) [116]. For node labeling, various classification models including naïve Bayes, decision trees [136], and support vector machines [31] were proposed to label the nodes using their observed attributes (e.g., words in papers to infer the topic of the paper).

More recent work exploit the relationships which exist between nodes, in particular the intra-dependencies they introduce, to perform these tasks. For entity resolution, approaches used knowledge that two nodes are predicted co-referent to collectively infer that related nodes may also be co-referent to each other [16, 160, 181]. Within the link prediction, approaches collective inferred which nodes share a link using concepts like triadic closure (two nodes predicted to share a link with a common node are likely to share an edge as well) [35, 86]. Finally, approaches have been proposed to collectively infer the labels of nodes with the assumption that the labels of related nodes are correlated [104, 114, 149].

6.2.2 Joint Models

While the vast majority of work in entity resolution, link prediction, and node labeling have looked at each as independent tasks, there has been limited work in recent years that have looked at how different pairs of these tasks are inter-

dependent. Taskar et al. [169] explored jointly performing link prediction and node labeling using Relational Markov Networks. Bhattacharya et al. [17] used a probabilistic generative model to perform entity resolution and node labeling. Wick et al. [182] performed entity resolution and node labeling using conditional random field. To our knowledge, however, previous work has not formulated the full complex structured prediction problem as interacting components in order to collectively infer a graph.

We note that graph identification is related to domain-specific joint inference problems such as information extraction in natural language processing [145], network mapping in computer networks [156], and biological network inference in bioinformatics [107]. While graph identification may provide a unifying paradigm for these problems and others, there are some important differences as well. Information extraction traditionally infers structured output from unstructured text (e.g., newspaper articles, emails), while graph identification is specifically focused on inferring structured data (i.e., the output graph) from other structured data (i.e., the input graph, perhaps produced from a noisy information extraction process). Network mapping and biological network inference are also related to graph identification, but they are mainly concerned with inferring only network topology.

6.3 Background

Throughout the paper, we use an uppercase letter to represent a random variable (e.g., Y) and a lowercase letter (e.g., y) to represent its value. Bold letters

represent a vector or set (e.g., $\mathbf{Y}$) and their values (e.g., $\mathbf{y}$).

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

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}_c) \tag{6.1}$$

where $Z = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}'_c)$ is a normalization constant. The potential functions are often represented more compactly as a log-linear combination over a set of features: $\phi_c(\mathbf{y}_c) = \exp\left(\sum_i w_i f_i(\mathbf{y}_c)\right) = \exp\left(\mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right)$. In this case, Equation 6.1 can be equivalently expressed as

$$P(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{c \in \mathcal{C}} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{y}_c)\right). \tag{6.2}$$

In many applications, we are interested in conditional distributions where a subset of the variables $\mathbf{X}$ are provided as *evidence*, and we predict a set of *target* variables $\mathbf{Y}$. A *conditional Markov network* defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$, where the partition function $Z(\mathbf{x})$ now depends on $\mathbf{x} : Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}'_c)$.

Note that evaluating the above equation requires that we compute $Z(\mathbf{x})$, which in turn, requires that we sum over all possible assignments to $\mathbf{y}'$. Since this is exponential in $|\mathbf{Y}|$, computing $Z(\mathbf{x})$ and hence the equation are generally intractable.

A common approximation is the pseudolikelihood [15]:

$$
\begin{aligned}
& P^*(\mathbf{y} \mid \mathbf{x}) \\
& \quad = \prod_i P(y_i | \mathbf{y}_{-i}, \mathbf{x}) \\
& \quad = \prod_i \frac{\exp\left(\sum_{c \in \mathcal{C}: y_i \in \mathbf{y}_c} \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right)}{Z(\mathbf{y}_c \setminus y_i, \mathbf{x})}
\end{aligned} \tag{6.3}
$$

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

6.4 Coupled Collective Classifiers

C^3 takes a graph $(\mathbf{V}, \mathbf{E})$ as input where $\mathbf{V}$ and $\mathbf{E}$ are respectively a set of vertices and directed edges[2]. Each vertex $v \in \mathbf{V}$ represents a reference to an entity, and each edge $(v_i, v_j) \in \mathbf{E}$ represents an interaction between references v_i and v_j. Each node v_i in the input graph has associated attributes A_i. For example, if a node represents a reference to a paper, the attributes may describe the words which appear in the paper. Edges (v_i, v_j) may also have associated attributes, denoted A_{ij}. An example of an edge attribute is the number of emails sent on a communication link from person v_i to v_j. We use $\mathbf{A} = \{A_i\} \cup \{A_{ij}\}$ where $i, j = 1, \ldots, |\mathbf{V}|$ to denote the attributes of all input nodes and edges.

[2] C^3 extends straightforwardly to graphs with more than one kind of edge and hypergraphs. We focus on the case of a single edge type for simplicity of presentation.

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

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

Given the above definitions and using Equation 6.3, we can represent the joint probability over the target variables $\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p$ given evidence $\mathbf{X}$ as follows:

[3]In practice, we do not instantiate all the $|\mathbf{V}|^2$ variables in $\mathbf{R}$ and $\mathbf{L}$. Section 6.5 describes how we use filtering techniques to only create variables for pairs that have some possibility of being co-referent/linked.

Table 6.1: Cora and Citeseer Feature Definition

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*

$$
\begin{aligned}
&P^{*}\left(\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p \mid \mathbf{x}\right) \\
=&\left(\prod_{r_p \in \mathbf{r}_p} P(r_p | \mathbf{y} \backslash r_p, \mathbf{x})\right)\left(\prod_{l_p \in \mathbf{l}_p} P(l_p | \mathbf{y} \backslash l_p, \mathbf{x})\right)\left(\prod_{n_p \in \mathbf{n}_p} P(n_p | \mathbf{y} \backslash n_p, \mathbf{x})\right).
\end{aligned}
\tag{6.4}
$$

6.4.1 Features

C^3 makes use of two kinds of features: *local* and *relational*. Local features capture the dependencies between a single predicted variable and evidence. For

Table 6.2: Enron Feature Definition

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 6.3: Discourse Opinion Feature Definition

Task	Type	Feature Description
ER	Local	· Discourse and dialog continuity features defined in [162]
	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 [162]
	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 [162]
	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*

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

Note that a wide gamut of dependencies can be cast in terms of C^3's features. This is essential for graph identification because it allows us to exploit the diverse set of dependencies which have been proposed for each of the underlying tasks. Previous work in entity resolution, for example, has proposed using a variety of attribute similarity measures between potentially co-referent pairs of nodes [36]. Similarity measures have also been proposed to quantify the set similarity of "neighborhoods"

of pairs of nodes [16]. Common definitions of a node's neighborhood include adjacent nodes, all nodes within a given shortest path distance, and all nodes which have an adjacent node in common (e.g., all papers which cite some common subset of papers). All of these definitions can be captured in our framework.

Work in link prediction also makes use of features based on attribute and neighborhood similarity. These features capture the assumption that many networks are homophilic, i.e., similar nodes are likely to share a link. Link prediction features also tend to rely on topology-based characteristics which capture the structural similarity (e.g., degree) or proximity (e.g., existence of paths) between two potentially adjacent nodes [99]. In multi-relational networks, link prediction may rely on features based on the attributes of links between the same pair of nodes (e.g., attributes of a communication edge between people imply something about their social relationship). For node labeling, features traditionally include the observed attributes of the given node, as well as observed and predicted values of nodes in its neighborhood. Table 6.1, 6.2, and 6.3 contain more examples of local and relational features, and shows the diversity of features that we used in our experimental evaluation.

We use $\mathcal{F}$ to denote the set of features used by C^3, and $\mathbf{y}_f$ to denote the random variables used in the definition of features $f \in \mathcal{F}$. Then, from Equation 6.3 and Equation 6.4, we can represent the joint probability over the target variables $\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p$ as follows:

$$P^*\left(\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p \mid \mathbf{x}\right) = \prod_{y \in \mathbf{r}_p \cup \mathbf{l}_p \cup \mathbf{n}_p} \frac{\exp\left(\sum_{f \in \mathcal{F}: y \in \mathbf{y}_f} w_f \cdot f(\mathbf{x}_f, \mathbf{y}_f)\right)}{Z(\mathbf{y}_f \setminus y, \mathbf{x})}. \tag{6.5}$$

6.4.2 Weight Learning

Observe that Equation 6.4 decomposes into three terms, one for each of the $\mathbf{R}_p$, $\mathbf{L}_p$ and $\mathbf{N}_p$ target variables. A feature that is defined over more than one type of variable appears in more than one of the terms with the same weight (e.g.,$f(\mathbf{R}_p, \mathbf{L}_p, \mathbf{x})$ appears in both $\prod_{r_p \in \mathbf{r}_p} P(r_p | \mathbf{y} \setminus r_p, \mathbf{x})$ and $\prod_{l_p \in \mathbf{l}_p} P(l_p | \mathbf{y} \setminus l_p, \mathbf{x})$). We simplify the equation further by assuming that the appearances of such a feature in a term are distinct from those in another term, thus allowing the weights of the feature to be different. This simplifies the weight learning algorithm by allowing it to find the optimal weights for each term separately.

In C^3, we are interested in inferring the most likely assignment of the variables (also known as the *maximum a posteriori* state). Hence, for each term $\prod_{v \in \mathbf{v}} P(v | \mathbf{y} \setminus v, \mathbf{x})$ ($\mathbf{v} \in \{\mathbf{r}_p, \mathbf{l}_p, \mathbf{n}_p\}$), we want to find feature weights that maximize the ratio $\frac{P(v|\mathbf{y}\setminus v, \mathbf{x})}{P(v'|\mathbf{y}\setminus v', \mathbf{x})}$ between the conditional probability of each correct assignment v and every incorrect assignment v'. Taking logs of the ratio, we see that we are equivalently maximizing the *margins* $\sum_{f \in \mathcal{F}: v \in \mathbf{y}_f} w_f \cdot (f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v) - f(\mathbf{x}_f, \mathbf{y}_f \setminus v, v'))$ for each $v \in \mathbf{v}$ and $v' \neq v$, i.e.,

$$\text{maximize } \gamma \quad \text{s.t. } \sum_{f \in \mathcal{F}} w_f^2 \leq 1 \text{ and}$$

$$\forall v \in \mathbf{v},\, \forall v' \neq v \;\; \Delta f_{(\mathbf{x}_f, \mathbf{y}_f)}(v, v') \geq \gamma$$

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

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

$$\text{minimize } \tfrac{1}{2}\textstyle\sum_{f\in\mathcal{F}} w_f^2 \;\; + \;\; K\sum_{v\in\mathbf{v}}\xi_v \;\; \text{s.t.}$$

$$\forall v \in \mathbf{v},\, \forall v' \neq v \;\; \Delta f_{(\mathbf{x}_f,\mathbf{y}_f)}(v,v') \geq 1-\xi_v$$

where K is a constant. The above is precisely the optimization that a multi-class support vector machine (SVM) [41] performs[4]. Hence we train three SVMs, one for each of the $\mathbf{R}_p$, $\mathbf{L}_p$, and $\mathbf{N}_p$ variables.

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

6.4.2.1 Semi-Supervised Learning

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

One difficulty with partially observed data is that we cannot compute the

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

values of relational features containing unlabeled variables. One solution to this problem is to use the observed variables to train a new set of SVMs containing only local features, one SVM for each of the $\mathbf{R}_p$, $\mathbf{L}_p$, and $\mathbf{N}_p$ variables. These are then used to infer the values of the target variables (recall that their values are not observed). With these inferred values, we can evaluate the relational features involving predicted variables, and hence learn feature weights that optimize the margins for the originally observed variables. Algorithm 2 contains the pseudocode for C^3's semi-supervised weight learning.

Algorithm 2 C^3 Semi-supervised Weight Learning

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

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

6.4.3 Inference

Algorithm 3 gives the pseudocode for C^3's inference procedure. Given a set of target variables $\mathbf{Y} = (\mathbf{R}_p, \mathbf{L}_p, \mathbf{N}_p)$ and evidence $\mathbf{x}$, we begin by using a local SVM

(i.e., one containing only local features) with learned weights, w^{local}, to infer the values of each of the $\mathbf{R}_p, \mathbf{L}_p$, and $\mathbf{N}_p$ variables. At this point, the variable assignments are based solely on the evidence $\mathbf{x}$. The algorithm then proceeds to capture the dependencies between the variables. It iteratively evaluates the relational features using the variable values inferred in the previous iteration, and uses a relational SVM (i.e., one containing local and relational features) with learned weights w (or w^k when using stacked learning) to infer new variable values for the current iteration. The algorithm terminates when the variable values converge, oscillate, or a user-specified maximum number of iterations is reached.

We note that our iterative approach is similar in spirit to the iterative classification algorithm (ICA) presented by Neville and Jensen [123] and the link-based classification work by Lu and Getoor [101]. We note that previous work in these methods have mainly looked only at the problem of node labeling, using simple aggregations for relational features. C^3 is a generalization of these approaches which use coupled classifiers to perform multiple tasks simultaneously, as well as using a richer set of relational features including aggregate, set similarity, and structure-based features.

6.4.4 Constructing the Output Graph

Given an assignment of values to the predicted variables, we can construct an output graph. We create an entity node in the output graph for each collection of co-referent references. We also create edges between the entities based on whether the

Algorithm 3 C^3 Inference

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

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

majority of their corresponding variables $\mathbf{L}$, defined over their references, indicate that the entities are linked. Finally, we can assign the label to an entity based on the values in $\mathbf{N}$ corresponding to its references. It is possible that assignments to these variables are inconsistent (i.e., N_i may not equal N_j even though references i and j are predicted co-referent). In these cases, we can define a procedure to resolve the inconsistencies prior to generating the output graph (e.g., enforce transitivity over co-referent pairs, add edges between entities whose references have an edge, and taking the mode label over the labels of its references). For the evaluation in this paper, we evaluate over the predicted variables $\mathbf{Y}$ both with and without applying these procedures.

6.4.5 C^3 Variants

Beyond the semi-supervised learning and inference procedures discussed in Section 6.4.2 and Section 6.4.3, we explored variants of C^3 based on various learning and inference paradigms. For inference, we consider three variants of C^3's inference procedure which exploit the ability of SVMs to not only return the most likely value for each variable, but also a probability distribution over all possible values [184]. Previous work has shown that approaches which deterministically assign the most likely value can converge to a poor local optimum [67]. To avoid this and potentially reach a global optimum, we can replace deterministic assignments in C^3 with samples drawn from the inferred probability distribution. The first variant, C^3-PS, performs this sampling for each variable in each iteration. With probability $1 - (i/maxIter)$ where i is the current iteration, we assign the variable to the sampled value for that variable in iteration i. Otherwise, we proceed as before and set the variable to the most likely value. Note that by setting the probability of sampling relative to each iteration, we are more likely to sample in early iterations while later iterations are more likely to use the most likely value.

Our second variant, C^3-GS, is based on the extensive work in Gibbs Sampling [66]. Gibbs sampling is widely regarded as one of the most accurate approximate inference procedures. Unfortunately, it is also very slow due to the number of samples that need to be collected. As with C^3-PS, we sample values from the returned probability distributions for each variable. We then assign variables to these sampled values for a fixed number of iterations. After this "burn-in" period,

we not only continue assigning the sampled values but also maintain a count of how often a value was sampled and assigned for a given variable. We perform this count for some predefined number of samples. After all the samples are collected, we assign each variable to the to the value most sampled for that variable. Algorithm 5 gives the pseudocode for C^3's Gibbs sampling procedure.

Our third variant uses the idea that collective performance is improved by explicitly identifying and preferentially exploiting the more certain relational information. McDowell et al. [115] refer to approaches which use this idea as "cautious" and showed that cautious approaches can significantly improve performance over their "aggressive" counterparts. In our cautious variant of C^3, denoted C^3-CI, we modify the inference such that we only update the top K ($K = (1/maxIter) *$ (number of random variables)) most confident predictions (i.e., probability of most likely class) for each task. Variables whose values are set in an iteration are not updated in later iterations.

We also propose a learning and inference variant based on the work in expectation maximization (EM) [44]. In this variant, C^3-EM, we do not learn a single set of SVMs to iteratively apply. We instead relearn a different set of SVMs at the beginning of each iteration using the output of the previous iteration. Algorithm 4 contains the pseudocode for C^3-EM. As with our first weight learning algorithm, we begin by learning and applying SVMs containing only local features to infer values of the target variables. These inferred values are then used in the relational features for learning weights for our first iteration, denoted w^1. To learn the weights of subsequent iterations, w^k, we apply SVMs using the feature weights of the previous

iteration, w^{k-1}, to update the inferred values of the target variables. The updated inferred values are then used in the relational features for learning w^k over the originally observed variables. This is repeated for each iteration for some pre-determined number of iterations. The assignments when applying the classifiers with the last set of weights are returned as output. We note that while we only use the values of unlabeled instances in computing the relational features, typical applications of EM would have learned classifiers over both the labeled and unlabeled instances. We explored the variant where labeled and unlabeled instances were used as training instances and and found that it performed poorly.

Algorithm 4 C^3 EM Learning and Inference

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

$\mathbf{w}^{local} \leftarrow LearnWeights(\mathbf{f}^{local}, \mathbf{y}^{observed}, \mathbf{x}, \text{SVM})$
$\mathbf{w}^0 \leftarrow \mathbf{w}^{local}$
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  if $i == 1$, **then** $\mathbf{f} \leftarrow \mathbf{f}^{local}$, **else** $\mathbf{f} \leftarrow \mathbf{f}^{local} \cup \mathbf{f}^{relational}$
  for each $Y \in \mathbf{Y}^{predicted}$
    $y^i \leftarrow InferValue(Y, \mathbf{f}, \mathbf{w}^{i-1}, \mathbf{x} \cup \mathbf{y}^{i-1} \cup \mathbf{y}^{observed}, \text{SVM})$
  $\mathbf{w}^i \leftarrow LearnWeights(\mathbf{f}, \mathbf{y}^{observed}, \mathbf{x} \cup \mathbf{y}^i \cup \mathbf{y}^{observed}, \text{SVM})$
until $i = maxIter$
for each $Y \in \mathbf{Y}^{predicted}$
  $y \leftarrow InferValue(Y, \mathbf{f}, \mathbf{w}^i, \mathbf{x} \cup \mathbf{y}^i \cup \mathbf{y}^{observed}, \text{SVM})$
return y

Algorithm 5 C^3 Gibbs Sampling Inference

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

$i \leftarrow 0$
for each $Y \in \mathbf{Y}$
  $y^i \leftarrow InferValue(Y, \mathbf{f}^{local}, \mathbf{w}^{local}, \mathbf{x},$ SVM)
Initialize sample counts $c[\mathbf{Y}, \cdot] = 0$
repeat
  $i \leftarrow i + 1$
  for each $Y \in \mathbf{Y}$
    $y^i \leftarrow InferValue(Y, \mathbf{f}, \mathbf{w}, \mathbf{x} \cup \{\mathbf{y}^{i-1} \setminus y^{i-1}\},$ SVM)
    if $i > burnIn$
      $c[Y, y^i] = c[Y, y^i] + 1$
until $i = maxIter$ **or** $\mathbf{y}$ values converge
for each $Y \in \mathbf{Y}$
  $y^i \leftarrow argmax_l\ c[Y, l]$
return $\mathbf{y}^i$

6.5 Experimental Evaluation

6.5.1 Datasets

We evaluate our approach using three sets of real-world networks: citations network, email communication, and discourse opinion networks.[5] We also develop a novel data generator to create synthetic networks for use in evaluating the scalability of C^3.

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

6.5.1.1 Citation Networks

We evaluate on two citation networks, CORA and CITESEER [149]. In a citation network, nodes represent papers and directed edges represent citations. The CORA network contains 2708 nodes with 5428 edges. The CITESEER network contains 3312 nodes with 4732 edges. The nodes of both networks also contain, after pruning, 500 binary attributes representing the presence of a word in a paper, as well as a label indicating the topic of a paper (7 possible labels in CORA and 6 in CITESEER). Because noisy versions of these networks are not readily available[6], we create noisy versions of these graphs (i.e., input graphs) which attempt to mimic the types of noise likely encountered during the extraction of a network from multiple sources.

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

[6]We note that while there are annotations for entity resolution (e.g., [16, 159]), link prediction (e.g., [99]), and node labeling (e.g., [149]) available, we are unable to use them directly since they are over different subsets of the network.

0.3, and 0.4 (denoted Low, Medium, and High Noise, respectively).

For entity resolution and link prediction, because the inferences are made over pairs of nodes, there are important scalability issues. If done naively, both entity resolution and link prediction require $O(|V|^2)$ predictions. Clearly this will be intractable for all but the smallest of graphs. In both tasks, a filtering step is often applied to limit the potential pairs that are considered [110, 169]. This is crucial for making the algorithm scalable, and has been shown to improve the accuracy of the predictions. The filtering step is referred to as *blocking* [59] or *canopies* [110]. Any method that can quickly identify the potential pairs while minimizing the false negatives can be used. In our setting, the blocking criterion for entity resolution filters potential pairs as nodes which have at least two nodes, adjacent via edges in the input graph, in common. For link prediction, the blocking criterion filters potential pairs as nodes which have an extracted edge between them. Note that while this substantially reduces the number of potential pairs, in our experiments there remain up to $120,000$ pairs for entity resolution and $34,000$ pairs for link prediction.

6.5.1.2 Email Communication Network

The second type of network we evaluate over is a corporate communication and social network, based on the ENRON dataset [90]. The input graph is an email communication network where the nodes correspond to email addresses, directed edges represent emails sent from one email address to another, and edge attributes

indicate the words used and the number of communications between those email addresses. The output graph is a social network where the nodes represent people, edges indicate a managerial relationships, and the node labels indicate people's titles. We also have annotations on which email addresses belong to the same person. The full network consists of 211 email address nodes with 2837 directed communication edges corresponding to 146 individuals with 5 job title labels and 139 managerial relationships among them. Candidate pairs for entity resolution are limited to pairs of email addresses which are at most a distance three away in the communication network. Similarly, candidate managerial relationships are limited to pairs of nodes which share a communication edge.

6.5.1.3 Discourse Opinion Network

We next evaluate over a discourse co-occurence and opinion reinforcement networks using annotations by Somasundaran et al.[162]. The input graph consists of a co-occurence network where the nodes represent opinions in a discourse, edges represent that the opinions occur in the same portion of the discourse, and attributes are those defined in [162] which capture discourse and dialogue continuity, opinion lexicons, dialog information, and unigram features of the text. The output graph consists of opinion and object nodes where the objects are linked to the opinions that refer to it, opinions are linked by reinforcement edges indicating whether or not the two opinions reinforce each other, and the node labels indicate the polarity (positive, negative, or neutral) of each opinion. The full network consists of 4606

opinion nodes corresponding to 3920 objects with 22925 co-occurence edges and 1045 reinforcement edges. Candidate pairs for entity resolution and link prediction are limited to opinions which co-occur.

6.5.1.4 Synthetic Networks

To test the scalability of our approach, we developed a novel synthetic data generator that creates a noisy network with ambiguous references which need to be merged to entities, missing labels which need to be classified, and missing edges which need to be predicted. The graphs and the attributes created by this synthetic data generation are modeled after the motivating problem presented in Section 6.5.1.1 where the desired output graph is a clean citation network where nodes are papers, edges are citation edges between those papers, and attributes represent the topic of that paper. Intuitively, the generator works by creating a synthetic output graph which mimics the structure and attributes of real-world networks. The generator then creates an observed input graph from the citation network output graph by adding different types of noise common to these networks.

The synthetic data generator begins by creating the structure of the network (i.e., the set of nodes and edges of the output graph). A number of network generation models have been proposed which create networks which exhibit properties observed in many real-world networks. For our experiments, we implemented the widely used Forest Fire generation model [98] which models many of these properties including 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 we generate 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 first set is for use with collective classification and includes the labels and attributes based on those labels. We use the label generation method described in [138] (5 labels, with 20% of the graph initially labeled randomly) to create the "topic" label of the paper nodes 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 [19]. The second set of attributes is used for link prediction and consist of 20 attributes generated using the method described in [138]. We generate these attributes for link prediction with the intuition that nodes with similar attributes are likely to share an edge. The last set of attributes are used for entity resolution and represent attributes that imply, non-uniquely, the entity it refers to (e.g., first author names non-uniquely imply the paper as multiple papers may have the same first author name). To generate this attribute, we use the method described in [16]. The resulting network is our synthetic output graph (citation network).

We create an input graph from our output graph by creating a noisy version of the output graph. We add noise in four ways. First, we add a "reference" paper for a paper entity for each of its citation edges.[7] Each input graph node initially has

[7]In these experiments, we allow for a maximum of 10 references per paper.

the same attributes and labels as the corresponding node in the output graph. We also create edges similar to those of the output graph by ensuring all input graph nodes have an edge "equivalent" to the edges of the corresponding output graph nodes. Equivalent edges are created by adding at least one edge from an input graph node, corresponding to a node v_o^j of the output graph, to an input graph node, corresponding to an output graph node v_o^k, if v_o^j and v_o^k share an edge. Once we generate the reference nodes, we add noise to the attributes of those nodes by removing the "topic" labels of all the nodes and randomly permuting the values of a subset of the other attributes. Finally, we add edge noise to the graph by randomly removing a percent of the existing edges (50% of the current number of edges) and replacing them with edges between randomly selected pairs of nodes in the graph; the resulting edges are our "noisy" observed edges. The resulting noisy network is our synthetic input graph. We limit the candidate pairs for entity resolution using *blocking* [59] over the attributes created for entity resolution. Similarly, we limit candidate pairs for link prediction to pairs of nodes which share a noisy edge.

6.5.2 Evaluation Metrics

The evaluation for these networks is semi-supervised; we train on the observed part of the network and predict the remaining parts of the network. We varied the percentage of missing annotations over the reference labels for node labeling and the potential pairs for entity resolution and link prediction, evaluating at 25%, 50%, and 75% for Cora, Citeseer and Discourse and 20%, 30%, and 40% for the

much smaller ENRON network (denoted Low, Medium, and High, respectively). We construct five random samples for each setting (and each noise level for CORA and CITESEER) using stratified snowball sampling and the results are average over those five samples.

We apply C^3 and the variants of C^3 described in Section 6.4 on the four real-world datasets. To explore the impact of the inter and intra-dependencies, we also define variants of C^3 which use different subsets of the full set of features. In the first variant, LOCAL, we use only features based on the observed attributes of the nodes (i.e., words, email address string). This is equivalent to commonly used approaches for entity resolution, link prediction, and node labeling which make predictions independently and base predictions on only observed attributes [31, 36]. The second variant, INTRA, performs C^3 using only the relational features which capture the intra-dependencies of the predictions (dependencies on predictions of the same type). This variant allows us to study the relative impact of capturing the collective propagation among target variables of the same type. The INTRA variant is also representative of approaches which perform collective entity resolution, collective link prediction, and collective node labeling as separate, unrelated tasks [123, 160, 16]. For all variants of C^3, we the LibSVM [31] implementation of support vector machines and use the features defined in Table 6.1, 6.2, and 6.3. We run C^3 until convergence or oscillation and for variants which require running to a maximum number of iterations, C^3-EM, C^3-PS, and C^3-CI, we set $maxIter = 20$. For C^3-GS, we set $burnIn = 100$ and $maxIter = 500$.

We also compare against two popular approaches to performing inference in-

volving multiple tasks: PIPELINE and Markov Logic Networks (MLN) [140]. The PIPELINE approach performs tasks one at a time and in a fixed order. At each stage of the PIPELINE, we perform collective inference for a particular task only, using a similar learning and inference procedure to C^3 for comparability, but with the intra-relational features for that task and the inter-relational features from tasks which occurred earlier. Consequently, while the intra-dependencies are captured at each stage, the flow of information in PIPELINE does not allow earlier stages to use the predictions of later stages. This baseline is sensitive to ordering so we consider all possible orderings (six in total for the three tasks in graph identification). To differentiate the results for the different orderings, we use the initials of each component in the corresponding order of the PIPELINE (i.e., PIPELINE with ordering ER, LP, NL is shown as PIPELINE-ELN). For space, in some cases we present only the performance of the best possible ordering (denoted PIPELINE*). The other approach we compare to, MLN, is a state-of-the-art joint inference model proposed by Richardson and Domingos [140]. For this comparison, we use an open source implementation of MLN called Alchemy[92].[8] Because dependencies in MLN are represented using first order logic, we define first order logic formulae to mimic features defined in Tables 6.1–6.3. We explored various data representations and parameters for Alchemy, including the option to perform MAP or marginal inference, and present the results for the best performing combination in terms of both runtime and performance.

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

We evaluated entity resolution, link prediction, and node labeling performance using the average F1 performance over the predictions for the target variables $\mathbf{Y}$, defined in Section 6.4. We note that due to the blocking used by all approaches for entity resolution and link prediction, there are a large number of pairs which none of the approaches explicitly predict over and implicitly are always predicted as not co-referent or not linked by all the approaches. To highlight the performance differences between the different approaches, we compute the entity resolution and link prediction F1 performance over only the random variables we are explicitly predicting over (i.e., the blocked pairs). We present the average F1 performance on the three tasks as well as overall (representing the average over the entity resolution, link prediction, and node labeling F1 performances) over the multiple levels of noise and annotation. We first evaluate over the random variables without applying the procedures for enforcing consistency. To explore these procedures when constructing the output graph, we present an evaluation of the random variables after applying of these procedures in Section 6.5.3.4. Next, we look at the runtime performance characteristics of C^3. We look at the convergence characteristics of C^3 in our experiments by analyzing the number of times convergence or oscillations are encountered and the average number of iterations required in our experiments. We also look at the average learning, inference, and overall runtime in our experiments to see how C^3 runtime performance compares to our baselines. Next, we discuss the potential in parallelization for C^3 and show runtime performance for varying degrees of parallelization. Finally, we explore scalability by looking at C^3 performance on large synthetic datasets.

6.5.3 Prediction Quality

We begin this section by discussing the quality of predictions for all algorithms prior to applying the graph construction procedures defined in Section 6.4.4. 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 6.5.3.4.

6.5.3.1 Comparison to Other Approaches

We first present the overall F1 performance of all algorithms (representing the average over the entity resolution, link prediction, and node labeling F1 performances) to summarize the performance over the multiple levels of noise, annotation, and datasets in Table 6.4. The best performance for each set is indicated in bold. We also perform statistical significance tests, using a paired-t test with significance $> 95\%$, over the F1 values for all pairs of approaches. The results are summarized in Table 6.5 which indicates the number of times one approach, shown in each row, significantly outperforms another approach, shown in the columns. We also present Table 6.6 a representative subset of the individual F1 performance for entity resolution, link prediction, and node labeling.

Comparing the performances of all the approaches, we see that C^3 and C^3-EM are overall the best performing. Looking at Table 6.5, we see that C^3 and C^3-EM significantly outperforms all the other approaches in most cases while there are no

instances where either does significantly worse than the other algorithms. Next, looking at the approaches which exploit varying subsets of the dependencies within and among the different tasks, we see that LOCAL has the worst performance, followed by INTRA, and PIPELINE*. The trend in performance is directly correlated with the amount of intra- and inter-dependencies used by each approach; the more intra- and inter-dependencies are exploited, the better the overall performance.

Comparing the performance of the two joint models, we find that C^3 significantly outperforms MLN performance. We found that despite multiple attempts to optimize the MLN, the performance of MLN in our experiments remained relatively poor. One possibility for this is that there is insufficient training data for the MLN weight learning given the number of dependencies and features involved. We may also need to look at extensions of the basic MLN model [177, 79]. Understanding the causes of the poor MLN performance and addressing those issues is part of our future work. Our experience with MLN, however, highlights the challenge in efficiently and successfully modeling all the dependencies to jointly infer the tasks involved in graph identification, and the advantages of using a simpler approach based on collections of coupled classifiers.

6.5.3.2 Varying Dependencies

Relating the performance of the INTRA and LOCAL approaches, we see that making use of the intra-dependencies can, by itself, significantly improve performance. This is consistent with previous work which looked at these tasks in iso-

lation and shows the importance of exploiting these types of dependencies. Similarly, comparing the relative performance of the INTRA to the PIPELINE* and C^3 approaches, we find that further making use of the inter-dependence yields a comparable, if not larger, improvement in performance with little impact on overall runtime. While using the inter-dependencies in these tasks has not been widely studied, the results show the importance of these types of dependencies. Relative to the PIPELINE* approach, we found the best performing PIPELINE* to be a competitive baseline. We note, however, that when we look at the per task performance and overall performance for various PIPELINE orderings in Table 6.6, there is a significant variance in the performance. Successful application of the PIPELINE* requires the non-trivial task of identifying which ordering is optimal which we accomplish by evaluating all possible orderings. Our C^3 approach, on the other hand, requires no ordering yet still significantly outperforms even the best performing PIPELINE*.

6.5.3.3 Comparison of Variants

We now look at our variants we considered for learning and inference, C^3-EM, C^3-PS, C^3-CI, and C^3-GS. While the variants based on sampling, C^3-PS and C^3-GS occasionally show improvement over C^3, none of the improvements are significant. In most cases, these variants actually result in significantly worse performance than C^3. The same is also true for our cautious variant, C^3-CI, which show improvement in few cases, but generally significantly worse. While this maybe addressed by running more iterations, particularly for C^3-GS, the additional cost

of running more iterations and these initial results do not support using these two variants over the standard C^3 inference procedure. The overall F1 performance of C^3-EM, on the other hand, generally outperforms C^3 with 13 cases significantly better. The improvement, however, 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 classifiers at every iteration, discussed further in Section 6.5.4.1, should be taken into considering before applying this variant.

6.5.3.4 Applying Graph Construction Procedures

As discussed in Section 6.4.4, the predicted co-references, links, and labels maybe inconsistent relative to some set of task and domain specific hard constraints. Entity resolution in some cases, for example, may require transitivity on the co-references (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 the predictions of all approaches evaluated in our experiments. In these situations, we can define a procedure to resolve the inconsistencies prior to generating the output graph. The procedure we need to apply for

Table 6.4: Overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performance) on the output of the different models. 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	0.745	0.900	0.854	0.798	0.516	0.648
	PIPELINE*	0.871	0.834	0.793	0.939	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**	0.899	0.550	**0.729**
	C^3-EM	0.882	**0.855**	**0.825**	**0.951**	**0.928**	**0.904**	0.544	**0.729**
	C^3-PS	0.880	0.851	0.817	0.947	0.924	0.890	0.539	0.679
	C^3-CI	**0.883**	0.853	0.817	0.949	0.926	0.897	0.552	0.693
	C^3-GS	0.881	0.852	0.817	0.949	**0.928**	0.897	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	0.768	0.921	0.888	0.849	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	0.934	0.900	0.862	**0.515**	0.658
	C^3-EM	**0.864**	**0.833**	**0.797**	**0.935**	**0.908**	**0.875**	**0.515**	**0.664**
	C^3-PS	0.860	0.827	0.782	0.930	0.896	0.855	0.497	0.478
	C^3-CI	0.860	0.826	0.781	0.932	0.899	0.861	0.512	0.621
	C^3-GS	0.858	0.823	0.777	0.931	0.898	0.854	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	0.770	0.708	0.868	0.816	0.741	0.448	0.351
	PIPELINE*	0.831	0.795	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	0.835	0.801	0.750	0.902	0.869	0.819	0.479	**0.483**
	C^3-EM	**0.844**	**0.813**	**0.770**	**0.910**	**0.883**	**0.841**	**0.493**	0.482
	C^3-PS	0.836	0.800	0.748	0.902	0.868	0.818	0.436	0.313
	C^3-CI	0.834	0.799	0.749	0.899	0.868	0.814	0.480	0.437
	C^3-GS	0.833	0.797	0.744	0.901	0.864	0.809	0.322	0.176

Table 6.5: Each row indicates the number of times the approach, in each row, significantly outperforms the average overall performance of the approaches in each column, over all three levels of noise and three levels of sampling (a maximum of 9 pairwise comparisons for CORA and CITESEER and a maximum of 3 for ENRON 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	1	0	6
C^3-*EM*	9	9	9	9	7	–	7	7	8
C^3-*PS*	9	9	9	9	0	0	–	1	6
C^3-*CI*	9	9	8	9	0	0	0	–	2
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	6	3	5
C^3-*EM*	9	9	9	9	5	–	8	6	6
C^3-*PS*	9	9	8	9	0	0	–	0	2
C^3-*CI*	9	9	9	9	0	0	2	–	3
C^3-*GS*	9	9	7	9	0	0	4	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	1	0
C^3-*EM*	3	2	0	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	1	3
MLN	0	0	0	–	0	0	0	0	0
C^3	3	3	3	3	–	0	3	3	3
C^3-*EM*	3	3	3	3	0	–	3	3	3
C^3-*PS*	3	1	0	3	0	0	–	0	3
C^3-*CI*	3	3	2	3	0	0	2	–	2
C^3-*GS*	1	0	0	3	0	0	0	0	–

Table 6.6: Average F1 performance over the entity resolution, link prediction, and node labeling output on the different models. We also compute the overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performance) on the different models. Bold indicates the highest value in a given column.

	ER	LP	NL	Average	ER	LP	NL	Average
	CORA				CITESEER			
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	0.841	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	0.809	0.892	0.916	0.828	0.879
PIPELINE-LEN	0.913	0.860	0.652	0.808	0.905	0.842	0.858	0.868
PIPELINE-LNE	0.916	0.860	0.621	0.799	0.908	0.842	0.828	0.859
PIPELINE-NEL	0.904	0.907	0.586	0.799	0.896	0.917	0.787	0.867
PIPELINE-NLE	0.916	0.890	0.586	0.797	0.912	0.898	0.787	0.866
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	0.919	0.918	0.862	0.900
C^3-EM	0.919	**0.913**	**0.665**	**0.833**	0.914	**0.935**	**0.874**	**0.908**
C^3-PS	0.917	0.911	0.652	0.827	0.907	0.920	0.862	0.896
C^3-CI	**0.922**	0.910	0.646	0.826	**0.921**	0.919	0.857	0.899
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	0.613	0.604
PIPELINE-ENL	0.891	**0.124**	0.513	0.509	0.552	0.655	0.602	0.603
PIPELINE-LEN	0.888	0.100	0.528	0.505	0.663	0.530	0.612	0.602
PIPELINE-LNE	0.894	0.100	0.404	0.466	0.663	0.530	0.597	0.597
PIPELINE-NEL	0.894	**0.124**	0.405	0.474	0.551	0.651	0.554	0.585
PIPELINE-NLE	0.894	**0.124**	0.405	0.474	0.665	0.533	0.554	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	0.618	0.658
C^3-EM	**0.896**	0.100	**0.548**	**0.515**	0.679	**0.690**	**0.623**	**0.664**
C^3-PS	0.826	**0.124**	0.541	0.497	0.431	0.406	0.596	0.478
C^3-CI	0.890	**0.124**	0.522	0.512	0.630	0.627	0.607	0.621
C^3-GS	0.527	**0.124**	0.499	0.383	0.201	0.194	0.546	0.314

resolving consistencies can vary depending on the data. For our experiments, we use the following procedure to resolve the inconsistencies for CORA, CITESEER, and ENRON: apply transitive closure over co-referent pairs, add edges between entities whose references have an edge, and taking the mode label of the labels over its references. For DISCOURSE, we define a separate procedure for its domain specific constraints: apply transitive closure over co-referent pairs, remove reinforcing edges between pairs not co-referent, and taking the mode label over the labels of opinions transitively co-referent and reinforcing.

We explore the impact of applying these procedures on the output of *all* algorithms prior to evaluation. For C^3, we also explore an alternative way these procedures are applied. Instead of just applying these procedures only after all the iterations in C^3 are completed, we can also apply these procedures at the end of every iteration. This ensures that at the end of each iteration in C^3, the predicted values are always consistent. We denote this variants C^3-CO. The results are provided in Tables 6.7 – 6.9.

Compared to the performance when the procedures are not applied, we generally see improvement for all the algorithms. This is especially true for LOCAL and INTRA which see improvements up to 80% and 61%, respectively. While LOCAL and INTRA do not capture either the intra- and inter-dependencies, the application of these procedures partially does. While we generally find improvement for all approaches when applying these procedures, we still find that the trends from Tables 6.4 – 6.6 remain. Approaches which explicitly take these dependencies into account still significantly outperform those which do not. Furthermore, C^3 and C^3-EM

are still the overall best performing for all amounts of noise, annotated data, and datasets. In comparison to C^3-CO variant, we found that applying the procedure once to the output of C^3, rather than per iteration, performed significantly better. The significant improvement of C^3-CO over C^3 without these procedures, however, does suggest that there maybe benefit in having more explicit support for ensuring consistency. We are currently exploring alternative ways to ensure consistency within the iterations of C^3.

6.5.4 Runtime Performance

6.5.4.1 Learning and Inference Time

In Table 6.10, we list the average learning, inference, and overall runtimes for the CORA experiments. These experiments were run on comparable servers with dual Intel Xeon 2.66Ghz processors and 48GB of memory. All implementations are in Java except for Alchemy which is in C++. Ordering the results based on the amount of intra- and inter-dependencies they capture, we see that there is a runtime cost associated with capturing more and more dependencies. Relative to the significant improvement in predictive performance, however, we see that the additional runtime required by C^3 compares favorably to the fastest algorithm evaluated while being an order of magnitude faster than the slowest algorithm. Comparing the two algorithms with the best predictive performance, there is a notable increase in learning time over C^3 when applying C^3-EM due to the need to relearn SVMs at every iteration. While C^3-EM shows significant improvement in prediction quality

Table 6.7: Overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performance) *after applying hard constraints* on the output of the different models . 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	0.863	0.813	0.760	0.932	0.902	0.861	0.743	0.637
	PIPELINE*	0.877	0.848	0.799	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	0.885	0.854	**0.818**	0.952	0.930	**0.905**	0.762	0.738
	C^3-*EM*	**0.886**	**0.857**	0.807	**0.954**	**0.931**	**0.905**	0.756	**0.744**
	C^3-*PS*	0.881	0.854	0.810	0.947	0.922	0.892	0.727	0.683
	C^3-*CI*	0.884	0.852	0.807	0.952	0.929	0.902	**0.764**	0.716
	C^3-*GS*	0.885	**0.857**	0.808	0.953	**0.931**	0.899	0.562	0.623
	C^3-*CO*	0.884	0.853	0.802	0.951	0.929	0.903	0.763	0.720
Medium % Unknown	LOCAL	0.780	0.695	0.622	0.853	0.794	0.752	0.655	0.429
	INTRA	0.844	0.787	0.710	0.913	0.876	0.830	0.701	0.521
	PIPELINE*	0.853	0.808	0.744	0.926	0.894	0.854	0.722	0.605
	MLN	0.495	0.602	0.565	0.488	0.463	0.439	0.306	0.334
	C^3	0.858	0.817	0.756	0.933	0.906	0.866	0.724	0.657
	C^3-*EM*	**0.862**	**0.819**	**0.765**	**0.936**	**0.909**	**0.875**	**0.731**	**0.668**
	C^3-*PS*	0.852	0.814	0.750	0.926	0.896	0.846	0.679	0.469
	C^3-*CI*	0.858	0.815	0.749	0.931	0.903	0.864	0.724	0.637
	C^3-*GS*	0.851	0.810	0.723	0.928	0.898	0.844	0.483	0.278
	C^3-*CO*	0.859	0.815	0.748	0.933	0.904	0.864	0.726	0.648
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	0.841	0.777	0.642	0.326
	PIPELINE*	0.827	0.776	0.698	0.896	0.859	0.806	0.660	0.418
	MLN	0.363	0.348	0.342	0.310	0.319	0.316	0.211	0.190
	C^3	0.829	0.782	0.708	0.901	0.866	0.814	0.659	**0.484**
	C^3-*EM*	**0.839**	**0.792**	**0.724**	**0.910**	**0.877**	**0.832**	**0.664**	0.481
	C^3-*PS*	0.829	0.777	0.703	0.894	0.858	0.799	0.596	0.295
	C^3-*CI*	0.827	0.776	0.704	0.897	0.864	0.807	0.662	0.454
	C^3-*GS*	0.820	0.767	0.668	0.891	0.854	0.781	0.358	0.119
	C^3-*CO*	0.828	0.780	0.704	0.897	0.861	0.803	0.659	0.469

Table 6.8: Each row indicates the number of times the approach, in each row, significantly outperforms the average overall performance *after applying hard constraints* of the approaches in each column, over all three levels of noise and three levels of sampling (a maximum of 9 pairwise comparisons for CORA and CITESEER and a maximum of 3 for ENRON and DISCOURSE).

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

Table 6.9: Average F1 performance *after applying hard constraints* over the entity resolution, link prediction, and node labeling output of the different models on all datasets for medium percentage unknown and medium noise for CORA and CITESEER. We also compute the overall F1 performance (representing the average over the entity resolution, link prediction, and node labeling F1 performance) on the different models. Bold indicates the highest value in a given column.

	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	0.624	0.787	0.872	0.919	0.836	0.876
PIPELINE-ELN	0.825	0.922	0.665	0.804	0.872	0.940	0.868	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	0.887	0.919	0.867	0.891
PIPELINE-LNE	0.852	0.914	0.657	0.808	0.896	0.919	0.866	0.894
PIPELINE-NEL	0.827	0.932	0.624	0.794	0.882	0.943	0.836	0.887
PIPELINE-NLE	**0.853**	0.916	0.625	0.798	0.896	0.932	0.837	0.888
MLN	0.568	0.854	0.384	0.602	0.174	0.742	0.474	0.463
C^3	**0.853**	**0.934**	0.665	0.817	**0.902**	**0.945**	0.870	0.906
C^3-EM	**0.853**	0.925	**0.679**	**0.819**	**0.902**	0.943	**0.881**	**0.909**
C^3-PS	0.845	**0.934**	0.664	0.814	0.874	0.944	0.869	0.896
C^3-CI	0.849	0.933	0.663	0.815	0.898	**0.945**	0.867	0.903
C^3-GS	0.832	0.933	0.667	0.810	0.876	0.944	0.875	0.898
C^3-CO	0.848	0.933	0.663	0.815	0.900	**0.945**	0.866	0.904
	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	0.622	0.721	0.556	0.575	0.605	0.579
PIPELINE-ENL	0.909	**0.639**	0.617	0.722	0.556	0.584	0.606	0.582
PIPELINE-LEN	0.906	0.631	0.622	0.720	0.691	0.519	0.605	0.605
PIPELINE-LNE	0.911	0.631	0.563	0.702	0.692	0.520	0.590	0.600
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	0.693	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**	0.622	0.724	0.699	0.665	0.608	0.657
C^3-EM	**0.914**	0.633	**0.646**	**0.731**	**0.710**	**0.682**	**0.611**	**0.668**
C^3-PS	0.807	0.593	0.638	0.679	0.430	0.412	0.565	0.469
C^3-CI	0.911	**0.639**	0.622	0.724	0.694	0.614	0.602	0.637
C^3-GS	0.510	0.419	0.520	0.483	0.192	0.196	0.445	0.278
C^3-CO	0.911	**0.639**	0.628	0.726	0.693	0.660	0.592	0.648

Table 6.10: Average learning, inference, and overall runtimes (in minutes) for each model over the experiments on CORA.

	Learning Time	Inference Time	Overall Time
LOCAL	1.8	0.5	2.3
INTRA	4.6	8.4	13.0
PIPELINE-ELN	4.7	7.2	11.9
PIPELINE-ENL	4.7	7.1	11.8
PIPELINE-LEN	4.9	7.2	12.1
PIPELINE-LNE	5.3	8.2	13.5
PIPELINE-NEL	4.9	8.1	13.1
PIPELINE-NLE	5.1	8.3	13.5
MLN	761.7	183.6	945.3
C^3	5.2	21.5	26.7
C^3-EM	47.9	23.7	71.6
C^3-PS	5.2	25.0	30.1
C^3-CI	5.2	10.6	15.8
C^3-GS	5.2	728.2	733.4

for some of the datasets, its inconsistency and the much greater learning time of C^3-EM suggest that the standard semi-supervised version should be considered first for most situations.

6.5.4.2 Convergence Results

An important characteristic affecting C^3 runtime is the number of iterations it requires during inference. In our experiments, we allowed C^3 to run until the predictions converge to a single set of assignments or until we detect an oscillation has occurred (i.e., the predictions exactly match those of an earlier iteration). We list the number of times each stopping criterion was encountered and the average number of iterations required for all datasets in Table 6.11. First, *all* the experiments we conducted with C^3 for varying levels of noise, annotations, and datasets either converged or began oscillating. Of the two stopping criterion, stopping due to the detection of an oscillation was more common with convergence to a single

Table 6.11: Number of times convergence or oscillation was reached in the experiments using C^3 for all datasets (a maximum entry of 45 for CORA and CITESEER and 15 for ENRON and DISCOURSE). We also present the average number of iterations performed prior to reaching convergence or oscillation. Note that all our C^3 experiments either converged or reached an oscillation point.

	# Converge	# Oscillate	Avg. # of Iterations
CITESEER	0	45	27.8
CORA	0	45	16.4
ENRON	14	1	4.2
DISCOURSE	0	15	9.9

value detected only for ENRON. Looking at the number of iterations required before reaching other stopping criteria, we find that C^3 typically requires very few iterations. On average, C^3 took as few as 4.2 iterations to run with the highest average number of iterations only 27.8. We are currently exploring theoretical bounds for the number of iterations required for encountering either stopping criterion. All current empirical evidence, however, indicate that a fast convergence or oscillation is typical of algorithms based on pseudolikelihood [15, 101, 123].

6.5.4.3 Parallelization Results

A notable characteristic of the C^3 learning and inference algorithms is the potential for significant portion of the algorithm to be run in parallel. For C^3 semi-supervised weight learning shown in Algorithm 2, the weights of the bootstrap classifiers for each of the three tasks can be learned in parallel. Next, we can apply these bootstrap classifiers in parallel to initialize the values of the target variables. The classifiers used iteratively can then be similarly learned in parallel. For C^3 inference we 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 in parallel without affecting the results.

We illustrate the parallelizability of C^3 by running a portion of the C^3 experiments on the Cora dataset with an implementation using Java threads. We present the learning, inference, and overall run times for C^3 varying the numbers of available threads in Figure 6.2. While the predicted values from these experiments are identical to those when run as a single process, we note that there is a substantial improvement in the overall runtime as we increase the number of threads. Increasing the number of threads to 2, for example leads to an immediate 1.8 time improvement in runtime. Comparing the improvements between the learning and inference times, we found that most of the improvement is from the faster inference time. While learning does benefit from more threads for bootstrapping, because our SVM learning algorithm does not support parallelization runtime improvement is limited to the number of local or relational classifiers we are learning. We note that there are approaches for parallelizing weight learning within the classifiers themselves [32]. We do not explore these algorithms in this paper but plan to explore them in future work. With inference, we find that can consistently see substantial improvement as we increase the number of threads. Unlike learning, where the number of classifiers we can learn in parallel is less than the number of available threads, the number of target variables we can infer in parallel is typically much larger than the number of available threads.

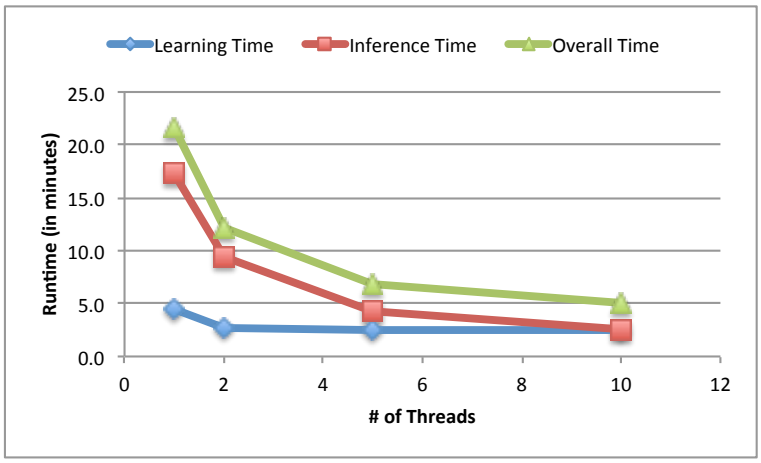


Figure 6.2: Learning, Inference, and Overall Time on Cora dataset for C^3 varying the numbers of available threads.

6.5.4.4 Scalability Results

We now evaluate the ability of C^3 to apply to large datasets. For these experiments, we generated increasingly larger synthetic networks (as described in Section 6.5.1.4) to study the characteristics of C^3 runtime performance. We perform experiments with C^3 using the same features and experimental settings for medium amounts of annotation as those used for the experiments on CORA and CITESEER. We present the learning, inference, and overall runtime of C^3 in input networks ranging in size from 2209 nodes and 15636 edges to networks up to 45522 nodes and 353228 edges in Figure 6.3. Although our implementation is not specifically designed for large networks, we can demonstrate that C^3 is able to scale well to such networks. Looking closely at the individual learning and inference times, however, an important thing to address is the increasing amount of runtime required for training. The ability of C^3 learning to scale is tied directly to the ability of its

underlying classifiers to scale. In our experiments, we use the LibSVM implementation of support vector machines [31] whose learning time is known to be quadratic to the number of training instances. This significantly limits the ability of our current implementation to learn from larger datasets. Fortunately, however, there has been significant progress in the theory and algorithms behind support vector machines we can directly apply. Beyond the algorithms for parallel learning of SVM discussed in Section 6.5.4.3, there are increasingly more efficient learning algorithms for SVM specifically for large data [27, 29, 151]. We plan on using a variety of these techniques in a system designed specifically for very large networks in future work.

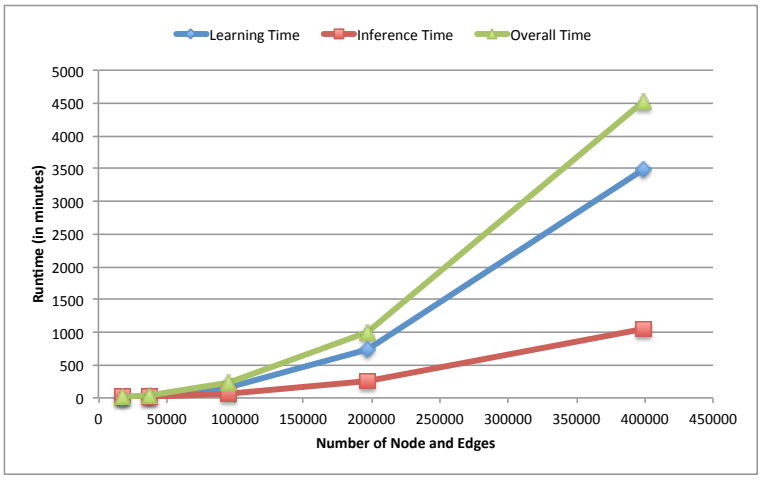


Figure 6.3: Learning, Inference, and Overall Time on synthetic dataset for C^3 as the number of nodes and edges in the input graph increase.

6.6 Conclusion

Graph identification is an important emerging problem. As more observational data describing networks becomes available, the need to properly map from

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

Chapter 7

GAIA

A fundamental challenge in exploring network data is the lack of a software system that allows for the easy application of various data mining algorithms over the diverse types of network data. This not only makes many previously proposed algorithms inaccessible to analysts interested in using these approaches, it also hinders progress in algorithm research by making it difficult to compare to and study previous approaches. We develop the Graph Alignment, Identification, and Analysis (GAIA) software library (http://linqs.cs.umd.edu/gaia) to address this challenge. In this chapter, we present the GAIA software library. We discuss the motivation behind its creation, the goals behind its design and implementation, and provide an overview of the supported tasks and utilities.

7.1 Introduction

Data from a variety of domains can naturally be represented as networks. In previous chapters, we showed a variety of problems where improving the quality of network data can have a significant impact. We have also shown developing algorithms which explicitly exploit both the attributes and relationships in networks, can significantly improve predictive performance. While there has been significant progress in studying network data and the algorithms for network data, a funda-

mental challenge still remains hindering the adoption and continued progress of this research: the lack of a software system that allows for easy representation, study, and manipulation of the diversity of real-world networks.

The current practice when working with and developing algorithms for network data involve ad-hoc implementations of code to load, represent, and manipulate networks. Not only does this result in wasted effort in the form of redundant code, the resulting code is often very specific in the types of networks, I/O formats, and tasks it can handle. Consequently, it is difficult to apply or compare to proposed approaches on new datasets. Furthermore, because implementations of algorithms designed for different tasks are spread across many incompatible systems, it is difficult to analyze and exploit the interplay between different tasks. In Chapter 6, we showed how important understanding the interplay between entity resolution, link prediction, and collective classification is to the problem of graph identification. Relying purely on implementations specific to only one of these tasks would have made it extremely difficult to explore this interplay for our problem.

The interactions between tasks go beyond joint inference tasks like graph identification however. Many tasks on network data rely on the output of algorithms for other tasks as features to function. Work in active learning in network data, for example, often relies on the probability distribution from collective classification models to decide which annotations to acquire [144, 150]. Similarly, the outputs of clustering models have been used as features in active learning [21, 102], classification [188], and link prediction models [3]. A system which supports multiple tasks allows people to apply a wider variety of algorithms to supplement their own

algorithms.

Many different tasks also have a common underlying approach. Entity resolution and link prediction can both be solved using a threshold approach based on various similarity measures [59, 99]. Similarly, relational clustering can be used to directly label nodes [68], predict edges [187], and predict co-reference [16]. A system which supports multiple tasks would not only be more efficient due to code reuse, it would also help us better understand the assumptions and relationships of different tasks.

In addition to using the output of other tasks during inference, the use of network task tools can also essential in understanding and evaluating the various algorithms. By supporting algorithms for network visualization and efficient computation of statistics like betweenness centrality, we can better understand the characteristic of the network we are evaluating. This provides context for understanding why algorithms work well with some networks and not on others. Output of other tasks can also be used in evaluating the approaches of others. For example, acquiring real-world networks for evaluation is often difficult or impractical. Consequently, researchers often use synthetically generated network data to evaluate their algorithms. By having a system which supports network generation algorithms [98], people can create more realistic networks for their evaluations.

As part of this dissertation, we developed the Graph Alignment, Identification, and Analysis (GAIA) software library to address this challenge. First, we survey of other available software packages for use on network data and describing their strengths and limitations in Section 7.2. In Section 7.3, we describe the GAIA

software, discussing the goals in its design and implementation. We also provide an overview of the available implementations and capabilities of GAIA. We discuss future work in GAIA and conclude in Section 7.4.

7.2 Related Work

Most previous machine learning and data mining software focus on working with local attributes of non-relational, independent data. For classification and ranking, there are libraries like LibSVM [31], SVM^{light} [82], PyBrain [147], Weka [69], and Apache Mahout [9]. More recently there has been software developed specifically for network data. While most implementations are single algorithm implementations used to supplement publications, there have been significant strides in more general packages. Junto [167] and Netkit [104] were developed for implementations of multiple label propagation based collective classification algorithms. Jung [171] and SNAP [172] include algorithms for network generation and for efficient computations of various network statistics. Pajek [13], Prefuse [71], and NodeXL [22] have support for various network visualization and layout algorithms. Of note also are implementations of complex statistical relational learning frameworks including Alchemy [92], Proximity [91], Factorie [111], and PSL[24]. While these software have been invaluable, there are significant limitations in using these software for certain networks and tasks. Many packages focus on specific types of networks. For example, Jung and Pajek are mainly focused on the network structure with minimal support for attributes. Netkit and Junto only support networks

with a single categorical attribute and no hypergraphs. These software also tend to focus on specific tasks (classification, visualization, or clustering only) or specific approaches (Alchemy, Proximity, and PSL only have implementations for Markov Logic Networks, Relational Dependency Networks, and Probabilistic Soft Logic, respectively) which make it difficult to compare approaches and study and exploit the interplay between various tasks.

7.3 GAIA Software

To address the limitations of previous software, GAIA is designed and developed with four main goals. First, GAIA supports a wide range of machine learning and analysis tasks on networks. Also, GAIA supports a variety of network types, attributes, and operations. Third, GAIA uses modularity and abstraction to allow for different parts of GAIA to be developed independently and to allow users to easily use these parts through simple, general interfaces. Finally, GAIA is designed to be easy to use for development with the goal of encouraging both adoption and further development. In the rest of this section, we discuss how our implementation of GAIA accomplishes these goals. At the same time, we will provide an overview of the capabilities of GAIA.

7.3.1 Algorithmic and Analysis Support

In the early design of GAIA, we aimed to identify the variety of tasks that would be applied on network data and ensuring a wide support for them. We did

an extensive exploration of network tasks people have explored in the literature. The survey paper by Getoor and Diehl [63] provide a good survey of these tasks. We present the lists of tasks they identified below updated with other tasks we identified. We indicate all the tasks which the current implementation of GAIA has algorithms for with a star (*).

1. Object-Related Tasks
    (a) Link-Based Object Ranking
    (b) Link-Based Object Classification*
    (c) Object Clustering/Group Detection/Relational Clustering*
    (d) Object Identification/Entity Resolution*
    (e) Active Learning and Inference*
2. Link-Related Tasks
    (a) Link Ranking
    (b) Link Classification*
    (c) Link Prediction*
    (d) Active Surveying*
3. Graph-Related Tasks
    (a) Graph Identification*
    (b) Subgraph Discovery

(c) Graph Classification*

(d) Generative Models for Graphs*

(e) Graph Representation and Storage*

(f) Graph Sampling*

(g) Graph Analysis and Visualization*

While algorithms for all these tasks are not currently available, GAIA was designed to provide a base infrastructure to support all of these. Implementations for all tasks are part of future work discussed in Section 7.4.

7.3.2 Graph Support

The underlying graph representation of the data is one of the most important part of any software system for networks. The capabilities of the graph representation define the data to which the software system can be applied on and what tasks can be performed. In order to be able to perform all the tasks listed in the previous section, the GAIA graph was designed to be both general and extensible. Unlike many software that can only support binary directed networks, GAIA supports hypergraphs with both directed and undirected edges. GAIA can also handle multiple edge and node types in the same graph to support not only bipartite graphs, but even more complex network structures. The GAIA graph also has extensive explicit support for attributes. The nodes, edges, and graphs can have strings, numeric, single category, and multi-categorical attributes. Beyond these general types, GAIA allows for easy addition of additional feature types and supports sparse data.

In addition to supporting a very large class of network data, the GAIA graph representation was also designed with an extensive set of operators to simplify accessing, transversing, and modifying the graph. For access, nodes, edges, and graphs can easily be accessed individually using their unique ID or as a group by their type. The graph can traversed by accessing the incident or adjacent nodes or edges, as well as by using various "neighbor" functions to identify a more complex set of nodes and edges. Similarly, beyond basic support for adding and removing nodes and edges, GAIA has support for more complex operations like adding and modifying features, removing nodes and edges by structural characteristics, and adding or removing nodes to hyperedges.

A general graph interface is of no use, however, if it is difficult to load new network data into GAIA. We developed a simple tab-delimited data format which supports all the types of graphs and attributes the GAIA graph supports. We collected a number of real-world datasets in this format for use in research, as well as to provide examples of this format. In addition, GAIA also supports for many common network data formats including the adjacency matrix representation, Pajek format [13], Dotty format [53], and GraphML [23].

7.3.3 Modular Architecture with Abstraction

Two fundamental principles in software engineering are modular design and abstraction. Modular design allows for the development of complex systems by breaking the larger problem into smaller modules. This in turn improves code reuse

since the modules can typically be used in other parts of the software. Whenever possible, components of GAIA which are common to multiple tasks are implemented as utilities. GAIA has an extensive list of utility methods and data structures. These include efficient implementations of utilities like for counting items, allowing for keyed access to Java collections (i.e., List, Sets), crawling websites, accessing databases, and working with probability distributions. We also have implementations of utilities for specific tasks. Entity resolution, for example, has utilities for enforcing transitive closure and converting to and from different representations of the entity resolution predictions. GAIA also has utilities for different ways of blocking [59, 110] commonly used in entity resolution, link prediction, and clustering.

The principle of abstraction aims to separate the desired behavior of software components from the implementation. A major benefit of ensuring proper abstraction in a system like GAIA is that we improve the usability of the system as a whole. Defining general interfaces which capture the core functionality needed for specific tasks allows users to easily understand the input and output of different tasks. Abstraction also allows the users to easily try different underlying implementations of the same abstraction to select the one that is best for their particular tasks. An example of this is the implementation of the GAIA graph object discussed earlier. The GAIA graph object is implemented and generally accessed using a Java interface. Users of GAIA develop using this interface and can either choose from one of the implementations of the graph interface in GAIA or create one that best suites their task. GAIA currently has two implementations of a graph. The first is an efficient in-memory graph object recommended for most use. The second is an im-

plementation which uses a Derby database to store and manage the graph for use in large datasets. Another example where this is highlighted in GAIA is for the tasks of collective classification. GAIA has a general interface for collective classification models supporting supervised, semi-supervised, and unsupervised models. Because of this interface, the included tool for running collective classification experiments can easily be set to use the largest variety of collective classification packages in a single software system. Aside from dozens of classifiers based on local attributes such as logistic regression [96], decision tree[136], and support vector machines [31], GAIA has support for wvRN [103], ICA [123, 100], stacked learning [93], RDN [124], and label propagation algorithms [167].

Note that an additional benefit of having a modular design with proper abstraction is that it is easy to add and modify implementations for different parts of the code. This includes the ability to add and use implementations from third party libraries. Whenever available, we tried to make it easy to use implementations from other software packages. For example, we provide a wrapper for the popular SimMetrics [175] string similarity library to allow for easy use of its string similarity measures for various tasks in GAIA. Similarly, we have wrappers for the classifiers in LibSVM [31] and Weka [69] allowing for ease of use of their classifiers. We even provide methods for converting to and from graph objects from third party libraries ranging from a simple adjacency matrix to the Jung [171] and Junto [167] graph objects.

7.3.4 Accessibility and Development

The final goal in the GAIA design and development is to ensure that the code can be easily used by new users, as well as accessible to future developers. An active community of users and developers are essential in making sure that GAIA continues evolving and is able to make an impact in research. Toward this goal, we developed GAIA using the Java programming language due to the ease of use and its availability on various platforms. Java is also the preferred language of many machine learning systems and has extensive support for using software written in other programming languages. Next, we made extensive use of Javadoc, Java's API documentation system, ensuring that all code and packages are well documented and can easily be used and understood. For implementations of specific algorithms, we include pointers to the paper which proposed that algorithm.

Another way we used to make GAIA more accessible is to provide example code and frameworks to run common tasks GAIA might be used for. This includes support for easily converting between different data formats, computing various statistics over the properties of a network, generating synthetic data, and performing various types of collective classification experiments. We also provide support for common tasks when performing experiments with GAIA including utilities for acquiring and processing data from outside sources, as well as computing various evaluation statistics like F-measure, AUC, and confusion matrix. Aside from the java code, we also provide bash scripts to simplify running and documenting experiments with GAIA, as well as releasing a number of network datasets ready for use

in GAIA.

GAIA is released open-source under the Apache License, Version 2.0. The code, documentation, scripts, and datasets are available via http://linqs.cs.umd.edu/gaia and as a project in the popular open software service GitHub [1].

7.4 Conclusions and Future Work

In this chapter, we discussed the need for a general software system for representing, studying, and manipulating network data. We discuss the benefits of such a system and discuss how current software packages are not able to realize those benefits. We then present the GAIA software library and tool and present the goals used in its design and implementation. While doing so, we provide an overview of the current capabilities of GAIA. GAIA, however, is still evolving and still has much unreached potential. To that end, we are working with various groups to contribute to GAIA. Implementations of many internally developed algorithms are already available in GAIA and the goal is to continue doing so for many more algorithms. We are also continuing to improve the documentation for GAIA, raising awareness of its availability and capabilities, and providing training to get new users and developers of GAIA.

[1] Available via https://github.com/linqs/GAIA

Chapter 8

Conclusion and Future Work

In this dissertation, we discussed the deficiencies common in network data and presented our work in addressing those deficiencies in the problems of entity resolution, link prediction, collective classification, and graph identification. Here, we summarize our contributions and discuss future directions of our research.

8.1 Summary of Contributions

Network data is ubiquitous and its analysis is essential in many domains. The problem with most gathered network data is that it is too noisy and incomplete to use directly. In this dissertation, we identified the most common deficiencies in these networks and describe entity resolution, link prediction, and collective classification can be used to address the problems. We described our work for each of these problems in turn, as well as our work in the task of graph identification which requires the joint application of these three problems.

For entity resolution, we studied the problem of resolving name references in email communication networks. In this work, we are given name mentions, email communications around these mentions, and a set of entities. The task is to infer, for a particular name mention, a ranking over the entities based on the likelihood each entity is the target of that mention. We developed novel unsupervised approaches

using summary statistics on the amount of traffic sent and received to each of the candidate entities relative to the time of the mention. Evaluating on manually annotated name references on a corporate dataset, we highlighted the importance of long term communication patterns to resolving name references and show that simple traffic models can achieve impressive name reference resolution performance.

For link prediction, we presented our work in the link prediction of social relationships. In this work, we are interested in a collaborative process between a human and machine where the goal of our algorithm is to focus the analyst's attention to (1) the most relevant communication relationship representing the relationship of interest and (2) the relevant message traffic that supports that relationship identification. We developed a novel supervised ranking approach for this task which minimize the number of rank violations in the ranking of the relationships. We empirically showed its utility on identifying subordinate-manager relationships given email communications from a major corporation. Using traffic and content-based features, the ranking method is able to cue the analyst to relevant communication relationships. We also developed a message ranker using content-based features and demonstrated its ability to highlight compelling evidence within the message traffic substantiating the social relationship.

For collective classification, we developed a novel method for active surveying for query-driven collective classification. While traditional collective classification aims to infer all missing labels, this work considers the setting in which one is primarily interested in labeling a particular subset of nodes (the query set). Given that acquiring the labeled instances and network structure used by semi-supervised

models are expensive, we applied active surveying to identify the nodes for which we can acquire labels and ego networks in order to maximize the classification performance on the query set. Leveraging common assumptions on feature and structural smoothness, we proposed a novel adaptive active surveying algorithm, ASQ2C, and empirically show its superior performance over standard active learning approaches on four real-world datasets.

These problems typically do not occur in isolation, however, and often all need to be addressed at the same time. We discussed how inherently inter-dependent these problems are and motivate the need to apply them jointly in a general problem we define as *graph identification*. We presented a novel approach to graph identification using a collection of Coupled Collective Classifiers, C^3, which in addition to capturing the variety of local features typically used for each task, can also capture the intra- and inter-dependency required. We discussed variants of C^3 using different learning and inference paradigms and show the superior performance of C^3, in terms of both prediction quality and runtime performance, over various previous approaches.

Finally, we presented the GAIA open-source software library and tool to fill the infrastructure need in algorithmic and analytical research on network data. We discussed the limitations of current systems in fulfilling this need and presented the goals and implementations in GAIA that make it easier not only to study specific tasks on network data, but also to study how these tasks are related. GAIA is available for download from http://linqs.cs.umd.edu/gaia.

8.2 Future Directions

In this dissertation, we looked at the tasks of entity resolution, link prediction, collective classification, and graph identification on network data. We developed algorithms to address these problems in various domains. There are multiple future research directions for all of our proposed solutions. We described these at the end of the corresponding chapters. In this section, we focus on future research directions beyond those discussed in earlier chapters.

First, our work in graph identification and GAIA made it clear that there are many research opportunities in studying the commonalities and interactions of various tasks. Not only has study in the interactions been limited, we have shown, through the example of graph identification, that considering these tasks together can yield significant improvement in performance. A specific interaction we would like to pursue is between the tasks of active learning and graph identification. For a complex task like graph identification, the amount and type of annotations available is much more complex than previous work in active learning methods have considered. Active learning for graph identification would need to balance between acquiring annotations to optimize the joint learning and inference of the three problems within graph identification. These annotations are also likely to require more complex cost structures causing many previous approaches to be inapplicable.

Another aspect we would like to pursue is the application of graph identification for large scale data. We showed that C^3 is scalable and can be parallelized using threads. For very large datasets, however, distributed frameworks like MapReduce

[43] maybe required. Identifying and resolving the issues involved in applying C^3 in a framework like MapReduce is a compelling future research direction. Related to large scale data, we are also interested in exploring large scale network data aggregated from multiple unknown sources (i.e., the source of the individual nodes and edges are not observed). Because of variability between different sources, the quality and amount of attribute and edge information in the data from one source maybe completely different from the data from another source. Aggregating from multiple social networks, for example, the nodes from some sources (e.g., LinkedIn, Facebook) may have more accurate name and age information than nodes from other sources (e.g., MySpace, Friendster). Similarly, while nodes from some sources have certain types of relationships, nodes from others may not (e.g., LinkedIn has "group member" relationships but Friendster does not). This variance in the quality and amount of information can have a significant impact on how well algorithms perform on the aggregate network. For example, entity resolution approaches using name similarity and a single threshold is unlikely to do well given that the optimal threshold for co-referent pairs between nodes that came from one source maybe different from pairs in another source. Explicitly reasoning about the hidden sources of the nodes and using that knowledge to vary the threshold for pairs conditioned on these sources should lead to improved performance.

Although not mentioned in this dissertation, we also did extensive work in visualizing network data. In addition to the basic visualization available in GAIA, we also participated in developing the DualNet [122] and G-Pare [155] visualization tools. The goal in these systems is not only to better understand network data,

but also to understand machine learning algorithms applied to them. By visualizing the output of various algorithms in network data, ideally the use can better see the biases, strengths, and weaknesses of these algorithms that would not be obvious to when looking at raw output or data files. In G-Pare, for example, we identified instances of the phenomenon common to collective classification approaches known as "flooding" [20]. By directly observing this phenomenon, not only can we see how sensitive proposed models are to flooding, we can also begin to identify interventions (through annotations, additional features, parameter changes) we can use resolve the problem. We plan to continue developing visualizations for analyzing the results of algorithms, as well as developing visualizations which let you observe and interact with machine learning algorithms *during* their application.

8.3 Conclusions

In this dissertation, we studied and developed approaches for the problems of entity resolution, link prediction, collective classification, and graph identification on network data. Through this work, we highlight the importance of exploiting both the relational information in networks and the dependencies between various tasks in inferring an accurate and complete network. With the growing amount and variety of network data available due improvement in technology and growth of services like social networking websites, being able to work and reason about network data is only going to become more important. This dissertation for improving the quality and accessibility of these networks hopefully marks a humble but important step toward being able toward correctly and more easily work with network data.

Bibliography

[1] Daniel Abadi. Comparing domain-specific and non-domain-specific anaphora resolution techniques. Master's thesis, Cambridge University Masters Dissertation, 2003.

[2] James Abello, Adam L. Buchsbaum, and Jeffery R. Westbrook. A functional approach to external graph algorithms. In *Proceedings of the Annual European Symposium on Algorithms*, 1998.

[3] Sisay Fissaha Adafre and Maarten de Rijke. Discovering missing links in wikipedia. In *Proceedings of the SIGKDD Workshop on Link Discovery*, 2005.

[4] Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

[5] Rka Albert, Bhaskar DasGupta, Riccardo Dondi, Sema Kachalo, Eduardo Sontag, Alexander Zelikovsky, and Kelly Westbrook. A novel method for signal transduction network inference from indirect experimental evidence. *Journal of Computational Biology*, 14:407–419, 2007.

[6] James Allan. *Topic Detection and Tracking*. Kluwer Academic Pub, 2002.

[7] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the International Conference on Very Large Databases*, 2002.

[8] Periklis Andritsos, Ariel Fuxman, and Renee J. Miller. Clean answers over dirty databases: A probabilistic approach. In *Proceedings of the International Conference on Data Engineering*, 2006.

[9] Apache Software Foundation, Isabel Drost, Ted Dunning, Jeff Eastman, Otis Gospodnetic, Grant Ingersoll, Jake Mannix, Sean Owen, and Karl Wettin. Apache mahout, 2010. http://mloss.org/software/view/144/.

[10] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Research*, 14(6):1170–1175, June 2004.

[11] Albert-László Barabasi and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[12] Nizar N. Batada, Teresa Reguly, Ashton Breitkreutz, Lorrie Boucher, Bobby-Joe Breitkreutz, Laurence D. Hurst, and Mike Tyers. Still stratus not altocumulus: Further evidence against the date/party hub distinction. *PLoS Biology*, 5(6):e154+, June 2007.

[13] Vladimir Batagelj and Andrej Mrvar. *Pajek - Analysis and Visualization of Large Networks*, volume 2265. Springer, January 2002.

[14] Nicolas Bertin, Nicolas Simonis, Denis Dupuy, Michael E Cusick, Jing-Dong J Han, Hunter B Fraser, Frederick P Roth, and Marc Vidal. Confirmation of organized modularity in the yeast interactome. *PLoS Biology*, 5(6):e153, 06 2007.

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

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

[17] Indrajit Bhattacharya, Shantanu Godbole, and Sachindra Joshi. Structured entity identification and document categorization: Two tasks with one joint models. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2008.

[18] M Bilenko, R Mooney, W Cohen, and P Ravikumar. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18:16–23, Jan 2003.

[19] Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2008.

[20] Mustafa Bilgic and Lise Getoor. Active inference for collective classification. In *AAAI Conference on Artificial Intelligence*, 2010.

[21] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proceedings of the International Conference on Machine Learning*, 2010.

[22] Elizabeth M. Bonsignore, Cody Dunne, Dana Rotman, Marc Smith, Tony Capone, Derek L. Hansen, and Ben Shneiderman. First steps to netviz nirvana: Evaluating social network analysis with nodexl. In *Proceedings of the International Conference on Computational Science and Engineering*, volume 4, pages 332–339. IEEE Computer Society Press, 2009.

[23] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. Graphml progress report, structural layer proposal. In *Proceedings of the International Symposium on Graph Drawing*, pages 501–512, Heidelberg, 2001.

[24] Matthias Broecheler and Lise Getoor. Probabilistic similarity logic. In *Proceedings of the International Workshop on Statistical Relational Learning*, 2009.

[25] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*, 2005.

[26] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Neural Information Processing Systems*. MIT Press, 2007.

[27] G. Caruana, Maozhen Li, and Man Qi. A mapreduce based parallel svm for large scale spam filtering. In *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, volume 4, pages 2659–2662, july 2011.

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

[29] Jair Cervantes, Xiaoou Li, Wen Yu, and Kang Li. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4–6):611–619, 2008.

[30] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the SIGMOD International Conference on Management of Data*, 1998.

[31] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[32] Edward Chang, Kaihua Zhu, Hao Wang, Hongjie Bai, Jian Li, Zhihuan Qiu, and Hang Cui. Parallelizing support vector machines on distributed computers. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 257–264. MIT Press, Cambridge, MA, 2008.

[33] Anton Chechetka and CarlosErnesto Guestrin. Focused belief propagation for query-specific inference. In *International Conference on Artificial Intelligence and Statistics*, May 2010.

[34] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, 22(13):1623–1630, 2006.

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

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

[37] A. Corrada-Emmanuel. Enron email dataset research, 2004. http://ciir.cs.umass.edu/~corrada/enron.

[38] Corinna Cortes and Mehryar Mohri. AUC optimization versus error rate minimization. *Advances in Neural Information Processing Systems*, 16:313–320, 2004.

[39] Corinna Cortes and Mehryar Mohri. Confidence intervals for the area under the ROC curve. *Advances in Neural Information Processing Systems*, 17:305–312, 2005.

[40] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21 – 27, jan 1967.

[41] Koby Crammer, Yoram Singer, Nello Cristianini, John Shawe-taylor, and Bob Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:2001, 2001.

[42] Aron Culotta, Michael Wick, Robert Hall, Matthew Marzilli, and Andrew McCallum. Canonicalization of database records using adaptive similarity measures. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2007.

[43] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the Symposium on Operating Systems Design & Implementation*, 2004.

[44] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[45] M. Deng, S. Mehta, F. Sun, and T. Chen. Inferring domain-domain interactions from protein-protein interactions. *Genome Research*, 12(10):1540–1548, October 2002.

[46] Giuseppe Di Battista, Thomas Erlebach, Alexander Hall, Maurizio Patrignani, Maurizio Pizzonia, and Thomas Schank. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking*, 15:267–280, 2007.

[47] Christopher Diehl, Lise Getoor, and Galileo Mark Namata. Name reference resolution in organizational email archives. In *SIAM Conference on Data Mining*, 2006.

[48] Christopher Diehl, Galileo Mark Namata, and Lise Getoor. Relationship identification for social network discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[49] Jana Diesner and Kathleen Carley. Exploration of communication networks from the Enron email corpus. In *Proceedings of the ICDM Workshop on Link Analysis, Counterterrorism and Security*, Newport Beach, CA, USA, April 21-23 2005.

[50] Jana Diesner and Kathleen M. Carley. Exploration of communications networks from the Enron email corpus. In *Proceedings of the SDM Workshop on Link Analysis, Counterterrorism and Security*, pages 3–14, 2005.

[51] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the SIGMOD International Conference on Management of Data*, 2005.

[52] J.-P. Eckmann, E. Moses, and D. Sergi. Dialog in e-mail traffic. *ArXiv Condensed Matter E-Prints*, April 2003.

[53] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz – open source graph drawing tools. *Graph Drawing*, pages 483–484, 2001.

[54] Tamer Elsayed, Doug Oard, and Galileo Mark Namata. Resolving personal names in email using context expansion. In *Annual Meeting of the Association of Computational Linguistics*, pages 265–268, June 2008.

[55] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Mathematics Institute Hungarian Academy of Science*, 5:17–61, 1960.

[56] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1999.

[57] Stephen Farrell, Christopher Campbell, and Suvda Myagmar. Relescope: An experiment in accelerating relationships. In *Extended Abstracts on Human Factors in Computing Systems*, 2005.

[58] Tom Fawcett and Foster J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316, 1997.

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

[60] Yoav Freund, Raj Iyer, Robert Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[61] Alexander Gammerman, Katy S. Azoury, and Vladimir Vapnik. Learning by transduction. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 148–155, 1998.

[62] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Machine Learning*, 3:679–707, 2003.

[63] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7:3–12, 2005.

[64] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[65] Lise Getoor, Eran Segal, Benjamin Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *Proceedings of the IJCAI Workshop on Text Learning: Beyond Supervision*, 2001.

[66] Walter R. Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in Practice*. Chapman & Hall CRC, 1996.

[67] Vincent Granville, Mirko Krivanek, and Jean-Paul Rasson. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, jun 1994.

[68] Andrew Guillory and Jeff Bilmes. Label selection on graphs. In *Neural Information Processing Society*, Vancouver, Canada, December 2009.

[69] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations Newsletter*, 11:10–18, November 2009.

[70] J. Heer. Exploring Enron: Visualizing ANLP results, 2004. http://jheer.org/enron/v1/.

[71] Jeffrey Heer, Stuart K. Card, and James A. Landay. Prefuse: A toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 421–430, New York, NY, USA, 2005.

[72] Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Advances in Neural Information Processing Systems*, 2008.

[73] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 97–102, 1999.

[74] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the SIGMOD International Conference on Management of Data*, 1995.

[75] Ralf Holzer, Bradley Malin, and Latanya Sweeney. Email alias detection using social network analysis. In *Proceedings of the ACM SIGKDD Workshop on Link Discovery: Issues, Approaches, and Applications*, Chicago, Illinois, USA, August 2005.

[76] Hailiang Huang and Joel S. Bader. Precision and recall estimates for two-hybrid screens. *Bioinformatics*, 25(3):372–378, 2009.

[77] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries*, 2005.

[78] Zan Huang and Dennis K. J. Lin. The Time-Series Link Prediction Problem with Applications in Communication Surveillance. *Informs Journal On Computing*, 21:286–303, 2008.

[79] Tuyen Huynh and Raymond Mooney. Max-margin weight learning for markov logic networks. In Wray Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5781 of *Lecture Notes in Computer Science*, pages 564–579. Springer Berlin / Heidelberg, 2009.

[80] Paul Jaccard. 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, 1901.

[81] M. A. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14:491–498, 1995.

[82] Thorsten Joachims. Making large-scale svm learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[83] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pages 133–142, 2002.

[84] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, New York, NY, USA, 2005. ACM Press.

[85] Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen. Exploiting relationships for domain-independent data cleaning. In *Proceedings of the SIAM International Conference on Data Mining*, 2005.

[86] Hisashi Kashima, Tsuyoshi Kato, Yoshihiro Yamanishi, Masashi Sugiyama, and Koji Tsuda. Link propagation: A fast semi-supervised learning algorithm

for link prediction. In *Proceedings of the SIAM International Conference on Data Mining*, 2009.

[87] Henry Kautz, Bart Selman, and Mehun Shah. Referral web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[88] P. Keila and David Skillicorn. Structure in the Enron email dataset. In *Proceedings of the ICDM Workshop on Link Analysis, Security and Counterterrorism*, pages 55–64, 2005.

[89] Bryan Klimt and Yimin Yang. The Enron corpus: a new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*, Pisa, Italy, September 20-24 2004.

[90] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *Conference on Email and Anti-Spam*, 2004.

[91] Knowledge Discovery Laboratory, University of Massachusetts Amherst. Proximity software. http://kdl.cs.umass.edu/proximity.

[92] Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, and Pedro Domingos. The alchemy system for statistical relational ai. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2006.

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

[94] Ankit Kuwadekar and Jennifer Neville. Relational active learning for joint collective classification models. In *Proceedings of the International Conference on Machine Learning*, 2011.

[95] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence datasets. In *Proceedings of the International Conference on Machine Learning*, 2001.

[96] S. le Cessie and J. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41:191–201, 1992.

[97] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2008.

[98] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2, 2007.

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

[100] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the International Conference on Machine Learning*, 2003.

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

[102] Sofus A. Macskassy. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2009.

[103] Sofus A. Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2003.

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

[105] Bradley Malin. Unsupervised name disambiguation via social network similarity. In *Proceedings of the SIAM International Conference on Data Mining*, Newport Beach, CA, USA, April 20-22 2005.

[106] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[107] Shawn Martin, Diana Roe, and Jean-Loup Faulon. Predicting protein-protein interactions using signature products. *Bioinformatics*, 21:218–226, 2005.

[108] Robert McArthur and Peter Bruza. Discovery of implicit and explicit connections between people using email utterance. In *Proceedings of the European Conference of Computer-supported Cooperative Work*, 2003.

[109] Andrew McCallum, Andres Corrada-Emmanuel, and Xuerui Wang. The author-recipient-topic model for topic and role discovery in social networks: Experiments with Enron and academic email. Technical Report UM-CS-2004-096, University of Massachusetts Amherst, December 2004.

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

[111] Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems*, 2009.

[112] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI Workshop on Information Integration on the Web*, 2003.

[113] Andrew Mccallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems*, 2004.

[114] Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious inference in collective classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[115] Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious collective classification. *Journal of Machine Learning Research*, 10:2777–2836, 2009.

[116] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[117] H. W. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil. Mips: a database for genomes and protein sequences. *Nucleic acids research*, 30:31–34, 2002.

[118] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the ACM Conference on Information and Knowledge Management*, 2008.

[119] Elena Nabieva, Kam Jim, Amit Agarwal, Bernard Chazelle, and Mona Singh. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics*, 21:302–310, 2005.

[120] Galileo Mark Namata, Stanley Kok, and Lise Getoor. Collective graph identification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[121] Galileo Mark Namata, Hossam Sharara, and Lise Getoor. A survey of link mining tasks for analyzing noisy and incomplete networks. In Jiawei Han Philip S. S. Yu and Christos Faloutsos, editors, *Link Mining: Models, Algorithms, and Applications*. Springer, 2010.

[122] Galileo Mark Namata, Brian Staats, Lise Getoor, and Ben Shneiderman. A dual-view approach to interactive network visualization. In *ACM Conference on Information and Knowledge Management*, 2007.

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

[124] Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

[125] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, October 1959.

[126] Howard B. Newcombe and James M. Kennedy. Record linkage: making maximum use of the discriminating power of identifying information. *Communications ACM*, 5(11):563–566, 1962.

[127] M. E. J. Newman. Mixing patterns in networks. *Physics Review E*, 67(2):026126, Feb 2003.

[128] Mark E. J. Newman, Albert L. Barabasi, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[129] Abhijit S. Ogale and Yiannis Aloimonos. Shape and the stereo correspondence problem. *International Journal of Computer Vision*, 65(3):147–162, December 2005.

[130] Hiroaki Ogata and Yoneo Yano. Collecting organizational memory based on social networks in collaborative learning. In *Proceedings of World Conference on the WWW and Internet*, pages 822–827, 1999.

[131] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[132] Joshua O'Madadhain and Padhraic Smyth. EventRank: A framework for ranking time-varying networks. In *Proceedings of the KDD Workshop on Link Discovery*, pages 9–16, 2005.

[133] Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems*, 2003.

[134] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[135] Alexandrin Popescul and Lyle H. Ungar. Statistical relational learning for link prediction. In *Proceedings of the IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[136] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[137] Matthew J. Rattigan and David Jensen. The case for anomalous link discovery. *SIGKDD Explorations Newsletter*, 7:41–47, 2005.

[138] Matthew J. Rattigan, Marc Maier, and David Jensen. Exploiting network structure for active inference in collective classification. Technical report, University of Massachusetts Amherst, 2007.

[139] Matthew J. Rattigan, Marc Maier, David Jensen Bin Wu, Xin Pei, JianBin Tan, and Yi Wang. Exploiting network structure for active inference in collective classification. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, pages 429–434, Washington, DC, USA, 2007. IEEE Computer Society.

[140] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[141] Dan Roth, Kevin Small, and Ivan Titov. Sequential learning of classifiers for structured prediction problems. In *Proceedings of the Conference on Artificial Intelligence and Statistics*, 2009.

[142] Dan Roth and Wen-Tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Conference on Computational Natural Language*, 2004.

[143] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning*, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[144] Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.

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

[146] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pages 269–278, New York, NY, USA, 2002.

[147] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 2010.

[148] Michael Schwartz and David Wood. Discovering shared interests among people using graph analysis of global electronic mail traffic. *Communications of the ACM*, 36:78–89, 1992.

[149] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[150] Burr Settles. Active learning literature survey. 1648, University of Wisconsin-Madison, 2009.

[151] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the International Conference on Machine Learning*, pages 807–814, 2007.

[152] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[153] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, 3:88, 2007.

[154] Hossam Sharara, Lise Getoor, and Myra Norton. Active surveying: A probabilistic approach for identifying key opinion leaders. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2011.

[155] Hossam Sharara, Awalin Sopan, Galileo Mark Namata, Lise Getoor, and Lisa Singh. G-pare: A visual analytic tool for comparative analysis of uncertain graphs. In *IEEE Conference on Visual Analytics Science and Technology*, 2011.

[156] Rob Sherwood, Adam Bender, and Neil Spring. Discarte: a disjunctive internet cartographer. *SIGCOMM Computer Communication Review*, 38(4):303–314, 2008.

[157] Jitesh Shetty and Jafar Adibi. The Enron email dataset: Database schema and brief statistical report. http://www.isi.edu/~adibi/Enron/Enron_Dataset_Report.pdf.

[158] Rohit Singh, Jinbo Xu, and Bonnie Berger. Struct2net: Integrating structure into protein-protein interaction prediction. In *Pacific Symposium on Biocomputing*, pages 403–414, 2006.

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

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

[161] David Skillicorn. Detecting unusual and deceptive communication in email. In *Centers for Advanced Studies Conference*, Richmond Hill, Ontario, Canada, October 17-20 2005.

[162] Swapna Somasundaran, Galileo Mark Namata, Lise Getoor, and Janyce Wiebe. Opinion graphs for polarity and discourse classification. In *TextGraphs-4: Graph-based Methods for Natural Language Processing*, August 2009.

[163] Xiaodan Song, Ching-Yung Lin, Belle L. Tseng, and Ming-Ting Sun. Modeling and predicting personal information dissemination behavior. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pages 479–488, 2005.

[164] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[165] Neil Spring, David Wetherall, and Thomas Anderson. Reverse engineering the internet. *SIGCOMM Computer Communication Review*, 34(1):3–8, 2004.

[166] Andras Szilagyi, Vera Grimm, Adrian K Arakaki, and Jeffrey Skolnick. Prediction of physical protein-protein interactions. *Physical Biology*, 2(2):S1–S16, 2005.

[167] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2009.

[168] Ben Taskar, Abbeel Pieter, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002.

[169] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems*, 2003.

[170] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26:2001, 2001.

[171] The JUNG Framework Development Team. Jung-java universal network/graph framework. http://jung.sourceforge.net.

[172] The SNAP Platform Development Team. Snap: Stanford network analysis platform. http://snap.stanford.edu.

[173] Henry L. Van Trees. *Detection, Estimation, and Modulation Theory.* John Wiley and Sons, 1968.

[174] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Communities and Technologies*, pages 81–96. Kluwer, B.V., 2003.

[175] UK Sheffield University. SimMetrics - open source similarity measure library.

[176] Jeffrey D. Ullman. Information integration using logical views. In Foto N. Afrati and Phokion G. Kolaitis, editors, *Proceedings of the International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.

[177] Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1106–1111. AAAI Press, 2008.

[178] Xuerui Wang, Natasha Mohanty, and Andrew McCallum. Group and topic discovery from relations and text. In *Proceedings of the KDD Workshop on Link Discovery*, pages 28–35, 2005.

[179] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[180] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Hector Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 219–232. ACM, 2009.

[181] Michael Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An entity-based model for coreference resolution. In *Proceedings of the SIAM International Conference on Data Mining*, 2009.

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

[183] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[184] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[185] R. Yan and A. G. Hauptmann. Efficient margin-based rank learning algorithms for information retrieval. In *International Conference on Image and Video Retrieval*, 2006.

[186] Haiyuan Yu, Philip M. Kim, Emmett Sprecher, Valery Trifonov, and Mark Gerstein. The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Computational Biology*, 3(4):e59+, April 2007.

[187] Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7):823–829, 2006.

[188] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 2003.

[189] Dong Zhang, Daniel Gatica-Perez, Deb Roy, and Samy Bengio. Modeling interactions from email communications. In *IEEE International Conference on Multimedia and Expo*, 2006.

[190] Lan Zhang, Sharyl Wong, Oliver King, and Frederick Roth. Predicting co-complexed protein pairs using genomic and proteomic data integration. *BMC Bioinformatics*, 5:38, 2004.

[191] Elena Zheleva, Lise Getoor, Jennifer Golbeck, and Ugur Kuter. Using friendship ties and family circles for link prediction. In *Proceedings of the SIGKDD Workshop on Social Network Mining and Analysis*, Lecture Notes in Computer Science, 2008.

[192] Ding Zhou, Eren Manavoglu, Jia Li, C. Lee Giles, and Hongyuan Zha. Probabilistic models for discovering e-communities. In *Proceedings of the International Conference on World Wide Web*, pages 173–182, 2006.

[193] Jianhan Zhu. *Mining Web Site Link Structure for Adaptive Web Site Navigation and Search.* PhD thesis, University of Ulster at Jordanstown, UK, 2003.

[194] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning.* Morgan and Claypool Publishers, 2009.

[195] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.