A Pipeline Approach to Graph Identification

Galileo Mark Namata NAMATAG@CS.UMD.EDU
Lise Getoor GETOOR@CS.UMD.EDU
Department of Computer Science, University of Maryland, College Park, MD 20742 USA

1. Introduction

There is a growing wealth of data describing networks of various types, including social networks, physical networks such as transportation or communication networks, and biological networks. At the same time, there is a growing interest in analyzing these networks, in order to uncover general laws that govern their structure and evolution, and 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, and missing information inferred. In this paper, we introduce the notion of *graph identification*, which explicitly models the inference of a "cleaned" output network from a noisy input graph. It is this output network that is appropriate for further analysis. We present an illustrative example and use the example to explore the types of inferences involved in graph identification, as well as the challenges and issues involved in combining those inferences. We then present a simple, general approach to combining the inferences in graph identification and experimentally show how the performance of graph identification is sensitive to the inter-dependencies among these inferences.

2. Motivation

Suppose we wish to understand and analyze the social network of a large organization. Specifically, we wish to explore the network which identifies the individuals in the organization, the close friendships between those individuals, and the roles of the individuals. For large organizations, it may be very difficult, if not impossible, to gather such a network directly. What may instead be available for such an organization are the archived email communications. Using these communications, we can generate a communication network where nodes represent email addresses, edges represent a 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 noisy and incomplete for our analysis. The nodes in the communication network do not accurately reflect the individuals in the organization (i.e., individuals typically have multiple email addresses). Moreover, the communication network neither reflect the relationships between individuals (i.e., users not only email close friends but also acquaintances and competitors) nor the attributes for our analysis. Although the communication network is not directly appropriate for our task, we can use the correspondence between the communication and the social network that we would like to infer. This requires identifying a subset of email addresses corresponding to the person nodes who use those addresses, friends (who are likely to email each other regarding social events), and their roles (reflected in the content of communications and/or with whom they communicate). We refer to this process, from the available noisy network to the network appropriate for our analysis as *graph identification*.

3. Definition

Graph identification involves identifying the output graph from a given input graph, and involves constructing the mapping from the input to the output graph. The *output graph* ($\mathcal{G}_{\mathcal{O}}$) is the graph that is appropriate for further analysis. Ideally, the output graph can be acquired directly. In most cases, however, the output would be too difficult or expensive to directly acquire. What may be available, instead, is another graph which reflects the output graph but is too noisy and incomplete to directly use for our analysis. We refer to this available graph as the *input graph* ($\mathcal{G}_{\mathcal{I}}$). Given these graphs, we define graph identification as the general problem of inferring the desired output graph given a noisy input graph. The process of doing graph identification results in a mapping from elements (i.e., nodes, edges, and/or attributes) of the input graph to nodes in the output graph, the application of an edge existence function between nodes in the output graph, and mapping attributes of the nodes and edges in the output graph to values.

By its nature, graph identification is domain dependent. The specific inferences needed to perform graph identification will vary based on what the input and output graphs and how the those graphs are related. In the scenario pre-

Presented at *ILP-MLG-SRL*, Leuven, Belgium, 2009.

sented in section 2, for example, we are interested in the friendship network of individuals in a company as our output graph while our available input graph is the company's email communication network. We know, given our domain, that the mapping from the elements of the input graph to the nodes of the output graph is a many to one mapping from the email address nodes of the communication graph to the person nodes of the social network. Specifically, individuals in our social network likely own at least one email address in the communication network. Thus, we can identify the set of person nodes by mapping all email address nodes which belong to the same person to a person node in the output graph. The problem of creating this type of mapping is commonly referred to as *entity resolution*. Once we define the mapping between the nodes of the input and output graph, we can identify the set of edges that exist between the nodes of the output graph. In our domain, we can use the domain knowledge that people who are friends likely email each other, likely have similar attributes (e.g., interests) and likely communicate using terms common to that type of relationship (e.g., invitations to social events). We can thus use this knowledge to define (or learn) an edge existence function for friendship edges which we can then apply over all people nodes in the output graph. The problem of defining (or learning) this existence function is commonly referred to as *link prediction*. Finally, we know that individuals who fill a particular role are likely to discuss their role in their communications. We also know that social networks are often homophilic such that individuals who fill the same role are likely to be friends. We can use this knowledge to define a mapping for the attribute value of a person's role. The problem of predicting the values of these types of attributes is commonly referred to as *collective classification*. Consequently, in our example in section 2, graph identification is performed by the application of entity resolution, link prediction, and collective classification over the input graph.

Although the specific inferences in graph identification is domain dependent, at its core graph identification is still the problem of performing three inferences: the node mapping, edge existence function, and attribute value mapping. An important aspect of graph identification is how these three inferences should be applied and how they interact. One method for doing graph identification involves applying a collection of local predictors for the three inferences (e.g., applying local entity resolution (ER), link prediction (LP), and collective classification (CC) models). Another method for doing graph identification involves defining a joint model, and extracting the output graph with the most likely configuration. For the rest of this paper, we will explore the former method. We will examine a simple, general way the local predictors can be combined, and show, in general, how inter-dependent these inferences are.

4. Pipeline Graph Identification

One method to perform graph identification is to apply a collection of local predictors. The benefit of this method is that we can use any previously defined models for each of the individual predictors. For our scenario in section 2, for example, this method allows us to apply a previously learned local entity resolution, link prediction, and classification models. When applying local predictors, however, we have to address the additional challenge of how to combine these predictors. In what order should these predictors be applied? When and how should the predictions be committed and information shared between these predictors?

In this paper, we present and analyze the most direct way of applying these inferences, in a pipeline (Roth & Yih, 2004). In the *pipeline approach*, we apply the predictors one at a time and in sequence. Referring to the example in section 2, we apply the entity resolution, link prediction, and collective classification in turn, as shown in Algorithm 1. In this approach, since all predictors are applied only once and in sequence, all predictions are committed at the end of each turn and are available for use in the next predictor in the pipeline.

Algorithm 1 Example Pipeline Graph Identification

Input: $\mathcal{G}_I, ER_{model}, LP_{model}, CC_{model}$
Output: $\mathcal{G}_O$
1: Create intermediate graph, $\mathcal{G}_n$
2: $\mathcal{G}_n \Leftarrow \mathcal{G}_I^{test}$
3: Apply ER_{model} on $\mathcal{G}_n$ (i.e., merge nodes)
4: Apply LP_{model} on $\mathcal{G}_n$ (i.e., add links)
5: Apply CC_{model} on $\mathcal{G}_n$ (i.e., add labels)
6: $\mathcal{G}_O^{test} \Leftarrow \mathcal{G}_n$

5. Experiment

In order to study the pipeline approach of graph identification, we experiment to see what strengths and weaknesses this approach has over different types of networks. First, we developed a novel synthetic data generator which allows us to create an input and output graph, modeled after the communication and social networks presented in section 2. In our generator, we can control the reference, link existence, and label ambiguity in the inference so that we can vary the ability of each predictor, ER, LP, and CC, respectively, to make an accurate inference. We generate networks with different ambiguity levels (Low, Med, High) for each type of ambiguity. and perform graph identification by applying Algorithm 1. In our experiments, we used three commonly used predictors for the ER, LP, and CC models. For entity resolution, we use collective relational clustering (CRC) (Bhattacharya & Getoor, 2007). In CRC, for a given pair of nodes we use the similarity between the node entity attributes of the input graph nodes for the feature similarity and the Jaccard-Coefficient similarity of their neighborhoods for the relational similarity. For link

prediction, we use logistic regression using a feature which measures the percent similarity of the mapped input graph node attributes of a given output graph node pair. Finally, for collective classification, we use the iterative classification algorithm (ICA) (Lu & Getoor, 2003), using logistic regression for the bootstrap and relational classifiers. For a given output graph node, we use aggregates of the attributes over the mapped input graph nodes in the bootstrap classifier. For the relational classifier, we use these aggregates, as well as the percent of neighboring nodes (i.e., nodes adjacent to a given node with a friendship edge) which have a specific label. We train predictors, for all combinations of ambiguity (27 in total), on one graph and test on another graph to compute the F1 performance averaged over six runs for each predictor. We provide specific details of the synthetic data generator and how we compute the F1 performance over our experiments in the supplementary material.[1]

5.1. Results and Discussion

The results are presented in Table 1. Due to space, we only show results where we vary one type of ambiguity while holding the others at medium. In general, we see that good performance of predictors early in the pipeline result in improved performance of later predictors. In fact, the best performances are seen when the entity ambiguity is low resulting in ER performing well. Good ER performance results in a more accurate set of person nodes, and thus a more accurate set of mappings for use by the features of LP and CC. We see the same trend in CC performance when link existence ambiguity is low. LP performance improves which results in more accurate links for the relational features used in CC. Note though the improvement in LP performance does not affect ER and the improvement in CC performance does not affect LP and ER. This is a weakness in the pipeline approach in that the flow of information is only one way. Ideally, in graph identification, the models and features used by the predictors should be able to make use of predictions from all other predictors to improve its performance (e.g., use labels and predicted edges in the ER feature relational similarity, use predicted labels in LP relational features). An obvious extension to address this weakness is an iterative pipeline approach where the pipeline is repeatedly applied over the network. We performed an initial study of the iterative pipeline approach but the results were inconclusive indicating a naive iterative approach may not be enough. This is a subject of future work. Next, we note that when predictors early in the pipeline perform poorly, the effect is reduced performance for all predictors later in the pipeline. In fact, the resulting reduction in performance can be drastic as shown when we increase reference ambiguity. Although link existence and label ambiguity is held constant, poor performance by ER early in the pipeline results in up to a 0.452 drop in LP and 0.198 in CC performance as both are forced to make predictions over people nodes whose mapped email address nodes inaccurately and incompletely reflect a person and that person node's friendships and attributes. Thus, in graph identification, we need to be aware of what the expected performance of each predictor is and how each predictor will impact the other predictors in the overall inference.

[1]Supplementary materials are available for download from http://linqs.cs.umd.edu/supplementary/namata-mlg09

Table 1. Average and standard deviation of F1 performance of ER, LP, and CC in the pipeline approach of various levels of ambiguity

	ER (F1)	*LP (F1)*	*CC (F1)*
Reference Ambiguity			
Low	0.758 ± 0.093	0.554 ± 0.089	0.715 ± 0.065
Med	0.627 ± 0.169	0.439 ± 0.148	0.677 ± 0.078
High	0.213 ± 0.033	0.102 ± 0.034	0.517 ± 0.110
Link Existence Ambiguity			
Low	0.627 ± 0.169	0.500 ± 0.161	0.705 ± 0.066
Med	0.627 ± 0.169	0.439 ± 0.148	0.677 ± 0.078
High	0.627 ± 0.169	0.101 ± 0.057	0.653 ± 0.052
Label Ambiguity			
Low	0.627 ± 0.169	0.439 ± 0.148	0.797 ± 0.069
Med	0.627 ± 0.169	0.439 ± 0.148	0.677 ± 0.078
High	0.627 ± 0.169	0.439 ± 0.148	0.517 ± 0.085

6. Conclusion and Future Work

In this paper, we introduce the notion of graph identification. We discuss the types of inferences involved in graph identification and explore those inferences using an illustrative example problem. We present a simple, general approach to applying these inferences in graph identification and show how important the inter-dependence of these inferences is in accurately predicting the output graph. In future work, we plan to further explore these inter-dependencies by exploring alternate ways to combine local predictors and comparing those methods to a full joint approach to graph identification.

Acknowledgments

The work was supported by NSF Grant No. 0746930.

References

Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, *1*, 1–36.

Lu, Q., & Getoor, L. (2003). Link-based classification. *Proceedings of the International Conference on Machine Learning*.

Roth, D., & Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. *Proc. of CoNLL-2004* (pp. 1–8). Boston, MA, USA.