6 Relational Markov Networks

Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller

One of the key challenges for statistical relational learning is the design of a representation language that allows flexible modeling of complex relational interactions. Many of the formalisms presented in this book are based on the directed graphical models (probabilistic relational models, probabilistic entity-relationship models, Bayesian logic programs). In this chapter, we present a probabilistic modeling framework that builds on undirected graphical models (also known as Markov random fields or Markov networks). Undirected models address two limitations of the previous approach. First, undirected models do not impose the acyclicity constraint that hinders representation of many important relational dependencies in directed models. Second, undirected models are well suited for discriminative training, where we optimize the conditional likelihood of the labels given the features, which generally improves classification accuracy. We show how to train these models effectively, and how to use approximate probabilistic inference over the learned model for collective classification and link prediction. We provide experimental results on hypertext and social network domains, showing that accuracy can be significantly improved by modeling relational dependencies.[1]

6.1 Introduction

We focus on supervised learning as a motivation for our framework. The vast majority of work in statistical classification methods has focused on "flat" data – data consisting of identically structured entities, typically assumed to be i.i.d. However, many real-world data sets are innately relational: hyperlinked webpages, cross-citations in patents and scientific papers, social networks, medical records, and more. Such data consists of entities of different types, where each entity type is

1. This chapter is based on work in [21, 22].

characterized by a different set of attributes. Entities are related to each other via different types of links, and the link structure is an important source of information.

Consider a collection of hypertext documents that we want to classify using some set of labels. Most naively, we can use a bag-of-words model, classifying each webpage solely using the words that appear on the page. However, hypertext has a very rich structure that this approach loses entirely. One document has hyperlinks to others, typically indicating that their topics are related. Each document also has internal structure, such as a partition into sections; hyperlinks that emanate from the same section of the document are even more likely to point to similar documents. When classifying a collection of documents, these are important cues that can potentially help us achieve better classification accuracy. Therefore, rather than classifying each document separately, we want to provide a form of *collective classification*, where we simultaneously decide on the class labels of all of the entities together, and thereby can explicitly take advantage of the correlations between the labels of related entities.

Another challenge arises from the task of predicting which entities are related to which others and what are the types of these relationships. For example, in a data set consisting of a set of hyperlinked university webpages, we might want to predict not just which page belongs to a professor and which to a student, but also which professor is which student's advisor. In some cases, the existence of a relationship will be predicted by the presence of a hyperlink between the pages, and we will have only to decide whether the link reflects an advisor-advisee relationship. In other cases, we might have to infer the very existence of a link from indirect evidence, such as a large number of coauthored papers.

We propose the use of a joint probabilistic model for an entire collection of related entities. Following the work of Lafferty et al. [13], we base our approach on discriminatively trained undirected graphical models, or *Markov networks* [17]. We introduce the framework of *relational Markov networks (RMNs)*, which compactly defines a Markov network over a relational data set. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex patterns over related entities. For example, we can represent a pattern where two linked documents are likely to have the same topic. We can also capture patterns that involve groups of links: for example, consecutive links in a document tend to refer to documents with the same label. As we show, the use of an undirected graphical model avoids the difficulties of defining a coherent generative model for graph structures in directed models. It thereby allows us tremendous flexibility in representing complex patterns.

Undirected models lend themselves well to discriminative training, where we optimize the conditional likelihood of the labels given the features. Discriminative training, given sufficient data, generally provides significant improvements in classification accuracy over generative training (see [23]). We provide an effective parameter estimation algorithm for RMNs which uses conjugate gradient combined with approximate probabilistic inference (belief propagation [17, 14, 12]) for estimating the gradient. We also show how to use approximate probabilistic inference

over the learned model for collective classification and link prediction. We provide experimental results on a webpage classification and social network task, showing significant gains in accuracy arising both from the modeling of relational dependencies and the use of discriminative training.

6.2 Relational Classification and Link Prediction

Consider hypertext as a simple example of a relational domain. A relational domain is defined by a schema, which describes entities, their attributes, and the relations between them. In our domain, there are two entity types: Doc and Link. If a webpage is represented as a bag of words, Doc would have a set of Boolean attributes Doc.$HasWord_k$ indicating whether the word k occurs on the page. It would also have the label attribute Doc.*Label*, indicating the topic of the page, which takes on a set of categorical values. The Link entity type has two attributes: Link.*From* and Link.*To*, both of which refer to Doc entities.

In general, a *schema* specifies of a set of entity types $\mathcal{E} = \{E_1, \ldots, E_n\}$. Each type E is associated with three sets of attributes: content attributes $E.\mathbf{X}$ (e.g., Doc.$HasWord_k$), label attributes $E.\mathbf{Y}$ (e.g., Doc.*Label*), and reference attributes $E.\mathbf{R}$ (e.g. Link.*To*). For simplicity, we restrict label and content attributes to take on categorical values. Reference attributes include a special unique key attribute $E.K$ that identifies each entity. Other reference attributes $E.R$ refer to entities of a single type $E' = Range(E.R)$ and take values in $Domain(E'.K)$.

An *instantiation* $\mathcal{I}$ of a schema $\mathcal{E}$ specifies the set of entities $\mathcal{I}(E)$ of each entity type $E \in \mathcal{E}$ and the values of all attributes for all of the entities. For example, an instantiation of the hypertext schema is a collection of webpages, specifying their labels, the words they contain, and the links between them. We will use $\mathcal{I}.\mathbf{X}$, $\mathcal{I}.\mathbf{Y}$, and $\mathcal{I}.\mathbf{R}$ to denote the content, label, and reference attributes in the instantiation $\mathcal{I}$; $\mathcal{I}.\mathbf{x}$, $\mathcal{I}.\mathbf{y}$, and $\mathcal{I}.\mathbf{r}$ to denote the values of those attributes. The component $\mathcal{I}.\mathbf{r}$, which we call an *instantiation skeleton* or *instantiation graph*, specifies the set of entities (nodes) and their reference attributes (edges). A hypertext instantiation graph specifies a set of webpages and links between them, but not their words or labels.

To address the link prediction problem, we need to make links first-class citizens in our model. Following Getoor et al. [7], we introduce into our schema object types that correspond to links between entities. Each link object ℓ is associated with a tuple of entity objects $(o_1, \ldots, o_k)$ that participate in the link. For example, a Hyperlink link object would be associated with a pair of entities — the linking page, and the linked-to page, which are part of the link definition. We note that link objects may also have other attributes; e.g., a hyperlink object might have attributes for the anchor words on the link.

As our goal is to predict link existence, we must consider links that exist and links that do not. We therefore consider a set of *potential* links between entities. Each potential link is associated with a tuple of entity objects, but it may or may

not actually exist. We denote this event using a binary *existence* attribute *Exists*, which is *true* if the link between the associated entities exists and *false* otherwise. In our example, our model may contain a potential link ℓ for each pair of webpages, and the value of the variable $\ell.Exists$ determines whether the link actually exists or not. The link prediction task now reduces to the problem of predicting the existence attributes of these link objects.

6.3 Graph Structure and Subgraph Templates

The structure of the instantiation graph has been used extensively to infer its importance in scientific publications [5] and hypertext [10]. Several recent papers have proposed algorithms that use the link graph to aid classification. Chakrabarti et al. [2] use system-predicted labels of linked documents to iteratively relabel each document in the test set, achieving a significant improvement compared to a baseline of using the text in each document alone. A similar approach was used by Neville and Jensen [16] in a different domain. Slattery and Mitchell [19] tried to identify directory (or hub) pages that commonly list pages of the same topic, and used these pages to improve classification of university webpages. However, none of these approaches provide a coherent model for the correlations between linked webpages. Thus, they apply combinations of classifiers in a procedural way, with no formal justification.

Taskar et al. [20] suggest the use of *probabilistic relational models (PRMs)* for the collective classification task. PRMs [11, 6] are a relational extension to Bayesian networks [17]. A PRM specifies a probability distribution over instantiations consistent with a given instantiation graph by specifying a Bayesian network-like template-level probabilistic model for each entity type. Given a particular instantiation graph, the PRM induces a large Bayesian network over that instantiation that specifies a joint probability distribution over all attributes of all of the entities. This network reflects the interactions between related instances by allowing us to represent correlations between their attributes.

In our hypertext example, a PRM might use a naive Bayes model for words, with a directed edge between $\mathsf{Doc}.Label$ and each attribute $\mathsf{Doc}.HadWord_k$; each of these attributes would have a *conditional probability distribution* $P(\mathsf{Doc}.HasWord_k \mid \mathsf{Doc}.Label)$ associated with it, indicating the probability that word k appears in the document given each of the possible topic labels. More importantly, a PRM can represent the interdependencies between topics of linked documents by introducing an edge from $\mathsf{Doc}.Label$ to $\mathsf{Doc}.Label$ of two documents if there is a link between them. Given a particular instantiation graph containing some set of documents and links, the PRM specifies a Bayesian network over all of the documents in the collection. We would have a probabilistic dependency from each document's label to the words on the document, and a dependency from each document's label to the labels of all of the documents to which it points. Taskar et al. [20] show that

this approach works well for classifying scientific documents, using both the words in the title and abstract and the citation-link structure.

However, the application of this idea to other domains, such as webpages, is problematic since there are many cycles in the link graph, leading to cycles in the induced "Bayesian network," which is therefore not a coherent probabilistic model. Getoor et al. [8] suggest an approach where we do not include direct dependencies between the labels of linked webpages, but rather treat links themselves as random variables. Each two pages have a "potential link," which may or may not exist in the data. The model defines the probability of the link existence as a function of the labels of the two endpoints. In this link existence model, labels have no incoming edges from other labels, and the cyclicity problem disappears. This model, however, has other fundamental limitations. In particular, the resulting Bayesian network has a random variable for each potential link — N^2 variables for collections containing N pages. This quadratic blowup occurs even when the actual link graph is very sparse. When N is large (e.g., the set of all webpages), a quadratic growth is intractable. Even more problematic are the inherent limitations on the expressive power imposed by the constraint that the directed graph must represent a coherent generative model over graph structures. The link existence model assumes that the presence of different edges is a conditionally independent event. Representing more complex patterns involving correlations between multiple edges is very difficult. For example, if two pages point to the same page, it is more likely that they point to each other as well. Such interactions between many overlapping triples of links do not fit well into the generative framework.

Furthermore, directed models such as Bayesian networks and PRMs are usually trained to optimize the joint probability of the labels and other attributes, while the goal of classification is a discriminative model of labels given the other attributes. The advantage of training a model only to discriminate between labels is that it does not have to trade off between classification accuracy and modeling the joint distribution over nonlabel attributes. In many cases, discriminatively trained models are more robust to violations of independence assumptions and achieve higher classification accuracy than their generative counterparts.

In our experiments, we found that the combination of a relational language with a probabilistic graphical model provides a very flexible framework for modeling complex patterns common in relational graphs. First, as observed by Getoor et al. [7], there are often correlations between the attributes of entities and the relations in which they participate. For example, in a social network, people with the same hobby are more likely to be friends.

We can also exploit correlations between the *labels* of entities and the relation type. For example, only students can be teaching assistants in a course. We can easily capture such correlations by introducing cliques that involve these attributes. Importantly, these cliques are informative even when attributes are not observed in the test data. For example, if we have evidence indicating an advisor-advisee relationship, our probability that X is a faculty member increases, and thereby our belief that X participates in a teaching assistant link with some entity Z decreases.

We also found it useful to consider richer subgraph templates over the link graph. One useful type of template is a *similarity* template, where objects that share a certain graph-based property are more likely to have the same label. Consider, for example, a professor X and two other entities Y and Z. If X's webpage mentions Y and Z in the same context, it is likely that the X-Y relation and the Y-Z relation are of the same type; for example, if Y is Professor X's advisee, then probably so is Z. Our framework accomodates these patterns easily, by introducing pairwise cliques between the appropriate relation variables.

Another useful type of subgraph template involves *transitivity* patterns, where the presence of an A-B link and of a B-C link increases (or decreases) the likelihood of an A-C link. For example, students often assist in courses taught by their advisor. Note that this type of interaction cannot be accounted for by just using pairwise cliques. By introducing cliques over triples of relations, we can capture such patterns as well. We can incorporate even more complicated patterns, but of course we are limited by the ability of belief propagation to scale up as we introduce larger cliques and tighter loops in the Markov network.

We note that our ability to model these more complex graph patterns relies on our use of an undirected Markov network as our probabilistic model. In contrast, the approach of Getoor et al. [8] uses directed graphical models (Bayesian networks and PRMs [11]) to represent a probabilistic model of both relations and attributes. Their approach easily captures the dependence of link existence on attributes of entities. But the constraint that the probabilistic dependency graph be a directed acyclic graph makes it hard to see how we would represent the subgraph patterns described above. For example, for the transitivity pattern, we might consider simply directing the correlation edges between link existence variables arbitrarily. However, it is not clear how we would then parameterize a link existence variable for a link that is involved in multiple triangles. See [20] for further discussion.

6.4 Undirected Models for Classification

As discussed, our approach to the collective classification task is based on the use of undirected graphical models. We begin by reviewing *Markov networks*, a "flat" undirected model. We then discuss how Markov networks can be extended to the relational setting.

6.4.1 Markov Networks

We use $\mathbf{V}$ to denote a set of discrete random variables and $\mathbf{v}$ an assignment of values to $\mathbf{V}$. A Markov network for $\mathbf{V}$ defines a joint distribution over $\mathbf{V}$. It consists of a qualitative component, an undirected dependency graph, and a quantitative component, a set of parameters associated with the graph. For a graph G, a *clique* is a set of nodes $\mathbf{V}_c$ in G, not necessarily maximal, such that each $V_i, V_j \in \mathbf{V}_c$ is connected by an edge in G. Note that a single node is also considered a clique.

Definition 6.1
Let $G = (\mathbf{V}, E)$ be an undirected graph with a set of cliques $C(G)$. Each $c \in C(G)$ is associated with a set of nodes $\mathbf{V}_c$ and a *clique potential* $\phi_c(\mathbf{V}_c)$, which is a non-negative function defined on the joint domain of $\mathbf{V}_c$. Let $\Phi = \{\phi_c(\mathbf{V}_c)\}_{c \in C(G)}$. The Markov net (G, Φ) defines the distribution $P(\mathbf{v}) = \frac{1}{Z} \prod_{c \in C(G)} \phi_c(\mathbf{v}_c)$, where Z is the *partition function* — a normalization constant given by $Z = \sum_{\mathbf{v}'} \prod \phi_c(\mathbf{v}'_c)$. ∎

Each potential ϕ_c is simply a table of values for each assignment $\mathbf{v}_c$ that defines a "compatibility" between values of variables in the clique. The potential is often represented by a log-linear combination of a small set of *features*:

$$\phi_c(\mathbf{v}_c) = \exp\{\sum_i w_i f_i(\mathbf{v}_c)\} = \exp\{\mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{v}_c)\} \ .$$

The simplest and most common form of a feature is the indicator function $f(\mathbf{V}_c) \equiv \delta(\mathbf{V}_c = \mathbf{v}_c)$. However, features can be arbitrary logical predicates of the variables of the clique, $\mathbf{V}_c$. For example, if the variables are binary, a feature might signify the parity or whether the variables are all the same value. More generally, the features can be real-valued functions, not just binary predicates. See further discussion of features at the end of section 6.4.

We will abbreviate log-linear representation as follows:

$$\log P(\mathbf{v}) = \sum_c \mathbf{w}_c \cdot \mathbf{f}_c(\mathbf{v}_c) - \log Z = \mathbf{w} \cdot \mathbf{f}(\mathbf{v}) - \log Z;$$

where $\mathbf{w}$ and $\mathbf{f}$ are the vectors of all weights and features.

For classification, we are interested in constructing discriminative models using *conditional Markov nets* which are simply Markov networks renormalized to model a conditional distribution.

Definition 6.2
Let $\mathbf{X}$ be a set of random variables on which we condition and $\mathbf{Y}$ be a set of target (or label) random variables. A *conditional Markov network* is a Markov network (G, Φ) which defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$, where $Z(\mathbf{x})$ is the partition function, now dependent on $\mathbf{x}$: $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod \phi_c(\mathbf{x}_c, \mathbf{y}'_c)$. ∎

Logistic regression, a well-studied statistical model for classification, can be viewed as the simplest example of a conditional Markov network. In standard form, for $Y = \pm 1$ and $\mathbf{X} \in \{0,1\}^n$ (or $\mathbf{X} \in \Re^n$), $P(y \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{y\mathbf{w} \cdot \mathbf{x}\}$. Viewing the model as a Markov network, the cliques are simply the edges $c_k = \{X_k, Y\}$ with potentials $\phi_k(x_k, y) = \exp\{y w_k x_k\}$. In this example, each feature is of the form $f_k(x_k, y) = y x_k$.

6.4.2 Relational Markov Networks

We now extend the framework of Markov networks to the relational setting. A *relational Markov network* specifies a conditional distribution over all of the labels

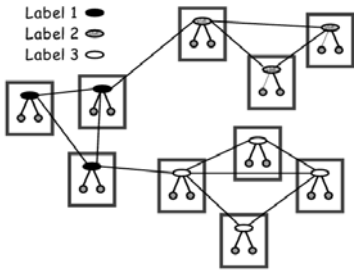


Figure 6.1 An unrolled Markov net over linked documents. The links follow a common pattern: documents with the same label tend to link to each other more often.

of all of the entities in an instantiation given the relational structure and the content attributes. (We provide the definitions directly for the conditional case, as the unconditional case is a special case where the set of content attributes is empty.) Roughly speaking, it specifies the cliques and potentials between attributes of related entities at a template level, so a single model provides a coherent distribution for any collection of instances from the schema.

For example, suppose that pages with the same label tend to link to each other, as in figure 6.1. We can capture this correlation between labels by introducing, for each link, a clique between the labels of the source and the target page. The potential on the clique will have higher values for assignments that give a common label to the linked pages.

To specify what cliques should be constructed in an instantiation, we will define a notion of a *relational clique template*. A relational clique template specifies tuples of variables in the instantiation by using a relational query language. For our link example, we can write the template as a kind of SQL query:

```
SELECT doc1.Category, doc2.Category
FROM Doc doc1, Doc doc2, Link link
WHERE link.From = doc1.Key and link.To = doc2.Key
```

Note the three clauses that define a query: the FROM clause specifies the cross product of entities to be filtered by the WHERE clause and the SELECT clause picks out the attributes of interest. Our definition of clique templates contains the corresponding three parts.

Definition 6.3
A *relational clique template* $C = (\mathbf{F}, \mathbf{W}, \mathbf{S})$ consists of three components:

- $\mathbf{F} = \{F_i\}$ — a set of entity variables, where an entity variable F_i is of type $E(F_i)$.
- $\mathbf{W}(\mathbf{F}.\mathbf{R})$ — a Boolean formula using conditions of the form $F_i.R_j = F_k.R_l$.
- $\mathbf{F}.\mathbf{S} \subseteq \mathbf{F}.\mathbf{X} \cup \mathbf{F}.\mathbf{Y}$ — a selected subset of content and label attributes in $\mathbf{F}$. ∎

For the clique template corresponding to the SQL query above, $\mathbf{F}$ consists of *doc1*, *doc2*, and *link* of types Doc, Doc, and Link, respectively. $\mathbf{W}(\mathbf{F}.\mathbf{R})$ is $link.From = doc1.Key \wedge link.To = doc2.Key$ and $\mathbf{F}.\mathbf{S}$ is $doc1.Category$ and $doc2.Category$.

A clique template specifies a set of cliques in an instantiation $\mathcal{I}$:

$$C(\mathcal{I}) \equiv \{c = \mathbf{f}.\mathbf{S} : \mathbf{f} \in \mathcal{I}(\mathbf{F}) \wedge \mathbf{W}(\mathbf{f}.\mathbf{r})\},$$

where $\mathbf{f}$ is a tuple of entities $\{f_i\}$ in which each f_i is of type $E(F_i)$; $\mathcal{I}(\mathbf{F}) = \mathcal{I}(E(F_1)) \times \ldots \times \mathcal{I}(E(F_n))$ denotes the cross product of entities in the instantiation; the clause $\mathbf{W}(\mathbf{f}.\mathbf{r})$ ensures that the entities are related to each other in specified ways; and finally, $\mathbf{f}.\mathbf{S}$ selects the appropriate attributes of the entities. Note that the clique template does not specify the nature of the interaction between the attributes; that is determined by the clique potentials, which will be associated with the template.

This definition of a clique template is very flexible, as the WHERE clause of a template can be an arbitrary predicate. It allows modeling complex relational patterns on the instantiation graphs. To continue our webpage example, consider another common pattern in hypertext: links in a webpage tend to point to pages of the same category. This pattern can be expressed by the following template:

```
SELECT doc1.Category, doc2.Category
FROM Doc doc1, Doc doc2, Link link1, Link link2
WHERE link1.From = link2.From and link1.To = doc1.Key
and link2.To = doc2.Key and not doc1.Key = doc2.Key
```

Depending on the expressive power of our template definition language, we may be able to construct very complex templates that select entire subgraph structures of an instantiation. We can easily represent patterns involving three (or more) interconnected documents without worrying about the acyclicity constraint imposed by directed models. Since the clique templates do not explicitly depend on the identities of entities, the same template can select subgraphs whose structure is fairly different. The RMN allows us to associate the same clique potential parameters with all of the subgraphs satisfying the template, thereby allowing generalization over a wide range of different structures.

Definition 6.4

A *relational Markov network* $\mathcal{M} = (\mathbf{C}, \Phi)$ specifies a set of clique templates $\mathbf{C}$ and corresponding potentials $\Phi = \{\phi_C\}_{C \in \mathbf{C}}$ to define a conditional distribution:

$$P(\mathcal{I}.\mathbf{y} \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) = \frac{1}{Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c),$$

where $Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})$ is the normalizing partition function:

$$Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) = \sum_{\mathcal{I}.\mathbf{y}'} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}'_c). \quad \blacksquare$$

Using the log-linear representation of potentials, $\phi_C(\mathbf{V}_C) = \exp\{\mathbf{w}_C \cdot \mathbf{f}_C(\mathbf{V}_C)\}$, we can write

$$\begin{aligned}\log P(\mathcal{I}.\mathbf{y} \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) &= \sum_{C \in \mathbf{C}} \sum_{c \in C(\mathcal{I})} \mathbf{w}_C \cdot \mathbf{f}_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c) - \log Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \\ &= \sum_{C \in \mathbf{C}} \mathbf{w}_C \cdot \mathbf{f}_C(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{r}) - \log Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \\ &= \mathbf{w} \cdot \mathbf{f}(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{r}) - \log Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}),\end{aligned}$$

where

$$\mathbf{f}_C(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{r}) = \sum_{c \in C(\mathcal{I})} \mathbf{f}_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c)$$

is the sum over all appearances of the template $C(\mathcal{I})$ in the instantiation, and $\mathbf{f}$ is the vector of all $\mathbf{f}_C$.

Given a particular instantiation $\mathcal{I}$ of the schema, the RMN $\mathcal{M}$ produces an *unrolled* Markov network over the attributes of entities in $\mathcal{I}$. The cliques in the unrolled network are determined by the clique templates C. We have one clique for each $c \in C(\mathcal{I})$, and all of these cliques are associated with the same clique potential ϕ_C. In our webpage example, an RMN with the link feature described above would define a Markov net in which, for every link between two pages, there is an edge between the labels of these pages. Figure 6.1 illustrates a simple instance of this unrolled Markov network.

Note that we leave the clique potentials to be specified using arbitrary sets of feature functions. A common set is the complete table of indicator functions, one for each instantiation of the discrete-valued variables in the clique. However, this results in a large number of parameters (exponential in the number of variables). Often, as we encounter in our experiments, only a subset of the instantiations is of interest or many instantiations are essentially equivalent because of symmetries. For example, in an edge potential between labels of two webpages linked from a given page, we might want to have a single feature tracking whether the two labels are the same. In the case of triad cliques enforcing transitivity, we might constrain features to be symmetric functions with respect to the variables. In the presence of continuous-valued variables, features are often a predicate on the discrete variables multiplied by a continuous value. We do not prescribe a language for specifying features (as does Markov logic; see chapter 11), although in our implementation, we use a combination of logical formulae and custom-designed functions.

6.5 Learning the Models

We focus here on the case where the clique templates are given; our task is to estimate the clique potentials, or feature weights. Thus, assume that we are given a set of clique templates $\mathbf{C}$ which partially specify our (relational) Markov network,

and our task is to compute the weights $\mathbf{w}$ for the potentials Φ. In the learning task, we are given some training set D where both the content attributes and the labels are observed. Any particular setting for $\mathbf{w}$ fully specifies a probability distribution $P_{\mathbf{w}}$ over D, so we can use the *likelihood* as our objective function, and attempt to find the weight setting that maximizes the likelihood (ML) of the labels given other attributes. However, to help avoid overfitting, we assume a prior over the weights (a zero-mean Gaussian), and use maximum a posteriori (MAP) estimation. More precisely, we assume that different parameters are a priori independent and define $p(w_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-w_i^2/2\sigma^2\right\}$. Both the ML and MAP objective functions are concave and there are many methods available for maximizing them. Our experience is that conjugate gradient performs fairly well for logistic regression and relational Markov nets. However, recent experience with conditional random fields (CRFs) suggests the L-BFGS method might be somewhat faster [18].

6.5.1 Learning Markov Networks

We first consider discriminative MAP training in the flat setting. In this case D is simply a set of i.i.d. instances; let d index over all labeled training data D. The discriminative likelihood of the data is $\prod_d P_{\mathbf{w}}(y_d \mid \mathbf{x}_d)$. We introduce the parameter prior, and maximize the log of the resulting MAP objective function:

$$\mathcal{L}(\mathbf{w}, D) = \sum_{d \in D} \left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_d, y_d) - \log Z(\mathbf{x}_d)\right) - \frac{||\mathbf{w}||_2^2}{2\sigma^2} + C \ .$$

The gradient of the objective function is computed as

$$\nabla \mathcal{L}(\mathbf{w}, D) = \sum_{d \in D} \left(\mathbf{f}(\mathbf{x}_d, y_d) - \boldsymbol{E}_{P_{\mathbf{w}}}[\mathbf{f}(\mathbf{x}_d, Y_d)]\right) - \frac{\mathbf{w}}{\sigma^2} \ .$$

The last term is the shrinking effect of the prior and the other two terms are the difference between the expected feature counts and the empirical feature counts, where the expectation is taken relative to $P_{\mathbf{w}}$:

$$\boldsymbol{E}_{P_{\mathbf{w}}}[\mathbf{f}(\mathbf{x}_d, Y_d)] = \sum_{y'} \mathbf{f}(\mathbf{x}_d, y'_d) P_{\mathbf{w}}(y'_d \mid \mathbf{x}_d) \ .$$

Thus, ignoring the effect of the prior, the gradient is zero when empirical and expected feature counts are equal.[2] The prior term gives the smoothing we expect from the prior: small weights are preferred in order to reduce overfitting. Note that the sum over y' is just over the possible categorizations for one data sample every time.

2. The solution of ML estimation with log-linear models is also the solution to the dual problem of maximum entropy estimation with constraints that empirical and expected feature counts must be equal [4].

6.5.2 Learning RMNs

The analysis for the relational setting is very similar. Now, our data set D is actually a single instantiation $\mathcal{I}$, where the same parameters are used multiple times — once for each different entity that uses a feature. A particular choice of parameters $\mathbf{w}$ specifies a particular RMN, which induces a probability distribution $P_{\mathbf{w}}$ over the unrolled Markov network. The product of the likelihood of $\mathcal{I}$ and the parameter prior define our objective function, whose gradient $\nabla\mathcal{L}(\mathbf{w}, \mathcal{I})$ again consists of the empirical feature counts minus the expected feature counts and a smoothing term due to the prior:

$$\mathbf{f}(\mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) - \boldsymbol{E}_{\mathbf{w}}[\mathbf{f}(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})] - \frac{\mathbf{w}}{\sigma^2},$$

where the expectation $E_{P_{\mathbf{w}}}[\mathbf{f}(\mathcal{I}.\mathbf{Y}, \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})]$ is

$$\sum_{\mathcal{I}.\mathbf{y}'} \mathbf{f}(\mathcal{I}.\mathbf{y}', \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) P_{\mathbf{w}}(\mathcal{I}.\mathbf{y}' \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \ .$$

This last formula reveals a key difference between the relational and the flat case: the sum over $\mathcal{I}.\mathbf{y}'$ involves the exponential number of assignments to all the label attributes in the instantiation. In the flat case, the probability decomposes as a product of probabilities for individual data instances, so we can compute the expected feature count for each instance separately. In the relational case, these labels are correlated — indeed, this correlation was our main goal in defining this model. Hence, we need to compute the expectation over the joint assignments to all the entities together. Computing these expectations over an exponentially large set is the expensive step in calculating the gradient. It requires that we run inference on the unrolled Markov network.

6.5.3 Inference in Markov Networks

The inference task in our conditional Markov networks is to compute the posterior distribution over the label variables in the instantiation given the content variables. Exact algorithms for inference in graphical models can execute this process efficiently for specific graph topologies such as sequences, trees, and other low treewidth graphs. However, the networks resulting from domains such as our hypertext classification task are very large (in our experiments, they contain tens of thousands of nodes) and densely connected. Exact inference is completely intractable in these cases.

We therefore resort to approximate inference. There is a wide variety of approximation schemes for Markov networks, including sampling and variational methods. We chose to use *belief propagation*(BP) for its simplicity and relative efficiency and accuracy. BP is a local message passing algorithm introduced by Pearl [17] and later related to turbo-coding by McEliece et al. [14]. It is guaranteed to converge to the correct marginal probabilities for each node only for singly connected Markov

networks. Empirical results [15] show that it often converges in general networks, and when it does, the marginals are a good approximation to the correct posteriors. As our results in section 6.6 show, this approach works well in our domain. We refer the reader to chapter 2 in this book for a detailed description of the BP algorithm.

6.6 Experimental Results

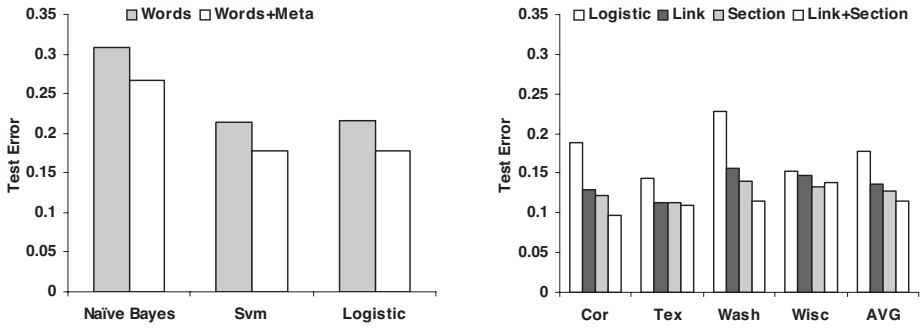
We present experiments with collective classification and link prediction, in both hypertext and social network data.

6.6.1 Experiments on WebKB

We experimented with our framework on the *WebKB* data set [3], which is an instance of our hypertext example. The data set contains webpages from four different computer science departments: Cornell, Texas, Washington, and Wisconsin. Each page has a label attribute, representing the type of webpage which is one of *course, faculty, student, project,* or *other*. The data set is problematic in that the category *other* is a grab bag of pages of many different types. The number of pages classified as *other* is quite large, so that a baseline algorithm that simply always selected *other* as the label would get an average accuracy of 75%. We could restrict attention to just the pages with the four other labels, but in a relational classification setting, the deleted webpages might be useful in terms of their interactions with other webpages. Hence, we compromised by eliminating all *other* pages with fewer than three outlinks, making the number of *other* pages commensurate with the other categories.[3] For each page, we have access to the entire HTML of the page and the links to other pages. Our goal is to collectively classify webpages into one of these five categories. In all of our experiments, we learn a model from three schools and test the performance of the learned model on the remaining school, thus evaluating the generalization performance of the different models.

Unfortunately, we cannot directly compare our accuracy results with previous work because different papers use different subsets of the data and different training/test splits. However, we compare to standard text classifiers such as naive Bayes, logistic regression, and support vector machines, which have been demonstrated to be successful on this data set [9].

3. The resulting category distribution is: course (237), faculty (148), other (332), research-project (82), and student (542). The number of remaining pages for each school are: Cornell (280), Texas (292), Washington (315), and Wisconsin (454). The number of links for each school are: Cornell (574), Texas (574), Washington (728) and Wisconsin (1614).



(a) (b)

Figure 6.2 (a) Comparison of Naive Bayes, Svm, and Logistic on WebKB, with and without metadata features. (Only averages over the four schools are shown here.) (b) Flat versus collective classification on WebKB: flat logistic regression with metadata, and three different relational models: Link, Section, and a combined Section+Link. Collectively classifying page labels (Link, Section, Section+Link) consistently reduces the error over the flat model (logistic regression) on all schools, for all three relational models.

6.6.1.1 Flat Models

The simplest approach we tried predicts the categories based on just the text content on the webpage. The text of the webpage is represented using a set of binary attributes that indicate the presence of different words on the page. We found that stemming and feature selection did not provide much benefit and simply pruned words that appeared in fewer than three documents in each of the three schools in the training data. We also experimented with incorporating metadata: words appearing in the title of the page, in anchors of links to the page, and in the last header before a link to the page [24]. Note that metadata, although mostly originating from pages linking into the considered page, are easily incorporated as features, i.e., the resulting classification task is still flat feature-based classification. Our first experimental setup compares three well-known text classifiers — Naive Bayes, linear support vector machines [4] (Svm), and logistic regression (Logistic) — using words and metawords. The results, shown in figure 6.2(a), show that the two discriminative approaches outperform Naive Bayes. Logistic and Svm give very similar results. The average error over the four schools was reduced by around 4% by introducing the metadata attributes.

4. We trained one-against-others SVM for each category and during testing, picked the category with the largest margin.

6.6.1.2 Relational Models

Incorporating metadata gives a significant improvement, but we can take additional advantage of the correlation in labels of related pages by classifying them collectively. We want to capture these correlations in our model and use them for transmitting information between linked pages to provide more accurate classification. We experimented with several relational models. Recall that logistic regression is simply a flat conditional Markov network. All of our relational Markov networks use a logistic regression model locally for each page.

Our first model captures direct correlations between labels of linked pages. These correlations are very common in our data: courses and research projects almost never link to each other; faculty rarely link to each other; students have links to all categories but mostly to courses. The Link model, shown in figure 6.1, captures this correlation through links: in addition to the local bag of words and metadata attributes, we introduce a relational clique template over the labels of two pages that are linked.

A second relational model uses the insight that a webpage often has internal structure that allows it to be broken up into *sections*. For example, a faculty webpage might have one section that discusses research, with a list of links to all of the projects of the faculty member, a second section might contain links to the courses taught by the faculty member, and a third to his advisees. This pattern is illustrated in figure 6.3. We can view a section of a webpage as a fine-grained version of Kleinberg's hub [10] (a page that contains a lot of links to pages of a particular category). Intuitively, if we have links to two pages in the same section, they are likely to be on similar topics. To take advantage of this trend, we need to enrich our schema with a new relation Section, with attributes *Key*, *Doc* (the document in which it appears), and *Category*. We also need to add the attribute *Section* to Link to refer to the section it appears in. In the RMN, we have two new relational clique templates. The first contains the label of a section and the label of the page it is on:

```
SELECT doc.Category, sec.Category
FROM Doc doc, Section sec
WHERE sec.Doc = doc.Key
```

The second clique template involves the label of the section containing the link and the label of the target page.

```
SELECT sec.Category, doc.Category
FROM Section sec, Link link, Doc doc
WHERE link.Sec = sec.Key and link.To = doc.Key
```

The original data set did not contain section labels, so we introduced them using the following simple procedure. We defined a section as a sequence of three or more links that have the same path to the root in the HTML parse tree. In the training set, a section is labeled with the most frequent category of its links. There is a sixth

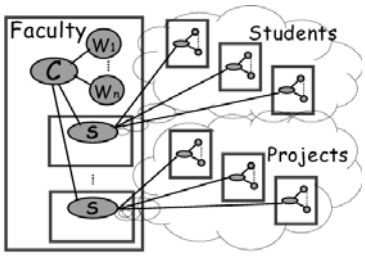


Figure 6.3 An illustration of the Section model.

category, *none*, assigned when the two most frequent categories of the links are less than a factor of 2 apart. In the entire data set, the breakdown of labels for the sections we found is: *course* (40), *faculty* (24), *other* (187), *research.project* (11), *student* (71), and *none* (17). Note that these labels are hidden in the test data, so the learning algorithm now also has to learn to predict section labels. Although not our final aim, correct prediction of section labels is very helpful. Words appearing in the last header before the section are used to better predict the section label by introducing a clique over these words and section labels.

We compared the performance of Link, Section, and Section+Link (a combined model which uses both types of cliques) on the task of predicting webpage labels, relative to the baseline of flat logistic regression with metadata. Our experiments used MAP estimation with a Gaussian prior on the feature weights with standard deviation of 0.3. Figure 6.2(b) compares the average error achieved by the different models on the four schools, training on three and testing on the fourth. We see that incorporating any type of relational information consistently gives significant improvement over the baseline model. The Link model incorporates more relational interactions, but each is a weaker indicator. The Section model ignores links outside of coherent sections, but each of the links it includes is a very strong indicator. In general, we see that the Section model performs slightly better. The joint model is able to combine benefits from both and generally outperforms all of the other models. The only exception is for the task of classifying the Wisconsin data. In this case, the joint Section+Link model contains many links, as well as some large tightly connected loops, so belief propagation did not converge for a subset of nodes. Hence, the results of the inference, which was stopped at a fixed arbitrary number of iterations, were highly variable and resulted in lower accuracy.

6.6.1.3 Discriminative vs. Generative

Our last experiment illustrates the benefits of discriminative training in relational classification. We compared three models. The Exists+Naive Bayes model is a completely generative model proposed by Getoor et al. [8]. At each page, a naive Bayes model generates the words on a page given the page label. A separate generative model specifies a probability over the existence of links between pages conditioned

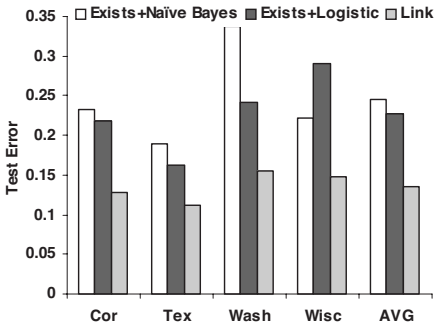


Figure 6.4 Comparison of generative and discriminative relational models. Exists+Naive Bayes is completely generative. Exists+Logistic is generative in the links, but locally discriminative in the page labels given the local features (words, meta-words). The Link model is completely discriminative.

on both pages' labels. We can also consider an alternative Exists+Logistic model that uses a discriminative model for the connection between page label and words — i.e., uses logistic regression for the conditional probability distribution of page label given words. This model has equivalent expressive power to the naive Bayes model but is discriminatively rather than generatively trained. Finally, the Link model is a fully discriminative (undirected) variant we have presented earlier, which uses a discriminative model for the label given both words and link existence. The results, shown in figure 6.4, show that discriminative training provides a significant improvement in accuracy: the Link model outperforms Exists+Logistic which in turn outperforms Exists+Naive Bayes.

As illustrated in table 6.1, the gain in accuracy comes at some cost in training time: for the generative models, parameter estimation is closed form while the discriminative models are trained using conjugate gradient, where each iteration requires inference over the unrolled RMN. On the other hand, both types of models require inference when the model is used on new data; the generative model constructs a much larger, fully connected network, resulting in significantly longer testing times. We also note that the situation changes if some of the data is unobserved in the training set. In this case, generative training also requires an iterative procedure (such as the expectation macimation algorihtm (EM)) where each iteration uses the significantly more expressive inference.

6.6.2 Experiments on extended WebKB

We collected and manually labeled a new relational data set inspired by WebKB [3]. Our data set consists of computer science department webpages from three schools: Stanford, Berkeley, and MIT. A total of 2954 pages are labeled into one of eight categories: faculty, student, research scientist, staff, research group, research project,

Table 6.1 Average train/test running times (seconds). All runs were done on a 700Mhz Pentium III. Training times are averaged over four runs on three schools each. Testing times are averaged over four runs on one school each.

	Links	Links+Section	Exists+NB
Training	1530	6060	1
Testing	7	10	100

course, and organization (organization refers to any large entity that is not a research group). *Owned pages*, which are owned by an entity but are not the main page for that entity, were manually assigned to that entity. The average distribution of classes across schools is: organization (9%), student (40%), research group (8%), faculty (11%), course (16%), research project (7%), research scientist (5%), and staff (3%).

We established a set of candidate links between entities based on evidence of a relation between them. One type of evidence for a relation is a hyperlink from an entity page or one of its owned pages to the page of another entity. A second type of evidence is a *virtual link*: We assigned a number of aliases to each page using the page title, the anchor text of incoming links, and email addresses of the entity involved. Mentioning an alias of a page on another page constitutes a virtual link. The resulting set of 7161 candidate links were labeled as corresponding to one of five relation types — advisor (faculty, student), member (research group/project, student/faculty/research scientist), teach (faculty/research scientist/staff, course), TA (student, course), part-of (research group, research project) — or "none," denoting that the link does not correspond to any of these relations.

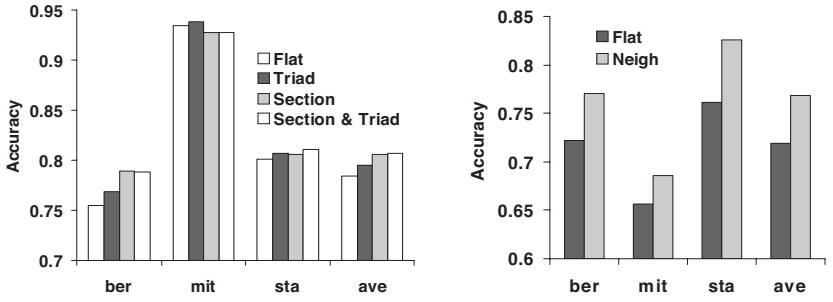
The observed attributes for each page are the words on the page itself and the "metawords" on the page — the words in the title, section headings, anchors to the page from other pages. For links, the observed attributes are the anchor text, text just before the link (hyperlink or virtual link), and the heading of the section in which the link appears.

Our task is to predict the relation type, if any, for all the candidate links. We tried two settings for our experiments: with page categories observed (in the test data) and page categories unobserved. For all our experiments, we trained on two schools and tested on the remaining school.

Observed entity labels We first present results for the setting with observed page categories. Given the page labels, we can rule out many impossible relations; the resulting label breakdown among the candidate links is: none (38%), member (34%), part-of (4%), advisor (11%), teach (9%), TA (5%).

There is a huge range of possible models that one can apply to this task. We selected a set of models that we felt represented some range of patterns that manifested in the data.

Link-Flat is our baseline model, predicting links one at a time using multinomial logistic regression. This is a strong classifier, and its performance is competitive



(a) (b)

Figure 6.5 (a) Relation prediction with entity labels given. Relational models on average performed better than the baseline Flat model. (b) Entity label prediction. Relational model Neigh performed significantly better.

with other classifiers (e.g., support vector machines). The features used by this model are the labels of the two linked pages and the words on the links going from one page and its owned pages to the other page. The number of features is around 1000.

The relational models try to improve upon the baseline model by modeling the interactions between relations and predicting relations jointly. The Section model introduces cliques over relations whose links appear consecutively in a section on a page. This model tries to capture the pattern that similarly related entities (e.g., advisees, members of projects) are often listed together on a webpage. This pattern is a type of similarity template, as described in section 6.3. The Triad model is a type of transitivity template, as discussed in section 6.3. Specifically, we introduce cliques over sets of three candidate links that form a triangle in the link graph. The Section & Triad model includes the cliques of the two models above.

As shown in figure 6.2(a), both the Section and Triad models outperform the flat model, and the combined model has an average accuracy gain of 2.26%, or 10.5% relative reduction in error. As we only have three runs (one for each school), we cannot meaningfully analyze the statistical significance of this improvement.

As an example of the interesting inferences made by the models, we found a student-professor pair that was misclassified by the Flat model as none (there is only a single hyperlink from the student's page to the advisor's) but correctly identified by both the Section and Triad models. The Section model utilizes a paragraph on the student's webpage describing his or her research, with a section of links to research groups and the link to his or her advisor. Examining the parameters of the Section model clique, we found that the model learned that it is likely for people to mention their research groups and advisors in the same section. By capturing this trend, the Section model is able to increase the confidence of the student-advisor relation. The Triad model corrects the same misclassification in a different way. Using the same example, the Triad model makes use of the information that both the student and

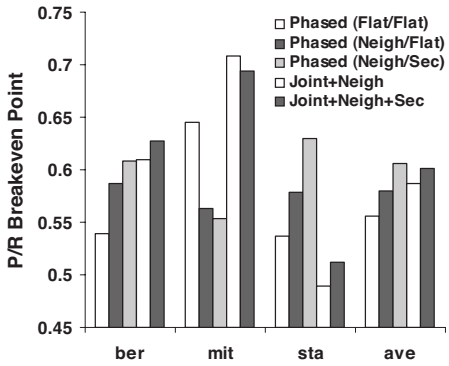
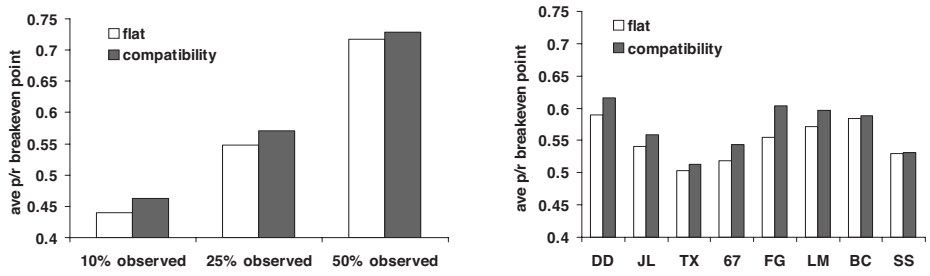


Figure 6.6 Relation prediction without entity labels. Relational models performed better most of the time, even though there are schools in which some models performed worse.

the teacher belong to the same research group, and the student TAed a class taught by his advisor. It is important to note that none of the other relations are observed in the test data, but rather the model bootstraps its inferences.

Unobserved entity labels When the labels of pages are not known during relations prediction, we cannot rule out possible relations for candidate links based on the labels of participating entities. Thus, we have many more candidate links that do not correspond to any of our relation types (e.g., links between an organization and a student). This makes the existence of relations a very low-probability event, with the following breakdown among the potential relations: none (71%), member (16%), part-of (2%), advisor (5%), teach (4%), TA (2%). In addition, when we construct a Markov network in which page labels are not observed, the network is much larger and denser, making the (approximate) inference task much harder. Thus, in addition to models that try to predict page entity and relation labels simultaneously, we also tried a two-phase approach, where we first predict page categories, and then use the predicted labels as features for the model that predicts relations.

For predicting page categories, we compared two models. The Entity-Flat model is a multinomial logistic regression that uses words and "metawords" from the page and its owned pages in separate "bags" of words. The number of features is roughly $10,000$. The Neighbors model is a relational model that exploits another type of similarity template: pages with similar URLs often belong to the same category or tightly linked categories (research group/project, professor/course). For each page, two pages with URLs closest in edit distance are selected as "neighbors," and we introduced pairwise cliques between "neighboring" pages. Figure 6.5(b) shows that the Neighbors model clearly outperforms the Flat model across all schools, by an average of 4.9% accuracy gain.



(a) (b)

Figure 6.7 (a) Average precision-recall breakeven point for 10%, 25%, 50% observed links. (b) Average precision-recall breakeven point for each fold of school residences at 25% observed links.

Given the page categories, we can now apply the different models for link classification. Thus, the Phased (Flat/Flat) model uses the Entity-Flat model to classify the page labels, and then the Link-Flat model to classify the candidate links using the resulting entity labels. The Phased (Neighbors/Flat) model uses the Neighbors model to classify the entity labels, and then the Link-Flat model to classify the links. The Phased (Neighbors/Section) model uses the Neighbors to classify the entity labels and then the Section model to classify the links.

We also tried two models that predict page and relation labels simultaneously. The Joint + Neighbors model is simply the union of the Neighbors model for page categories and the Flat model for relation labels given the page categories. The Joint + Neighbors + Section model additionally introduces the cliques that appeared in the Section model between links that appear consecutively in a section on a page. We train the joint models to predict both page and relation labels simultaneously.

As the proportion of the "none" relation is so large, we use the probability of "none" to define a precision-recall curve. If this probability is less than some threshold, we predict the most likely label (other than none); otherwise we predict the most likely label (including none). As usual, we report results at the precision-recall breakeven point on the test data. Figure 6.6 shows the breakeven points achieved by the different models on the three schools. Relational models, both phased and joint, did better than flat models on the average. However, performance varies from school to school and for both joint and phased models, performance on one of the schools is worse than that of the flat model.

6.6.3 Social Network Data

The data set we used has been collected by a portal website at a large university that hosts an online community for students [1]. Among other services, it allows students to enter information about themselves, create lists of their friends, and browse the social network. Personal information includes residence, gender, major, and year, as well as favorite sports, music, books, social activities, etc. We focused on the task of predicting the "friendship" links between students from their personal information

and a subset of their links. We selected students living in sixteen different residences or dorms and restricted the data to the friendship links only within each residence, eliminating interresidence links from the data to generate independent training/test splits. Each residence has about fifteen to twenty-five students and an average student lists about 25% of his or her housemates as friends.

We used an eight-fold train-test split, where we trained on fourteen residences and tested on two. Predicting links between two students from just personal information alone is a very difficult task, so we tried a more realistic setting, where some proportion of the links is observed in the test data, and can be used as evidence for predicting the remaining links. We used the following proportions of observed links in the test data: 10%, 25%, and 50%. The observed links were selected at random, and the results we report are averaged over five folds of these random selection trials.

Using just the observed portion of links, we constructed the following flat features: for each student, the proportion of students in the residence that list him/her and the proportion of students he/she lists; for each pair of students, the proportion of other students they have as common friends. The values of the proportions were discretized into four bins. These features capture some of the relational structure and dependencies between links: Students who list (or are listed by) many friends in the observed portion of the links tend to have links in the unobserved portion as well. More importantly, having friends in common increases the likelihood of a link between a pair of students.

The Flat model uses logistic regression with the above features as well as personal information about each user. In addition to the individual characteristics of the two people, we also introduced a feature for each match of a characteristic; for example, both people are computer science majors or both are freshmen.

The Compatibility model uses a type of similarity template, introducing cliques between each pair of links emanating from each person. Similarly to the Flat model, these cliques include a feature for each match of the characteristics of the two potential friends. This model captures the tendency of a person to have friends who share many characteristics (even though the person might not possess them). For example, a student may be friends with several computer science majors, even though he is not a CS major himself. We also tried models that used transitivity templates, but the approximate inference with 3-cliques often failed to converge or produced erratic results.

Figure 6.7(a) compares the average precision-recall breakpoint achieved by the different models at the three different settings of observed links. Figure 6.7(b) shows the performance on each of the eight folds containing two residences each. Using a paired t-test, the Compatibility model outperforms Flat with p-values 0.0036, 0.00064, and 0.054 respectively.

6.7 Discussion and Conclusions

We propose an approach for collective classification and link prediction in relational domains. Our approach provides a coherent probabilistic foundation for the process of collective prediction, where we want to classify multiple entities and links, exploiting the interactions between the variables. We have shown that we can exploit a very rich set of relational patterns in classification, significantly improving the classification accuracy over standard flat classification.

We show that the use of a probabilistic model over link graphs allows us to represent and exploit interesting subgraph patterns in the link graph. Specifically, we have found two types of patterns that seem to be beneficial in several places. Similarity templates relate the classification of links or objects that share a certain graph-based property (e.g., links that share a common endpoint). Transitivity templates relate triples of objects and links organized in a triangle.

Our results use a set of relational patterns that we have discovered to be useful in the domains that we have considered. However, many other rich and interesting patterns are possible. Thus, in the relational setting, even more so than in simpler tasks, the issue of feature construction is critical. It is therefore important to explore the problem of automatic feature induction, as in [4].

Finally, we believe that the problem of modeling link graphs has numerous other applications, including analyzing communities of people and the hierarchical structure of organizations, identifying people or objects that play certain key roles, predicting current and future interactions, and more.

References

[1] L. Adamic, O. Buyukkokten, and E. Adar. A social network caught in the web. http://www.hpl.hp.com/shl/papers/social/, 2002.

[2] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM International Conference on Management of Data*, 1998.

[3] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the National Conference on Artificial Intelligence*, 1998.

[4] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (4):380–393, 1997.

[5] L. Egghe and R. Rousseau. *Introduction to Informetrics*. Elsevier, Amsterdam, 1990.

[6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the International Joint Conference on*

Artificial Intelligence, 1999.

[7] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proceedings of the International Conference on Machine Learning*, 2001.

[8] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *Proceedings of the IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

[9] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, 1999.

[10] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[11] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the National Conference on Artificial Intelligence*, 1998.

[12] F. Kschischang and B. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal of Selected Areas in Communications*, 16(2):219–230, 1998.

[13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[14] R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

[15] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1999.

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

[17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[18] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, 2003.

[19] S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *Proceedings of the International Conference on Machine Learning*, 2000.

[20] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[21] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002.

[22] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of Neural Information Processing Systems*, 2003.

[23] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[24] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2):219–241, 2002.