Edge Prediction via Relational Markov Networks

Emmanouel Liodakis
liodakis@stanford.edu

March 15, 2011

1 Introduction

In the last decade the computer science community has increasingly relied on graphical networks to model user interaction in a variety of different scenarios. For example, in social networks nodes have been used to represent users and their accompanying edges to represent friendships. In publication networks authors are represented by nodes and coauthored papers by edges.

In general these models span large networks with thousands, if not millions, of nodes. As research groups gain access to large datasets that model these networks, one of the questions that seems to emerge is: Is there a way to predict future edges in a graphical network? Specifically, given a network G at time t is there a way to predict the edges that will be created in the network between time t and $t + 1$. We define this question as the edge-prediction problem.

Note that this problem is also related to the problem of inferring missing edges from a partially observed network. In these networks we assume that only a subset of all interactions are visible, it is then the goal of the algorithm to infer the missing links.[1] While this problem is related to the one at hand, it deals with a graph G' at time t and does not focus on the evolution inherent to many of these large networks.

While research has been done by looking simply at the nodes and and edges of the graph[2], we hoped to expand on this research by not only using the graph structure, but also additional data associated with each connection to make more informed predictions. Specifically, we attempt to infer connections by representing the graphical network as a Relational Markov network (RMN) and then learn on this network, using the data to predict probability scores for new edges being created between time t and $t + 1$.

2 Model

Our model builds upon the methods described in [3] but includes the textual content of the emails sent in order to improve prediction accuracy. Our model is built on two hypotheses:

1. Two employees are more likely to be linked if they send similar emails. One example of this occurring is with human resource representatives and recruiters. They tend to know each other because of their profession, and they tend to send similar emails (especially to potential employees).

2. Whether an email is personal or professional in nature affects how people related to the sender and recipient are linked in the social graph. Professional emails tend to be sent within the same department, and people within a specific department tend to know each other.

2.1 Relational Markov Networks

We utilize the Relational Markov network (RMN) as described in [4]. A RMN describes potentials between variables (the attributes of objects) at a template level, so that one set of parameters can describe multiple potentials. Formally, we define RMNs in terms of relational clique templates.

Definition 1. *A relational clique template* $C = (F, W, S, \phi, \Theta)$ *consists of five components:*

- $F = \{F_i\}$ *- a set of logical variables, where each* F_i *is typed.*
- W *- a boolean formula on the variables* F_i*.*
- S *- a subset of attributes of* F*.*
- ϕ_C *- a potential parameterized by* Θ*.*

A relational clique template describes the following process: "For every value of F such that the expression W is satisfied, define a potential ϕ_C over the variables S parameterized by Θ." The semantics used to describe RMNs are very similar to the "SELECT" query used in SQL.

Definition 2. *A Relational Markov network* $\mathcal{M}$ *specifies a set of clique templates* $\mathbf{C}$ *with corresponding potentials* ϕ_C *for each* $C \in \mathbf{C}$ *to define a distribution:*

$$P(\mathcal{I}.X) = \frac{1}{Z} \prod_{c \in C} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.X_c) \quad (1)$$

where $\mathcal{I}.X$ *is an instantiation of the network* $\mathcal{M}$*, and* $C(\mathcal{I})$ *is the set of cliques for which the relational clique template* C *applies to the instantiation* $\mathcal{I}$*.*

2.1.1 Classification Using RMNs

The RMN allows us to apply Markov networks to classification tasks. In standard classification tasks, the labels that are predicted

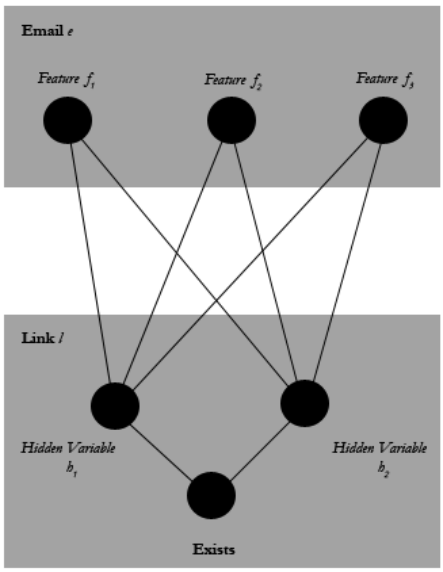

Figure 1: Template Example

are conditionally independent of each other given the parameters of the model. In RMNs, the labels that we are trying to predict are nodes in a large unrolled Markov network, so it is very likely that the values that we are trying to predict are dependent on each other.

To train RMNs for classification, we have a training network and a test network, which are instantiations of the specific RMN we are trying to train. We learn parameters over the training network (using maximum likelihood, gradient methods, or EM if there is missing data). To predict, we perform maximum a posteriori estimation with the parameters learned during fixed training, so our predicted labels maximize the probability of the data given the parameters.

2.1.2 RMNs in our Model

A significant portion of our model relates features of emails to link predictions. We model the link between two employees as an object type, called `link`, with a single observed at-

tribute
`link.exists` (which describes whether a social link exists between two employees), and several hidden attributes $\texttt{link.hidden}_i$.

We also have a type for emails (which we will refer to as `email`). Each email contains some visible attributes (such as sender, recipient, message length, along with other features extracted from the bodies of the emails).

For each link object, we define a potential between each hidden attribute and the observed attribute. In addition, we have potentials connecting features of the emails to the hidden attributes of link objects. Unfortunately, connecting every email to every possible link between two employees creates a network on which it is is computationally expensive to perform learning, so we have to restrict the pairs of emails and links over which we define potentials.

More formally, our undirected model uses two relational clique templates: one to define potentials between the features of email objects and the hidden attributes of links between employees, and one to define potentials between the hidden attributes and the observed (`exists`) attribute of a link. Refer to Figure 3.

2.2 Specifics

2.2.1 Feature Selection

We extracted several features from each email:

1. The LSA score of the email
2. The percentage of characters in the email that was numeric
3. The number of total characters in the email

These items were discretized so that we could run them through discrete inference libraries.

2.2.2 Relational Clique Templates

As described above, we used relational clique templates in two parts of our RMN:

1. In connecting every hidden attribute of a link object to the `exists` attribute for that object, and
2. In connecting features of emails to hidden attributes of link objects.

For (1), for every link variable `l`, we have a potential $\phi_C(\texttt{l.hidden}_i, \texttt{l.exists})$ for each hidden variable $\texttt{l.hidden}_i$.

For (2), we tried two different templates:

1. We learned similarities between the user's emails by running Latent Semantic Analysis (LSA) on the bodies of a hold-out set of emails. LSA produces a vector embedded in $\mathbb{R}^n$ for each email in the set. We compared the Euclidean distances of the emails, and picked all pairs of emails for which their distances were under a certain threshold value. For each pair of emails $\texttt{e}_1, \texttt{e}_2$ that satisfied this threshold value, we created potentials $\phi_C(\texttt{e}_1\texttt{.feature}_i, \texttt{l}_{1,2}\texttt{.hidden}_j)$ and $\phi_C(\texttt{e}_2\texttt{.feature}_i, \texttt{l}_{1,2}\texttt{.hidden}_j)$ for each feature i and each hidden attribute j. The rationale behind this template is the hypothesis that the messages of similar emails are similar, and thus more likely to be linked. We called this method "EmailNeighbor."
2. We leveraged the social network that was already present at training time, by defining a potential (similar to above) between each email, and pairs of employees $(\texttt{l}_\texttt{s}, \texttt{l}_\texttt{r})$, where $\texttt{l}_\texttt{s}$ is a neighbor of the sender of the email, and $\texttt{l}_\texttt{r}$ is a neighbor of the recipient of the email. We called this template "GraphNeighbor."

Because of time constraints, we used these template frameworks as heuristics to determine which two people are more likely to be linked, rather than modelling relationships between properties of emails and employees.

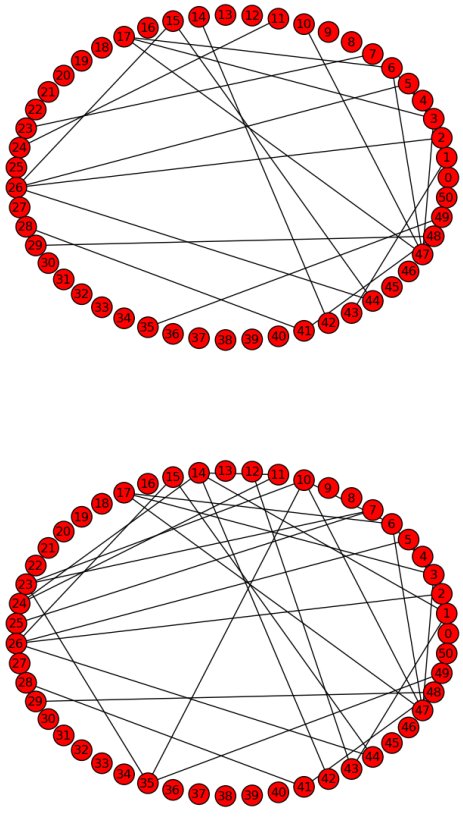

Figure 2:
Training Network before November, 2001 [above]
Training Network after November, 2001 [below]

We used this to our advantage because any email that was not connected to a link object did not need to be modeled (our RMN distribution does not change when conditioned on that email).

3 Experimental Procedure

3.1 Dataset

The project is based on the Enron email dataset. [5] The dataset contains 517,431 emails from 151 employees. Each email contains the sender and receiver fields, along with the date and time, subject, and body text fields.

We tested our model on a well-connected subset of this dataset: We took roughly 50 employees and all of the emails sent between the employees. We further separated the employees into a training set and a testing set. For training, the data we used to build the RMN consisted of the emails that were passed between employees in the training set before November 11, 2001, and emails that were sent by employees of the training set to other employees before November 11, 2001. The labels were the new links formed between November 11, 2001 and December 31, 2004.

We considered two employees to be connected at a time point t if one of them had sent the other an email before t.

3.2 Baseline

In order to test the predictive power of the model we were first required to establish a benchmark. We used Liben-Nowell and Kleinberg's proximity metrics to calculate baseline edge prediction probabilities. Specifically, we implemented both graph distance and common neighbor metrics to calculate the scores of new edges. The first metric, graph distance, calculates a score for a new edge $e_{i,j}$ between nodes n_i and n_j based on the distance of the shortest path between the two nodes through undirected edges in the training graph. The second metric, closest neighbor, is based on the intuition that a future edge is more likely between two nodes if they share a large number of neighboring nodes. Therefore, the metric calculates the number of neighbors (distance one away) for both n_i and n_j. The metric's score is then based on the size of the set calculated from the intersection between the two sets. In Figure 3 we present, as a percentage, the number of accurate edge predictions over total edge predictions for each metric averaged across the cross validation folds. In accordance with the

results presented by Liben-Nowell and Kleinberg, the closest neighbor metric provides the better edge predictor.[3]

Both of these metrics only rely on the structure of the graph and thus serve as a good baseline, but are far from the best future edge predictors. Intuitively, an algorithm that incorporates both the email subjects and bodies in its prediction of future edges would perform better.

3.3 Experiments

Due to time constraints, we ran two experiments. The first experiment compared the EmailNeighbor and GraphNeighbor templates to the baselines. The second experiment tested the effectiveness of having hidden attributes for each link when used for classification. The motivation for using hidden attributes is this: when classifying whether a social relationship exists, there are variables that are not necessarily observed but do affect the existence of the link.

We tested the model with no hidden variables, with one hidden variable, and with two hidden variables. The hidden variables we used were all binary.

4 Results, Analysis

Compared to the baseline metrics, the two clique templates performed better than the baseline in terms of accuracy.

One of the reasons that the EmailNeighbor template performed better relative to the baseline and the GraphNeighbor template is that it takes into account links between nodes that may not have been connected originally. On the other hand, the baseline methods and the GraphNeighbor template cannot predict links between all users if they are in separate connected components.

Modeling hidden variables did seem to have a significant effect on the accuracy of our classification. Without hidden variables (the

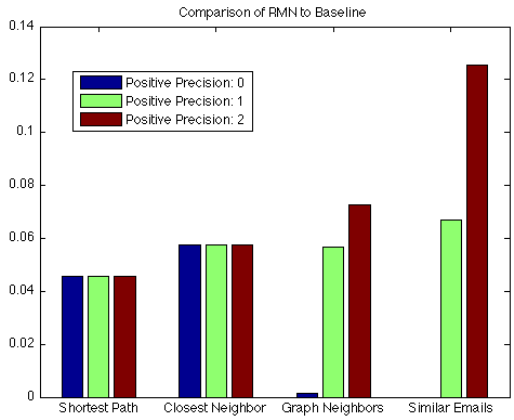

Figure 3: Comparison of RMN to Baseline

email features were directly connected to the `exists` attribute of the link object), the models achived close to 0% accuracy, with the accuracy increasing only as the number of hidden variables used increased. This seems intuitive in that more correlations can be modeled with more hidden variables. Having zero variables would mean that the model was parameterized by one potential. Because social networks are in general fairly connected, the training and test networks we created were not the best networks to test our models on. Our graphs were much more disconnected (they often had multiple nodes that were not connected to any other point in the graph) before November 2001. There are a couple of factors that contribute to this. The first is that the number of messages sent by each employee in the dataset is highly non-uniform. Some employees sent thousands of emails over the time period, while others only sent one or two. In addition, the rate of emails being sent was also not uniform. Around late 2001, when the Enron scandal occurred, there was a massive spike in the number of emails sent.

5 Conclusion

As we increased the number of hidden variables between the feature variables and the prediction variable our model began performing better than the baseline. Without hidden variables, the model overfit the relational clique template parameters to the training data. This caused the model to perform poorly. However, as we increased the number of hidden variables, we introduced (non-relational) potentials for the classification task, which in turn allowed the model to learn parameters that generalized well.

The relatively low performance of the overall model (it only achieved an accuracy of 13%) can be directly attributed to the data set it was run on. As can be seen in Figure 2, many of the nodes within the networks that we were training on did not have any edges. This intuitively would keep overall performance low. In order to increase performance we would have to more carefully separate the training and testing cross-validation folds. Training on larger networks (with more nodes and edges) would also have helped with this problem. One possible avenue to pursue would be to use sampling for inference to be able to learn larger networks. Overall, the model's general behavior compared to that of the baseline followed our theoretical expectation. With more representative data we are confident that the networks overall positive precision would have also increased.

6 Acknowledgement

We would like to thank Professor Daphne Koller and and her teaching staff from Stanford University. This analysis was prepared as a final project for CS228: Probabilistic Graphical Models.

References

[1] Taskar,B. Wong, M. Abbeel, P Koller, D. "Link Prediction in Relational Data". Stanford University [link]

[2] Sarukkai, R. "Link prediction and path analysis using Markov chains". Yahoo, Inc. [link]

[3] Liben-Nowell, D. Kleinberg, J. "The Link-Prediction Problem for Social Networks". Journal of the American Society for Information Science and Technology. [link]

[4] Taskar, B. Abbeel, P. Koller, D. "Discriminative Probabilistic Models for Relational Data". Stanford University [link]

[5] Shetty, J. Adibi, J. "The Enron Email Dataset: Database Schema and Brief Statistical Report". [link]