Structure Learning on Large Scale Common Sense Statistical Models of Human State

William Pentney
Department of Computer Science & Engineering
University of Washington
bill@cs.washington.edu

Matthai Philipose
Intel Research Seattle
matthai.philipose@intel.com

Jeff Bilmes
Department of Electrical Engineering
University of Washington
bilmes@ee.washington.edu

Abstract

Research has shown promise in the design of large scale common sense probabilistic models to infer human state from environmental sensor data. These models have made use of mined and preexisting common sense data and traditional probabilistic machine learning techniques to improve recognition of the state of everyday human life. In this paper, we demonstrate effective techniques for structure learning on graphical models designed for this domain, improving the SRCS system of (Pentney *et al.* 2006) by learning additional dependencies between variables. Because the models used for common sense reasoning typically involve a large number of variables, issues of scale arise in searching for additional dependencies. We describe how we use data mining techniques to address this problem and show experimentally that these techniques improve the accuracy of state prediction. We present techniques to improve prediction the unlabeled as well as the labeled variable case. At a high level, we demonstrate progress towards an old goal of AI, learning new commonsense facts about daily life from sensor data.

Introduction

In recent years, researchers have made increasing progress in designing systems to recognize the state of the everyday human environment, or *human state*, from sensor data. Such systems would be of great use in many areas, such as elder care, where they could be used to monitor and evaluate patients with cognitive disorders. Many systems have been developed that attempt to recognize everyday human activity through the deployment of dense sensors, e.g. (Wyatt, Philipose, & Choudhury 2005; Tapia, Intille, & Larson 2004; Lukowicz *et al.* 2004; Lester *et al.* 2005). Developments in lightweight and inexpensive sensor technology have made it possible to more easily collect useful information about everyday life that may be used for inference and reasoning.

In (Pentney *et al.* 2006), a system called SRCS, for State Recognition using Common Sense, was introduced, which makes use of statistical inference techniques, preexisting common sense data, and observations collected from wearable sensors to reason about human state. To build a model, SRCS makes use of common sense information from the Open Mind Indoor Commonsense (OMICS) database (Gupta & Korchendorfer 2004), which collects common sense facts in natural language form from Internet users, such as "People who are hungry eat" or "A towel is used in the bathroom". SRCS uses these facts, along with World Wide Web data mined with the KnowItAll system of (Etzioni *et al.* 2004), to create a statistical model over random variables about human state, such as "The user is in the kitchen" or "The duster is dirty". In (Pentney *et al.* 2007), it was shown that conventional machine learning techniques can improve the performance of the SRCS.

However, considerable room for improvement of the system's prediction exists. The commonsense facts provided by OMICS offer many useful relationships between objects, actions, locations, and situations in everyday life. However, because the OMICS database represents facts collected in a somewhat haphazard fashion from Internet users, many gaps in this information exist. For example, the form of the OMICS database used in SRCS does not provide any relationship between the use of a pen and the action "write".

In this work, we consider the improvement of a common sense reasoning model about human state through structure learning techniques. Using a small amount of sparsely labeled data, we seek to learn new relationships that will be valuable in human state prediction, effectively filling some of the "gaps" in existing common sense models, as well as removing unnecessary relationships. A major issue with structure learning in graphical models is the scale of the problem, made worse by the very large SRCS model, which has thousands of variables and features; we discuss how we reduce the structural search space for our model using data mining techniques, and select a set of candidate variables to search from using sparsely labeled data. In our experiments, we show results demonstrating improved prediction of queries about daily life with our techniques, and explore the improvement in performance from using unlabeled variables in the training data for learning in addition to the labeled variables. Our results demonstrate that these techniques have the capacity both to improve prediction of human state, and to augment models with new common sense information about daily life. We present some conclusions and suggestions for future work.

Structure Learning on Undirected Graphical Models

A *Markov random field (MRF)* is described by a set of random variables $X = x_1, x_2, ...x_n$, a set of *feature functions* or *factors* $F = \{f_1, f_2, ...f_m\}$, where each f_i is a function mapping a subset of variables $X_{f_i} \subset X$ to a nonnegative real number, and a set of linear parameters $\Lambda = \lambda_1, \lambda_2, ...\lambda_m$. This model can be depicted by a graph in which each feature f_i represents a clique amongst all the variables in X it is dependent upon. The parameter λ_i represents a linear weight on the feature f_i. The distribution the model represents is defined by $P(\hat{X}) = \frac{1}{Z}\exp(\sum_i \lambda_i f_i(\hat{X}_{f_i}))$, where $Z = \sum_{\hat{X}} \exp(\sum_i \lambda_i f_i(\hat{X}_{f_i}))$ is a normalization constant. Parameter learning on MRFs is often done using the technique of *maximum likelihood (ML)* learning. Given a set of labeled examples $X_1, ...X_D$, the goal of ML learning is to find the set of parameters Λ^* that maximize the log likelihood $LL(X_1, ...X_D|F, \Lambda) = \log \prod_i P(X_i|\Lambda) = \sum_i \log P(X_i|\Lambda)$. An additional regularization term is also commonly used, inducing a tradeoff between likelihood and simplicity of a model. In (Pentney *et al.* 2007), a Gaussian regularization term was used to prevent overfitting and reduce overflow/underflow issues. The optimized objective function was $RLL(X_1, ...X_D|F, \Lambda) = LL(X_1, ...X_D|\Lambda) - N(\Lambda|0, \eta)$, where $N(\Lambda|0, \eta)$ is a Gaussian function of Λ with mean zero and variance η. To encourage the removal of features from the graph altogether, one can introduce a regularization term based on the L1 norm of Λ, instead of a Gaussian regularization term, and attempt to maximize the regularized log-likelihood

$$RLL(X_1, ...X_D|F, \Lambda) = LL(X_1, ...X_D|\Lambda) - \alpha\|\Lambda\|_1$$

The derivative of the L1-norm-regularized likelihood with respect to a given weight λ_i will be

$$\frac{\partial RLL(X_1, ...X_D|F, \Lambda)}{\partial \lambda_i} = f_i(X_{f_i}) - E(X_{f_i}) - \alpha \operatorname{sign}(\lambda_i)$$

Since the gradient of this regularized likelihood contains a subtracted constant based on the sign of λ_i, it can be seen that the L1-norm regularization encourages all weights in the graph to move towards zero on each iteration of learning. α is a parameter controlling the weight of the regularization term.

L1-norm regularization has been explored by many researchers as a means of reducing the complexity of machine learning problems and favoring models with minimal parameters; one notable example is the LASSO algorithm for linear regression (Tibshirani 1996). L1-norm regularization for learning in graphical models has been explored in (Lee, Ganapathi, & Koller 2006) and (Schmidt, Niculescu-Mizil, & Murphy 2007), among others. There is a slight problem with this formulation - the gradient function given above is discontinuous at zero, but proper gradient descent requires a continuous function and gradient, and this function can thus not be optimized with this method. We tackled this problem by using stochastic gradient descent with the continuous approximation $\frac{\partial|\Lambda|_1}{\partial\lambda_i} = \frac{2}{\pi}\tan^{-1}(\frac{\lambda_i}{\epsilon_{thresh}})$ with a small parameter ϵ_{thresh} and remove features with $\lambda_i < \epsilon_{thresh}$.

Incremental Feature Introduction

L1-norm regularization may be used in conjunction with local search techniques to define a structure learning algorithm for undirected graphical models. Many researchers have performed structure learning on graphical models through the use of *incremental feature introduction* (Della Pietra, Della Pietra, & Lafferty 1997; McCallum 2003; Lee, Ganapathi, & Koller 2006). This involves performing iterations of parameter learning, perhaps with some form of regularization, followed by addition of a single feature of maximal expected utility according to some measure; this process may be repeated until added features provide no further utility. Algorithm 1 defines the conventional incremental-feature-introduction-based structure learning framework.

Algorithm 1 LEARN-STRUCTURE$(X, F, \Lambda, \hat{X})$

Require: MRF with variable set X, initial featureset F, weight vector Λ and training data $\hat{X}$

1: **for all** iterations i in $1, 2, \ldots$ **do**
2: Learn maximum-likelihood parameters $\hat{\Lambda}$ on (X, F)
3: G = GENERATE-CANDIDATE-SET(X,F,$\hat{X}$)
4: **for all** candidate features $\phi \in G$ **do**
5: $s(\phi)$ = FEATURE-SCORE(g,X,F,$\hat{\Lambda}$,$\hat{X}$)
6: **end for**
7: Select feature $\phi^* = \operatorname{argmax}_\phi s(\phi)$ and add ϕ^* to F
8: Terminate loop if $s(\phi^*) < \epsilon$
9: **end for**
10: **return** F, Λ

The routine GENERATE-CANDIDATE-SET returns a set of *candidate features* G, each of which is a feature on X and which may be considered as a possible addition to the MRF. The routine FEATURE-SCORE returns a score on a feature according to some appropriately chosen measure. A common scoring method is the *gain-based method*, in which expected utility of a candidate is measured as the *information gain* of the feature. The information gain of feature f is $\operatorname{argmax}_{\lambda_f}|RLL(X|F \cup f, \Lambda \cup \lambda_f) - RLL(X|F, \Lambda)$, i.e., the maximal change in log-likelihood when the feature f is introduced, given the optimal value of the parameter λ_f. For our purposes, we will use the gain-based method, which has been used in other scenarios as well for successful feature introduction.

The SRCS System

In (Pentney *et al.* 2006), a system called SRCS (State Recognition with Common Sense) was introduced which provides a framework for inexpensively collecting and reasoning over large amounts of commonsense knowledge about the everyday world. SRCS collects information from publicly available large-scale commonsense databases such

as the OMICS database, and parses the facts contained therein to produce a probabilistic graphical model which may reason over random variables about the state of the human environment. Additionally, SRCS uses the KnowItAll web mining system, described in (Etzioni *et al.* 2004), to bootstrap its knowledge and evaluate the reliability of the facts it parses. In (Pentney *et al.* 2007) parameter learning using stochastic gradient descent to maximize likelihood, with a small amount of labeled data was used to improve the accuracy of SRCS's prediction of state. A full description of SRCS may be found in the cited papers; here we will briefly describe some of its features.

Each random variable in SRCS represents a Boolean fact relating everyday objects, actions, and states, such as `state(plate, dirty)` (the semantic meaning of which is "The plate is dirty".) Feature functions representing Horn clauses, such as `personIn(hungry)` $\Rightarrow$ `action(eat)` (i.e. "A hungry person will eat"), connect the random variables; each Horn clause represented has two 0-1 indicator feature functions, one set to 1 only when the clause is satisfied, and another only set to 1 when the clause is violated. These Horn clauses are initially collected by automatically parsing the OMICS database into a form reflecting relationships between Boolean variables. The Horn clause `personIn(hungry)` $\Rightarrow$ `action(eat)`, is created from the OMICS fact `people(hungry,eat)`, which represents the same basic concept. Each Horn clause is used to represent two indicator feature functions f_i and $\bar{f}_i$, one which is only true when the clause is satisfied and the other only true when the clause is violated. These features are each given a weight λ_i and $\bar{\lambda}_i$; initial values of these weights are collected using the KnowItAll system of (Etzioni *et al.* 2004), which uses Web data to find correlations between the concepts represented by SRCS's random variables.

KnowItAll is an information retrieval system designed to extract and evaluate widely known facts (and therefore also common sense) from the web. At the heart of KnowItAll is a template-based system that works as follows. To evaluate instances of a particular relationship, say `people(Action,Context)`, KnowItAll uses a small number of examples of the relation to induce a number of text templates that exemplify the relation. Using normalized counts of incidences of these patterns in the web, KnowItAll is able to produce a measure of how reliable a particular Horn clause c may be, given as a score $score(c) \in [0, 1]$. SRCS uses these scores as an initial model for semi-supervised learning of human state giving sparsely labeled data. For features f_i and $\bar{f}_i$ representing clause c, this initial model is set with parameters $\lambda_i = score(c)$ and $\bar{\lambda}_i = -score(c)$.

SRCS uses a graphical model to represent the state of the everyday human environment over time, in which the state of the world at time t will be represented by the set of Boolean random variables $X_t = x_{1_t}, x_{2_t}, ..., x_{n_t}$. Additionally, feature functions connect each random variable x_{i_t} and $x_{i_{t+1}}$ to represent a conditional dependency between the state of the variable x_i at t and its state at $t + 1$. In particular, we choose these feature functions such that

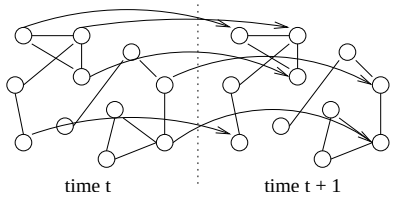

Figure 1: Outline of the structure of SRCS's graphical model. Each variable represents a Boolean variable (e.g. `state(plate,dirty)`), and each clique within a time slice is a 0-1 indicator feature representing a Horn clause. Directed arrows represent conditional probabilities on a variable x_{i_t+1} given x_{i_t}.

$P(x_{i_{t+1}} = k | x_{i_t} = k) = 1 - \epsilon$ for some ϵ and all k (i.e. given no other information, variables will stay in the same state with high probability over time). This structure is representable as a dynamic graphical model, with representations of the state vector X at time t and $t + 1$, as seen in Figure 1. Given evidence vectors $E_i \subset X_i$, this representation is sufficient for computing the conditional distribution $P(x_{i_t} | E_1 ... E_T)$ via smoothing, not unlike that which can be performed on dynamic Bayesian networks. In practice, this distribution is actually approximated using the loopy belief propagation algorithm of (Pearl 1988); empirically, we find this approximation to be sufficient, as it often is.

The full SRCS model contains 50,000+ variables and 130,000+ features; for the experiments here, we used a reduced model like that used in (Pentney *et al.* 2007), which contained all variables a depth of $d = 2$ edges in the model's graph from the nodes used for training/testing, resulting in a model containing 8,764 nodes and 24,465 features per timeslice.

Reducing Search Space Size with Data Mining Techniques

One significant challenge in structure learning on a graph of the scale we are dealing with is the issue of *candidate generation* - that is, how to find which features to add to the graph for maximal effectiveness. Certain techniques have been used for effective generation of candidate features under certain constraints; these constraints may involve the cardinality or type of features generated as well as the number of nodes in the graphical model involved.

Because the timeslice MRF in the SRCS model is very large, these methods of candidate generation and testing, by themselves, are infeasible as a means of learning structure on our timeslice MRF. Even if we restricted feature generation only to pairwise features of one specific type (e.g. Horn clauses), there would be millions of potential features to consider in the graph - too many to consider for repeated iterations of incremental feature introduction. Adding features of larger cardinality or of different types would make the problem more difficult still. Clearly we need to find an efficient means of drastically reducing the size of the candidate feature space.

Fortunately, we have some foreknowledge of the structure of our problem. Our feature space consists of Horn clauses, and the graph contains a great many random variables that are quite likely to have no significant relationship of this form to each other. Since our features represent logical rules, however, if we may find a technique for quickly finding rules that are somewhat likely to be true in our domain, this by itself may be sufficient to give us a set of viable candidate features - we shall call them *candidate candidates*. Finding such rules is a simple matter of finding Horn clauses that are at least *somewhat* likely to be true, according to our training data.

Our issue is thus finding a good set of candidate candidates which covers almost all useful features without being so large as to be infeasible in practice. The problem of finding frequently satisfied logical sentences in a large database - a collection of labeled training data may be viewed as such a database - has been previously considered and researched in the database community. We consider a technique related to the *apriori algorithm* of (Agrawal & Srikant 1994), a commonly used algorithm for finding frequently occuring itemsets in a database. We have a set of timeslices $\hat{X} = \hat{X}_1, ...\hat{X}_D$ each representing an assignment of random variables $X = x_1, ...x_n$, and we wish to find a subset of frequently satisfied Horn clauses in the training data relating the variables x_i. We find a set of candidate candidates L through an iterative process. Given a threshold value γ, we first find a set L_1 of all Horn clauses of length 1 - i.e. single variables, representing a consequent with no antecedents - that receive a "score" over some threshold γ on the training data. This is the initial set of *candidate clauses*. At each iteration k, we have a set of candidate clauses L_{k-1}. We consider every Horn clause $c = x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_{k-1}} \Rightarrow x_{i_k}$ of length k, where each $x_{i_j} \in X$, such that every clause of length $k-1$ that may be generated from the variables in c is found in L_{k-1}. Let L_k consist of all such Horn clauses scoring γ or higher. Say we are given variable set X, labeled data $\hat{X} = \hat{X}_1, ...\hat{X}_D$, where each $\hat{X}_i$ is a labeling of X, and γ. Let $count(c) = |\{\hat{X}_i \in \hat{X} | c \text{ is satisfied by } \hat{X}_i\}$ be our scoring function. The algorithm is thus as follows:

Algorithm 2 GENERATE-CANDIDATE-SET$(X, F, \hat{X}, \gamma)$

1: $L_1 = \{\{x_i\} | x_i \in X, count(\{\Box \Rightarrow x_i\}) \geq \gamma\}$
2: **for all** $(k = 2; L_{k-1} \neq; k++)$ **do**
3: $\quad C_k = \{(x_{i_1} \wedge ... \wedge x_{i_{k-1}} \Rightarrow x_{i_k}) : x_{i_1}, ...x_{i_k} \in X, \forall s \subset \{x_{i_1}, ...x_{i_k}\}, |s| = k-1$ we have $s \in L_{k-1}\}$
4: $\quad L_k = \{c \in C_k | count(c) \geq \gamma\}$
5: **end for**
6: **return** $L = (\bigcup_k L_k) - (F \cap \bigcup_k L_k)$

We use this routine for GENERATE-CANDIDATE-SET, with parameter γ added, in LEARN-STRUCTURE. Each Horn clause generated represents a candidate 0-1 indicator feature for incremental feature introduction; clauses already represented by a feature in the model are discarded. This algorithm is not guaranteed to find every Horn clause resulting in information gain - an event requiring many variables set to "true" as a precondition might plausibly be missed by this routine. Intuitively, however, we suspect that such complex relationships will be uncommon, and we find that the apriori algorithm discovers a great many clauses offering significant benefit, and which are intuitively sensible. For example, when run in our experiments, it discovers the relationship `used(pen)` $\Rightarrow$ `action(write with)`, a relationship not collected from the parsed OMICS data.

Having selected a set of candidate candidates with the apriori algorithm, we may then score each feature for inclusion for an iteration of incremental feature introduction. However, in our context, we have a very large number of variables, many of which may not be labeled in the training data. We would like to identify a subset of variables for which candidate candidate generation would be particularly beneficial.

Selecting Variables for Candidate Search

We also seek to narrow the number of variables to consider in our candidate search, which we will call *candidate variables*. Variables that are labeled for any time slice in the training data may be considered for inclusion in candidate candidates; as mentioned, the training data we deal with will generally be sparsely labeled, so the set of such variables will be rather small.

In addition, however, we will consider some of the unlabeled variables as well. The parameter learning model presented in (Pentney *et al.* 2007) made use of the initial model given by KnowItAll mining, combined with Gaussian regularization, to compute maximum likelihood on sparsely labeled data. To expand the candidate variable space, we will also seek to add variables that are not labeled, but are predicted with a generally high degree of confidence by this learned model. If a learned model predicts such variables well, it may be useful to apply maximum a posteriori (MAP) labeling to these variables and include them in structure learning. To compute the confidence of the labeling of nodes, we compute the *mean entropy* of the distributions given by filtering. Given partially labeled time slices $E_1, ...E_D$, for each fully unlabeled variable x_i, we compute $H(x_i) = -\frac{1}{D}\sum_{t=1...D}\sum_{x_{i_t}} P(x_{i_t}|E_1, ...E_t)\log P(x_{i_t}|E_1, ...E_t)$. The higher the confidence in the labeling of x_i by the learned model, the lower the value of $H(x_i)$. (If x_i is fully labeled, then $H(x_i) = 0$). We will select a threshold $\bar{h}$. The set of variables V to consider for candidate candidate generation will consist of (1) all labeled variables, and (2) all variables whose mean entropy, as computed from a model with trained parameters, falls below $\bar{h}$. As we increase $\bar{h}$, the set of candidate variables grows; Figure 2 graphs the quantity of variables whose mean entropy falls below $\bar{h}$ as a function of $\bar{h}$ given a trained SRCS model on our data set, showing how the candidate variable set size increases with $\bar{h}$.

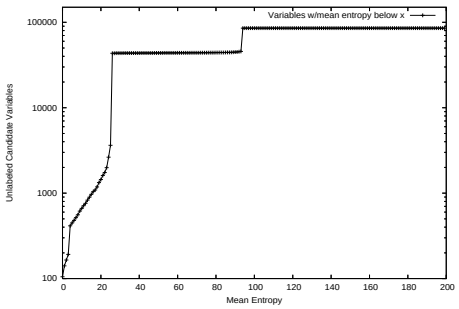

Figure 2: Graph of mean entropy $\bar{h}$ vs. variables with mean entropy below $\bar{h}$ with trained model.

Weighted Probabilistic Apriori Algorithm

To integrate the partially labeled variables into our candidate candidate generation process, we introduce a probabilistic version of the apriori algorithm. We are given a set of partially labeled time slices $E_1, ...E_D$, where each $E_i \subset X_i$. We are given candidate variable set $V \subset X$. For each $x_i \in C_i$ and each timeslice t, we may compute $P(x_{i_t}|E_1, ...E_t)$. (If x_{i_t} is labeled as y, then $P(x_{i_t} = y|E_1, ...E_t) = 1$.) To compute $count(c)$ for a Horn clause $c = x_{i_1} \wedge x_{i_2} \wedge ... \Rightarrow x_{i_{|c|}}$ in this version of the apriori algorithm, we use the MAP labeling given for each of the nodes x_{i_1} ... $x_{i_{|c|}}$ by the learned model, and use that to determine whether c is satisfied. Let $\mathbf{1_{c,t}}$ be the indicator showing that c is satisfied under this labeling. Computing the conditional distributions for each $P(x_{i_t})$ requires performing inference over the labeled data set, which is not computationally expensive relative to the learning process.

Additionally, we make one other modification. Because, in practice, true instances of variables are relatively rare in the data (most random variables are false at any given time), this count metric may inappropriately favor clauses which are often satisfied by false antecedents, providing increased accuracy, but not increased precision or recall with respect to true settings. To address this, we use a technique similar to that used in(Pentney *et al.* 2007) for weighting in parameter learning. We thus give more weight to certain slices containing true instances of queries that are less frequently true in the training data. For the labeled variables $q_1...q_l$, let t_{q_i} be the number of true appearances of q_i in a timeslice X_i in the training data, and $\alpha(q_i) = \frac{D-t_{q_i}}{t_{q_i}}$. In computing counts, we then weight the term representing slice t in $count(c)$ by multiplying it by $\sum_{j=1}^{l} \mathbf{1}_{q_j,t}\alpha(q_j)$ where $\mathbf{1}_{q_j,t}$ is 1 if q_j is labeled true in slice t and 0 otherwise. This gives more weight to a slice if it contains true instances of a query, particularly if true instances of that query are uncommon. Thus the count for clause c is computed as $count(c) = \frac{1}{Z_c}\sum_{t=1}^{D} \mathbf{1}_{c,t} \sum_{j=1}^{l} \mathbf{1}_{q_j,t}\alpha(q_j)$, where $Z_c = \sum_{t=1}^{D} \sum_{j=1}^{l} \mathbf{1}_{q_j,t}\alpha(q_j)$.

Our structure learning algorithm thus runs as follows:

- Run parameter learning as in (Pentney *et al.* 2007), with a Gaussian prior for regularization, to learn a model for prediction. Compute the mean entropies of all unlabeled variables using this model.
- Compute a candidate variable set V given mean entropy threshold $\bar{h}$.
- Run the weighted probabilistic apriori algorithm to select a set of candidate candidates G.
- Perform structure learning with incremental feature introduction using the set of candidate features given by G at each iteration.

This version of the algorithm, combined with our method of selecting variables, thus presents us with parameters to set to set for learning: the threshold γ for acceptance of a clause as a candidate candidate and the entropy threshold $\bar{h}$ for inclusion of a variable in the candidate variable set. We chose $\gamma = .5$ and $\bar{h} = 30$, on the basis of their empirical performance; this choice provided a reasonably large set of candidate candidates (≈600) while still making the resulting learning problem tractable.

Experiments

For our experimental evaluation, we used 70-75 minutes of traces of household object use in an experimental setting as worn by three users while performing various daily activities in a simulated home environment, as used in (Pentney *et al.* 2006). These traces were divided into time slices of 2.5 seconds; reasoning was to be performed over each slice.

For these activities, we then selected a set of 24 Boolean variables in the collected SRCS database which represented these variables, or were semantically very close to them, such as `stateof(cereal, prepared)`. We then recorded their "truth" value as being true or false for each interval of time in the trace.

We measured the accuracy of our system in predicting the value of the 24 queries, measuring prediction accuracy, precision and recall with respect to specifically labeling the true instances of each query (since variables are usually false, these measures are often more relevant), and compute the F-measure, the harmonic mean of these two values. We performed training on ≈25% of the data using parameter learning as described in (Pentney *et al.* 2007), and performed prediction with the learned model. We then tested SRCS's prediction on the same setup, but with the use of structure learning with the weighted probabilistic apriori algorithm as described. We used the parameter $\alpha = 4$ for our L1-regularized learning. The results may be seen in Figure 3. We see that structure learning significantly improves precision and recall of prediction. In using structure learning, we also found that the number of features in the model learned through structure learning was reduced from 24,465 in the initial model to 17,719, demonstrating that our model is simpler, yet offers better performance.

In addition, we may consider some of the candidate features that are introduced. A sampling of some of the features discovered and added to the model may be seen in Figure 4. These features represent intuitively useful concepts, and some of these features appear to significantly aid in the

Learning method	Accuracy	Precision	Recall	F-measure
Parameter, no structure	94.87%	69.31	43.53	53.47
Structure, apriori	95.67%	76.86	50.23	60.75

Figure 3: Results for experiments with and without structure learning, as described.

- `used(spoon)` $\Rightarrow$ `action(eat)`
- `location(pantry)` $\Rightarrow$ `used(cereal)`
- `used(pill)` $\Rightarrow$ `action(swallow)`
- `used(shampoo)` $\Rightarrow$ `location(shower)`
- `likelyaction(shower with)` $\wedge$ `used(soap)` $\Rightarrow$ `location(bathroom)`
- `used(pen)` $\Rightarrow$ `action(write a letter)`

Figure 4: Some of the features added to the SRCS model by the apriori algorithm.

prediction of some queries for which good features are missing, compared to parameter learning alone.

We would also like to determine whether structure learning can help on unlabeled queries, further demonstrating the benefit of a common sense model over one which merely measures . To do this, we perform an additional experiment in which we will train SRCS on data in which only twelve of the 24 queries are labeled, and perform testing on seven other labeled queries whose labels were not given. The same experiment was performed in (Pentney *et al.* 2007), but with only parameter learning; we compare to those results. To see if we benefit from the addition of unlabeled variables to the learning, we will compare structure learning in two setups: one in which unlabeled variables are not added to the candidate variable set for learning, and one in which we add unlabeled variables using a mean entropy threshold of $\bar{h} = 30$. In Figure 5, we compare prediction with respect to precision, recall, and F-measure. We see that learning provides improvements in prediction on some of the unlabeled queries; adding unlabeled variables improves certain queries as well. This indicates that we may learn a more general model of the everyday environment from sparse information with a good prior model - in our case, the model given by KnowItAll. Furthermore, it demonstrates that, given confident predictions with a good model, there can be value in adding unlabeled variables for structure learning.

How general are these results? To determine how general, we collected further results in which we randomly selected 12 of the 24 queries to use for training, and tested on the other 12. We performed structure learning with $\bar{h} = 30$, and then evaluated the results. The mean and variance of accuracy and F-measure on the 12 testing queries over 20 such randomized trials are shown in Figure 6. We see that performance is inconsistent. This is not surprising; in many random partitions, many of the training queries are unhelpful in providing improved information about the testing queries, which may not be adequately inferred by the initial parameter-trained model (e.g. information about brushing teeth is generally unhelpful in predicting that the user is in the kitchen). In the previous experiment, by contrast, some training queries were contextually related to the testing queries. This suggests that structure learning with our technique can improve performance on unlabeled queries as well, but that this performance may rely upon some dependence in the initial trained model between those variables being trained and those being tested upon. In future work, we would like to consider the use of other techniques to better find dependencies between variables, even with very sparse labeling, and hone SRCS's model of common sense. Additionally, we would like to consider means of finding optimal choices of parameters for regularization and learning.

Mean Acc	Var Acc	Mean F-meas	Var F-meas
93.21%	23.16%	37.86	71.19

Figure 6: Mean and variance of accuracy and F-measure over twenty randomized trials.

Conclusions

We have presented a means of performing structure learning on a large scale model of human state. Using labeled data for feature introduction, we are able to learn new dependencies in a large graphical model, learning more efficiently by reducing the search space of features. Through the use of these techniques, we can efficiently perform structure learning that significantly improves prediction on a common sense reasoning system about human state, as well as collect new common sense data about daily life.

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In Bocca, J. B.; Jarke, M.; and Zaniolo, C., eds., *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 487–499. Morgan Kaufmann.

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

Etzioni, O.; Cafarella, M.; Downey, D.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proc. of 19th Annual AAAI Conference*.

Gupta, R., and Korchendorfer, M. J. 2004. Common sense data acquisition for mobile indoor robots. In *Proceedings of AAAI*.

Lee, S.; Ganapathi, V.; and Koller, D. 2006. Struture learning of markov networks using l1-regularization. In *Proceedings of NIPS*.

Lester, J.; Choudhury, T.; Kern, N.; Borriello, G.; and Hannaford, B. 2005. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Lukowicz, P.; Ward, J.; Junker, H.; Stager, M.; Troster, G.; Atrash, A.; and Starner, T. 2004. Recognizing workshop ac-

Query	Param learning			Struct learning, no unlabeled			Struct learning w/unlabeled		
	Prec	Rec	F-meas	Prec	Rec	F-meas	Prec	Rec	F-meas
activity(brushing teeth)	100.00	83.82	91.19	100.00	83.82	89.43	100.00	83.82	91.19
location(shower)	100.00	31.13	47.48	100.00	37.27	54.29	97.93	42.41	59.19
action(eat)	100.00	18.91	31.81	100.00	53.13	69.38	100.00	54.17	70.27
action(add milk to)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
stateof(tea kettle, hot)	100.00	38.24	55.32	100.00	38.24	55.32	86.67	38.24	53.06
stateof(window, dirty)	4.82	6.56	5.55	8.54	11.48	9.79	8.54	11.48	9.79
location(greenhouse)	100.00	48.56	65.37	100.00	84.62	91.67	100.00	84.62	91.67
Aggregate	72.12	32.46	51.04	72.64	43.84	52.84	70.44	44.96	53.59

Figure 5: Results of the seven unlabeled queries with parameter and structure learning on twelve queries, with and without inclusion of unlabeled candidate variables.

tivity using body worn microphones and accelerometers. In *Proc. of PERVASIVE 2004, LNCS 3001*, 18–32.

McCallum, A. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of UAI*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffman.

Pentney, W.; Popescu, A.; Wang, S.; Kautz, H.; and M.Philipose. 2006. Sensor-based understanding of daily life via large-scale use of common sense. In *Proceedings of AAAI*.

Pentney, W.; Philipose, M.; Bilmes, J.; and Kautz, H. 2007. Learing large scale common sense models of daily life. In *Proceedings of AAAI*.

Schmidt, M.; Niculescu-Mizil, A.; and Murphy, K. 2007. Learning graphical model structure using l1-regularization paths. In *Proceedings of AAAI*.

Tapia, E. M.; Intille, S. S.; and Larson, K. 2004. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, 158–175.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)* 58 No. 1:267–288.

Wyatt, D.; Philipose, M.; and Choudhury, T. 2005. Unsupervised activity recognition using automatically mined common sense. In *AAAI*, 21–27.