Learning Large Scale Common Sense Models of Everyday Life

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

Henry Kautz
Department of Computer Science
University of Rochester
kautz@cs.rochester.edu

Abstract

Recent work has shown promise in using large, publicly available, hand-contributed commonsense databases as joint models that can be used to infer human state from day-to-day sensor data. The parameters of these models are mined from the web. We show in this paper that learning these parameters using sensor data (with the mined parameters as priors) can improve performance of the models significantly. The primary challenge in learning is scale. Since the model comprises roughly 50,000 irregularly connected nodes in each time slice, it is intractable either to completely label observed data manually or to compute the expected likelihood of even a single time slice. We show how to solve the resulting semi-supervised learning problem by combining a variety of conventional approximation techniques and a novel technique for simplifying the model called *context-based pruning*. We show empirically that the learned model is substantially better at interpreting sensor data and an detailed analysis of how various techniques contribute to the performance.

Introduction

Sensor-based methods for inferring the state of people and their environment have a variety of applications such as elder care management (Wilson *et al.* 2005) and institutional workflow management. A system that could tell whether an elderly person living alone has a cold, has taken medication or is depressed, for instance, could substantially reduce the financial burden of care. A key challenge in building such systems is the need for models that relate low-level sensor signals (e.g., vision) to high-level concepts (e.g., depression). The conventional approach to acquiring such models is to apply machine learning techniques to labeled sensor data, given a "structure" prior on the dependencies between variables. The structure itself is often provided by application developers in a manual fashion. However, if many of the tens of thousands of aspects of daily life are to be tracked, the variables and their relationships amount to a "commonsense" encoding of daily life, and manual specification becomes impractical for a single individual.

Fortunately, common sense does not vary much across time, people or contexts, so that once captured, it can be re-used many times. Well-known efforts such as Cyc (Lenat & Guha 1990) and OpenMind/OMICS (Gupta & Kochenderfer 2004) have therefore devoted themselves to accumulating and codifying commonsense information about daily life. Using these databases for interpreting real-world sensor data presents two challenges:

- How should ground terms in the database (e.g., `use(knife)`) be mapped to sensor signals (e.g., pixels)?
- How should errors, uncertainty and conflict within the database be represented and handled during reasoning?

The recent work of Pentney *et al.* (2006) on the SRCS (State Recognition using Common Sense) system provides an interesting viewpoint. First, since emerging *dense* sensor networks (Tapia, Intille, & Larson 2004; Fishkin, Philipose, & Rea 2005) can directly report high-level object-use data, and since the OMICS database is grounded extensively in terms of object use, SRCS provides a commonsense interpretation of the world by connecting dense sensors to a model representing the OMICS database. Second, SRCS attaches weights automatically mined from the web to relations in OMICS and uses these weights to provide a soft interpretation of the relations in the database. In particular, it converts the database into a large dynamic probabilistic graphical model where the mined weights become feature weights. The resulting system uses OMICS automatically to interpret real-world activity data with modest success.

In restricting itself to using web-mined feature weights, SRCS omits a more conventional approach to obtaining the weights, that of learning the weights using observed data. This paper shows that it is both feasible and useful to augment the weights using machine learning techniques on observed data. Learning is challenging because of the size, complexity and scope of the SRCS graphical model. The model is a dynamic model that reasons over thousands of time steps with a roughly 50,000 node graph at each time step. Unlike graphs representing images in computer vision, for instance, the graph has extremely non-uniform structure and a large fraction of hidden nodes. Finally, given that the graph represents a wide variety of random variables about the state of the world at each step (from peoples' emotions to the state of their roof) it is realistic to expect manual labeling of no more than a very small fraction of the nodes at each time step. The end result is a very large semi-supervised learning problem (Zhu 2005) over a mixed directed/undirected probabilistic graphical model.

Although general approaches to semi-supervised learning of graphical models do exist, solving large problems requires considerable sophistication in exploiting problem structure in practice. In this light, this paper makes three main contributions. First, it formulates the commonsense model learning problem as maximum likelihood estimation on a graphical model with mined priors, and applies a variety of existing techniques to make learning tractable. Second, it presents a simple novel technique called *context-based pruning* to speed up inference substantially. Context-based pruning captures the intuition that we may be able to associate with sets of observations a subset of variables (called a context) that are likely to be affected by these observations. By identifying, and avoiding reasoning about, irrelevant contexts, it may be possible to substantially reduce the number of variables considered during inference. Third, it provides a detailed evaluation that shows that learning is both reasonably fast and improves the SRCS model significantly, and that many of our design choices contribute significantly to this result. To our knowledge, this work is the first to show how to improve a large common sense database by observing sensor data of human activity.

Preliminaries

The model on which we perform learning is a slight variation of the SRCS model (Pentney *et al.* 2006). We are given a set of observations $o = o_1, o_2, ...o_r$; in our context, these observations will be Boolean random variables. We wish to infer the state of a set of *queries* $s = s_1, s_2, ...s_{n-r}$, also Boolean random variables, for a given slice of time t. Collectively, we will speak of the state of the observations and queries as the set of variables $\mathbf{x} = \mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_n}$, where $x_1...x_r = o_1, ...o_r$ and $x_{r+1}...x_n = s_1, ...s_{n-r}$, and the set of the observations and queries at some specific point in time t as the *state vector* $\mathbf{x_t} = \mathbf{x_{1_t}}, \mathbf{x_{2_t}}, \mathbf{x_{3_t}}...\mathbf{x_{n_t}}$.

We assume that we have a set of weighted propositional Horn clauses $(c_1, q_1), (c_2, q_m), \ldots (c_m, q_m)$, where each clause c_i is a relation over Boolean variables about the state of the environment, weighted by quality score q_i in [0,1]. For instance, the clause `used(bowl)` $\rightarrow$ `action(make cereal)` may have weight 0.9, and represent the proposition "if the bowl is being used, the user is making cereal."

Given a set of Horn clauses, we produce a probabilistic graphical model designed to model the state of the environment over time. As in SRCS, time slice t may be modeled with a Markov random field, in which $\mathbf{x_t}$ will represent the state of the world at t. Some of the variables x_{i_t} can be grounded in actual observations; if we can observe the use of a bowl at time t, for instance, the variable representing `used(bowl)` may be provided as evidence. In a slight deviation for SRCS (which produces one feature per Horn clause), we represent each Horn clause c_i by two binary-valued feature functions $\phi_i(x_{c_i})$ and $\phi_{m+i}(x_{c_i})$, where x_{c_i} is the set of all variables x_j which appear in c_i. The function $\phi_i(x_{c_i})$ is a *positive feature*, and set to 1 if all variables in x_{c_i} are true (i.e. both the left and right sides of the Horn clause are true, and thus c_i is satisfied), and 0 otherwise. The function $\phi_{m+i}(x_{c_i})$, conversely, is a *negative feature* and set to 1 if the assignment to x_{c_i} violates the Horn clause and 0 otherwise. We assign weight $\lambda_i = q_i$ to positive feature $\phi_i(x_{c_i})$, and $\lambda_{m+i} = -q_i$ to negative feature $\phi_{m+i}(x_{c_i})$. For simplicity, we may sometimes refer to $\phi_i(\mathbf{x})$, where $\mathbf{x}$ is an entire state vector; this should be read as $\phi_i(\hat{\mathbf{x}}_{\mathbf{c_i}})$, where $\hat{\mathbf{x}}_{\mathbf{c_i}}$ is the assignment imposed on x_{c_i} by $\hat{\mathbf{x}}$. The full set of all such feature functions $\phi = \phi_1, \phi_2, ...\phi_{2m}$ will represent the relationships between different variables at time t.

We follow SRCS in modeling in how variables x_t change over time. For simplicity, we will assume that, given that $x_{i_{t-1}} = p$, $x_{i_t} = p$ with some fixed probability σ_p. This is represented with another set of feature functions $\psi = \{\psi_1, \psi_2, ...\psi_n\}$, where ψ_i is a function of $(x_{i_{t-1}}, x_{i_t})$, $\psi_i(p,p) = \sigma_p$, and $\psi_i(p,q) = 1 - \sigma_p$ where $p \in \{0,1\}$, $p \neq q$. Intuitively, this simplifying model means that, given no other evidence, variables will remain in the state they are, but with declining probability over time.

The probability of the state x_t given the state x_{t-1} as evidence may be calculated in this model as

$$P(\mathbf{x_t}|\mathbf{x_{t-1}}) = \frac{\mathbf{1}}{\mathbf{Z_{x_{t-1}}}} \exp((\sum_{\mathbf{i}} \lambda_{\mathbf{i}}\phi_{\mathbf{i}}(\mathbf{x_t})) + \sum_{\mathbf{i}} \mu_{\mathbf{i}}\psi_{\mathbf{i}}(\mathbf{x_t}, \mathbf{x_{t-1}}))$$

where $Z_{x_{t-1}} = \sum_{\mathbf{x_t}} \exp((\sum_i \lambda_i \phi_i(\mathbf{x_t})) + \sum_{\mathbf{i}} \mu_{\mathbf{i}}\psi_{\mathbf{i}}(\mathbf{x_t}, \mathbf{x_{t-1}}))$ is a normalizing constant summing all assignments to x_t given x_{t-1}, and $(\lambda, \mu) = ((\lambda_1, \lambda_2, ...\lambda_{2m}), (\mu_1, \mu_2, ...\mu_n))$ represent parameters on the features ψ, ϕ. This distribution factorizes with respect to a mixed directed/undirected graph with directed links between adjacent time slices.

A primary challenge in providing accurate inference using this model is the discovery of a good set of parameters (λ, μ). In SRCS, the q_i were found from Google searches, with the intention of giving higher scores to those predicates which represent more significant relationships, and low or zero scores to those that provide little useful information. Once the λ_i were set to q_i and μ_i to a default high "smoothing value", SRCS made no effort to re-estimate parameters λ or μ. In contrast, we use these initial values of the parameters as priors that aid in re-estimating them given additional labeled data.

Learning

To improve the model used, we attempt to optimize the model λ, μ for maximum log-likelihood based on training data. That is, given an actual trace of observed states $\hat{\mathbf{x}} = \hat{\mathbf{x}}_\mathbf{1}, \hat{\mathbf{x}}_\mathbf{2}, ..., \hat{\mathbf{x}}_\mathbf{d}$ and some prior on the state of the environment $\hat{\mathbf{x}}_\mathbf{0}$, we wish to optimize the value $P(\hat{\mathbf{x}})$; we will do this by instead optimizing $LL(\lambda, \mu) = \log P(\hat{\mathbf{x}}|\lambda, \mu)$. The distribution reflected by our model possesses the Markov property, so $\log P(\hat{\mathbf{x}}|\lambda,\mu) = \sum_{\mathbf{t=1}}^{\mathbf{d}} \log \mathbf{P}(\hat{\mathbf{x}}_\mathbf{t}|\hat{\mathbf{x}}_{\mathbf{t-1}},\lambda,\mu)$. The gradient of LL with respect to each λ_j is

$$\frac{\partial LL}{\partial \lambda_j} = \sum_t \phi_j(\hat{\mathbf{x}}_\mathbf{t}) + \sum_{\hat{\mathbf{x}}_{\mathbf{c_i}}} \mathbf{P}(\hat{\mathbf{x}}_{\mathbf{c_i}})\phi_\mathbf{j}(\hat{\mathbf{x}}_{\mathbf{c_i}})$$

and with respect to each μ_j is

$$\frac{\partial LL}{\partial \mu_j} = \sum_t \psi_j(\hat{\mathbf{x}}_\mathbf{t},\hat{\mathbf{x}}_{\mathbf{t-1}}) + \sum_{\mathbf{x_{i_t}},\mathbf{x_{i_{t-1}}}} \mathbf{P}(\mathbf{x_{i_t}},\mathbf{x_{i_{t-1}}})\psi_\mathbf{j}(\mathbf{x_{i_t}},\mathbf{x_{i_{t-1}}})$$

The second terms of these equations are equal to $E(\phi_j(\hat{\mathbf{x}}))$ and $E(\psi_j(\hat{\mathbf{x}}))$, respectively, and may be calculated efficiently by computing the marginal probabilities of each $\hat{x}_{c_i}$ using the well-known belief propagation (BP) algorithm (Pearl 1988). We may then optimize LL by finding the model parameters (λ,μ) for which the gradient is zero using stochastic gradient descent (SGD) methods. SGD optimization has been shown to be effective for optimization of conditional random fields in other contexts (Vishwanathan *et al.* 2006). It should be noted that while some other researchers have found the L-BFGS algorithm to be effective for optimization (Sutton, Rohanimanesh, & McCallum 2004), we found SGD to provide much faster convergence in our experiments.

Let $\nabla\theta_k(\mathbf{x})$ be the gradient of the likelihood function on training instance $\mathbf{x}$ under model θ_k. The standard SGD algorithm begins with an initial model $\theta_0 = (\lambda,\mu)$ and iteratively updates the model by selecting a single training example $\mathbf{x}$, calculating the gradient $\nabla\theta_{k-1}(\mathbf{x})$, and updating the model with the equation

$$\theta_k = \theta_{k-1} - \eta_k \nabla\theta_{k-1}(\mathbf{x})$$

where η_k is the *gain factor* for iteration k. We select training instances for each iteration by selecting a random permutation of an entire set of traces and proceeding through this permutation, one iteration per trace. The gain factor controls the rate of convergence of SGD, and much research has been concerned with effective selection and iterative updating of η_k. We begin with $\eta_1 = .1$ and follow a technique similar to that of (Y.Lecun *et al.* 1998) for gain selection. Our algorithm typically ran for 500-550 iterations.

In practice, we will probably not have access to a fully labeled data trace $\hat{\mathbf{x}}$; providing correct labels for every query about the state of the world at every point in a period of time can be quite expensive, if not impossible. We will thus assume that each slice $\hat{\mathbf{x}}_i$ may be only partially labeled (perhaps very sparsely so), with some set of variables y_t unlabeled at each t. The computation of each term in LL is then

$$\log P(\hat{\mathbf{x}}_\mathbf{t}|\hat{\mathbf{x}}_{\mathbf{t-1}}) = \log[\sum_{\hat{\mathbf{y}}_\mathbf{t}} \exp((\sum_\mathbf{i} \lambda_\mathbf{i}\phi_\mathbf{i}(\hat{\mathbf{x}}_\mathbf{t}|_{\hat{\mathbf{y}}_\mathbf{t}})) + \sum_i \mu_i\psi_i(\hat{\mathbf{x}}_\mathbf{t}|_{\hat{\mathbf{y}}_\mathbf{t}},\hat{\mathbf{x}}_{\mathbf{t-1}}|_{\hat{\mathbf{y}}_{\mathbf{t-1}}}))] - \log \mathbf{Z}_{\hat{\mathbf{x}}_{\mathbf{t-1}}}$$
$$= \log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}},\hat{\mathbf{x}}_\mathbf{t}\setminus\mathbf{y_t}} - \log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}}}$$

where $\hat{\mathbf{x}}_\mathbf{t}|_{\hat{\mathbf{y}}_\mathbf{t}}$ is the partial assignment $\hat{\mathbf{x}}_\mathbf{t}$ with the unlabeled variables set to $\hat{y}_t$. Unfortunately, both $\log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}},\hat{\mathbf{x}}_\mathbf{t}\setminus\mathbf{y_t}}$ and $\log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}}}$ can be prohibitively expensive to calculate, even with many standard approximation techniques such as MCMC. To efficiently calculate an approximation to both $\log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}},\hat{\mathbf{x}}_\mathbf{t}\setminus\mathbf{y_t}}$ and $Z_{\hat{\mathbf{x}}_{t-1}}$, we use the Bethe free energy approximation technique, described in (Yedidia, Freeman, & Weiss 2004). This technique approximates the value of $\log Z_{\hat{\mathbf{x}}_{t-1}}$ using marginal probabilities of the variables x_{i_t} and feature functions ϕ_j,ψ_j, and this approximation takes the form

$$\log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}}} = \sum_{\hat{\mathbf{x}}_\mathbf{t}} -[\sum_{\phi_j} P(\phi_j)\log\lambda_j\phi_j(\hat{\mathbf{x}}_\mathbf{t}) +$$
$$\sum_{\psi_j} P(\psi_j)\log\mu_j\psi_j(\hat{\mathbf{x}}_\mathbf{t},\hat{\mathbf{x}}_{\mathbf{t-1}}) - \sum_{\phi_\mathbf{j}} \mathbf{P}(\phi_\mathbf{j})\log\mathbf{P}(\phi_\mathbf{j}) +$$
$$\sum_{\psi_j} P(\psi_j)\log P(\psi_j)] + \sum_j (d_j - 1)\sum_k P(\hat{x}_{k_t})\log P(\hat{x}_{k_t})$$

where d_k is the *degree* of variable x_k, or the number of features ϕ_j and ψ_j dependent upon x_k. Since the marginals can generally be calculated correctly and efficiently using BP on the entire model with no variables fixed, this can be computed efficiently for learning, and we have generally found it to be a good approximation in practice. Similarly, the Bethe approximation of $\log Z_{\hat{\mathbf{x}}_{\mathbf{t-1}},\hat{\mathbf{x}}_\mathbf{t}\setminus\mathbf{y_t}}$ may be calculated by performing BP with the variables fixed by x_t and x_{t-1} clamped to the appropriate values.

To improve our optimization procedure, we make two more changes. First, it should be noted that our optimization procedure may result in overfitting. To limit this, we actually optimize the function $RLL(\lambda,\mu) = LL(\lambda,\mu) - N(\lambda,\mu)$, where $N(\lambda,\mu)$ is a Gaussian function of the vector (λ,μ) with mean zero and variance η (set to 5 in practice). This term penalizes excessively large weights in the vectors λ and μ. We then have $\frac{\partial RLL}{\partial\lambda_j} = \frac{\partial LL}{\partial\lambda_j} - \frac{\lambda_j}{\eta}$ and $\frac{\partial RLL}{\partial\mu_j} = \frac{\partial LL}{\partial\mu_j} - \frac{\mu_j}{\eta}$, and may optimize as described.

Second, in practice, because true instances of variables are relatively rare, the optimization procedure may favor a model which is disproportionately likely to label variables as false. 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 query variables $q_1...q_l$, let t_{q_i} be the number of true appearances of q_i in a timeslice in the training data $\hat{\mathbf{x}} = \hat{\mathbf{x}}_\mathbf{1}...\hat{\mathbf{x}}_\mathbf{d}$, and $\alpha(q_i) = \frac{d-t_{q_i}}{t_{q_i}}$. In learning, we then weight the term $P(\hat{\mathbf{x}}_\mathbf{t}|\hat{\mathbf{x}}_{\mathbf{t-1}})$ by multiplying it by $\sum_{j=1}^{l} 1_{q_j,i}\alpha(q_j)$ where $1_{q_j,i}$ is 1 if q_j is labeled true in x_i 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.

Discovering Context

While one can perform inference on a full double timeslice graph with BP to infer the probability of the state of the environment, it can be computationally quite expensive. In (Pentney *et al.* 2006) this issue was resolved through pruning the graph to a subgraph G', defined by the union of all

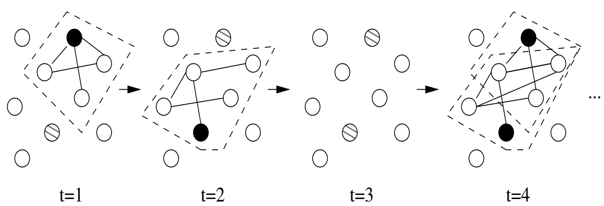

Figure 1: Example of our dynamic graphical model with use of context for inference. Shaded nodes represent observations, and are true when black; the dashed regions are the observations' contexts. When observations are true, features in those observations' contexts are applied to that timeslice. The directed arrows represent the temporal features between slices.

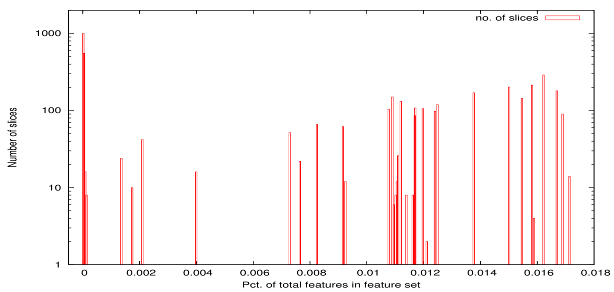

Figure 2: Distribution of sizes of feature sets over time slices with context-based pruning.

breadth-first traversals to a depth d on each fact p_i (in the paper, $d = 2$); this eliminates a large number of the nodes and features in the graph. While this is an effective technique, it requires foreknowledge of what predicates one wishes to query over before constructing the graph.

To improve efficiency in another manner, we take advantage of intuition about the structure of our graph. Our collection of everyday commonsense predicates contains many predicates whose respective states are likely to be mostly unrelated. For example, if we observe that a kitchen implement has been used, this is unlikely to affect the state of the bathroom sink, whose intuitive "context" is different. Thus, if observations relating to the bathroom sink have not changed at this time as well, it is unlikely that the bathroom sink will be affected by the new observations this time slice. Given this intuition, we seek to partition the timeslice graph into subgraphs based on the features defined upon the graph. We refer to this as *context-based pruning*.

Most variables, at any given point in time, are false. Given an observation o represented by random variable x_i, it is our goal to find an appropriate set of nodes C_o which are most likely to become true as a result of o being true. To find these nodes, we calculate the point-wise mutual information between each x_i and o for true values of x_i o in a training set $\hat{\mathbf{x}}$. The point-wise mutual information between x_i and o for values $\hat{x}_i$ and $\hat{o}$ in $\hat{\mathbf{x}}$ is calculated as $PMI(\hat{x}_i, \hat{o}) = \log \frac{P(x_i = \hat{x}_i, o = \hat{o})}{P(x_i = \hat{x}_i)P(o = \hat{o})}$.

If x_i and o are both fully labeled in the training data, these values may be computed in the training data $\hat{\mathbf{x}}$ by counts, i.e. $P(x_i = \hat{x}_i, o = \hat{o}) = \frac{1}{d} 1_{\hat{x}_i, \hat{o}}$, where $1_{\hat{x}_i, \hat{o}} = 1$ if $x_i = \hat{x}_i$ and $o = \hat{o}$ and 0 otherwise. However, x_i might not be labeled in the training data. In this case, we may estimate $PMI(\hat{x}_i, \hat{o})$ by estimating $P(x_i = \hat{x}_i, o = \hat{o}), P(x_i = \hat{x}_i)$ and $P(o = \hat{o})$. We estimate these values by performing BP on our existing model with all labeled x_i fixed appropriately.

To find the context of an observation o, we then select a threshold ϵ_o and let the context of o be $C_o = \{x_i : PMI(x_i, o) > \epsilon_o\}$. In our experimental environment, which contained 24 distinct observations, the size of the resulting contexts ranges from a couple of nodes to several thousand. This technique permits the pruning of many features from the graph at each timeslice that are unlikely to carry meaningful information about queries based on observations, and many nodes which are not sufficiently relevant to any possible observation may be pruned from the graph.

We produce a set of contexts $C = (C_{o_1}, C_{o_2}, ...C_{o_r})$ for the full set of observations by calculating each C_{o_i} based on the training data as described. Let $\zeta(C) = \{\phi_i : x_{c_i} \subset C\}$, $\mathbf{O_t}$ be the set of all observations o_i set to true in assignment $\mathbf{x_t}$, and $\zeta_{\mathbf{x_t}} = \zeta(\bigcup_{o_i \in \mathbf{O_t}} C_{o_i})$. To perform inference, we then compute the probability

$$P(\mathbf{x_t}|\mathbf{x_{t-1}}) = \frac{\mathbf{1}}{\mathbf{Z_{x_{t-1}, \zeta_{x_t}}}} \exp((\sum_{\mathbf{i}, \phi_\mathbf{i} \in \zeta_{\mathbf{x_t}}} \lambda_\mathbf{i} \phi_\mathbf{i}(\mathbf{x_t})) + (\sum_i \mu_i \psi_i(\mathbf{x_t}, \mathbf{x_{t-1}})))$$

Note that here the normalizing constant $Z_{\mathbf{x_{t-1}}, \zeta_{\mathbf{x_t}}}$ is dependent on both x_{t-1} and $\zeta_{\mathbf{x_t}}$. To summarize, here we are performing inference over the same graphical model as in our initial model, only with all features ϕ_i dependent upon nodes that are *not* in the context of the observations true at time t removed from the model. The features between time slices $t-1$ and t for each node remain in the equation. As with the initial model, we use BP to efficiently approximate the marginals of $\mathbf{x_t}$ in this model.

Under this scheme, contexts whose variables are more highly correlated are produced. Intuitively, this corresponds to the discovery of subsets of predicates which have strong relationships with each other, and likely share a similar real-life context (e.g. random variables relating to kitchen implements are likely to be more correlated with `location(kitchen)`).

Figure 1 depicts context-based pruning. Different time slices, as the figure shows, will have different feature sets, and sometimes even have no features in $\zeta_{\mathbf{x_t}}$ within the timeslice. In Figure 2, we plot the sizes of $\zeta_{\mathbf{x_t}}$ as a percentage of total graph features over the time slices in our experimental data. The average number of features, not including temporal features, used for inference at each time slice was ≈ 725, compared with 89,356 for the full, unpruned model.

Experiments

In our experimental evaluation, we will try to answer each of the following questions: (1) Do learning techniques help

No.	Method	Learn	Prec	Recall	Acc	F-measure	Train time	Inf. time
1	Full graph	No	10.84%	30.12%	78.84%	15.94	-	8:13
2	$d = 2$	No	24.85%	51.34%	81.81%	33.48	-	2:42
3	$d = 2$	FL/SGD	19.12%	45.87%	83.47%	26.99	39:38	2:41
4	$d = 2$	Bethe/SGD	69.31%	43.53%	94.87%	53.47	54:52	2:41
5	Context	No	16.43%	72.18%	73.58%	26.76	-	1:12
6	Context	Bethe/SGD	36.66%	53.28%	90.76%	42.87	43:13	1:12

Figure 3: A comparison of inference methods by accuracy, precision, recall, and F-measure.

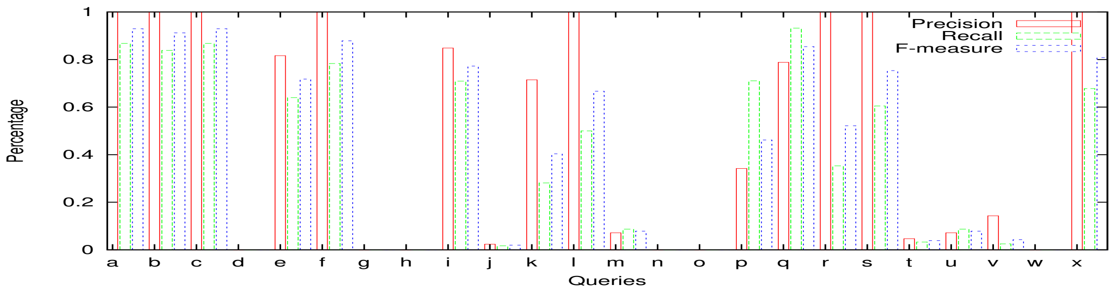

Figure 4: Precision, recall, and F-measure numbers for each query with learning and inference on the graph pruned to $d = 2$.

in state prediction, and how? (2) How does learning help on unlabeled queries? (3) What sort of queries can we predict well? (4) How much does context-based pruning help with scaling? (5) Does PMI-based context-based pruning help?

For our experimental evaluation, we used traces of household object usage 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). A total of approximately 70-75 minutes of data was collected, with traces divided into time slices of 2.5 seconds. For these activities, a set of 24 variables in the collected set of queries, such as `stateof(cereal,prepared)`, were labeled as true or false for each time slice.

To answer question (1), we ran inference algorithms in different settings, measuring the accuracy of our system in predicting the value of the 24 queries. Since the majority ($\approx$93%) of queries are false, and intuitively we are somewhat more interested in being able to predict cases where a query is *true*, we also measure precision and recall with respect to labeling the true instances of each query, and compute the F-measure, the harmonic mean of the two values (a standard measure of effectiveness in information retrieval). For experiments involving learning, we performed training on $\approx$25% of the data using SGD, and performed prediction. We used two versions of learning: one in which we set all unlabeled values to false, and one in which we find marginals for unlabeled values using BP. We used MAP estimation for prediction in the learned case; when learning techniques were not used, we used the thresholding technique used in SRCS to compute predictions from the resulting marginal probabilities computed.

Results of our prediction can be seen in Figure 3. In order by row, this table displays results for prediction (1) on the full, unpruned model; (2) without learning and with the graph pruned to depth $d = 2$ from the selected queries; (3) with learning by setting all unlabeled values in the (partially labeled) training data to false; (4) with learning using the Bethe method; (5) using context-based pruning without learning; and (6) using context-based pruning with learning using the Bethe method. Note that, due to some slight changes in the SRCS model, results are slightly different from (Pentney *et al.* 2006) for (2), although the same experiment is run in that paper. We see that adding learning to the system provides notable increases in both accuracy and F-measure. Using the Bethe approximation for learning provides far better precision than false-labeled learning; while faster, false-labeled learning offers an inaccurate depiction of the relationship between labeled values, and thus learns a worse model.

To answer (2), we consider the precision, recall, and F-measure values for each of the 24 queries with learning added; a plot of each may be seen in Figure 4. As we can see, performance is somewhat uneven, with some queries being predicted extremely well and others with less accuracy. The poor performance of some queries is, in some cases, due to difficulty in measuring their accuracy based on use of household objects alone. For instance, query (o) is the random variable `stateof(window, dirty)`; while the system can learn to predict a clean window (i.e. `stateof(window, dirty) = false`) based on usage of window cleaner and/or towels, predicting a dirty window is more difficult and does not correspond well to any objects used in the data. A more sophisticated model which takes into account observations for some period both before and after the current timeslice may adequately account for queries such as these. Similar problems can be seen in queries (d) (`stateof(teeth,clean)`), (h) (`stateof(cereal,`

Query	No learning		Learning	
	Acc	F-mea	Acc	F-mea
activity(brushing teeth)	96.12%	94.11	99.47%	91.19
location(shower)	89.86%	0.00	92.99%	47.48
action(eat)	87.61%	25.36	97.12%	31.81
action(add milk to)	99.04%	0.00	99.04%	0.00
stateof(tea kettle, hot)	72.26%	31.92	95.97%	55.32
stateof(window, dirty)	96.89%	0.00	93.47%	5.55
location(greenhouse)	61.65%	33.28	94.87%	65.37
Aggregate	86.20%	31.74	96.13%	51.04

Figure 5: Accuracy and F-measure for inference with model trained on half (12) of the labeled queries on seven other queries.

Method	Avg. time/slice (s)	No. features
full graph	15.576	130642
$d = 1$ subgraph	2.44	4370
$d = 2$ subgraph	5.12	24465
$d = 3$ subgraph	9.38	49825
context discovery	1.08	725 (avg.)

Figure 6: Time for inference per slice for different inference methods, and the number of features per slice on the model.

`prepared)`), and (n) (`stateof(cereal,eaten)`). It seems that much of the poorer performance can be attributed to (1) the somewhat subjective labeling of the data (assertions like "the window is clean" can be a matter of judgment in some cases), (2) gaps in the commonsense knowledge in the system, and (3) queries that are not well linked to the relatively coarse observations provided by our system. Improvement on these types of queries may demand a more complex model or supplemental sources of observations.

We attempt to answer (3) by training a model with only 12 of the 24 queries, with the graph pruned to $d = 2$. This graph contained seven other labeled queries whose labels were not given; we test performance on predicting these queries against that of the model without learning. We see in Figure 5 that most of the queries have improved accuracy and F-measure, even though their labels were not given, showing that our commonsense learning can help improve prediction of unlabeled variables as well.

We answer (4) by measuring the average time inference per slice for different inference methods. An example of the time speedup provided by context-based pruning is given in Figure 6. Finding contexts imposes a one-time overhead cost, but improves the efficiency of inference when used for prediction. Here we compare the average amount of time spent on inference per timeslice in our full graph; the time spent on inference per timeslice in a graph pruned with d set to 1, 2, and 3; and the time spent on inference per timeslice on a graph pruned via context-based pruning. Inference using context takes less time than any of these methods, on average; in a real-life common sense reasoning application,

Method	Acc	F-mea	Time/slice
$(d = 2)$-based context	79.72%	22.26	1.01
PMI-based context	90.65%	22.98	0.96

Figure 7: PMI-based contexts outperform simple $(d = 2)$ contexts.

which could conceivably have an even larger number of variables to reason over, this speedup could be quite useful, even at the cost of some accuracy.

Finally, to answer (5) and show the value of our technique for context-based pruning, we test our PMI-based algorithm for context-based pruning against a "dumber" algorithm which merely sets a context for every node by selecting every node a depth of $d = 2$ away from each observation as its context. We perform learning on 12 of the queries, and then perform inference to predict all 24. The results are in Figure 7. We see that our algorithm provides increased accuracy in these experiments, suggesting the value of our more sophisticated context-based pruning technique.

Conclusions

We have presented a system for learning models of the everyday human environment using statistical techniques and preexisting common sense data. We have shown that the learning techniques can provide considerable improvements in prediction within our model, exceeding previously established techniques. We have also presented a technique for performing discovery of context for common sense information using partially labeled data to provide more efficient inference, and demonstrated the utility of this technique. We believe this system has strong potential as a means of enabling statistical reasoning about the everyday human environment on a large scale.

References

Fishkin, K. P.; Philipose, M.; and Rea, A. 2005. Hands-on RFID: Wireless wearables for detecting use of objects. In *ISWC 2005*, 38–43.

Gupta, R., and Kochenderfer, M. J. 2004. Common sense data acquisition for indoor mobile robots. In *AAAI*, 605–610.

Lenat, D., and Guha, R. V. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

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*.

Sutton, C.; Rohanimanesh, K.; and McCallum, A. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. of 21st International Conference on Machine Learning (ICML)*.

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

Vishwanathan, S.; Schraudolph, N.; Schmidt, M.; and Murphy, K. 2006. Accelerated training of conditional random fields with stochastic meta-descent. In *Proceedings of ICML*.

Wilson, D.; Consolvo, S.; Fishkin, K.; and Philipose, M. 2005. In-home assessment of the activities of daily living of the elderly. In *Extended Abstracts of CHI 2005: Workshops - HCI Challenges in Health Assessment*.

Yedidia, J.; Freeman, W.; and Weiss, Y. 2004. Constructing free energy approximations and generalized belief propagation algorithms. *Mitsubishi Electronic Research Labs Technical Report TR 2002-35*.

Y.Lecun; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86 (11):2278–2324.

Zhu, X. 2005. Semi-supervised learning literature survey. *Computer Sciences TR 1530, University of Wisconsin, Madison*.