Comparison of Different Methods for Next Location Prediction

Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer

Institute of Computer Science, University of Augsburg,
Eichleitnerstr. 30, 86159 Augsburg, Germany
{petzold, bagci, trumler, ungerer}@informatik.uni-augsburg.de

Abstract. Next location prediction anticipates a person's movement based on the history of previous sojourns. It is useful for proactive actions taken to assist the person in an ubiquitous environment. This paper evaluates next location prediction methods: dynamic Bayesian network, multi-layer perceptron, Elman net, Markov predictor, and state predictor. For the Markov and state predictor we use additionally an optimization, the confidence counter. The criterions for the comparison are the prediction accuracy, the quantity of useful predictions, the stability, the learning, the relearning, the memory and computing costs, the modelling costs, the expandability, and the ability to predict the time of entering the next location. For evaluation we use the same benchmarks containing movement sequences of real persons within an office building.

1 Introduction

Can the movement of people working in an office building be predicted based on room sequences of previous movements? In our opinion people follow some habits, but interrupt their habits irregularly, and sometimes change their habits. Moreover, moving to another office fundamentally changes habits too. Thus location prediction methods need to exhibit some features: high prediction accuracy, a short training time, retention of prediction in case of irregular habitual interrupts, but an appropriate change of prediction in case of habitual changes.

Location predictions with such features could be used for a number of applications in ubiquitous and mobile environments.

- Smart doorplates that are able to direct visitors to the current location of an office owner based on a location-tracking system and predict if the office owner is soon coming back [14].
- Similarly, next location prediction within a smart building can be used to prepare the room which is presumably entered next by a habitant, e.g. by phone call forwarding.
- Outdoor movement patterns can be used to predict the next region a person will enter.
- Elevator prediction could anticipate at which floor an elevator will be needed next.

W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 909–918, 2006.

– Routing prediction for cellular phone systems may predict the next radio cell a cellular phone owner will enter based on his previous movement behaviour.

We considered the first application in more detail and used benchmarks with movement data of four persons over several months. The benchmarks are called Augsburg Indoor Location Tracking Benchmarks. They are publicly available [9], and are applied to evaluate several prediction techniques and to compare the efficiency of these techniques with exactly the same evaluation set-up and data.

Our aim is to investigate how far machine learning techniques can dynamically predict room sequences and time of room entry independent of additional knowledge. Of course the information could be combined with contextual knowledge as e.g. the office time table or personal schedule of a person, however, in this paper we focus on dynamic techniques without contextual knowledge.

Time of arrival at the predicted location depends on the sojourn time at the current location plus the rather constant time to move to the predicted location. The sojourn time was modelled into the presented Bayesian network. We tested also a time prediction which calculated the mean and the median of the previous sojourn times within a location. The best results were reached by the median. The time prediction is independent of the location prediction method and can easily be combined with any of the regarded methods. Therefore we restrict this comparison to location prediction only.

Several prediction techniques are proposed in literature — namely Bayesian networks, Markov models or Hidden Markov models, various neural network approaches, and the state predictor methods. The challenge is to transfer these algorithms to work with context information. In this paper we choose five approaches, a dynamic Bayesian network, a multi-layer perceptron, an Elman net, a Markov predictor, and a state predictor. In the case of the Markov predictor and the state predictor we use additionally a version which is optimized by confidence estimation.

There are a lot of methodological problems for a fair comparison of such diverse methods. The models are different and hard to compare. We chose the same set-up to model all methods and for each method the best model that we could find. Moreover we had the choice either to combine all persons within a single model thus potentially making improvements by detecting correlations between person movements or to model each person separately. We chose the latter simpler model because simulations with the combined model using the Augsburg Benchmarks showed no improvements.

The main criterion for comparison is the average prediction accuracy of the different methods. Another question concerns the model and the modelling costs of the technique. Which parameters exist and influence the model? What happens if one parameter is changing? We call this the stability of the techniques. Can the model simply be extended by more or other locations? The answer to this question allows to assess how well the model can be transferred to other applications.

Further interesting questions concern the efficiency of training of a predictor, before the first useful predictions can be performed, and of retraining, i.e. how long it takes until the predictor adapts to a habitual change and provides again useful predictions. Predictions are called useful if a prediction is accurate with a certain confidence level. Moreover, memory and performance requirements of a predictor are of interest in particular for mobile appliances with limited performance ability and power supply.

The next section states related work on context prediction. Section 3 introduces shortly the five approaches and the applied location models. For detailed information about the basic techniques use the stated references. Section 4 gives the evaluation results. The paper ends with the conclusions.

2 Related Work

The Adaptive House project [7] of the University of Colorado developed a smart house that observes the lifestyle and desires of the inhabitants and learned to anticipate and accommodate their needs. Occupants are tracked by motion detectors and a neural network approach is used to predict the next room the person will enter and the activities he will be engaged. Patterson et al. [8] presented a method of learning a Bayesian model of a traveller moving through an urban environment based on the current mode of transportation. The learned model was used to predict the outdoor location of the person into the future.

Markov chains are used by Kaowthumrong et al. [5] for active device selection. Ashbrook and Starner [1] used location context for the creation of a predictive model of user's future movements based on Markov models. They propose to deploy the model in a variety of applications in both single-user and multi-user scenarios. Their prediction of future location is currently time independent, only the next location is predicted. Bhattacharya and Das [2] investigate the mobility problem in a cellular environment. They deploy a Markov model to predict future cells of a user. An architecture for context prediction was proposed by Mayrhofer [6] combining context recognition and prediction. Active LeZi [4] was proposed as good candidate for context prediction.

There are several publication of our group which present next location prediction in an office building. In [10] we proposed the basic state predictor technique which is similar to the Markov predictor, but an automaton is used for the prediction. In [11] an enhancement by confidence estimation techniques is presented. Vintan et al. [15] applied a multi-layer perceptron and Petzold et al. [12] proposed a dynamic Bayesian network to predict indoor movements of several persons.

The contribution of this paper is the comparison of five different prediction methods including the new Elman net approach and the confidence estimation applied to the Markov predictor. According to our knowledge no comparative studies of different methods with the same evaluation setups and benchmarks exist.

3 Prediction Methods

Figure 1 shows the next location prediction principle which is used by each investigated model. The input consists only of the sequence of the last visited locations and the entry time of these locations. The output is the possible next location and the appropriate entry time.

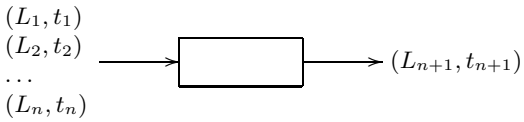


Fig. 1. Next location prediction

Dynamic Bayesian Network

In order to predict the next location of a person, a dynamic Bayesian network was chosen. Additionally the time is predicted when the person is probably entering the next location. In different simulations we looked for the best settings [12]. We detected that the prediction of next location is independent of the time parameter like the time of day and the weekday. Therefore we chose this proposed dynamic Bayesian network without these time dependencies for the comparison. As history we elected 2 for a better comparison based on similar memory costs.

Multi-Layer Perceptron

For next location prediction we chose the simplest multi-layer perceptron with one hidden layer and used a modified back-propagation algorithm for learning [15]. In principal each location would be represented by a single input and a single output neuron. However, we chose a binary encoding because it saves computing costs. This fact is interesting for mobile devices which must achieve some energy and real-time restrictions. The optimal parameter values for the network structure and the learning algorithm were determined by many simulation runs and are summarized in table 1.

Table 1. Optimal parameter values of the multi-layer perceptron

parameter	investigated values	optimal value
network structure		
history	[1;6]	2
number of hidden neurons	{5;7;...;15}	9
learning algorithm		
threshold	{0.1;0.3;...;0.9}	0.1
learning rate	[0.05;0.30]	0.10
number of backward steps	[1;5] and unlimited	1

Elman Net

The Elman net is another neural network method which expands the multi-layer perceptron by another hidden layer – the context layer. The context neurons provide storage for the activation states of the hidden neurons. This generates a dependency between two propagations within the net, since the hidden neurons get information from the input and the context neurons across the weighted connections to perform the next step. The number of the context neurons corresponds with the number of hidden neurons. To find the optimal parameters of the net many simulations were performed. Table 2 shows the investigated and the optimal values of the parameters separated in parameter of the network structure and the learning algorithm. Since the Elman net is a recurrent network, the information about previous locations is modelled in the context cells. Therefore the history consists only of the current location.

Table 2. Optimal parameter values of the Elman net

parameter	investigated values	optimal values
network structure		
encoding	binary, one to one	one to one
number of hidden neurons	[5;20]	5
history	[1;5]	1
learning algorithm		
initialization	random, fix	fix
activation function	$\tanh(x)$, $\frac{1}{1+\exp(-x)}$	$\tanh(x)$
learning cycles	[5;150]	31
learning rate η	[0.1;0.7]	0.1
momentum α	[0.00;0.05;...;0.95]	0.00

Markov Predictor

Markov models seem a good approach for the next location prediction based on location histories. A Markov model regards a pattern of the last visited locations of a user to predict the next location. The length of the regarded pattern is called the order. Thus a Markov model with order 3 uses the last three visited locations. For all patterns the model stores the probabilities of the next location which is calculated from the whole sequence of the visited locations by the user. A simple Markov model is the Markov predictor [3,13]. A Markov predictor stores for every pattern the frequencies of the next locations. For the comparison we chose an order of 2. Furthermore we will compare a Markov predictor which is optimized by confidence estimation [11].

State Predictor

A disadvantage of the Markov predictor is its bad relearning capability because of the frequency counter. After a habit change the new habit must be followed as

often as the previous habit before the prediction is changed. The state predictors [10] prevent this problem. They use a finite automaton which is called two-state predictor for every pattern thus replacing the frequency counter of the Markov predictor. A state predictor with order 2 is used in the comparison.

The basic state predictor method can be significantly improved by some confidence estimation techniques [11]. One of the proposed methods, the confidence counter method, is independent of the used prediction algorithm. This method estimates the prediction accuracy with a saturation counter. Figure 2 shows a two-bit counter that consists of 4 states.

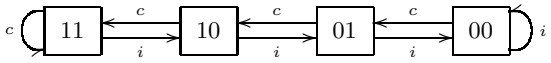


Fig. 2. Confidence counter

The initial state is state 10. Let s be the current state of the confidence counter. If a prediction result is proved as correct (c) the counter will be incremented, that means the state graph changes from state s into the state $s + 1$. If $s = 11$ the counter keeps the state s. Otherwise if the prediction is incorrect (i) the counter switches into the state $s - 1$. If $s = 00$ the counter keeps the state s. If the counter is in the state 11 or 10 the predictor is assumed as confident, otherwise the predictor is unconfident and the prediction result will not be supplied.

For the state predictor the prediction accuracy will be considered separately for every pattern. The confidence counter can also be applied with other techniques, in the evaluation a Markov predictor using the counter will be considered.

4 Evaluation

To evaluate the five techniques we chose the Augsburg Indoor Location Tracking Benchmarks taken from the Context Database of the University of Linz [9]. These benchmarks consist of two sets, the summer and the fall data. The used benchmarks contain the movements of four persons in an office building. The prediction accuracy is calculated for every person for all predictions from all rooms except the own office. For our comparison we tried to use models with similar memory costs. Therefore we didn't always choose the best setting for every technique. In fact we elected history 2 for the Markov predictor and the state predictor, which perform better with longer history but at the expense of large tables.

In the following we will compare all techniques on the basis of different criterions. The results are summarized in table 3.

Table 3. Comparison on the basis of the criterions

	Bayesian network	multi-layer perceptron	Elman net	Markov predictor	state predictor	Markov predictor with counter	state predictor with counter
accuracy (%) (quantity (%))	78.82 (89.89)	76.45 (≈ 100)	79.68 (100)	76.53 (90.47)	70.89 (90.47)	81.14 (78.40)	81.88 (74.38)
stability (%)	29.67	32.59	71.57	24.67	29.97	24.95	23.99
learning	fast	slow	slow	fast	fast	fast	fast
relearning	slow	slow	slow	slow	fast	slow	fast
memory (bit)	6,500	3,880	7,215	36,960	2,730	37,380	3,150
computing costs	inefficient chain rule	training until $E < t$, otherwise one propagation	training over many learning cycles, otherwise one propagation	table look-up	table look-up	table look-up	table look-up
modelling costs	medium	high	high	low	low	low	low
expandability	yes	no	no	yes	yes	yes	yes
time prediction	integrated	parallel	parallel	parallel	parallel	parallel	parallel

Prediction accuracy. The prediction accuracy is calculated with the fall data; the summer data is used for the training. We assume that a prediction is needed after every location change. That means the number of requested predictions p is equal to the number of location changes. The Bayesian network, the Markov predictor and the state predictor cannot predict the next location if the current pattern occurs the first time. The number of predictions which cannot be delivered by these three techniques were denoted by p_n. In contrast the neural networks deliver a prediction if the code of the output vector corresponds to a location. Thus the Elman net predicts always a location ($p_n = 0$). Now we can determine the number of deliverable predictions $p_d = p - p_n$. Let c be the number of correct predictions then the prediction accuracy a is calculated as follows:

$$a = \frac{c}{p_d}$$

It isn't essential to make a prediction in our application, rather a prediction is an added value. Therefore we consider in the calculation of the accuracy only predictions which provide a result.

Table 3 shows the average prediction accuracy of the four persons for all techniques. If we consider the five techniques without confidence counter enhancement, the Elman net reaches the highest average prediction accuracy. If we

consider the state predictor and Markov predictor using the confidence counter and compare them with all other methods, the state predictor with confidence counter delivers now the best accuracy.

Quantity. The number of deliverable predictions p_d is smaller than the number of requested predictions p. This gap can be determined by the quantity q:

$$q = \frac{p_d}{p}$$

The Elman net reaches a quantity of 100% because the net produces always an output vector. The quantity of the multi-layer perceptron is nearly 100% since not every code of the output vector corresponds with a location. The Bayesian network, the Markov predictor and the state predictor reach nearly the same quantity. With an optimization like the confidence estimation the quantity decreases.

Stability. The stability shows the impact of the change of a parameter. For the Bayesian network the history and the time parameters are a possibility to optimize the prediction accuracy. The multi-layer perceptron and the Elman net hold a multitude of parameters which can influence the prediction accuracy. Therefore the Elman net shows the worst stability. The Markov predictor and the state predictor give only the possibility to choose the order. Table 3 shows as stability the difference between the minimum and the maximum of the prediction accuracies reached with different parameters.

Learning. The learning phase of the neural networks takes a long time since the networks must be trained before they can be effectively used. The Bayesian network, the Markov predictor and the state predictor with and without confidence counter could make a prediction already after the second occurrence of a pattern.

Relearning. The neural networks, the Markov predictor without and with confidence counter need a long time for relearning. The Bayesian network relearns also slow. The state predictor with and without confidence counter relearns after two changes the new habit.

Memory costs. For the memory costs we calculated the minimal number of bits which will be needed to store the current state of the technique. For the evaluation all models were chosen to exhibit similar memory costs. In general, the memory costs of both neural networks are very low and independent from the number of location changes of a person. The Bayesian network needs only a small memory space, but the memory depends on the number of location changes. The state predictor requires the least memory. Against this the Markov predictor has the highest memory costs since the Markov predictor stores all frequencies. The costs of both predictors are dependent on the number of location changes. The confidence counter arises the costs insignificantly. Table 3 shows the memory costs for an upper limit of 500 location changes.

Computing costs. Because of the training process the computing costs of the neural networks are very high. The Elman net needs many learning cycles

and the multi-layer perceptron will be trained until the error is less than a threshold. The computing costs during the use of both neural networks are low because both execute only one backward propagation. The Bayesian network calculates the probabilities by the chain rule resulting in relatively high computation costs. The computing costs of the Markov predictor and the state predictor with and without the confidence counter consist of one table look-up.

Modelling efforts. For modelling the Bayesian network possible dependencies of the used variables must be extracted from the available data. Both neural networks require a high effort for modelling generated by the search for the optimal parameters. The costs for modelling the Markov predictor and the state predictor are low. A decision will only be needed concerning the length of the order. If a confidence counter is used the number of the counter states and the barrier must be determined.

Expandability. The expandability means the possibility to use the model with more locations. In the used benchmarks there are 15 locations. If we use a scenario with locations like the cells in a mobile network, the neural network models cannot be reused. A new modelling process with search for the optimal parameter is necessary. That means the neural network models cannot be reused without additional costs in another application. The Bayesian network, the Markov predictor and the state predictor with and without confidence counter can be expanded for more context without additional costs.

5 Conclusion

The paper compared five prediction techniques on the basis of different criterions. The comparison of the different techniques showed that there isn't an ultimative prediction technique. The user must decide which is the most important criterion for the application.

The Elman net reached the highest prediction accuracy, since it is a recurrent neural network which is affected by previous inputs. But both neural networks require high modelling costs, additional costs to expand for more contexts, and show the lowest stability. If the time prediction is the most important criterion the Bayesian network must be chosen. The state predictor should be applied if the prediction accuracy or the memory costs are the main facts. Compared to the Markov predictor, the state predictor relearns faster and uses less memory. The use of the confidence counter improves the prediction accuracy of state and Markov predictors.

Next step could be the test of a hybrid predictor which uses different techniques in parallel. A selector within the hybrid predictor selects the estimated best prediction among these different predictors. With the hybrid predictor the advantages of the different methods can be joined. A further question is: can the confidence estimation also improve the prediction accuracy of the other approaches. In this paper we considered only next location prediction. A further investigation should be to expand the techniques to predict locations at a certain time in future.

References

1. Daniel Ashbrook and Thad Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.
2. Amiya Bhattacharya and Sajal K. Das. LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks. *Wireless Networks*, 8:121–135, 2002.
3. I-Cheng K. Chen, John T. Coffey, and Trevor N. Mudge. Analysis of Branch Prediction via Data Compression. In *ASPLOS VII*, pages 128–137, Cambridge, Massachusetts, USA, October 1996.
4. Karthik Gopalratnam and Diane J. Cook. Active LeZi: An Incremental Parsing Algorithm for Sequential Prediction. In *Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 38–42, St. Augustine, Florida, USA, May 2003.
5. Khomkrit Kaowthumrong, John Lebsack, and Richard Han. Automated Selection of the Active Device in Interactive Multi-Device Smart Spaces. In *Workshop at UbiComp'02: Supporting Spontaneous Interaction in Ubiquitous Computing Settings*, Gothenburg, Sweden, 2002.
6. Rene Mayrhofer. An Architecture for Context Prediction. In *Advances in Pervasive Computing*. Austrian Computer Society (OCG), April 2004.
7. Michael C. Mozer. The Neural Network House: An Environment that Adapts to its Inhabitants. In *AAAI Spring Symposium on Intelligent Environments*, pages 110–114, Menlo Park, CA, USA, 1998.
8. Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring High-Level Behavior from Low-Level Sensors. In *5th International Conference on Ubiquitous Computing*, pages 73–89, Seattle, WA, USA, 2003.
9. Jan Petzold. Augsburg Indoor Location Tracking Benchmarks. Context Database, Institute of Pervasive Computing, University of Linz, Austria. http://www.soft.uni-linz.ac.at/Research/Context_Database/index.php, January 2005.
10. Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer. Global and Local Context Prediction. In *Artificial Intelligence in Mobile Systems 2003 (AIMS 2003)*, Seattle, WA, USA, October 2003.
11. Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer. Confidence Estimation of the State Predictor Method. In *2nd European Symposium on Ambient Intelligence*, pages 375–386, Eindhoven, The Netherlands, November 2004.
12. Jan Petzold, Andreas Pietzowski, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer. Prediction of Indoor Movements Using Bayesian Networks. In *Location- and Context-Awareness (LoCA 2005)*, Oberpfaffenhofen, Germany, May 2005.
13. Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 1985.
14. Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer. Smart Doorplate. In *First International Conference on Appliance Design (1AD)*, Bristol, GB, May 2003. Reprinted in Pers Ubiquit Comput (2003) 7: 221-226.
15. Lucian Vintan, Arpad Gellert, Jan Petzold, and Theo Ungerer. Person Movement Prediction Using Neural Networks. In *First Workshop on Modeling and Retrieval of Context*, Ulm, Germany, September 2004.

Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior

Brian D. Ziebart, Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bziebart@cs.cmu.edu, amaas@andrew.cmu.edu, anind@cs.cmu.edu, dbagnell@ri.cmu.edu

ABSTRACT

We present *PROCAB*, an efficient method for Probabilistically Reasoning from Observed Context-Aware Behavior. It models the context-dependent utilities and underlying reasons that people take different actions. The model generalizes to unseen situations and scales to incorporate rich contextual information. We train our model using the route preferences of 25 taxi drivers demonstrated in over 100,000 miles of collected data, and demonstrate the performance of our model by inferring: (1) decision at next intersection, (2) route to known destination, and (3) destination given partially traveled route.

Author Keywords

Decision modeling, vehicle navigation, route prediction

ACM Classification Keywords

I.5.1 Pattern Recognition: Statistical; I.2.6 Artificial Intelligence: Parameter Learning; G.3 Probability and Statistics: Markov Processes

INTRODUCTION

We envision future navigation devices that learn drivers' preferences and habits, and provide valuable services by *reasoning* about driver behavior. These devices will incorporate real-time contextual information, like accident reports, and a detailed knowledge of the road network to help mitigate the many unknowns drivers face every day. They will provide context-sensitive route recommendations [19] that match a driver's abilities and safety requirements, driving style, and fuel efficiency trade-offs. They will also alert drivers of unanticipated hazards well in advance [25] – even when the driver does not specify the intended destination, and better optimize vehicle energy consumption using short-term turn prediction [10]. Realizing such a vision requires new models of non-myopic behavior capable of incorporating rich contextual information.

UbiComp'08, September 21-24, 2008, Seoul, Korea.

In this paper, we present PROCAB, a method for **P**robabilistically **R**easoning from **O**bserved **C**ontext-**A**ware **B**ehavior. PROCAB enables reasoning about context-sensitive user actions, preferences, and goals – a requirement for realizing ubiquitous computing systems that provide the right information and services to users at the appropriate moment. Unlike other methods, which directly model action sequences, PROCAB models the negative *utility* or *cost* of each action as a function of contextual variables associated with that action. This allows it to model the *reasons* for actions rather than the actions themselves. These reasons *generalize* to new situations and differing goals. In many settings, reason-based modeling provides a *compact* model for human behavior with fewer relationships between variables, which can be learned more precisely. PROCAB probabilistically models a distribution over all behaviors (*i.e.*, sequences of actions) [27] using the principle of maximum entropy [9] within the framework of inverse reinforcement learning [18].

In vehicle navigation, roads in the road network differ by type (*e.g.*, interstate vs. alleyway), number of lanes, and speed limit. In the PROCAB model, a driver's utility for different combinations of these road features is learned, rather than the driver's affinity for particular roads. This allows generalization to locations where a driver has never previously visited.

Learning good context-dependent driving routes can be difficult for a driver, and mistakes can cause undesirable travel delays and frustration [7]. Our approach is to learn driving routes from a group of experts – 25 taxi cab drivers – who have a good knowledge of alternative routes and contextual influences on driving. We model context-dependent taxi driving behavior from over 100,000 miles of collected GPS data and then recommend routes that an efficient taxi driver would be likely to take.

Route recommendation quality it is difficult to assess. Instead, we evaluate using three prediction tasks:

- **Turn Prediction**: What is the probability distribution over actions at the next intersection?
- **Route Prediction**: What is the most likely route to a specified destination?
- **Destination Prediction**: What is the most probable destination given a partially traveled route?

We validate our model by comparing its prediction accuracy on our taxi driver dataset with that of existing approaches. We believe average drivers who will use our envisioned smart navigation devices choose less efficient routes, but visit fewer destinations, making route prediction somewhat harder and destination prediction significantly easier.

In the remainder of the paper, we first discuss existing probabilistic models of behavior to provide points of comparison to PROCAB. We then present evidence that preference and context are important in route preference modeling. Next, we describe our PROCAB model for probabilistically modeling context-sensitive behavior compactly, which allows us to obtain a more accurate model with smaller amounts of training data. We then describe the data we collected, our experimental formulation, and evaluation results that illustrate the the PROCAB model's ability to reason about user behavior. Finally, we conclude and discuss extensions of our work to other domains.

BACKGROUND

In this section, we describe existing approaches for modeling human actions, behavior, and activities in context-rich domains. We then provide some illustration of the contextual sensitivity of behavior in our domain of focus, modeling vehicle route preferences.

Context-Aware Behavior Modeling

An important research theme in ubiquitous computing is the recognition of a user's current activities and behaviors from sensor data. This has been explored in a large number of domains including classifying a user's posture [17], tracking and classifying user exercises [5], classifying a user's interruptibility [6] and detecting a variety of daily activities within the home [26, 24]. Very little work, in comparison, looks at how to predict what users want to do: their goals and intentions. Most context-aware systems attempt to model context as a proxy for user intention. For example, a system might infer that a user wants more information about a museum exhibit because it observes that she is standing near it.

More sophisticated work in context-awareness has focused on the problem of predicting where someone is going or going to be. Bhattacharya and Das used Markov predictors to develop a probability distribution of transitions from one GSM cell to another [3]. Ashbrook and Starner take a similar approach to develop a distribution of transitions between learned significant locations from GPS data [2]. Patterson *et al*. used a Bayesian model of a mobile user conditioned on the mode of transportation to predict the users future location [20]. Mayrhofer extends this existing work in location prediction to that of context prediction, or predicting what situation a user will be in, by developing a supporting general architecture that uses a neural gas approach [15]. In the Neural Network House, a smart home observed the behaviors and paths of an occupant and learned to anticipate his needs with respect to temperature and lighting control. The occupant was tracked by motion detectors and a neural network was used to predict the next room the person was going to enter along with the time at which he would enter and leave the home [16].

However, all of these approaches are limited in their ability to learn or predict from only small variations in user behavior. In contrast, the PROCAB approach is specifically designed to learn and predict from small amounts of observed user data, much of which is expected to contain context-dependent and preference-dependent variations. In the next section, we discuss the importance of these variations in understanding route desirability.

Understanding Route Preference

Previous research on predicting the route preferences of drivers found that only 35% of the 2,517 routes taken by 102 different drivers were the "fastest" route, as defined by a popular commercial route planning application [13]. Disagreements between route planning software and empirical routes were attributed to contextual factors, like the time of day, and differences in personal preferences between drivers.

We conducted a survey of 21 college students who drive regularly in our city to help understand the variability of route preference as well as the personal and contextual factors that influence route choice. We presented each participant with 4 different maps labeled with a familiar start and destination point. In each map we labeled a number of different potentially preferable routes. Participants selected their preferred route from the set of provided routes under 6 different contextual situations for each endpoint pair. The contextual situations we considered were: early weekday morning, morning rush hour, noon on Saturday, evening rush hour, immediately after snow fall, and at night.

	Route						
Context	A	B	C	D	E	F	G
Early morning	6	6	4	1	2	2	0
Morning rush hour	8	4	5	0	1	2	1
Saturday noon	7	5	4	0	1	2	2
Evening rush hour	8	4	5	0	0	3	1
After snow	7	4	4	3	2	1	0
Midnight	6	4	4	2	1	4	0

Table 1. Context-dependent route preference survey results for one particular pair of endpoints

The routing problem most familiar to our participants had the most variance of preference. The number of participants who most preferred each route under each of the contextual situations for this particular routing problem is shown in Table 1. Of the 7 available routes to choose from (A-G) under 6 different contexts, the route with highest agreement was only preferred by 8 people (38%). In addition to route choice being dependent on personal preferences, route choice was often context-dependent. Of the 21 participants, only 6 had route choices that were context-invariant. 11 participants used two different routes depending on the context, and 4 participants employed three different context-dependent routes.

Our participants were not the only ones in disagreement over the best route for this particular endpoint pair. We generated route recommendations from four major commercial mapping and direction services for the same endpoints to compare against. The resulting route recommendations also demonstrated significant variation, though all services are context- and preference-independent. Google Maps and Microsoft's MapPoint both chose route E, while MapQuest generated route D, and Yahoo! Maps provided route A.

Participants additionally provided their preference towards different driving situations on a five-point Likert scale (see Table 2). The scale was defined as: (1) very strong dislike, avoid at all costs; (2) dislike, sometimes avoid; (3) don't care, doesn't affect route choice; (4) like, sometimes prefer; (5) very strong like, always prefer.

	Preference				
Situation	1	2	3	4	5
Interstate/highway	0	0	3	14	4
Excess of speed limit	1	4	5	10	1
Exploring unfamiliar area	1	8	4	7	1
Stuck behind slow driver	8	10	3	0	0
Longer routes with no stops	0	4	3	13	1

Table 2. Situational preference survey results

While some situations, like driving on the interstate are disliked by no one and being stuck behind a slow driver are preferred by no one, other situations have a wide range of preference with only a small number of people expressing indifference. For example, the number of drivers who prefer routes that explore unfamiliar areas is roughly the same as the number of drivers with the opposite preference, and while the majority prefer to drive in excess of the speed limit and take longer routes with no stops, there were a number of others with differing preferences. We expect that a population covering all ranges of age and driving ability will possess an even larger preference variance than the more homogeneous participants in our surveys.

The results of our formative research strongly suggest that drivers' choices of routes vary greatly and are highly dependent on personal preferences and contextual factors.

PROCAB: CONTEXT-AWARE BEHAVIOR MODELING

The variability of route preference from person to person and from situation to situation makes perfectly predicting every route choice for a group of drivers extremely difficult. We adopt the more modest goal of developing a *probabilistic model* that assigns as much probability as possible to the routes the drivers prefer. Some of the variability from personal preference and situation is explained by incorporating contextual variables within our probabilistic model. The remaining variability in the probabilistic model stems from influences on route choices that are unobserved by our model.

Many different assumptions and frameworks can be employed to probabilistically model the relationships between contextual variables and sequences of actions. The PROCAB approach is based on three principled techniques:

- Representing behavior as sequential actions in a *Markov Decision Process* (MDP) with parametric cost values
- Using *Inverse Reinforcement Learning* to recover cost weights for the MDP that explain observed behavior
- Employing the *principle of maximum entropy* to find cost weights that have the least commitment

The resulting probabilistic model of behavior is context-dependent, compact, and efficient to learn and reason about. In this section, we describe the comprising techniques and the benefits each provides. We then explain how the resulting model is employed to efficiently reason about context-dependent behavior. Our previous work [27] provides a more rigorous and general theoretical derivation and evaluation.

Markov Decision Process Representation

Markov Decision Processes (MDPs) [21] provide a natural framework for representing sequential decision making, such as route planning. The agent takes a sequence of *actions* ($a \in A$), which transition the agent between *states* ($s \in S$) and incur an action-based *cost*[1] ($c(a) \in \Re$). A simple deterministic MDP with 8 states and 20 actions is shown in Figure 1.

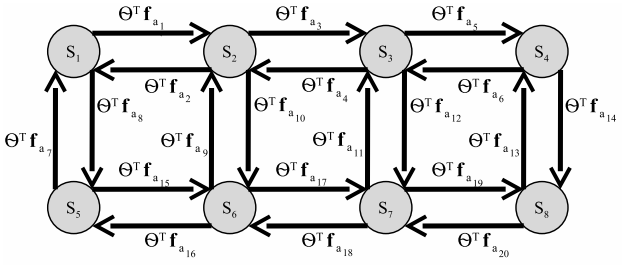


Figure 1. A simple Markov Decision Process with action costs

The agent is trying to minimize the sum of costs while reaching some destination. We call the sequence of actions a *path*, ζ. For MDPs with parametric costs, a set of *features* ($\mathbf{f}_a \in \Re^k$) characterize each action, and the cost of the action is a linear function of these features parameterized by a *cost weight* vector ($\theta \in \Re^k$). Path features, $\mathbf{f}_\zeta$, are the sum of the features of actions in the path: $\sum_{a \in \zeta} \mathbf{f}_a$. The path cost is the sum of action costs (Figure 1), or, equivalently, the cost weight applied to the path features.

$$\text{cost}(\zeta|\theta) = \sum_{a \in \zeta} \theta^\top \mathbf{f}_a = \theta^\top \mathbf{f}_\zeta$$

The MDP formulation provides a natural framework for a number of behavior modeling tasks relevant to ubiquitous computing. For example, in a shopping assistant application, shopping strategies within an MDP are generated that optimally offset the time cost of visiting more stores with the benefits of purchasing needed items [4].

[1]The negation of costs, *rewards*, are more common in the MDP literature, but less intuitive for our application.

The advantage of the MDP approach is that the cost weight is a compact set of variables representing the reasons for preferring different behavior, and if it is assumed that the agent acts sensibly with regard to incurred costs, the approach gerneralizes to previously unseen situations.

Inverse Reinforcement Learning

Much research on MDPs focuses on efficiently finding the optimal behavior for an agent given its cost weight [21]. In our work, we focus on the inverse problem, that of finding an agent's cost weight given demonstrated behavior (*i.e.*, traversed driving routes). Recent research in *Inverse Reinforcement Learning* [18, 1] focuses on exactly this problem. Abbeel and Ng showed that if a model of behavior matches feature counts with demonstrated feature counts, $\tilde{\mathbf{f}}$ (Equation 1), then the model's expected cost matches the agent's incurred costs[1].

$$\sum_{\text{Path } \zeta_i} P(\zeta_i)\mathbf{f}_{\zeta_i} = \tilde{\mathbf{f}} \tag{1}$$

This constraint has an intuitive appeal. If a driver uses 136.3 miles of interstate and crosses 12 bridges in a month's worth of trips, the model should also use 136.3 miles of interstate and 12 bridges in expectation for those same start-destination pairs, along with matching other features of significance. By incorporating many relevant features to match, the model will begin to behave similarly to the agent. However, many distributions over paths can match feature counts, and some will be very different from observed behavior. In our simple example, the model could produce plans that avoid the interstate and bridges for all routes except one, which drives in circles on the interstate for 136 miles and crosses 12 bridges.

Maximum Entropy Principle

If we only care about matching feature counts, and many different distributions can accomplish that, it is not sensible to choose a distribution that shows preference towards other factors that we do not care about. We employ the mathematical formulation of this intuition, the principle of maximum entropy [9], to select a distribution over behaviors. The probability distribution over paths that maximizes Shannon's information entropy while satisfying constraints (Equation 1) has the least *commitment* possible to other non-influential factors. For our problem, this means the model will show no more preference for routes containing specific streets than is required to match feature counts. In our previous example, avoiding all 136 miles of interstate and 12 bridge crossings except for one single trip is a very low entropy distribution over behavior. The maximum entropy principle much more evenly assigns the probability of interstate driving and bridge crossing to many different trips from the set. The resulting distribution is shown in Equation 2.

$$P(\zeta|\theta) = \frac{e^{-cost(\zeta|\theta)}}{\sum_{\text{path } \zeta'} e^{-cost(\zeta'|\theta)}} \tag{2}$$

Low-cost paths are strongly preferred by the model, and paths with equal cost have equal probability.

Compactness Advantage

Many different probabilistic models can represent the same dependencies between variables that pertain to behavior – context, actions, preference, and goal. The main advantage of the PROCAB distribution is that of compactness. In a more compact model, fewer parameters need to be learned so a more accurate model can be obtained from smaller amounts of training data.

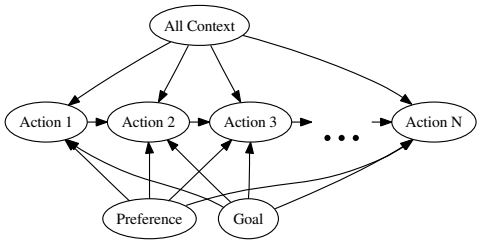


Figure 2. Directed graphical model of sequential context-sensitive, goal-oriented, preference-dependent actions

Directly modeling the probability of each action given all other relevant information is difficult when incorporating context, because a non-myopic action choice depends not only on the context directly associated with that action (*e.g.*, whether the next road is congested), but context associated with all future actions as well (*e.g.*, whether the roads that a road leads to are congested). This approach corresponds to a directed graphical model (Figure 2) [22, 14] where the probability of each action is dependent on all contextual information, the intended destination, and personal preferences. If there are a large number of possible destinations and a rich set of contextual variables, these conditional action probability distributions will require an inordinate amount of observed data to accurately model.

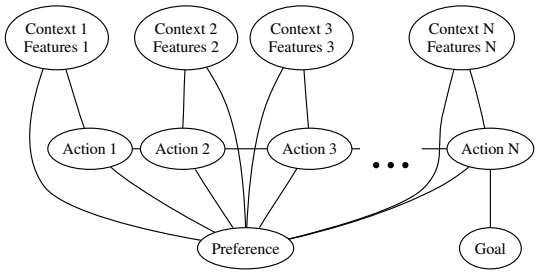


Figure 3. Undirected graphical model of sequential context-sensitive, goal-oriented, personalized actions

The PROCAB approach assumes drivers imperfectly minimize the cost of their route to some destination (Equation 2). The cost of each road depends only on the contextual information directly associated with that road. This model corresponds to an undirected graphical model (Figure 3), where cliques (*e.g.*, "Action 1," "Context 1," and "Preference") define road costs, and the model forms a distribution over paths based on those costs. If a road becomes congested, its cost will increase and other alternative routes will become more probable in the model without requiring any other road costs to change. This independence is why the PROCAB model

is more compact and can be more accurately learned using more reasonable amounts of data.

Probabilistic Inference

We would like to predict future behavior using our PROCAB model, but first we must train the model by finding the parameter values that best explain previously demonstrated behavior. Both prediction and training require probabilistic inference within the model. We focus on the inference of computing the expected number of times a certain action will be taken given known origin, goal state, and cost weights[2]. A simple approach is to enumerate all paths and probabilistically count the number of paths and times in each path the particular state is visited.

Algorithm 1 Expected Action Frequency Calculation

Inputs: cost weight θ, initial state s_o, and goal state s_g
Output: expected action visitation frequencies $D_{a_{i,j}}$

Backward pass

1. Set $Z_{s_i} = 1$ for valid goal states, 0 otherwise
2. Recursively compute for T iterations

$$Z_{a_{i,j}} = e^{-\text{cost}(a_{i,j}|\theta)} Z_{s:a_{i,j}}$$

$$Z_{s_i} = \sum_{\text{actions } a_{i,j} \text{ of } s_i} Z_{a_{i,j}} + \mathbf{1}_{s_i \in S_{\text{Goal}}}$$

Forward pass

3. Set $Z'_{s_i} = 1$ for initial state, 0 otherwise
4. Recursively compute for T iterations

$$Z'_{a_{i,j}} = Z'_{s_i} e^{-\text{cost}(a_{i,j}|\theta)}$$

$$Z'_{s_i} = \sum_{\text{actions } a_{j,i} \text{ to } s_i} Z'_{a_{j,i}} + \mathbf{1}_{s_i = s_{\text{initial}}}$$

Summing frequencies

5. $D_{a_{i,j}} = \dfrac{Z'_{s_i} e^{-\text{cost}(a_{i,j}|\theta)} Z_{s_j}}{Z_{s_{\text{initial}}}}$

Algorithm 1 employs a more tractable approach by finding the probabilistic weight of all paths from the origin (o) to a specific action (a), $Z'_a = \sum_{\zeta_{o \to a}} e^{-\text{cost}(\zeta)}$, all paths from the action to the goal (g)[3], $Z_a = \sum_{\zeta_{a \to g}} e^{-\text{cost}(\zeta)}$ and all paths from the origin to the goal, $Z_o = Z'_g = \sum_{\zeta_{o \to g}} e^{-\text{cost}(\zeta)}$. Expected action frequencies are obtained by combining these results (Equation 3).

$$D_a = \frac{Z_a Z'_a e^{-\text{cost}(a)}}{Z_o} \tag{3}$$

Using dynamic programming to compute the required Z values is exponentially more efficient than path enumeration.

[2]Other inferences, like probabilities of sequence of actions or one specific action are obtained using the same approach.

[3]In practice, we use a finite T, which considers a set of paths of limited length.

Cost Weight Learning

We train our model by finding the parameters that maximize the [log] probability of (and best explain) previously observed behavior.

$$\theta^* = \underset{\theta}{\text{argmax}} \log \prod_i \frac{e^{-\text{cost}(\tilde{\zeta}_i|\theta)}}{\sum_{\text{path } \zeta_j} e^{-\text{cost}(\zeta_j|\theta)}} \tag{4}$$

Algorithm 2 Learn Cost Weights from Data

Stochastic Exponentiated Gradient Ascent

Initialize random $\theta \in \Re^k, \gamma > 0$
For $t = 1$ to T :
  For random example, compute D_a for all actions a (using Algorithm 1)
  Compute gradient, ∇F from D_a (Equation 5)
  $\theta \leftarrow \theta e^{\frac{\gamma}{t} \nabla F}$

We employ a gradient-based method (Algorithm 2) for this convex optimization. The gradient is simply the difference between the demonstrated behavior feature counts and the model's expected feature counts, so at the optima these feature counts match. We use the action frequency expectations, D_a, to more efficiently compute the gradient (Equation 5).

$$\tilde{\mathbf{f}} - \sum_{\text{Path } \zeta_i} P(\zeta_i) \mathbf{f}_{\zeta_i} = \tilde{\mathbf{f}} - \sum_a D_a \mathbf{f}_a \tag{5}$$

The algorithm raises the cost of features that the model is over-predicting so that they will be avoided more and lowers the cost of features that the model is under-predicting until convergence near the optima.

TAXI DRIVER ROUTE PREFERENCE DATA

Now that we have described our model for probabilistically reasoning from observed context-aware behavior, we will describe the data we collected to evaluate this model. We recruited 25 Yellow Cab taxi drivers to collect data from. Their experience as taxi drivers in the city ranged from 1 month to 40 years. The average and median were 12.9 years and 9 years respectively. All participants reported living in the area for at least 15 years.

Collected Position Data

We collected location traces from our study participants over a 3 month period using global positioning system (GPS) devices that log locations over time. Each participant was provided one of these devices[4], which records a reading roughly every 6 to 10 seconds while in motion. The data collection yielded a dataset of over 100,000 miles of travel collected from over 3,000 hours of driving. It covers a large area surrounding our city (Figure 4). Note that no map is being overlaid in this figure. Repeated travel over the same roads leaves the appearance of the road network itself.

[4]In a few cases where two participants who shared a taxi also shared a GPS logging device

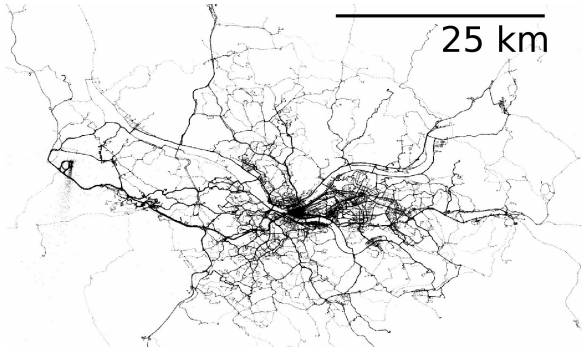


Figure 4. The collected GPS datapoints

Road Network Representation

The deterministic action-state representation of the corresponding road network contains over 300,000 states (*i.e.*, road segments) and over 900,000 actions (*i.e.*, available transitions between road segments). There are characteristics describing the speed categorization, functionality categorization, and lane categorization of roads. Additionally we can use geographic information to obtain road lengths and turn types at intersection (*e.g.*, straight, right, left).

Fitting to the Road Network and Segmenting

To address noise in the GPS data, we fit it to the road network using a particle filter. A particle filter simulates a large number of vehicles traversing over the road network, focusing its attention on particles that best match the GPS readings. A motion model is employed to simulate the movement of the vehicle and an observation model is employed to express the relationship between the true location of the vehicle and the GPS reading of the vehicle. We use a motion model based on the empirical distribution of changes in speed and a Laplace distribution for our observation model.

Once fitted to the road network, we segmented our GPS traces into distinct trips. Our segmentation is based on time-thresholds. Vehicles with a small velocity for a period of time are considered to be at the end of one trip and the beginning of a new trip. We note that this problem is particularly difficult for taxi driver data, because these drivers may often stop only long enough to let out a passenger and this can be difficult to distinguish from stopping at a long stoplight. We discard trips that are too short, too noisy, and too cyclic.

MODELING ROUTE PREFERENCES

In this section, we describe the features that we employed to model the utilities of different road segments in our model, and machine learning techniques employed to learn good cost weights for the model.

Feature Sets and Context-Awareness

As mentioned, we have characteristics of the road network that describe speed, functionality, lanes, and turns. These combine to form path-level features that describe a path as numbers of different turn type along the path and road mileage at different speed categories, functionality categories, and numbers of lanes. To form these path features, we combine the comprising road segments' features for each of their characteristics weighted by the road segment length or intersection transition count.

Feature	Value
Highway	3.3 miles
Major Streets	2.0 miles
Local Streets	0.3 miles
Above 55mph	4.0 miles
35-54mph	1.1 miles
25-34 mph	0.5 miles
Below 24mph	0 miles
3+ Lanes	0.5 miles
2 Lanes	3.3 miles
1 Lane	1.8 miles

Feature	Value
Hard left turn	1
Soft left turn	3
Soft right turn	5
Hard right turn	0
No turn	25
U-turn	0

Table 3. Example feature counts for a driver's demonstrated route(s)

The PROCAB model finds the cost weight for different features so that the model's feature counts will match (in expectation) those demonstrated by a driver (*e.g.*, as shown in Table 3). when planning for the same starting point and destination. Additionally, unobserved features may make certain road segments more or less desirable. To model these unobservable features, we add unique features associated with each road segment to our model. This allows the cost of each road segment to vary independently.

We conducted a survey of our taxi drivers to help identify the main contextual factors that impact their route choices. The perceived influences (with average response) on a 5-point Likert Scale ranging from "no influence" (1) to "strong influence" (5) are: *Accidents* (4.50), *Construction* (4.42), *Time of Day* (4.31), *Sporting Events* (4.27), and *Weather* (3.62). We model some of these influences by adding real-time features to our road segments for *Accident*, *Congestion*, and *Closures* (Road and Lane) according to traffic data collected every 15 minutes from Yahoo's traffic service.

We incorporate sensitivity to time of day and day of week by adding duplicate features that are only active during certain times of day and days of week. We use *morning rush hour*, *day time*, *evening rush hour*, *evening*, and *night* as our time of day categories, and *weekday* and *weekend* as day of week categories. Using these features, the model tries to not only match the total number of *e.g.*, interstate miles, but it also tries to get the right amount of interstate miles under each time of day category. For example, if our taxi drivers try to avoid the interstate during rush hour, the PROCAB model assigns a higher weight to the joint interstate and rush hour feature. It is then less likely to prefer routes on the interstate during rush hour given that higher weight.

Matching all possible contextual feature counts highly constrains the model's predictions. In fact, given enough contextual features, the model may exactly predict the actual demonstrated behavior and *overfit* to the training data. We avoid this problem using *regularization*, a technique that relaxes the feature matching constraint by introducing a penalty term ($-\sum_i \lambda_i \theta_i^2$) to the optimization. This penalty

prevents cost weights corresponding to highly specialized features from becoming large enough to force the model to perfectly fit observed behavior.

Learned Cost Weights

We learn the cost weights that best explain a set of demonstrated routes using Algorithm 2. Using this approach, we can obtain cost weights for each driver or a collective cost weight for a group of drivers. In this paper, we group the routes gathered from all 25 taxi drivers together and learn a single cost weight using a training set of 80% of those routes.

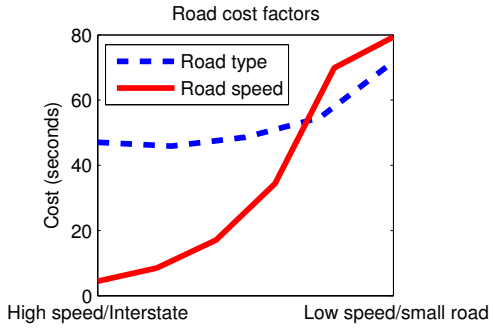


Figure 5. Speed categorization and road type cost factors normalized to seconds assuming 65mph driving on fastest and largest roads

Figure 5 shows how road type and speed categorization influence the road's cost in our learned model.

NAVIGATION APPLICATIONS AND EVALUATION

We now focus on the applications that our route preference model enables. We evaluate our model on a number of prediction tasks needed for those applications. We compare the PROCAB model's performance with other state of the art methods on these tasks using the remaining 20% of the taxi route dataset to evaluate.

Turn Prediction

We first consider the problem of predicting the action at the next intersection given a known final destination. This problem is important for applications such as automatic turn signaling and fuel consumption optimization when the destination is either specified by the driver or can be inferred from his or her normal driving habits. We compare PROCAB's ability to predict 55,725 decisions at intersections with multiple options[5] to approaches based on Markov Models.

We measure the accuracy of each model's predictions (*i.e.*, percentage of predictions that are correct) and the average log likelihood, $\frac{1}{\#decisions} \sum_{\text{decision } d} \log P_{\text{model}}(d)$ of the actual actions in the model. This latter metric evaluates the model's ability to probabilistically model the distribution of decisions. Values closer to 0 are closer to perfectly modeling the data. As a baseline, guessing uniformly at random (without U-turns) yields an accuracy of 46.4% and a log likelihood of -0.781 for our dataset.

[5]Previous Markov Model evaluations include "intersections" with only one available decision, which comprise 95% [22] and 28% [11] of decisions depending on the road network representation.

Markov Models for Turn Prediction

We implemented two Markov Models for turn prediction. A Markov Model predicts the next action (or road segment) given some observed conditioning variables. Its predictions are obtained by considering a set of previously observed decisions at an intersection that match the conditioning variables. The most common action from that set can be predicted or the probability of each action can be set in proportion to its frequency in the set[6]. For example, Liao *et al.* [14] employ the destination and mode of transport for modeling multi-modal transportation routines. We evaluate the Markov Models employed by Krumm [11], which conditions on the previous K traversed road segments, and Simmons *et al.* [22], which conditions on the route destination.

	Non-reducing		Reducing	
History	**Accu.**	**Likel.**	**Accu.**	**Likel.**
1 edge	**85.8%**	-0.322	85.8%	-0.322
2 edge	85.6%	$\mathbf{-0.321}$	86.0%	$\mathbf{-0.319}$
5 edge	84.8%	-0.330	**86.2%**	-0.321
10 edge	83.4%	-0.347	86.1%	-0.328
20 edge	81.5%	-0.367	85.7%	-0.337
100 edge	79.3%	-0.399	85.1%	-0.356

Table 4. *K*-order Markov Model Performance

Evaluation results of a K-order Markov Model based on road segment histories are shown in Table 4. We consider two variants of the model. For the *Non-reducing* variant, if there are no previously observed decisions that match the K-sized history, a random guess is made. In the *Reducing* variant, instead of guessing randomly when no matching histories are found, the model tries to match a smaller history size until some matching examples are found. If no matching histories exist for $K = 1$ (*i.e.*, no previous experience with this decision), a random guess is then made.

In the non-reducing model we notice a performance degradation as the history size, K, increases. The reduction strategy for history matching helps to greatly diminish the degradation of having a larger history, but still we still find a small performance degradation as the history size increases beyond 5 edges.

Destinations	Accuracy	Likelihood
1x1 grid	**85.8%**	-0.322
2x2 grid	85.1%	$\mathbf{-0.319}$
5x5 grid	84.1%	-0.327
10x10 grid	83.5%	-0.327
40x40 grid	78.6%	-0.365
100x100 grid	73.1%	-0.416
2000x2000 grid	59.3%	-0.551

Table 5. Destination Markov Model Performance

We present the evaluation of the Markov Model conditioned on a known destination in Table 5. We approximate each unique destination with a grid of destination cells, starting from a single cell (1x1) covering the entire map, all the way

[6]Some chance of selecting previously untaken actions is added by *smoothing* the distribution.

up to 4 million cells (2000x2000). As the grid dimensions grow to infinity, the treats each destination as unique. We find empirically that using finer grid cells actually degrades accuracy, and knowing the destination provides no advantage over the history-based Markov Model for this task.

Both of these results show the inherent problem of *data sparsity* for directed graphical models in these types of domains. With an infinite amount of previously observed data available to construct a model, having more information (*i.e.*, a larger history or a finer grid resolution) can never degrade the model's performance. However, with finite amounts of data there will often be few or no examples with matching conditional variables, providing a poor estimate of the true conditional distribution, which leads to lower performance in applications of the model.

PROCAB Turn Prediction

The PROCAB model predicts turns by reasoning about paths to the destination. Each path has some probability within the model and many different paths share the same first actions. An action's probability is obtained by summing up all path probabilities that start with that action. The PROCAB model provides a compact representation over destination, context, and action sequences, so the exact destination and rich contextual information can be incorporated without leading to data sparsity degradations like the Markov Model.

	Accuracy	**Likelihood**
Random guess	46.4%	−0.781
Best History MM	86.2%	−0.319
Best Destination MM	85.8%	−0.319
PROCAB (no context)	91.0%	−0.240
PROCAB (context)	**93.2%**	**−0.201**

Table 6. Baseline and PROCAB turn prediction performance

We summarize the best comparison models' results for turn prediction and present the PROCAB turn prediction performance in Table 6. The PROCAB approach provides a large improvement over the other models in both prediction accuracy and log likelihood. Additionally, incorporating time of day, day of week, and traffic report contextual information provides improved performance. We include this additional contextual information in the remainder of our experiments.

Route Prediction

We now focus on route prediction, where the origin and destination of a route are known, but the route between the two is not and must be predicted. Two important applications of this problem are route recommendation, where a driver requests a desirable route connecting two specified points, and unanticipated hazard warning, where an application that can predict the driver will encounter some hazard he is unaware of and warn him beforehand.

We evaluate the prediction quality based on the amount of matching distance between the predicted route and the actual route, and the percentage of predictions that match, where we consider all routes that share 90% of distance as matches. This final measure ignores minor route differences, like those caused by noise in GPS data. We evaluate a previously described Markov Model, a model based on estimated travel time, and our PROCAB model.

Model	**Dist. Match**	**90% Match**
Markov (1x1)	62.4%	30.1%
Markov (3x3)	62.5%	30.1%
Markov (5x5)	62.5%	29.9%
Markov (10x10)	62.4%	29.6%
Markov (30x30)	62.2%	29.4%
Travel Time	72.5%	44.0%
PROCAB	**82.6%**	**61.0%**

Table 7. Evaluation results for Markov Model with various grid sizes, time-based model, and PROCAB model

Markov Model Route Planning

We employ route planning using the previously described destination-conditioned Markov Model [22]. The model recommends the most probable route satisfying origin and destination constraints. The results (Table 7, *Markov*) are fairly uniform regardless of the number of grid cells employed, though there is a subtle degradation with more grid cells.

Travel Time-Based Planning

A number of approaches for vehicle route recommendation are based on estimating the travel time of each route and recommending the fastest route. Commercial route recommendation systems, for example, try to provide the fastest route. The Cartel Project [8] works to better estimate the travel times of different road segments in near real-time using fleets of instrumented vehicles. One approach to route prediction is to assume the driver will also try to take this most expedient route.

We use the distance and speed categorization of each road segment to estimate travel times, and then provide route predictions using the fastest route between origin and destination. The results (Table 7, *Travel Time*) show a large improvement over the Markov Model. We believe this is due to data sparsity in the Markov Model and the Travel time model's ability to generalize to previously unseen situations (*e.g.*, new origin-destination pairs).

PROCAB Route Prediction

Our view of route prediction and recommendation is fundamentally different than those based solely on travel time estimates. Earlier research [13] and our own study have shown that there is a great deal of variability in route preference between drivers. Rather than assume drivers are trying to optimize one particular metric and more accurately estimate that metric in the road network, we implicitly learn the metric that the driver is actually trying to optimize in practice. This allows other factors on route choice, such as fuel efficiency, safety, reduced stress, and familiarity, to be modeled. While the model does not explicitly understand that one route is more stressful than another, it does learn to imitate a driver's avoidance of more stressful routes and features associated with those routes. The PROCAB model provides

Model	Percentage of trip observed					
	10%	20%	40%	60%	80%	90%
MM 5x5	9.61	9.24	8.75	8.65	8.34	8.17
MM 10x10	9.31	8.50	7.91	7.58	7.09	6.74
MM 20x20	9.69	8.99	8.23	7.66	6.94	6.40
MM 30x30	10.5	9.94	9.14	8.66	7.95	7.59
Pre 40x40 MAP	7.98	7.74	6.10	5.42	4.59	4.24
Pre 40x40 Mean	7.74	7.53	5.77	4.83	4.12	3.79
Pre 80x80 MAP	11.02	7.26	5.17	4.86	4.21	3.88
Pre 80x80 Mean	8.69	7.27	5.28	**4.46**	**3.95**	3.69
PRO MAP	11.18	8.63	6.63	5.44	3.99	**3.13**
PRO Mean	**6.70**	**5.81**	**5.01**	4.70	3.98	3.32

Table 8. Prediction error of Markov, Predestination, and PROCAB models in kilometers

increased performance over both the Markov Model and the model based on travel time estimates, as shown in Table 7. This is because it is able to better estimate the utilities of the road segments than the feature-based travel time estimates.

Destination Prediction

Finally, we evaluate models for destination prediction. In situations where a driver has not entered her destination into an in-car navigation system, accurately predicting her destination would be useful for proactively providing an alternative route. Given the difficulty that users have in entering destinations [23], destination prediction can be particularly useful. It can also be used in conjunction with our model's route prediction and turn prediction to deal with settings where the destination is unknown.

In this setting, a partial route of the driver is observed and the final destination of the driver is predicted. This application is especially difficult given our set of drivers, who visit a much wider range of destinations than typical drivers. We compare PROCAB's ability to predict destination to two other models, in settings where the set of possible destinations is not fully known beforehand.

We evaluate our models using 1881 withheld routes and allow our model to observe various amounts of the route from 10% to 90%. The model is provided no knowledge of how much of the trip has been completed. Each model provides a single point estimate for the location of the intended destination and we evaluate the distance between the true destination and this predicted destination in kilometers.

Bayes' Rule

Using Bayes' Rule (Equation 6), route preference models that predict route (B) given destination (A) can be employed along with a prior on destinations, $P(A)$, to obtain a probability distribution over destinations, $P(A|B)$[7].

$$P(A|B) = \frac{P(B|A)P(A)}{\sum_{A'} P(B|A')P(A')} \propto P(B|A)P(A) \quad (6)$$

[7]Since the denominator is constant with respect to A, the probability is often expressed as being proportionate to the numerator.

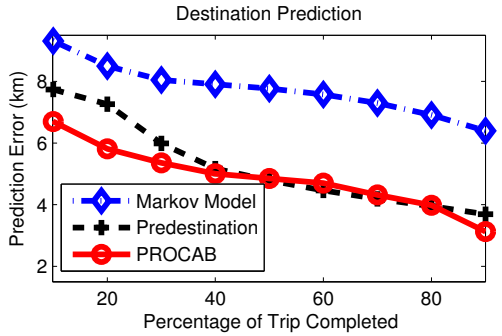


Figure 6. The best Markov Model, Predestination, and PROCAB prediction errors

Markov Model Destination Prediction

We first evaluate a destination-conditioned Markov Model for predicting destination region in a grid. The model employs Bayes' Rule to obtain a distribution over cells for the destination based on the observed partial route. The prior distribution over grid cells is obtained from the empirical distribution of destinations in the training dataset. The center point of the most probable grid cell is used as a destination location estimate.

Evaluation results for various grid cell sizes are shown in Table 8 (MM). We find that as more of the driver's route is observed, the destination is more accurately predicted. As with the other applications of this model, we note that having the most grid cells does not provide the best model due to data sparsity issues.

Predestination

The Predestination sytem [12] grids the world into a number of cells and uses the observation that the partially traveled route is usually an *efficient* path through the grid to the final destination. Using Bayes' rule, destinations that are opposite of the direction of travel have much lower probability than destinations for which the partially traveled route is efficient. Our implemention employs a prior distribution over destination grid cells conditioned on the starting location rather than the more detailed original Predestination prior. We consider two variants of prediction with Predestination. One predicts the center of the most probable cell (*i.e.*, the *Maximum a Posteriori* or *MAP* estimate). The other, *Mean*, predicts the probabilistic average over cell center beliefs. We find that Predestination shows significant improvement (Table 8, Pre) over the Markov Model's performance.

PROCAB Destination Prediction

In the PROCAB model, destination prediction is also an application of Bayes' rule. Consider a *partial path*, $\zeta_{A\to B}$ from point A to point B. The destination probability is then:

$$P(\text{dest}|\zeta_{A\to B},\theta) \propto P(\zeta_{A\to B}|\text{dest},\theta)P(\text{dest}) \propto \frac{\sum_{\zeta_{B\to\text{dest}}} e^{-\text{cost}(\zeta|\theta)}}{\sum_{\zeta_{A\to\text{dest}}} e^{-\text{cost}(\zeta|\theta)}} P(\text{dest}) \quad (7)$$

We use the same prior that depends on the starting point (A) that we employed in our Predestination implementation. The

posterior probabilities are efficiently computed by taking the sums over paths from points A and B to each possible destination (Equation 7) using the forward pass of Algorithm 1.

Table 8 (PRO) and Figure 6 show the accuracy of PROCAB destination prediction compared with other models. Using averaging is much more beneficial for the PROCAB model, likely because it is more sensitive to particular road segments than models based on grid cells. We find that the PROCAB model performs comparably to the Predestination model given a large percentage of observed trip, and better than the Predestination model when less of the trip is observed. PROCAB's abilities to learn non-travel time desirability metrics, reason about road segments rather than grid cells, and incorporate additional contextual information may each contribute to this improvement.

CONCLUSIONS AND FUTURE WORK

We presented PROCAB, a novel approach for modeling observed behavior that learns context-sensitive action utilities rather than directly learning action sequences. We applied this approach to vehicle route preference modeling and showed significant performance improvements over existing models for turn and route prediction and showed similar performance to the state-of-the-art destination prediction model. Our model has a more compact representation than those which directly model action sequences. We attribute our model's improvements to its compact representation, because of an increased ability to generalize to new situations.

Though we focused on vehicle route preference modeling and smart navigation applications, PROCAB is a general approach for modeling context-sensitive behavior that we believe it is well-suited for a number of application domains. In addition to improving our route preference models by incorporating additional contextual data, we plan to model elder drivers and other interesting populations and we plan to model behavior in other context-rich domains, including modeling the movements of people throughout dynamic work environments and the location-dependent activities of families. We hope to demonstrate PROCAB as a domain-general framework for modeling context-dependent user behavior.

ACKNOWLEDGMENTS

The authors thank Eric Oatneal and Jerry Campolongo of Yellow Cab Pittsburgh for their assistance, Ellie Lin Ratliff for helping to conduct the study of driving habits, and John Krumm for his help in improving this paper and sharing GPS hardware hacks. Finally, we thank the Quality of Life Technology Center/National Science Foundation for supporting this research under Grant No. EEEC-0540865.

REFERENCES

1. P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, pages 1–8, 2004.
2. D. Ashbrook and T. Starner. Using gps to learn significant locations and prediction movement. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.
3. A. Bhattacharya and S. K. Das. Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks. *Wireless Networks*, 8:121–135, 2002.
4. T. Bohnenberger, A. Jameson, A. Krüger, and A. Butz. Location-aware shopping assistance: Evaluation of a decision-theoretic approach. In *Proc. Mobile HCI*, pages 155–169, 2002.
5. K. Chang, M. Y. Chen, and J. Canny. Tracking free-weight exercises. In *Proc. Ubicomp*, pages 19–37, 2007.
6. J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. B. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction*, 12(1):119–146, 2005.
7. D. A. Hennesy and D. L. Wiesenthal. Traffic congestion, driver stress, and driver aggression. *Aggressive Behavior*, 25:409–423, 1999.
8. B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. K. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *ACM SenSys*, pages 125–138, 2006.
9. E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.
10. L. Johannesson, M. Asbogard, and B. Egardt. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):71–83, March 2007.
11. J. Krumm. A markov model for driver route prediction. *Society of Automative Engineers (SAE) World Congress*, 2008.
12. J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *Proc. Ubicomp*, pages 243–260, 2006.
13. J. Letchner, J. Krumm, and E. Horvitz. Trip router with individualized preferences (trip): Incorporating personalization into route planning. In *Proc. IAAI*, pages 1795–1800, 2006.
14. L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.
15. R. M. Mayrhofer. *An architecture for context prediction*. PhD thesis, Johannes Kepler University of Linz, Austria, 2004.
16. M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *AAAI Spring Symposium*, pages 110–114, 1998.
17. B. Mutlu, A. Krause, J. Forlizzi, C. Guestrin, and J. Hodgins. Robust, low-cost, non-intrusive sensing and recognition of seated postures. In *Proc. UIST*, pages 149–158, 2007.
18. A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. ICML*, pages 663–670, 2000.
19. K. Patel, M. Y. Chen, I. Smith, and J. A. Landay. Personalizing routes. In *Proc. UIST*, pages 187–190, 2006.
20. D. J. Patterson, L. Liao, D. Fox, and H. A. Kautz. Inferring high-level behavior from low-level sensors. In *Proc. Ubicomp*, pages 73–89, 2003.
21. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.
22. R. Simmons, B. Browning, Y. Zhang, and V. Sadekar. Learning to predict driver route and destination intent. *Proc. Intelligent Transportation Systems Conference*, pages 127–132, 2006.
23. A. Steinfeld, D. Manes, P. Green, and D. Hunter. Destination entry and retrieval with the ali-scout navigation system. *The University of Michigan Transportation Research Institute No. UMTRI-96-30*, 1996.
24. E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Proc. Pervasive*, pages 158–175, 2004.
25. K. Torkkola, K. Zhang, H. Li, H. Zhang, C. Schreiner, and M. Gardner. Traffic advisories based on route prediction. In *Workshop on Mobile Interaction with the Real World*, 2007.
26. D. Wyatt, M. Philipose, and T. Choudhury. Unsupervised activity recognition using automatically mined common sense. In *Proc. AAAI*, pages 21–27, 2005.
27. B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, 2008.

PreHeat: Controlling Home Heating Using Occupancy Prediction

James Scott[1], A.J. Bernheim Brush[2], John Krumm[2], Brian Meyers[2]
Mike Hazas[3], Steve Hodges[1], Nicolas Villar[1]

[1]Microsoft Research Cambridge
Cambridge, UK
{jws|shodges|nvillar}@microsoft.com

[2]Microsoft Research Redmond
Redmond, WA USA
{ajbrush|jckrumm|brianme}@microsoft.com

[3]Lancaster University
Lancaster, UK
hazas@comp.lancs.ac.uk

ABSTRACT
Home heating is a major factor in worldwide energy use. Our system, PreHeat, aims to more efficiently heat homes by using occupancy sensing and occupancy prediction to automatically control home heating. We deployed PreHeat in five homes, three in the US and two in the UK. In UK homes, we controlled heating on a per-room basis to enable further energy savings. We compared PreHeat's prediction algorithm with a static program over an average 61 days per house, alternating days between these conditions, and measuring actual gas consumption and occupancy. In UK homes PreHeat both saved gas and reduced MissTime (the time that the house was occupied but not warm). In US homes, PreHeat decreased MissTime by a factor of 6-12, while consuming a similar amount of gas. In summary, PreHeat enables more efficient heating while removing the need for users to program thermostat schedules.

Author Keywords
Energy, environment, home heating, sensing, prediction.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Algorithms, Experimentation, Human Factors.

INTRODUCTION
Home heating uses more energy than any other residential energy expenditure including air conditioning, water heating, and appliances [1]. This makes increasing the efficiency of home heating an important goal for saving money and reducing our ecological footprint. Although programmable thermostats provide the technology to reduce this problem, they are underutilized. Surveys have found that fewer than 50% of US households have programmable thermostats, and even worse, the US Environmental Protection Agency estimates that 30% or more of US households with programmable thermostats are not using their thermostat's programming feature, so they are not saving the 10%-30% claimed for such devices [2, 6].

Fundamentally, home heating is a trade-off between energy use and warmth. By leaving their thermostat set permanently to a warm temperature, households incur increased energy use costs; by using a programmed thermostat to only heat for some of the time, households can use less energy, but occupants may be cold if the program is wrong or while waiting for the house to heat.

Our home heating system, PreHeat, aims both to eliminate the need for manual user programming of a thermostat schedule and to improve the efficiency of home heating (i.e., improving the tradeoff achieved between energy use and time in which occupants are cold). PreHeat uses occupancy sensing and historical occupancy data to estimate the probability of future occupancy, allowing the home to be heated only when necessary.

We experimentally evaluated PreHeat in five family homes. In three US homes, we controlled whole-house forced air heating systems and used active RFID for occupancy sensing. In two UK homes, we controlled radiators and underfloor heating per room, and we sensed occupancy using motion sensors. During the study, we alternated days between using PreHeat and using a schedule, with an average 61 study days per house. We found that PreHeat did indeed achieve a better tradeoff between gas consumption and MissTime [8] – the amount of time someone was home and the house was not warm. US homes used about the same amount of gas, but had a 6x-12x reduction in MissTime. In the UK houses (with per-room control), improvements in both metrics were achieved.

We acknowledge that there are many complementary ways to save energy in home heating, including better insulation, new architectural standards, and persuasive/informative approaches. Our contribution focuses on better heating scheduling. By deploying our system into real homes, and by directly comparing it to the previous control systems, we provide evidence that our novel predictive heating algorithm can improve on the tradeoff between energy consumption and MissTime, and can do so without requiring homeowners to program complex occupancy schedules.

UbiComp'11, September 17–21, 2011, Beijing, China.

BACKGROUND AND RELATED WORK

In this paper, we focus on the control of two types of *central heating* systems common in Europe and North America. The first type is *hot water space heating*, in which a boiler heats water to 60°C or more and circulates it around the house. Heat in each room is distributed from floor-embedded pipes (*underfloor heating*) or by wall-mounted *radiators*. The second type, more common in North America, is *forced air heating*, in which a furnace heats air and circulates it through ductwork to rooms.

For both types of systems, the temperature is often controlled using a single thermostat located in a central area of the home such as a living room or hallway. The thermostat takes temperature readings and switches the central heating on as needed to maintain a user-defined *setpoint* temperature. Systems with radiators can also be fitted with a thermostatic valve at each radiator, which can be manually set from "1" to "5" to define a separate target temperature for each room. In some houses, particularly those with underfloor heating and some forced-air systems, the system control is split into *zones*, each of which has its own thermostat.

Modern thermostats are typically programmable, allowing the user to specify a *schedule* of times during the day or week when the home should be heated to the setpoint. Outside of these times, the thermostat may use a lower *setback* temperature to reduce the heating energy required, or, as is common in the UK, simply be turned off.

More sophisticated thermostats exist: some are controllable via the Internet; some measure the heat-up latency of the house and compensate the schedule; and others are reactive to occupancy as sensed by entryway or motion sensors. These are expensive, not commonly installed, and as others have reported [3, 8], the latter two tend to heat unnecessarily, actually increasing the energy required.

Researchers have studied the use of domestic heating systems for some time. In 1978, Sonderegger observed that the total energy required by a home's heating system is heavily determined by the particular people living there, dwarfing the effects of insulation or heating infrastructure [9]. Other work has aimed to reduce heating energy using feedback, e.g. via increased frequency and detail of utility bills [10], or real-time energy displays [4]. Feedback and more direct interventions such as "persuasive" interfaces are complementary to our work: while PreHeat may lower energy use by heating less during unoccupied times, a persuasive display may "nudge" an occupant to lower the setpoint and wear a sweater.

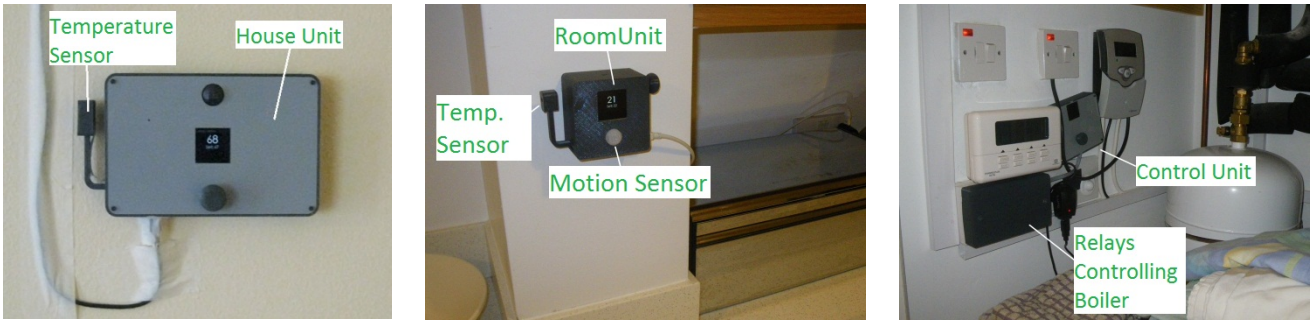
Closest to our work are approaches that have aimed to improve control beyond what off-the-shelf products currently offer. Mozer et al. describe a "Neurothermostat" which utilizes a hybrid occupancy predictor, making use of an available daily schedule and a neural network which was trained on five consecutive months of occupancy data [7]. Although real occupancy data from one house is used in the analysis, a first order approximation of the heating system and house is used to model temperature and energy consumption. Mozer et al. show that the Neurothermostat results in a lower unified cost, where energy and occupant "comfort" are expressed as a combined figure, in dollars. We believe that the equivalency of comfort and energy is problematic (the relationship between setpoint deviation and dollars is nonlinear, depends upon the individual, and changes over time), so we prefer to characterize separately the deviation from setpoint temperature at occupied times, and the energy required.

Gupta et al. propose using live data from mobile phones or in-vehicle GPS devices to control home heating and cooling [3]. Their method works by ensuring the home can always be brought to the setpoint in the time it would take the person to travel home from the current location. Using a simulation based on look-up tables extrapolated from three days of temperature readings, they compute the savings possible in four houses based on two months of GPS data. While their scheme does ensure the house is always at setpoint upon arrival, it typically results in lower savings than programmable or manual thermostats.

Lu et al. formulate a hidden Markov model to predict occupancy and control HVAC systems [8]. They collected occupancy data in eight US households for one to two weeks. Using leave-one-out cross-validation to train and test the HMM, they evaluate their approach's MissTime (i.e. total occupied time not at setpoint) and energy savings for each day in a week using the US Dept. of Energy's EnergyPlus simulator. Their forward-looking approach is designed to control specialized two-stage HVAC systems and employs a second "deep" setback (10°C/50°F). By contrast, we show how our system performs in real households using single setback temperatures and single-stage, gas-fired equipment commonly deployed today.

Our occupancy prediction algorithm is itself an improvement over the above approaches: it results in more favorable trade-offs between miss time and energy than the GPS-reactive system [3]. Krumm and Brush [5] also presented an occupancy prediction algorithm that gives probabilities of occupancy at different times of day. However, this algorithm computes a representative Sunday, Monday, etc. for each day of the week, without being able to respond to changing occupancy patterns as PreHeat does.

More broadly, and unlike any of the above work, we evaluate the performance of PreHeat in situ in five houses, using custom embedded sensing and real-time control of the central heating. Previous evaluations have relied upon simulations [3, 7, 8]. While industry standard simulators such as EnergyPlus may go beyond simple first-order approximations, we question the efficacy of simulators for characterizing either (1) the daily energy requirement of heating infrastructure – savings of smaller than 10% can be important, so daily errors of even 3 kWh can muddy analysis; or (2) the deviation from setpoint at occupied



(a) (b) (c)

Figure 1: Photographs of (a) House Unit installed in US house replacing thermostat on wall, (b) Room Unit in UK room, (c) Control Unit beside old house thermostat in UK, wired into boiler control circuitry.

times (i.e. MissTime) – this requires simulator granularity finer than five minutes, better than 1°C, and for individual rooms (when using per-room prediction and control). In contrast, our deployment allows us to characterize PreHeat using real temperatures, gas readings, and occupancy sensors.

EXPERIMENTAL HEATING SYSTEM

We implemented an experimental system, designed to facilitate comparison between PreHeat and existing heating controls. We deployed this system in five homes: three in Seattle, USA (referred to as US1, US2, US3) and two in Cambridge, UK (UK1, UK2) during the winter from January – April, 2011. We chose to deploy in homes of project researchers or our colleaagues, so people knew how to cope if any problems occurred. All of the homes were family homes with two adults and one or more children – ranging from one toddler (UK1) to three school-age children (US1). All US homes used forced air heating, UK1 used both radiators (4 of 10 independently heatable rooms) and underfloor heating (8 of 10 rooms, 2 with radiators too), and UK2 used radiators in all 14 rooms.

Hardware

We built custom hardware, shown in Figure 1, to control the heating systems. In US houses, we replaced the existing thermostat for whole-house control with a "House Unit" built using our in-house-developed prototyping platform, Microsoft .NET Gadgeteer. The core of this unit is a mainboard based on GHI Electronics' Embedded Master module with ARM7 CPU, running custom software in C#. Additionally there were a number of peripheral modules. The House Unit measured temperature using a Sensiron SHT15 sensor (accuracy +/- 0.1°C), and used an Avago ASSR-1611 solid state relay to control the furnace through the existing wiring. It also included a rotary encoder, RGB light level sensor, a 128x128 OLED display and a passive infra-red motion sensor (Panasonic PIR-AMN34111J, 10m max range and 110 degree beam angle).

In UK houses, we controlled the heating on a per-room basis by installing "Room Units" in each space with independent heating. These were similar to "House Units", but instead of a relay had an 868MHz transmitter (TX868-785 from elv.de) to control wireless radiator valves (HHFHT-8V from HouseHeat) which replaced the existing thermostatic radiator valves on the radiators in each room.

In the UK we also deployed "Control Units" that did not have any sensors, but included either Avago ASSR-1611 relays for controlling per-room underfloor heating valves or Omron G6D relays providing 240V control of the house boiler. Turning on heating in any individual room involved activating the boiler (if it was not already active for another room's heating) and per-room valves – either on radiators or for the underfloor heating. These units also controlled the hot water heating for sinks, showers, etc., using a static schedule reflecting the previous hot water schedule.

All hardware was powered through USB mains power adaptors, aside from the battery-powered wireless radiator valves. Power optimization is possible (c.f. battery powered wireless security sensors), but is not our focus. All units had ZigBee radio modules (the "XBee ZB") for wireless mesh communication with a central server PC in each house, running custom software described in the next section.

To evaluate PreHeat, we logged actual gas usage. In the UK, we deployed gas meter readers based on the RFXPulse sensor from RFXCom. In the US, we used daily meter readings provided by the local energy company website.

Occupancy sensing

In US houses, we sensed when occupants were home by using an RFID receiver plugged into the server and placing a small Active RFID tag (RF8315T-s manufactured by Ananiah Electronics) on the house keys of each adult using the house, including the nanny for US3. The option was provided to each household to have RFID tags for kids, but none of the houses accepted this offer, since the kids were in general not expected to be home alone. Visitors were not provided with RFID tags. Each tag sent its identity to the receiver every 5s when in range. The server was placed such that the 8m nominal range included the whole "front hall" area of the house, where we asked participants to leave their keys. We also asked participants to take their keys whenever they left the house (compliance results are described in the Deployment section).

In UK houses, where we could control heating at the room level, we added motion sensors that could determine per room occupancy. These detected motion by any occupant: adults, children or visitors – note that neither UK house had pets. This allowed our predictive system in the UK houses to heat for and predict arrival of any type of occupant, not just those with RFID tags. UK occupants also had RFID tags deployed which were only used during system evaluation, and not for prediction.

Our occupancy sensors (RFID and motion) generated events at discrete points in time. From this data, we derived periods of occupancy by filling in gaps between two sensed events within a certain time difference (2 minutes for RFID, 5 minutes for motion sensing during the day, and 30 minutes for motion sensing during pre-defined sleep hours).

Software

The PC we deployed in each house ran our custom software which managed the network of devices, collected sensor data and ran the heating algorithm. The heating algorithm was provided with a collection of preset parameters. These comprised (for each day of the week): Sleep and Wake times, a Sleep setpoint temperature to use between those times, and Occupied[1] and Away setpoints. In the UK, each room could have its own Occupied temperature. For our experiments, we deployed different heating algorithms, and we switched between algorithms at the preset Sleep time (which all houses chose to be the same time on all seven days of the week). The experimental system was designed so that the only difference between conditions was the algorithm's choice of which setpoint to use at a particular time. Our study used the following algorithms:

Scheduled: Equivalent to a seven-day programmable thermostat, it used preconfigured times for Leave and Return for each of the seven week days. The Away setpoint is used between Leave and Return times, and the Occupied setpoint is used otherwise (other than during Sleep hours).

AlwaysOn: Forced the use of the Occupied setpoint all the time including between Sleep and Wake.

PreHeat: Our prediction algorithm which chooses between Occupied and Away depending on current and predicted occupancy – more details later in this section.

Heating in Anticipation of Occupancy

If a space (whole house in US or room in UK) was deemed occupied by the heating algorithm, this determined the setpoint to use. However, even if a space was not occupied, it may require heating in order to be warm for future need. To achieve this, the system looked ahead up to three hours into the future. If a higher setpoint was expected by the heating algorithm, then the system evaluated the current temperature, target temperature, look ahead duration, and the HeatRate, which is the warming rate of the house in degrees per hour, to decide whether it needed to start heating at the present time. For example, with a HeatRate of 3°C/hour, space temperature of 20°C, and a requirement for 22°C in one hour's time, no heating is yet needed. Twenty minutes later, if the space temperature remained 20°C, then the system would activate the heat in order to warm up in time for the future (predicted or scheduled) occupancy.

User Interface and Overrides

We provided a user interface for occupants on the House Unit/Room Units themselves. This showed the current space temperature and the setpoint. Just as with normal heating controls, we provided a menu option on the units that gave occupants a way to override the system in case they were too cold. We also used the light sensor to automatically dim the screen when the space was dark, to make our system "bedroom friendly".

System Reliability

Since our system ran in real homes, we conducted extensive in-situ testing and implemented a number of features to promote reliability, including the ability to remotely log in to the servers, automatic emails if problems occurred, an auto-restart program to recover from software crashes, etc.

We also implemented a "failsafe mode." In the rare case that the unit was unable to contact the server, while it attempted to recover communication it would act as a local thermostat using the Occupied setpoint. As we later detail, the system achieved an average 99.8% uptime in the study.

PreHeat Occupancy Prediction Algorithm

The PreHeat prediction algorithm works in two ways. First, it uses occupancy-reactive heating; when a space is occupied, it uses the Occupied setpoint (or Sleep setpoint at night). Second, when a space is not occupied, it predicts when it will next be occupied by matching the occupancy data from the current day against historical occupancy data.

We represent space occupancy as a binary vector for each day, where each element represents occupancy in a 15-minute interval, as shown in Figure 2. In the UK such spaces are individual rooms, in the US we use a single whole-house space – no distinction is made by the algorithm. The vector element is 1 if there is any occupancy during the interval or 0 otherwise. As a day progresses, we maintain a partial occupancy vector from midnight up to the current time. To predict future occupancy, we use this partial occupancy vector to find similar days in the past. Specifically, we compute the Hamming distance between the current partial day and the corresponding parts of all the past occupancy vectors. (The Hamming distance simply counts the number of unequal corresponding binary vector elements.) We then pick the K nearest past days for making the prediction. Based on initial experiments, we found K=5 proved to be a good choice for high prediction accuracy.

[1] Although our system supported having different morning and evening setpoints (mirroring many US thermostats), all of our households chose the same value for these, so we refer to a single Occupied setpoint to simplify discussion.

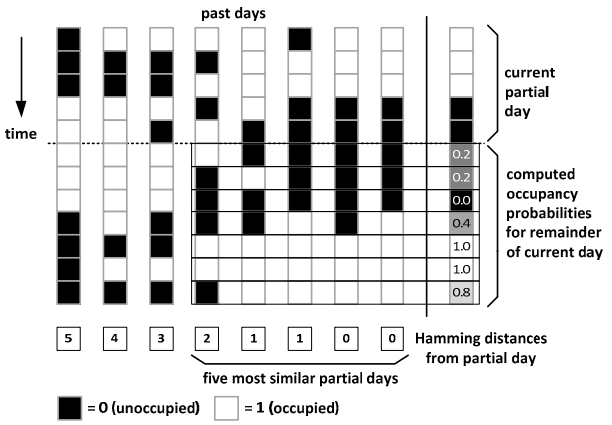


Figure 2: PreHeat prediction algorithm. Each vertical set of blocks represents one day of occupancy split into 15-minute periods. Given a partially observed current day, the algorithm finds the five best matches in the past and averages the remainder of those matched days to compute probabilities for future occupancy.

The predicted occupancy probability for a future time is simply the mean of the corresponding occupancy values in the K nearest past days.

There are two slight variations of this basic algorithm that we implemented to improve accuracy. The first variation distinguishes between weekdays and weekends. Weekday predictions are only computed from past weekdays, and similarly for weekends. This helps accuracy, because households often have quite different occupancy patterns on the two types of days. The second variation augments the beginning of each occupancy vector with four hours of occupancy data from the previous day. This gives predictions near the beginning of the day some extra basis on which to compare to previous days. We also pad the end of each occupancy vector with four hours of occupancy data from the following day. This avoids complexities in making predictions that span midnight.

While in this algorithm we treated every historical day equally for prediction purposes, we could instead automatically adapt to changing occupancy schedules by preferring more recent days when matching.

One advantage of an algorithm like ours is that it gives occupancy probabilities. This gives us the freedom to set a probability threshold for declaring when a space will be occupied. For our experiments, we set this threshold to a neutral value of ½. But, a more environmentally-conscious user may set the threshold higher to ensure that the system is more confident of future occupancy before heating. In contrast, a user more concerned about being warm would choose a lower probability threshold. We demonstrate the effect of this tradeoff in the results section below.

In testing, we found that PreHeat did not perform well in per-room situations for a certain class of rooms – those which are only occupied for very short periods of time and at random intervals, e.g. bathrooms and hallways. These spaces tended to be predicted as "never occupied". We therefore decided to use whole-house occupancy (determined by using the mathematical union of signals from all motion sensors) to predict and heat these rooms. Intuitively, whenever you are home it is possible that you might use the bathroom. In addition to these "whole house spaces", this left 6 individually controlled rooms out of 10 for UK1 and 8 rooms out of 10 for UK2.

DEPLOYMENT

Deployment allowed us to evaluate the heating algorithms using real weather conditions, central heating systems, and human occupancy behavior. Our three phase deployment lasted three to four months in each house, with an average of 61 days per house in Phase 2 (see Table 1), the main comparison between PreHeat and Scheduled heating.

Phase 0: Debug/Acclimatization (≥ 7 days)

In this phase we installed our system running the Scheduled algorithm. We showed adult members of the household how to use the UI and attached RFID tags to keys. To get the most accurate schedule we used the household's current thermostat settings as a baseline and asked household members to update the program if necessary. For per-room heating in the UK we asked participants to provide "Occupied" setpoint temperatures for each room. To model typical systems in the UK which simply turn off between heating periods rather than use a setback, we did not ask for Sleep or Away temperatures, but used a low value of 5°C, which never triggered any heating during our study.

The goal of this phase was to make sure our system was running smoothly in the home and to give people time to adjust their setpoint temperatures or scheduled times if desired. Although changes were permitted in later phases, no household requested to make such a change. We used the average observed heating rate from this phase to set the HeatRate parameter for each house (ranging from 1.5°C/hour in UK1 to 3°C/hour in US2). Phase 0 lasted a minimum of a week and was longest in US1, US2 and UK1 where we did initial in-situ testing.

Phase 1: Initial Data Collection (14 days)

PreHeat's prediction algorithm requires some occupancy history data to work. We therefore ran a 14 day phase without Prediction, which gave the subsequent prediction phase enough historical data to work accurately. In a real deployment of PreHeat, the system might bootstrap by simply heating during all non-sleep hours until it determined that its predictions were accurate enough to start using Away. In our post hoc analysis, however, we found that prediction accuracy was adequate after just one or two days of occupancy data. Given that prior research [3, 8] has used AlwaysOn as a baseline, during Phase 1 we decided to alternate between the AlwaysOn and Scheduled conditions.

Phase 2: PreHeat vs. Scheduled (48-72 days, 61 average)

This comparison was our main focus and comprised the majority of days in the study (see Table 1). In order to balance any effect weather or household schedule changes

may have on our study, we alternated each day between two conditions: using PreHeat's prediction algorithm and the Scheduled algorithm. Using neighborhood weather data provided to each US house by their utility company and from nearby public weather stations in the UK, we determined that, for all houses, the average outdoor temperature for PreHeat days differed from Scheduled days by less than 0.3°C.

Our primary metrics for comparing PreHeat and Scheduled were gas consumption and the MissTime metric used by Lu et. al [8]. They defined MissTime as the total time in minutes that the home is occupied but the temperature was more than 1°C below the Occupied setpoint. We used the same definition, but applied it per-space (room in the UK, house in the US). For gas consumption, we used automated meter readings of the volume of gas used as described earlier. Because the system was deployed in our own houses, we purposefully avoided any subjective metrics.

Households US2, US3, and UK2 each went on vacation for about a week during the study. Since at such times it is usual to do some exceptional thermostat programming (e.g. turn it off), we collected a list of such days and excluded them from Phase 2. This is primarily because vacation times are "unfair" to the Scheduled condition which heats regardless. We took advantage of vacation days to run additional AlwaysOn days in these houses. We continued to collect occupancy data during vacations, as PreHeat is robust to having these periods in the historical data set.

Reliability During Deployment

During the study period we had a small number of technical problems which caused us to remove three days from the analysis (N.B. In Table 1 and elsewhere, we have excluded these days already). In UK1, an electrical fuse trip caused two days to be excluded. In UK2, the XBee radio module attached to the PC "hung" and required manual power cycling, causing one day to be excluded.

Of the study days used for analysis, the system was in a not-fully-functional state for an average of three minutes per day (uptime 99.8%). House UK2 had the most problems, averaging eight minutes per day where units were in "failsafe" mode. Most problems were due to XBee, and the system recovered automatically.

Ensuring Occupancy Correctness

Because our MissTime metric is only as valid as the occupancy data used, we took extensive steps to ensure accuracy. Each server directed an email to a house occupant and a project member every morning with a summary of the previous day's per-space occupancy. The recipients looked for any discrepancies, consulting family members if necessary. The documented errors included both "social" failures due to forgetting to carry an RFID tag, and "technical" failures where some aspect of the sensing failed. We also ran data integrity checks on the 37,000 occupancy records generated over the course of the study to find failure cases: In the US, we examined days where RFID tags disappeared many times (a radio range failure). In the UK, we used the RFID data (which was not used for heating prediction purposes) to highlight times when the house was occupied but no motion sensors were active.

Using the documented errors and integrity checks we found a set of 57 issues that could affect the analysis and corrected them in a "ground truth" occupancy table. In the US, there were 17 corrections, of which 14 were "social", and 12 of those occurred in US1. This household had keyless entry so the RFIDs were not as convenient. In the UK, 21/40 corrections were due to occupants being asleep outside the Sleep/Wake period (and thus difficult to detect using motion). In future work, we hope to address sleep detection. Issues with a motion sensor in UK2 not completely covering a room resulted in 18/40 corrections.

We compared the MissTime metric using the original table against the ground truth table and found that only a single instance actually altered the result set (this impacted UK2 - the corrected data is reported in this paper). In the remaining instances, the house was warm enough to prevent the error from impacting the occupants. It is important to note that the PreHeat algorithm did NOT use corrected data; it predicted based on live data that included any errors. This is more representative of a real-life deployment.

PREHEAT VS SCHEDULED RESULTS

Figure 3 and Table 1 summarize how the measured gas consumption and MissTime metrics differ between the Scheduled and PreHeat conditions in each house.

In the UK, PreHeat performed better than Scheduled on both metrics. PreHeat saved energy by not only selecting better heating times, but also by heating rooms different amounts – more details to follow in the Per-Room Heating section. Gas usage decreased from Scheduled by 18% in UK1 and 8% in UK2. In both conditions, the UK houses had comparatively little MissTime, but PreHeat also succeeded in decreasing this by 38% and 60%.

In the US houses, MissTime improved by a large factor (84%, 88% and 92% reductions, a factor of 6-12 decrease!).

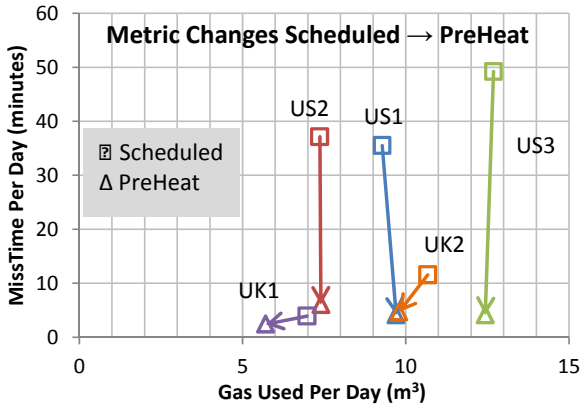


Figure 3: PreHeat improves the trade-off between gas used and MissTime compared to Scheduled

	US1	US2	US3	UK1	UK2
Phase 2 Days (Alternating PreHeat and Scheduled)	72	64	58	62	48
Average Daily Occupancy	89%	69%	67%	73%	68%
Savings in MissTime PreHeat vs. Scheduled	88%	84%	92%	38%	60%
Savings in Gas Used PreHeat vs. Scheduled	-5% (worse)	-1%	2%	18% (better)	8%
Savings in Gas Used PreHeat vs. AlwaysOnModel	3%	17%	10%	27%	35%

Table 1: PreHeat decreased MissTime in all houses. Gas Used decreased in PreHeat in the UK homes with per room heating and was equivalent or slightly higher for the three US homes.

This is further shown by Figure 4 which illustrates that MissTime happens unevenly through the study – over half the days had no MissTime, but one Scheduled day in five (in US houses) had over one hour. PreHeat, unlike a static schedule, is able to dynamically heat day-by-day to more closely match the occupancy. To give one example, in US2, PreHeat heated more on weekends and less on weekdays than Scheduled, while being adaptive to instances of absence on weekends or occupancy on weekdays. PreHeat was able to achieve these large improvements in MissTime while using a similar amount of gas – saving slightly in US3 and using slightly more in US1 and US2. In the Occupancy Prediction section, we discuss how PreHeat could instead be tuned to favor energy savings at the expense of a smaller improvement in MissTime.

Many More Overrides in Scheduled

Our system's user interface allowed occupants to override the current action. We classify overrides into two categories: "time" overrides when the system was not heating and the user overrode it to heat to the normal Occupied temperature, and "temperature" overrides when a non-standard temperature was set. During the study, there were 23 time overrides, 21 of which occurred in the Scheduled condition. The "time" overrides in Scheduled happened when people were home on vacation, sick, or otherwise home at times when Scheduled was not heating the house and they felt cold. The overrides caused more energy to be used, but without these overrides, the MissTime metric would have been (even) higher. The occupancy-reactive element of PreHeat reduces the need for overrides. There were two "time" overrides in the PreHeat condition, which were due to occupancy sensing failures – a guest in US2 who did not have an RFID tag, and a resident of US1 left her RFID tag in the car. The small number of "time" overrides in PreHeat supports our claim that PreHeat requires less programming of a heating schedule by occupants (since overriding is a form of programming).

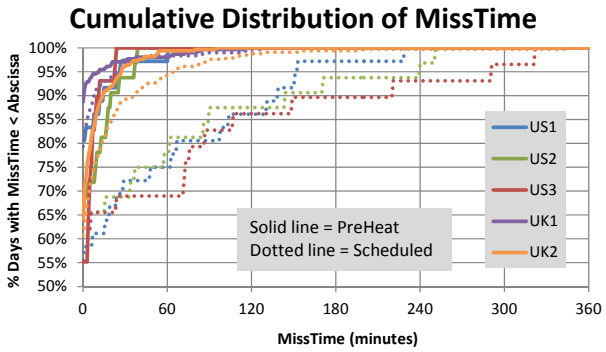


Figure 4: Occurrences of long MissTimes are much reduced with PreHeat, particularly for US houses. (Note Y-axis intercept at 50%)

There were nine "temperature" overrides during Phase 2. These types of overrides can occur in either condition if an occupant wanted a deviation from the normal setpoint. In US3, participants overrode the temperature to be 1-3 degrees higher than the setpoint seven times: three in Scheduled and four in PreHeat. In UK2, an occupant overrode the temperature to be lower than the setpoint twice, both in PreHeat, because it was sunny and he didn't feel the heat needed to be on.

Per-Room Heating

Although we did not study the effect of per-room heating through a direct comparison, we can still get a measure of how well PreHeat tailored its behavior to individual room occupancy patterns.

Figure 5 shows UK1's six per-room-controlled spaces (the other four spaces used whole-house occupancy for heating as previously described), illustrating average occupancy in each space and the amount of time that the Occupied setpoint was used in both PreHeat and Scheduled conditions. We see that Scheduled heats around the same amount in each space (variation is due to turning on a little earlier or later in each space depending on how cold it gets when away/asleep). PreHeat tracks occupancy fairly well.

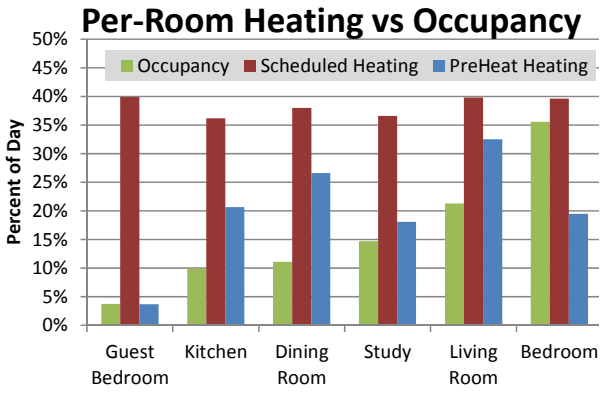


Figure 5: PreHeat heats higher-occupied rooms more, while Scheduled does not adapt per-room.

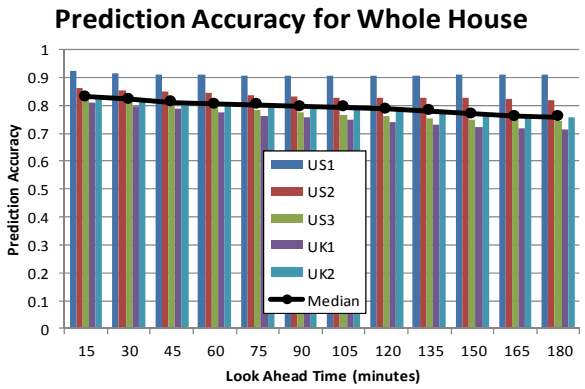


Figure 6: Prediction accuracy varied only slightly with the look ahead time.

Note that the rightmost room is the bedroom where all occupants sleep, so this space needs less heating than occupancy might suggest.

OCCUPANCY PREDICTION

Occupancy prediction is a key part of PreHeat. This section explores how accurate the predictions are compared to actual occupancy.

Figure 6 shows the overall prediction accuracy for different look ahead times for the five houses. We compute accuracy for a given look ahead time as simply the number of correct occupancy predictions (either true or false) divided by the number of attempted predictions. The graph reveals that prediction accuracy for all houses was generally high. The median line shows that prediction accuracy only slightly decreases for longer look ahead times.

Since the system needs advance notice of the need for heat in order for the space to reach the desired setpoint in time, we should evaluate the prediction based on how well it can achieve this goal. Examining each daytime heating instance during Phase 2, we find that 91% of the time, the system needed 90 minutes or less advance notice. For the remainder of our prediction assessments, we evaluate based on this 90-minute look ahead time.

Figure 7 shows the prediction accuracies for 90 minutes into the future. For comparison, it also shows the prediction accuracy of the manually programed Scheduled condition, which was worse by a median 10 percentage points. We suspect that the improvement from using PreHeat would be much greater for the general population, many of whom do not keep up to date programs on their thermostats or do not have programmable thermostats [3].

As one might expect, prediction accuracy varied with the time of day. Accuracy is highest during sleep time and drops during the day when occupancy is naturally less predictable. Because prediction accuracy is high during sleep times, we have eliminated sleep times from all our prediction accuracy assessments. Thus, all our prediction

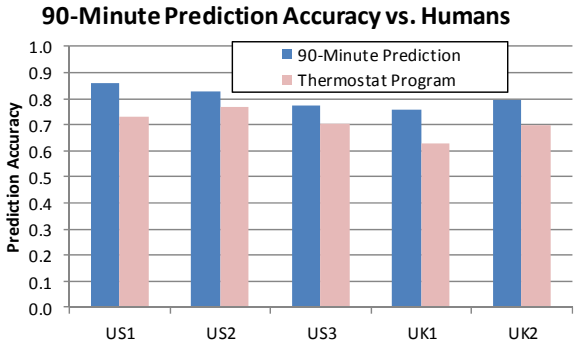


Figure 7: Our prediction algorithm was consistently more accurate than carefully programmed thermostats.

assessments, including those in Figure 6 and Figure 7, pertain only to non-sleep times, eliminating the uninteresting boost in accuracy we would otherwise get.

We also evaluated how prediction accuracy varied with the day of the week. As expected, weekend days are worse, with Sunday being the better of the two. We attribute this to the fact that our households generally had parents with regular working hours and children with regular school hours during the week.

Our prediction algorithm computes occupancy probabilities that are then subjected to a threshold to make a concrete occupancy prediction. For our study, this threshold was set to a neutral value of ½. The effect of this threshold is shown in the receiver operating characteristic (ROC) curve in Figure 8. For different values of the probability threshold, these curves show the tradeoff between false positives (mistakenly predicting positive occupancy when the home would actually be empty) and true positives (correctly predicting positive occupancy). Our selected threshold value is indicated by a dot on each ROC curve.

The ideal operating point is in the upper left corner, where the false positive rate is 0.0 and the true positive rate is 1.0.

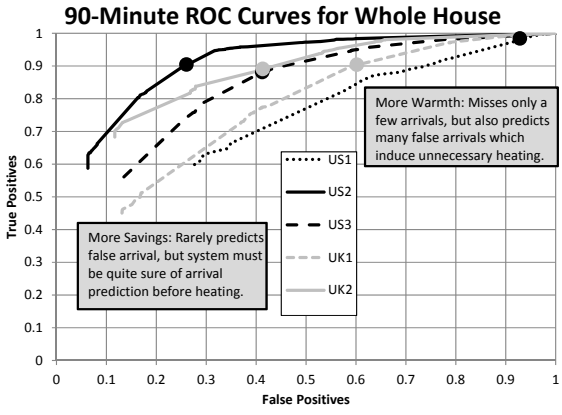


Figure 8: These ROC curves demonstrate the tradeoff in prediction errors due to adjusting the probability threshold.

In reality, as with almost all detection problems, increasing the true positive rate comes at the expense of increasing the false positive rate. An environmentally-conscious user, who is more concerned about energy rather than warmth, would operate with a high probability threshold, which is toward the left side of the ROC curve. This would give fewer mistaken heating events at the expense of missing times when the heat should actually be turned on. A more warmth-sensitive user would operate with a lower probability threshold, which would cause heating more often during times of both occupancy and non-occupancy.

Our study results show that PreHeat improved MissTime in all houses by substantial amounts. This indicates that our selected threshold was generally closer to the "warmth" end of the curves than the "savings" end, as verified in Figure 8. The operating point for US1 is biased far toward warmth. This is because this house was almost continuously occupied, resulting in prediction probabilities that were generally high (including sleep time, US1 was occupied 89% of the day vs. a median of 69% for the other houses). Even with a relatively high false positive rate of 93%, the house was occupied so much that its overall 90-minute prediction accuracy was highest of all the houses at 86%.

Finally, we also looked at prediction on a per-person basis, by applying our algorithm to each individual RFID tag using the tag's unique ID. Running our algorithm in this way showed that our prediction accuracy is even higher, with a median accuracy of 97% compared to a whole-household median of 80%. This means our algorithm could be used to accommodate different temperature preferences when only one of the home's occupants is present.

DISCUSSION

We now discuss how PreHeat compares with AlwaysOn and to occupancy-reactive heating, and share some observations from our experiences of living with PreHeat.

Comparison with Other Heating Algorithms

Our main study directly compared PreHeat with Scheduled based on actual measurements. We now discuss how PreHeat compares to other algorithms.

AlwaysOn

Prior research [e.g. 3, 8] frequently uses the energy necessary to keep the house at a permanent setpoint (AlwaysOn) as a baseline for comparison. Given its past inclusion, we also provide this comparison baseline; however, we believe the comparison between PreHeat and Scheduled to be more meaningful.

Since there were not sufficient days in a single winter to run three conditions, we elected to only gather AlwaysOn data during Phase 1 and vacations. We then used linear regression on this data to model the gas consumed given the average outside daily temperature. The linear regression showed average daily temperature was a very good predictor of gas used for the US houses, where more than 92% of the variance in the data was accounted for in the linear regression equation (all $R^2 > 0.92$). Modeling did not work as well in the UK, with $R^2 = 0.69$ in UK1 and $R^2 = 0.78$ in UK2. However, the models still give us some ability to compare PreHeat to AlwaysOn.

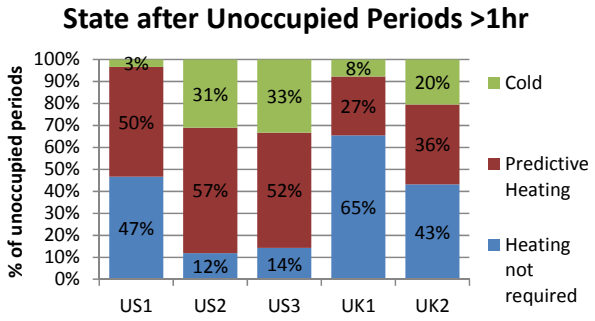


Figure 9: Predictive heating was important in all houses. While occupancy-reactive heating would have sufficed in 36% of cases (average across households) where heating was not required, predictive heating was used to achieve warmth on return for an average of 45% of cases.

For each PreHeat day we used the average daily temperature as input to the linear regression equation. We show the comparison between the model's result and the actual gas used by PreHeat in Table 1. As expected, PreHeat uses less gas than the AlwaysOn model (ranging from a 3% to a 35% decrease). Although the difference in US1 is quite small (3%) we attribute this to the fact that the house was only empty 11% of the day which does not provide much opportunity for savings.

Occupancy-reactive

The PreHeat system incorporates both occupancy-reactive and predictive heating. We wished to evaluate how often just occupancy-reactive heating would have been sufficient in our study, and thereby get a measure of the value of the predictive element. We therefore looked for instances when someone entered a space that had been unoccupied for more than 60 minutes on PreHeat days. As Figure 9 shows, occupancy-reactive heating would have worked 36% of the time (average for all houses) without any MissTime – i.e., the space was still warm enough when someone returned. However, in 45% of the instances PreHeat heated the unoccupied space prior to the person's return and so occupancy-reactive heating would have caused MissTime to occur. Thus we can conclude that predictive heating plays a very significant part in the PreHeat system.

Living With PreHeat

We quantitatively compared the Scheduled and PreHeat algorithms because of the potential for bias when using our own system. However, based on our experience with living with PreHeat, we report a few qualitative examples of how PreHeat gracefully adapted to our homes.

PreHeat Better Handles Weekend Chaos: During the setup phase, we heard that weekend schedules varied more than weekday schedules, and that it was difficult to program the thermostat for weekends. Households adopted one of

two strategies for their (previous) thermostat programs, either heating all day on the weekends (US1, US3) or using a shorter away duration than their weekday schedule (US2, UK1, UK2). These strategies were carried over into the study's Scheduled condition. In either case, PreHeat better handled weekend heating without requiring manual effort by household members. For households with a weekend setback, the house stayed warm when they were home on PreHeat days without an override. Households that left the heating on all day had the potential to save energy on weekend days if they left home.

PreHeat Supports More Complicated Occupancy Patterns: Programmable thermostats typically allow people to schedule one away period per day. In contrast, PreHeat can predict multiple away periods. For example, by the end of the study, PreHeat was correctly predicting that US2 would come home and then leave again for a regular Friday evening appointment, and similarly for UK1 on Tuesday evenings. More valuable was PreHeat's ability to predict on a per-room basis in the UK. Even if households had the ability to program on a per room basis, it is unlikely they would make the effort to maintain up-to-date programs in each room separately. We also observed that PreHeat handled occupancy patterns that re-occurred, but not on a weekly basis. For example, PreHeat smoothly handled the fact that a member of US2 arrived home early roughly every other Tuesday afternoon.

PreHeat Adapts to Changing Schedules: In US3, the Nanny who stayed at home after school with the kids took another job during the study. As is likely typical after this type of change, it did not occur to the occupants of US3 that they should change their heating schedule. However, PreHeat adapted over time and began correctly predicting later arrival times at the home.

CONCLUDING REMARKS

By predicting future occupancy from historical data and current occupancy, the PreHeat system provides a better trade-off between energy use and MissTime (the amount of time an occupied space is cold) than a thermostat program, and does so without requiring a user to program an occupancy schedule (which past research has shown that many users fail to do). We evaluated PreHeat using a real deployment in five family homes during winter 2011, alternating days between PreHeat and scheduled heating to provide a direct comparison with measured gas consumption and MissTime.

In three US homes, we found that PreHeat reduced MissTime by a factor of 6-12 while using around the same amount of gas. Across two UK homes, PreHeat halved MissTime and also reduced gas usage by 8% and 18% - this is because UK homes used PreHeat on a per-room basis, so it was able to make additional savings by heating rooms adaptively at different times of day, again without requiring any programming of per-room schedules.

Our research suggests several interesting directions for future work. We want to investigate improvements to the PreHeat algorithm, e.g. using other sources of data such as location from phones, or further exploring per-person predictions. Sleep detection would assist with automatically determining times for setpoint changes. A more sophisticated heating model could also enable more savings, e.g. by predicting departures and "pre cooling" since houses stay warm for some time. Finally, we plan to explore exposing the high-level tradeoffs PreHeat offers between the likelihood of being warm when arriving home unexpectedly and energy consumption to users so they can customize based on their personal preferences.

ACKNOWLEDGEMENTS

Thanks to Kori and Jeff Quinn, and to our families, for being our guinea pigs!

REFERENCES

1. EIA, USE.I.A. Table 2.5 "Household Energy Consumption and Expenditures by End Use and Energy Source, Selected Years 1978 – 2005." Retrieved in April 2011 at http://www.eia.doe.gov/emeu/aer/consump.html.
2. EPA, USE.P.A., Summary of Research Findings From the Programmable Thermostat Market. Available from: http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/thermostats/Summary.pdf
3. Gupta, M., S.S. Intille, and K. Larson. "Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges." *Proc. of Pervasive,* 2009.
4. Hargreaves, T., Nye, M., and Burgess, J. "Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors." *Energy Policy* 38:10, pp. 6111-6119, 2010.
5. Krumm, J. and Brush, A. "Learning Time-Based Presence Probabilities", *Proc. of Pervasive 2011*, 2011.
6. Meier, A., Aragon, C., Hurwitz, B., Mujumdar, D., Peffer, T., Perry, D., Pritoni, M. "How People Actually Use Thermostats", *Proc. of ACEEE 2010.*
7. Mozer, M.C., Vidmar, L., and Dodier, R.M. "The Neurothermostat: PreHeat Optimal Control of Residential Heating Systems." *Advances in Neural Information Processing Systems* 9, pp. 953-959, 1997.
8. Lu, J, Sookoor, T., Srinivasan V., Gao, G, Holben, B., Stankovic, J., Field, E., Whitehouse, K. "The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes." *Proc. of SenSys,* 2010.
9. Sonderegger, R. "Movers and stayers: The resident's contribution to variation across houses in energy consumption for space heating." *Energy and Buildings* 1:3, pp. 313-324, 1978.
10. Wilhite, H. and Ling, R. "Measured energy savings from a more informative energy bill." *Energy and Buildings* 22:2, pp. 145-155, 1995.