

---

# Learning annotated hierarchies from relational data

---

Daniel M. Roy,  
CSAIL, MIT  
droy@mit.edu

Charles Kemp, Vikash K. Mansinghka, and Joshua B. Tenenbaum  
Dept. of Brain & Cognitive Sciences, MIT, Cambridge, MA 02139  
{ckemp, vkm, jbt}@mit.edu

## Abstract

The objects in many real-world domains can be organized into hierarchies, where each internal node of a hierarchy picks out a category of objects. Given a collection of features and relations defined over a set of objects, an *annotated* hierarchy includes a specification of the categories that are most useful for describing each individual feature and relation. We define a generative model for annotated hierarchies and the features and relations that they describe, and develop a Markov chain Monte Carlo scheme for learning annotated hierarchies. We show that our model discovers interpretable structure in several real-world data sets.

## 1 Introduction

Researchers in AI and cognitive science [1, 7] have proposed that hierarchies are useful for representing and reasoning about the objects in many real-world domains. Each node in a hierarchy represents a category that includes all of its descendants, and one of the reasons that hierarchies are valuable is that they compactly specify categories at many levels of resolution. Consider, for example, the simple hierarchy shown in Figure 1a, which picks out five categories relevant to a typical university department: employees, staff, faculty, professors, and assistant professors.

Suppose that we are given a large data set describing the features of these employees and the interactions among these employees. Each of the five categories will account for some aspects of the data, but different categories will be needed for understanding different features and relations. “Faculty,” for example, is the single most useful category for describing the employees that publish papers (Figure 1b), but three categories may be needed to describe the social interactions among the employees (Figure 1c). In order to understand the structure of the department, it is important not only to understand the hierarchical organization of the employees, but to understand which levels of this hierarchy are appropriate for describing each feature and each relation. Suppose, then, that an *annotated hierarchy* is a hierarchy along with a specification of the levels in the hierarchy that are relevant to each feature and relation.

The idea of an annotated hierarchy is one of the oldest proposals in cognitive science, and researchers including Collins and Quillian [1] and Keil [7] have argued that semantic knowledge is organized into representations with this form. Previous treatments of annotated hierarchies, however, usually suffer from two limitations. First, annotated hierarchies are usually hand-engineered, and there are few proposals describing how they might be learned from data. Second, annotated hierarchies typically capture knowledge only about the features of objects: relations between objects are rarely considered. We address both problems by defining a generative model for objects, features, relations, and hierarchies, and showing how it can be used to recover an annotated hierarchy from raw data.

Our generative model for feature data assumes that the objects are located at the leaves of a rooted tree, and that each feature is generated from a partition of the objects “consistent” with the tree. A *tree-consistent partition* of the objects with respect to a rooted tree is a partition of the objects into disjoint classes such that the objects in each class corresponds exactly with the leaves descending from some node in the tree (Figure 1a,b). Therefore, a tree-consistent partition can be encoded uniquely as the set of nodes whose leaf descendants comprise the classes. The simplest tree-consistent partition is the singleton set containing the root node, which places all objects into a single class. The most complex tree-consistent partition is the set of all leaves, which assigns each object to its own class. Given the tree-consistent partition for a particular feature, its value for one

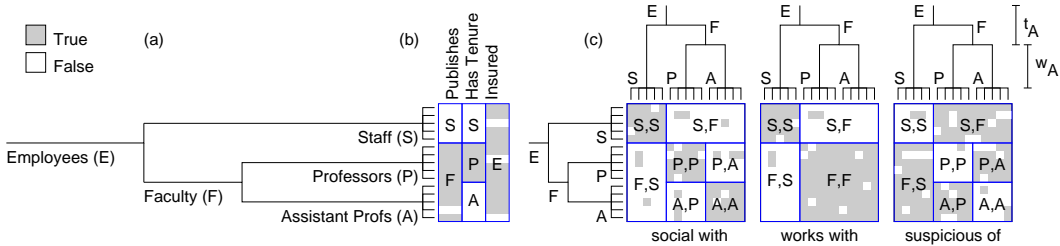


Figure 1: (a) A hierarchy over 15 members of a university department: 5 staff members, 5 professors and 5 assistant professors. (b) Three binary features, each of which is associated with a different tree-consistent partition of the objects. Each class in each partition is labeled with the corresponding node in the tree. (c) Three binary relations, each of which is associated with a different tree-consistent partition of the set of object pairs. Each class in each partition is labeled with the corresponding pair of nodes.

object is conditionally independent of its value for all objects in different classes. Within each class, the feature is independently and identically distributed across the objects. Therefore, finding the categories in the tree most relevant to a feature can be formalized as finding the simplest tree-consistent partition that best accounts for the distribution of the feature (Figure 1b). We therefore define an annotated hierarchy as a hierarchy together with a set of tree-consistent partitions for all features.

Although most discussions of annotated hierarchies focus on features, much of the data available to human learners comes in the form of relations. Understanding the structure of social groups, for instance, involves inferences about relations like *admires*( $\cdot, \cdot$ ), *friend of*( $\cdot, \cdot$ ) and *brother of*( $\cdot, \cdot$ ). Like the feature case, our generative model for relational data assumes that each (binary) relation is generated from a tree-consistent partition of the set of all *ordered pairs of objects*, which can be viewed as an annotation of the tree. Each class in a tree-consistent partition now corresponds to a pair of nodes (Figure 1c). As in the feature case, the partition for each relation asserts independence between classes. Within each class, the relation will exist with some probability between every object pair in the class. Again, every partition must be “consistent” with the same tree in the feature case. As in the feature case, finding the categories in the tree most relevant to a relation can be formalized as finding the tree-consistent partition that best accounts for the distribution of the relation. Intuitively, this is the simplest partition in which the object pairs within each class have roughly the same probability of exhibiting the relation. The final piece of our generative model is a prior distribution over rooted trees representing hierarchies. At a high level, the best hierarchy will be the one that provides categories with which the features and relations can be most compactly summarized.

Like other methods for discovering structure in data, our approach may be useful both as a tool for data analysis and as a model of human learning. After describing our approach, we apply it to several data sets inspired by problems faced by human learners. Our first analysis suggests that the model recovers coherent domains given objects and features from several domains (animals, foods, tools and vehicles). Next we show that the model discovers interpretable structure in kinship data, and in data representing relationships between ontological kinds.

## 2 A generative model for features and relations

Our approach is organized around a generative model for feature data and relational data. For simplicity, we present our model for feature and relational data separately, focusing on the case where we have a single binary feature or a single binary relation. After presenting our generative model, we describe how it can be used to discover annotated hierarchies from data.

We begin with the case of a single binary feature and define a joint distribution over three entities: a rooted, weighted, binary tree  $T$  with the  $\mathcal{O}$  objects at the leaves; a tree-consistent partition of the objects; and feature observations organized into a column of data,  $\mathbf{d}$ , where  $\mathbf{d}_o$  is the value of the feature for object  $o$ . Let  $e_n$  be the edge (with weight  $w_n$ ) connecting the node  $n$  to its parent. For brevity, we identify the root of the tree  $T$  with  $T$  itself and each leaf,  $o$ , with the object it represents. When clear from context, we also identify the subtree rooted at a node  $n$  with  $n$  itself. A tree-consistent partition  $\pi$  is represented by a set of subtrees  $\{n_1, n_2, \dots, n_k\}$ , such that each object belongs to exactly one subtree in  $\pi$ . In Figure 1b, three tree-consistent partitions associated with the hierarchy are represented and each class in each partition is labeled with the corresponding subtree.

The joint distribution  $P(T, \pi, \mathbf{d} | \lambda, \gamma_f)$  is induced by the following generative process:

- i. Sample a tree  $T$  from a uniform distribution over rooted binary trees with  $\mathcal{O}$  leaves (each leaf will represent an object and there are  $\mathcal{O}$  objects).
- ii. For each edge  $e_n$ , generate its weight,  $w_n$ , according to an exponential prior distribution with parameter  $\lambda$ , i.e.  $p(w_n|\lambda) = \lambda e^{-\lambda w_n}$ .
- iii. Generate a tree-consistent partition  $\pi_f = \{n_1, n_2, \dots, n_k\}$  according to the following, recursively-defined, mutation process starting at the node  $n = T$ :
  1. If  $n$  is a leaf, let  $\pi_f := \pi_f \cup \{n\}$ .
  2. With probability  $1 - e^{-w_n}$ , let  $\pi_f := \pi_f \cup \{n\}$ ;
  3. Otherwise, recurse on the left and right children of  $n$ .

Intuitively, if an edge in the tree has a large weight, the process is more likely to stop at the edge and group all the data below it. For the *publishes* feature in Figure 1b, the tree-consistent partition is  $\{F, S\}$ .

- iv. For each subtree  $n \in \pi_f$ , generate a parameter  $\theta_n \sim \text{Beta}(\gamma_f, \gamma_f)$ , where  $\theta_n$  is the probability that objects in the subtree  $n$  exhibit the feature  $f$ . Returning to the *publishes* example in Figure 1b, two parameters,  $\theta_F$  and  $\theta_S$ , would be drawn for this feature.
- v. For each object  $o$ , draw  $\mathbf{d}_o \sim \text{Bernoulli}(\theta_n)$ , where  $n \in \pi_f$  is the subtree containing  $o$ .

Consider now the case where we have a single binary relation defined over all ordered pairs of objects  $\{(o_i, o_j)\}$ . In the relational case, our joint distribution is defined over a rooted, weighted, binary tree; a tree-consistent partition of *ordered pairs* of objects; and observed, relational data represented as a matrix  $\mathbf{D}$  where  $\mathbf{D}_{i,j} = 1$  if the relation holds between  $o_i$  and  $o_j$ .

Given a pair of subtrees  $(n_i, n_j)$ , let  $n_i \times n_j$  be the set of all pairs of objects  $(o_i, o_j)$  such that  $o_i$  is an object in the subtree  $n_i$  and  $o_j$  is an object in the subtree  $n_j$ . With respect to pairs of objects, a tree-consistent partition,  $\pi$ , is a set of pairs of subtrees  $\{(n_{i_1}, n_{i_2}), (n_{i_3}, n_{i_4}), \dots\}$  such that, for every pair of objects  $(o_1, o_2)$ , there exists exactly one pair  $(n_i, n_j) \in \pi$  such that  $(o_1, o_2) \in n_i \times n_j$ . If we reordered the columns and rows of the matrix  $\mathbf{D}$  according to an in-order traversal of the binary tree  $T$ , each tree-consistent partition would split the matrix into contiguous, rectangular blocks (see Figure 1c, where each rectangular block is labeled with its subtree pair). Assuming we have already generated a rooted, weighted binary tree, we now specify the generative process for a single binary relation (c.f. steps iii through v in the feature case):

- iii. Generate a tree-consistent partition  $\pi_r = \{(n_{i_1}, n_{i_2}), \dots, (n_{i_{2k}}, n_{i_{2k+1}})\}$  according to the following, recursively-defined, mutation process defined over a pair of trees  $(n, m)$ , starting at the node  $n = m = T$ :

Let  $t_n$  be the total sum of the weights of all edges above edge  $n$  on the path to the root (i.e. including the weight of the root but not the weight of  $n$ ). Let  $n^* = n$  if  $t_n + w_n < t_m + w_m$  and  $n^* = m$  otherwise. That is,  $n^*$  is the ‘‘lighter’’ node. Let  $w^* = \max\{t_n, t_m\}$ . Then,

  1. If  $n$  and  $m$  are both leaves or if  $n^*$  is a leaf, then  $\pi_r := \pi_r \cup \{(n, m)\}$ .
  2. With probability  $1 - e^{-2(w_{n^*} - w^*)}$ , let  $\pi_r := \pi_r \cup \{(n, m)\}$ .
  3. Otherwise, recurse on the left and right children of  $n^*$ , keeping the other node fixed.

Intuitively, if a pair of edges in the tree both have large weights, the process is more likely to group all pairs of objects below those two edges. In Figure 1c, the tree-consistent partition for the *works with* relation is  $\{(S, S), (S, F), (F, S), (F, F)\}$ .

- iv. For each pair of subtrees  $(n_i, n_j) \in \pi_r$ , generate the parameter,  $\theta_{n_i, n_j} \sim \text{Beta}(\gamma_r, \gamma_r)$ , where  $\theta_{n_i, n_j}$  is the probability that the relation holds between any pair of objects in  $n_i \times n_j$ . For the *works with* relation in Figure 1c, parameters would be drawn for each of the four classes in the tree-consistent partition.
- v. For each pair of objects  $(o_i, o_j)$ , draw  $\mathbf{D}_{i,j} \sim \text{Bernoulli}(\theta_{n_i, n_k})$ , where  $(n_i, n_k) \in \pi_r$  is the subtree containing  $(o_i, o_j)$ . That is, the probability that the relation holds between a pair is the same for all pairs in a given class.

For data sets with multiple relations and features, we assume that all relations and features are conditionally independent given the weighted tree  $T$ .

## 2.1 Inference

Given a data set of features and/or relations defined on a set of objects, we would like to determine the maximum *a posteriori* (MAP) weighted tree, having marginalized over the tree-consistent partitions and class parameters. Given the MAP tree, we then determine the MAP tree-consistent partition for each feature and relation. We find an approximate MAP tree using a Markov chain Monte Carlo (MCMC) scheme to search the space of weighted trees. The marginal likelihood of each candidate tree is efficiently computed by a dynamic program.

The posterior probability of the weighted tree,  $T$ , given data  $D = (\{\mathbf{d}_f\}_{f=1}^{\mathcal{F}}, \{\mathbf{D}_r\}_{r=1}^{\mathcal{R}})$  over  $\mathcal{O}$  objects,  $\mathcal{F}$  features and  $\mathcal{R}$  relations and parameters  $\lambda$  and  $\gamma = (\{\gamma_f\}_{f=1}^{\mathcal{F}}, \{\gamma_r\}_{r=1}^{\mathcal{R}})$  is

$$P(T|D, \lambda, \gamma) \propto P(T|\lambda) P(D|T, \gamma) = \prod_{i=1}^{2\mathcal{O}-1} \lambda e^{-\lambda w_{n_i}} \prod_{f=1}^{\mathcal{F}} P(\mathbf{d}_f|T, \gamma_f) \prod_{r=1}^{\mathcal{R}} P(\mathbf{D}_r|T, \gamma_r)$$

We first present a dynamic program  $\mathbb{T}_f(n) \equiv \mathbb{T}_f(T, n, \mathbf{d}_f, \gamma_f)$  for calculating  $\mathbb{T}_f(T) = P(\mathbf{d}_f|T, \gamma_f)$ , the marginal probability of the data associated with feature  $f$ , conditioned on the weighted tree and the hyperparameter  $\gamma_f$ . The program  $\mathbb{T}$  is defined in terms of  $\mathbb{M}_f(n) \equiv \mathbb{M}_f(T, n, \mathbf{d}_f, \gamma_f)$ , the marginal probability of the data associated with the objects in the subtree rooted at the node  $n$ . For our binary data sets,  $\mathbb{M}_f(n)$  is the standard marginal likelihood for the beta-binomial model. If parts of the data are missing completely at random, they can simply be ignored.

First observe that, for all objects (i.e. leaf nodes)  $o$ ,  $\mathbb{T}_f(o) = \mathbb{M}_f(o)$ . Assume the mutation process has reached node  $n$ . At this point, the process either terminates with probability  $1 - e^{-w_n}$ , in which case the contribution to  $\mathbb{T}_f$  will be  $\mathbb{M}_f(n)$ , or it continues onto both children of  $n$ ,  $n.\text{left}$  and  $n.\text{right}$ . If it continues, then the possible ways in which the data under node  $n$  will be grouped include every tree-consistent partition of the objects below  $n.\text{left}$  paired with every tree-consistent partition of  $n.\text{right}$ . Altogether, this yields the following recurrence:

$$\mathbb{T}_f(n) = \begin{cases} (1 - e^{-w_n})\mathbb{M}_f(n) + e^{-w_n}\mathbb{T}_f(n.\text{left})\mathbb{T}_f(n.\text{right}) & \text{if } n \text{ is an internal node} \\ \mathbb{M}_f(n) & \text{otherwise.} \end{cases}$$

It is important to note that, the program  $\mathbb{T}$  is effective only if  $\mathbb{M}$  can be efficiently computed over trees. Fortunately, this includes many standard conjugate-exponential models, allowing this model to handle a wide variety of data.

In the relational case, we define a program  $\mathbb{T}_r(n, m) \equiv \mathbb{T}_r(T, n, m, \mathbf{D}_r, \gamma_r)$  over pairs of nodes  $(n, m)$  that calculates  $\mathbb{T}_r(T, T) = P(\mathbf{D}_r|T, \gamma_r)$ , the marginal probability of the data associated with relation  $r$ , conditioned on the weighted tree and the hyperparameter  $\gamma_r$ . The program  $\mathbb{T}_r$  is defined in terms of  $\mathbb{M}_r(n, m) \equiv \mathbb{M}_r(T, n, m, \mathbf{d}_f, \gamma_r)$ , the marginal probability of the relational data associated with the pairs of objects in  $n \times m$ . For relations,  $\mathbb{M}_f(n, m)$  is the beta-binomial. Let  $w^* = \max\{t_n, t_m\}$ , where  $t_n$  is the total sum of the weight of the edges above the edge  $n$ . Let  $n^* = n$  if  $t_n + w_n < t_m + w_m$  and  $n^* = m$  otherwise. That is,  $n^*$  is the ‘‘lighter’’ node. If  $n$  and  $m$  are both leaves or  $n^*$  is a leaf, then  $\mathbb{T}_r(n, m) = \mathbb{M}_r(n, m)$ . Otherwise,

$$\begin{aligned} & \mathbb{T}_r(n, m) \\ &= \begin{cases} (1 - e^{-2(w_n - w^*)})\mathbb{M}_r(n, m) + e^{-2(w_n - w^*)}\mathbb{T}_r(n.\text{left}, m)\mathbb{T}_r(n.\text{right}, m) & \text{if } n \text{ is lighter} \\ (1 - e^{-2(w_m - w^*)})\mathbb{M}_r(n, m) + e^{-2(w_m - w^*)}\mathbb{T}_r(n, m.\text{left})\mathbb{T}_r(n, m.\text{right}) & \text{if } m \text{ is lighter} \end{cases} \end{aligned}$$

For all the results in this paper, we fixed the hyperparameters of all beta distributions to  $\gamma = 0.5$  (i.e. the asymptotically least informative prior) and report the MAP tree and MAP tree-consistent partitions conditioned on the tree. The MCMC scheme we use to search the space of weighted trees is adapted from [13]. In particular, we cycle through three Metropolis-Hastings moves:

- i. **Subtree Pruning and Regrafting:** Choose a node  $n$  uniformly at random (except the root). Choose another node  $n'$  that is not a descendant of  $n$ . Sample  $u \sim \text{Uniform}(0, 1)$ . Detach  $n$  from its parent and reattach it above  $n'$ . Then set  $w_n := uw_{n'}$  and set  $w_{n'} := (1 - u)w_{n'}$ . Note that this move yields a symmetric proposal, so that the Metropolis-Hastings acceptance ratio for this move reduces to the ratio of the marginal likelihoods of the tree before and after the move.
- ii. **Edge Weight Adjustment:** Choose a node  $n$  uniformly at random (including the root). Propose a new value for  $w_n$  distributed as  $\text{Normal}(\log(w_n), 1)$ .

- iii. Subtree Swapping: Choose a node  $n$  uniformly at random (except the root). Choose another node  $n'$  such that neither  $n$  nor  $n'$  are descendants of each other, and swap  $n$  and  $n'$ . (This is a symmetric proposal.)

The first two moves suffice to make the chain ergodic; subtree swapping is included to improve mixing. The initial tree topology was generated by running standard, agglomerative, hierarchical clustering (HC) with the Hamming distance metric.<sup>1</sup> Given the initial topology, the weights of all edges were set to a fixed, constant value and the chain was left to mix for roughly 10,000 iterations (after which point it rarely improved on the MAP topology). Given the initial topology, the MCMC procedure refines the topology and adjusts the weights to find the salient categories in the data. Given a weighted tree, the MAP tree-consistent partition for each feature and relation can be efficiently calculated by a straightforward modification of the above dynamic programs, replacing sums with max operations and maintaining a list of nodes representing the MAP tree-consistent partition at each subtree.

## 2.2 Related Work

There are several methods that discover hierarchical structure in feature data. Hierarchical clustering [4] has been successfully used for analyzing both biological data [17] and psychological data, but cannot learn the annotated hierarchies that we consider. Bayesian hierarchical clustering [6] is a recent alternative with a principled probabilistic foundation, and could be extended to learn annotated hierarchies from feature data. An extended version of this method may produce results similar to our own approach when only feature data are available. Our model for feature data is most closely related to methods for Bayesian phylogenetics [13]. These methods typically assume that features are generated directly by a stochastic process over a tree. Our model adds an intervening layer of abstraction by assuming that partitions are generated by a stochastic process over a tree, and that features are generated from these partitions. By introducing a partition for each feature, we gain the ability to annotate a hierarchy with the levels most relevant to each feature.

There are several methods for discovering hierarchical structure in relational data [5, 12], but none of these methods provides a general purpose solution to the problem we consider. Most of these methods take a single relation as input, and assume that the hierarchy captures an underlying community structure: in other words, objects that are often paired in the input are assumed to lie nearby in the tree. Our approach handles multiple relations simultaneously, and allows a more flexible mapping between each relation and the underlying hierarchy. Different relations may depend on very different regions of the hierarchy, and some relations may establish connections between categories that are quite distant in the tree (see Figure 4).

Many non-hierarchical methods for relational clustering have also been developed [8, 9, 15, 16]. One family of approaches is based on the stochastic blockmodel [14], which handles multiple relations simultaneously, and does not assume that each relation has underlying community structure. The blockmodel, however, does not discover hierarchical structure; instead it partitions the objects into a set of non-overlapping categories. Our relational model is an extension of the blockmodel that discovers a nested set of categories, and that discovers in addition which categories are useful for understanding each relation in the data set.

## 3 Results

We applied our model to three problems inspired by tasks that human learners are required to solve. Our first application used data collected in a feature listing task by Cree and McRae [2]. Participants in this task listed the features that came to mind when they thought about a given object: when asked to think about a lemon, for example, subjects listed features like “yellow,” “sour,” and “grows on trees.”<sup>2</sup> We analyzed a subset of the full data set including 60 common objects and the 100 features most commonly listed for these objects. The 60 objects are shown in Figure 2, and were chosen to represent four domains: animals, food, vehicles and tools.

Figure 2 shows the MAP tree identified by our algorithm. The model discovers the four domains as well as superordinate categories (e.g. “living things”, including fruits, vegetables, and animals) and subordinate categories (e.g. “wheeled vehicles”). Figure 2 also shows MAP partitions for 10 representative features. The model discovers that some features are associated only with certain parts of the tree: “is juicy” is associated with the fruits, and “is metal” is associated with the man-made items. Discovering domains is a fundamental cognitive problem that may be solved early

<sup>1</sup>For the purpose of initialization only, we flatten the relational data into features.

<sup>2</sup>Note that some of the features are noisy — according to these data, onions are not edible, since none of the participants chose to list this feature for onion.

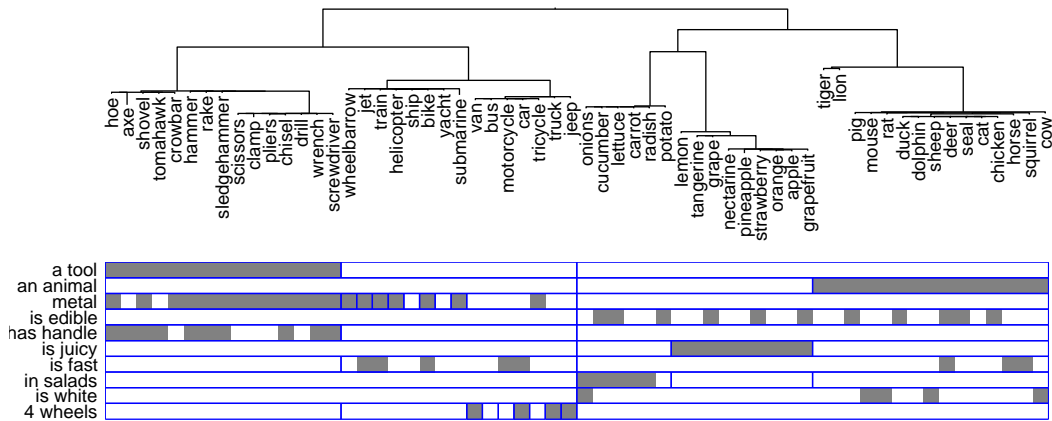


Figure 2: The MAP tree recovered from a data set including 60 objects from four domains. MAP partitions for several features are shown: the model discovers, for example, that “is juicy” is associated with only one part of the tree. The weight of each edge in the tree is proportional to its vertical extent.

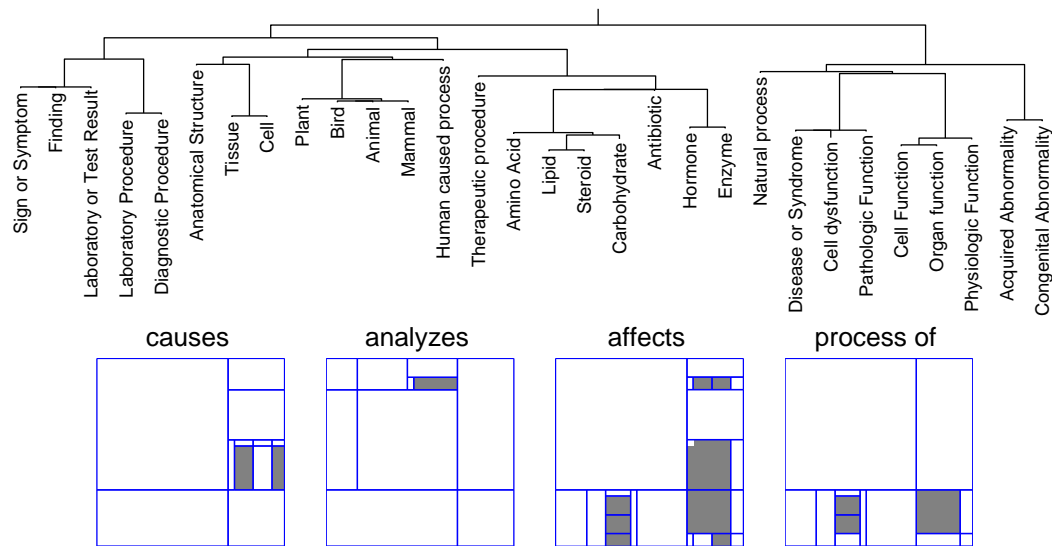


Figure 3: The MAP tree recovered from 49 relations between entities from a biomedical data set. Four of the relations are shown, and the entities along the rows and the columns are arranged according to an in-order traversal of the MAP tree. For instance, “sign or symptom” corresponds to the top row and the leftmost column of each relation.

in development [10], but that is ignored by many cognitive models, which consider only carefully chosen data from a single domain (e.g. data including only animals and only biological features). By organizing the 60 objects into domains and identifying a subset of features that are associated with each domain, our model begins to suggest how infants may parse their environment into coherent domains of objects and features.

Our second application explores the acquisition of ontological knowledge, a problem that has been previously discussed by Keil [7]. We demonstrate that our model discovers a simple biomedical ontology given data from the Unified Medical Language System (UMLS) [11]. The full data set includes 135 entities and 49 binary relations, where the entities are ontological categories like ‘Sign or Symptom’, ‘Cell’, and ‘Disease or Syndrome,’ and the relations include verbs like *causes*, *analyzes* and *affects*. We applied our model to a subset of the data including the 30 entities shown in Figure 3.

The MAP tree is an ontology that captures several natural groupings, including a category for “living things” (plant, bird, animal and mammal), a category for “substances” (amino acid, lipid, antibiotic, enzyme etc.) and a category for abnormalities. The MAP partitions for each relation identify the relevant categories in the tree relatively cleanly: the model discovers, for example, that the distinction between “living things” and “substances” is irrelevant to the second place of the relation *causes*,

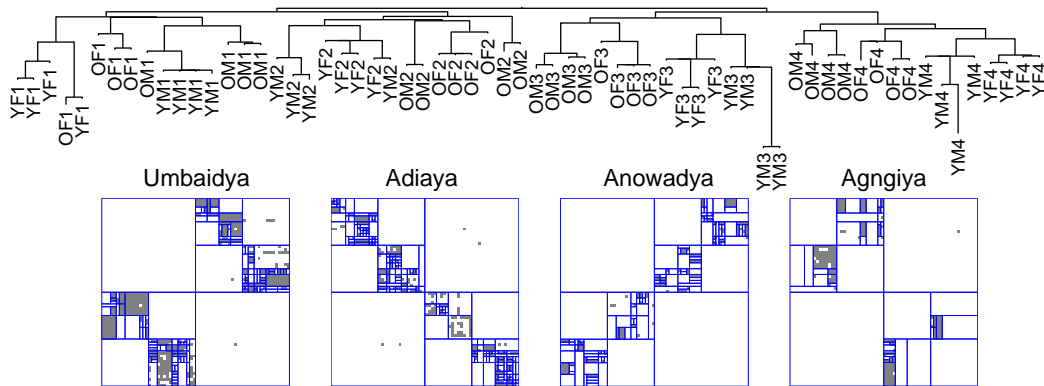


Figure 4: The MAP tree recovered from kinship relations between 64 members of the Alywarra tribe. Individuals have been labelled with their age, gender and kinship section (e.g. “YF1” is a young female from section 1). MAP partitions are shown for four representative relations: the model discovers that different relations depend on the tree in very different ways.

since neither of these category can be caused by any entity in the data set. This distinction, however, is relevant to the first place of *causes*: substances can cause abnormalities and dysfunctions, but living things cannot. Note that the MAP partitions for *causes* and *analyzes* are rather different: one of the reasons why discovering separate tree-consistent partitions for each relation is important is that different relations can depend on very different parts of an ontology.

Our third application is inspired by the problem children face when learning the kinship structure of their social group. This problem is especially acute for children growing up in Australian tribes, which have kinship systems that are more complicated in many ways than Western kinship systems, but which nevertheless display some striking regularities. We focus here on data from the Alywarra tribe [3]. Denham [3] collected a large data set by asking 104 tribe members to provide kinship terms for each other. Twenty-six different terms were mentioned in total, and four of them are represented in Figure 4. More than one kinship term may describe the relationship between a pair of individuals — since the data set includes only one term per pair, some of the zeros in each matrix represent missing data rather than relationships that do not hold. For simplicity, however, we assume that relationships that were never mentioned do not exist.

The Alywarra tribe is divided into four kinship sections, and these sections are fundamental to the social structure of the tribe. Each individual, for instance, is permitted only to marry individuals from one of the other sections. Whether a kinship term applies between a pair of individuals depends on their sections, their ages and their genders [3, 8]. We analyzed a subset of the full data set including 64 individuals chosen to equally represent all four sections, both genders, and people young and old. The MAP tree divides the individuals perfectly according to kinship section, and discovers additional structure within each of these sections. Group three, for example, is split first according to age and then according to gender. The MAP partitions for each relation indicate that different relations depend very differently on the structure of the tree. *Adiadya* refers to a classificatory brother or sister: that is, to a younger member of one’s own kinship section, even if he or she is not a biological sibling. The MAP partition for this relation contains fine-level structure only along the diagonal, indicating that the model has discovered that the term only applies between individuals from the same kinship section. *Umbaidya* can be used only between members of sections 1 and 3, and members of sections 2 and 4. Again the MAP partition indicates that the model has discovered this structure. In some places the MAP partitions appears to overfit the data: the partition for *Umbaidya*, for example, appears to capture some of the noise in this relation. This result may reflect the fact that our generative process is not quite right for these data: in particular, it does not capture the idea that some of the zeroes in each relation represent missing data.

## 4 Conclusions

We developed a probabilistic model that assumes that features and relations are generated over an annotated hierarchy, and showed how this model can be used to recover annotated hierarchies from raw data. Three applications of the model suggested that it is able to recover interpretable structure in real-world data, and may help to explain the computational principles which allow human learners to acquire hierarchical representations of real-world domains.

Our approach opens up several avenues for future work. A hierarchy specifies a set of categories, and an annotated hierarchy indicates which of these categories are important for understanding specific features and relations. A natural extension is to learn sets of categories where the categories are no longer hierarchically organized, but possess some other kind of structure, such as factorial structure [16]. For example, the kinship data we analyzed may be well described by a set of categories where each individual belongs to a kinship section, a gender, and an age group.

Our generative model for relational data extends naturally to data sets that specify relationships between two distinct sets of objects. Given information, say, about the book-buying habits of a set of customers, this extension of our model could discover a hierarchical representation of the customers and a hierarchical representation of the books, and discover the categories of books that tend to be preferred by different kinds of customers. In addition, we could easily extend our model to handle higher-order relations, non-binary data, or multiple trees, each describing disjoint subsets of the features and relations. A challenging but intriguing extension might then consider learning the number of these trees and which features and relations they describe.

Although statistical relational learning has been studied for many years, recent advances in machine learning have greatly expanded the scope of relational models. Models of human learning, however, have yet to explore the cognitive implications of these techniques. Our approach represents one attempt to understand how humans acquire structured knowledge about objects, features, and relations, and future work in this direction should find further cognitive applications of machine-learning techniques.

## References

- [1] A. M. Collins and M. R. Quillian. Retrieval Time from Semantic Memory. *JVLVB*, 8:240–248, 1969.
- [2] G. Cree and K. McRae. Analyzing the factors underlying the structure and computation of the meaning of chipmunk, chisel, cheese, and cello (and many other concrete nouns). *J. Exp. Gen.*, 132:163–201, 2003.
- [3] W. Denham. *The detection of patterns in Alyawarra nonverbal behaviour*. PhD thesis, U. of Wash., 1973.
- [4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley.
- [5] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
- [6] K. Heller and Z. Ghahramani. Bayesian Hierarchical Clustering. Technical report, Gatsby Computational Neuroscience Unit, 2005.
- [7] F. C. Keil. *Semantic and Conceptual Development*. Harvard University Press, Cambridge, MA, 1979.
- [8] C. Kemp, T. L. Griffiths, and J. B. Tenenbaum. Discovering Latent Classes in Relational Data. Technical Report AI Memo 2004-019, MIT, 2004.
- [9] J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *NCAI*, pages 798–804, 2002.
- [10] J. M. Mandler and L. McDonough. Concept formation in infancy. *Cog. Devel.*, 8:291–318, 1993.
- [11] A. T. McCray. An upper level ontology for the biomedical domain. *Comp. Func. Genom.*, 4:80–84, 2001.
- [12] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. *Proceedings of the Text Mining and Link Analysis Workshop, IJCAI*, 2003.
- [13] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In *Molecular Systematics, 2nd. edition*, 1996.
- [14] Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *JASA*, 82:8–19, 1987.
- [15] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge Press, 1994.
- [16] A. P. Wolfe and D. Jensen. Playing multiple roles: discovering overlapping roles in social networks. In *Proc. ICML workshop on statistical relational learning and its connections to other fields*, 2004.
- [17] K. Y. Yeung, M. Medvedovic, and R. E. Bumgarner. Clustering gene-expression data with repeated measurements. *Genome Biology*, 2003.