Data-Driven Activity Prediction: Algorithms, Evaluation Methodology, and Applications

Bryan Minor
Washington State University
Pullman, WA 99163, USA
bminor@eecs.wsu.edu

Janardhan Rao Doppa
Washington State University
Pullman, WA 99163, USA
jana@eecs.wsu.edu

Diane J. Cook
Washington State University
Pullman, WA 99163, USA
cook@eecs.wsu.edu

ABSTRACT

We consider a novel problem called *Activity Prediction*, where the goal is to predict the future activity occurrence times from sensor data. In this paper, we make three main contributions. First, we formulate and solve the activity prediction problem in the framework of imitation learning and reduce it to simple regression learning problem. This approach allows us to leverage powerful regression learners; is easy to implement; and can reason about the relational and temporal structure of the problem with negligible computational overhead. Second, we present several evaluation metrics to evaluate a given activity predictor, and discuss their pros and cons in the context of real-world applications. Third, we evaluate our approach using real sensor data collected from 24 smart home testbeds. We also embed the learned predictor into a mobile device based activity prompter and evaluate the app on multiple participants living in smart homes. Our experimental results indicate that the activity predictor learned with our approach performs better than the baseline methods, and offers a simple and reliable approach to prediction of activities from sensor data.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

Keywords

activity prediction; smart environments; digital prompting; regression learning

1. INTRODUCTION

Learning and understanding observed activities is at the center of many fields of study. An individual's activities affect that individual, society, and the environment. Over the past decade, the maturing of data mining and pervasive computing technologies has made it possible to automate activity learning from sensor data. This activity information is now commonly utilized in applications from security systems to computer games. As a result of this technology push and application pull, robust approaches exist for labeling activities that occurred in the past or may be occurring in the present. In this paper, we propose to extend this recent work to look at activities that will occur in the future.

We study a novel problem called *Activity Prediction*, where the goal is to predict the future activity occurrence times from sensor data, and introduce a data-driven method for performing activity prediction. Activity prediction is valuable for providing activity-aware services such as energy-efficient home automation, prompting-based interventions, and anomaly detection. However, activity prediction faces challenges not found in many other data mining tasks: the sensor readings are noisy, activity labels provided by activity recognition algorithms are subject to error, and the data contains rich spatial and temporal relationships that must be exploited to be able to make highly-accurate predictions.

We formulate and solve the activity prediction problem as an instance of the imitation learning framework, where the training data serves as the demonstrations provided by the expert. We provide a reduction of activity prediction learning to simple regression learning, which allows us to leverage powerful off-the-shelf regression learners to learn an effective activity predictor that can reason about relational and temporal structure among the activities in an efficient manner. Our approach naturally facilitates life-long learning setting, where the predictor can be improved and adapted based on the new data from the users.

Selecting performance metrics for activity prediction is challenging because there are multiple parameters that influence the desirability of the algorithm's performance. We provide several evaluation metrics and discuss their usefulness in the context of real-world applications. We evaluate our prediction algorithms on twenty-four smart home sensor datasets and find that our proposed imitation-based methods not only outperform baseline predictors but predict a majority of the activities within minutes of their actual occurrence. In addition, we embed our activity predictor inside an activity prompting algorithm and demonstrate the effectiveness of the prompting app for multiple participants living in smart homes.

2. PROBLEM SETUP

We consider the problem of *Activity Prediction* from sensor event data. Let $A = \{a_1, a_2, \cdots, a_T\}$ be the set of all activities, where a_i corresponds to the i^{th} activity class. Given features $\mathbf{x} \in \Re^d$ extracted from the sensor event data

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783408.

at time t_e as input, the activity predictor needs to generate $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T)$ as output, where $\hat{y}_i \in \Re$ is the predicted relative next occurrence time of activity a_i, or the predicted number of time units that will pass until a_i occurs again. Figure 1 provides an illustration of the activity prediction problem.

Our training data consists of a sequence of raw sensor events $\Lambda = (\lambda_1, \lambda_2, \cdots, \lambda_N)$, where λ_i corresponds to sensor readings generated at time t_i. We assume that an activity recognition (AR) algorithm is available to label each sensor event with its corresponding activity class and we use this information to train the activity predictor. An activity recognition algorithm learns a mapping from Λ to the corresponding activity label, a_Λ. We employ the AR algorithm [4] which yields 95% recognition accuracy via 3-fold cross validation on the activities evaluated in this paper.

We further assume the availability of a *feature function* Φ that computes a d-dimensional feature vector $\Phi(\lambda_i) \in \Re^d$ for any sensor event λ_i using the context of recent sensor events and a non-negative *loss function* L such that $L(x, \hat{y}, y^*) \in \Re^+$ is the loss associated with labeling a particular input $\mathbf{x} \in \Re^d$ by output $\hat{\mathbf{y}} \in \Re^T$ when the true output is $\mathbf{y}^* \in \Re^T$ (e.g., RMSE). Our goal is to return a function/predictor whose predicted outputs have low expected loss.

3. LEARNING ALGORITHMS

In this section we describe two algorithms for learning activity predictors: 1) The Independent Predictor (IP), a simple baseline approach, and 2) The Recurrent Activity Predictor (RAP), which is intended to improve on the baseline.

3.1 Independent Predictor

The *Independent Predictor* is our baseline activity predictor. As the name suggests, this predictor completely ignores the relational and temporal structure of the problem, and makes predictions using only the information from the most recent sensor events at a given time. The independent predictor is trained as follows. For each sensor event data λ_i in the training sequence Λ, we extract the features $\mathbf{x}_i = \Phi(\lambda_i) \in \Re^d$ listed in Table 7 (input) and the ground-truth activity predictions $\mathbf{y^*}_i \in \Re^T$ (output) from the labeled activity segments (see Figure 1 for an illustration). The aggregate set of input-output pairs $\{\mathbf{x}_i, \mathbf{y}^*_i\}_{i=1}^N$ (training examples) is given to a multi-output regression learner to learn the activity predictor by minimizing the given loss function L.

This approach is simple and the test-time complexity of the predictor is very low, which is valuable for making real-time predictions. However, the main weakness of this approach is that the local sensor event data may not provide sufficient information to make highly-accurate activity predictions.

3.2 Recurrent Activity Predictor

Notice that the independent predictor only uses the local sensor event data at a given time to make its predictions. To address this weakness, one could consider joint models by reasoning over the relationships between different activities and accounting for the temporal structure of the problem. It is important to note that the activity prediction problem can be viewed as a generalization of sequence labeling, where each output token is a vector of T real values corresponding to the next activity occurrence time of each activity (T is the number of activities).

A natural solution would be to define a graphical model encoding the relationships between input and output variables at different time steps and learn the parameters from the training data [21]. However, such a graphical model may be very complex (high tree-width) and can pose severe learning and inference challenges. We may consider simplifying the model to allow for tractable learning and inference, but that can be detrimental to prediction accuracy. An alternate solution is to employ a heuristic inference method (e.g., loopy belief propagation or variational inference) with the complex model. Even though these methods have shown some success in practice, it is very difficult to characterize their solutions and to predict when they will work well for a new problem. Therefore, we provide a much simpler, but effective solution that is based on imitation learning.

Recurrent Predictor. The *recurrent predictor* employs both the local features computed from the recent sensor event window and context features which try to capture the activity predictions from a small history window to make its predictions. The main advantage of a recurrent predictor, compared to the graphical model solution, is that it allows us to encode arbitrary relationships between activities and the temporal structure as context features and is highly efficient in terms of training and testing.

Imitation Learning Approach. We formulate and solve the activity prediction problem in the framework of imitation learning. In traditional imitation learning, the goal of the learner is to learn to imitate the behavior of an expert performing a sequential-decision making task (e.g., playing a video game) in a way that generalizes to similar tasks or situations. Typically this is done by collecting a set of trajectories of the expert's behavior (e.g., games played by the expert) on a set of training tasks. Then supervised learning is used to find a predictor that can replicate the decisions made on those trajectories. Often the supervised learning problem corresponds to learning a mapping from states to actions and off-the-shelf classification tools can be used. In recent work, imitation learning techniques are successfully applied to solve a variety of structured prediction tasks in natural language processing and computer vision [11, 6, 5, 25, 30, 22, 23].

In our activity predictor learning problem, the expert corresponds to the loss function L (available for training data) and the expert behavior corresponds to predicting the best output $\mathbf{y}^*_i \in \Re^T$ at each time step i (see Figure 1). To make predictions, the activity predictor uses both local features $\Psi_{local}(i) = \Phi(\lambda_i)$ (see Table 7) and prediction context features $\Psi_{context}(i)$, including the previous activity predictions from a small history window. The context features $\Psi_{context}(i)$ we employ in this work include the predicted labels $\hat{y} \in \Re^T$ for all the T activities and for each history window. If we use a history context of H previous window, the context feature vector will be of size $H \cdot T$.

Algorithm 1 provides the pseudo-code of our approach for recurrent activity predictor learning via exact imitation of the loss function. At each time step i, we compute the joint features $\Psi_i = \Psi_{local}(i) \oplus \Psi_{context}(i)$ (input) and the best activity predictions $\mathbf{y}^*_i \in \Re^T$ (output) from the training data, where $\oplus$ refers to the vector concatenation operator. Note that for the exact imitation training, context features consist of ground-truth labels from the previous windows. The

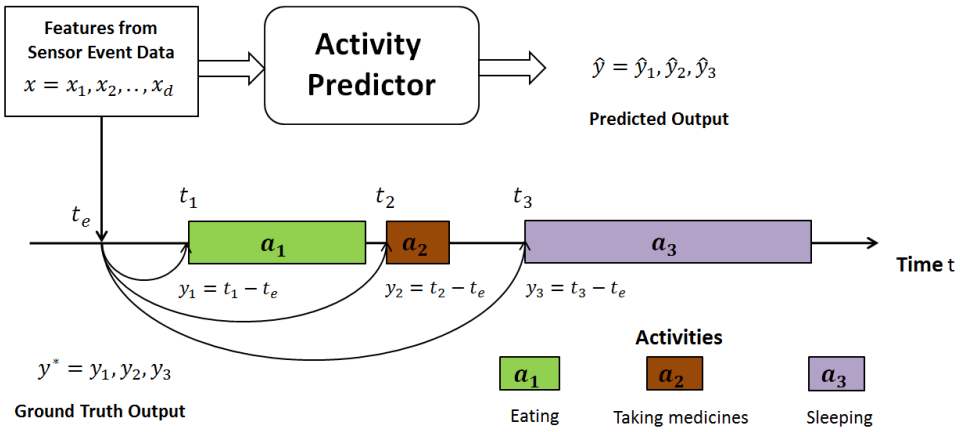

Figure 1: A high-level overview of the activity prediction problem. Given features $\mathbf{x} \in \Re^d$ extracted from the sensor event data at time t_e as input, the activity predictor needs to predict the relative occurrence time of each activity. In this example, we have three activities: a_1 (eating); a_2 (taking medicines); and a_3 (sleeping). The starting times of activities a_1, a_2, and a_3 are t_1, t_2, and t_3, respectively. Therefore, the ground-truth output is $\mathbf{y}^* = (y_1, y_2, y_3)$, where $y_i = t_i - t_e$ stands for the correct relative next occurrence time of activity a_i.

Algorithm 1 RAP Learning via Exact Imitation

Input: Λ = Training sequence of sensor event data labeled with activity segments, L = Loss function
Output: $\mathcal{F}$, the recurrent predictor

```
Initialize the set of regression examples D = ∅
for each time step i = 1 to |Λ| do
    Compute local features Ψ_local(i) = Φ(λ_i)
    Compute context features Ψ_context(i)
    Compute joint features Ψ_i = Ψ_local(i) ⊕ Ψ_context(i)
    Compute best output y*_i ∈ ℜ^T using the loss function
    Add regression example (Ψ_i, y*_i) to D
end for
F = Multi-Output-Regression-Learner(D)
return learned predictor F
```

aggregate set of input-output pairs $\{\mathbf{x}_i, \mathbf{y}_i^*\}_{i=1}^N$ (training examples) is given to a multi-output regression learner to learn the recurrent activity predictor by minimizing the given loss function L. If we can learn a function $\mathcal{F}$ that is consistent with these imitation examples, then it can be proved that the learned function will generalize and perform well on new instances [17, 26].

One issue with exact imitation training is error propagation: errors in early time steps can propagate to downstream decisions and can lead to poor global performance [14]. If the error propagation problem arises, we could employ more advanced imitation learning algorithms including DAgger [26] to learn robust predictors. DAgger is an iterative algorithm that can be viewed as generating a sequence of predictors (one per iteration), where the first iteration corresponds to exact imitation training. In each subsequent iteration, DAgger makes decisions with the predictor from the previous iteration, and generates additional training examples to learn to recover from any errors. A new predictor is learned from the aggregate set of training examples. In the end, the final predictor is selected based on a validation set. DAgger also has nice theoretical properties and can be seen as a no-regret online learning algorithm [26]. If we deploy the learned recurrent predictor in a real-life application, then the predictor can be adapted online based on feedback from the users, and the DAgger algorithm can be employed to naturally facilitate a *life-long learning* setting.

3.3 Multi-Output Regression Learner

Recall that both of our activity predictor learning algorithms need a cost-sensitive multi-output regression learner. The multi-output regression learning problem is the regression analog of the multi-label classification problem, where individual labels are real-values instead of binary labels [7].

In principle, we can employ any multi-output regression learner. However, inspired by the Binary Relevance (BR) classifier for multi-label classification [7], we decompose the multi-output regression problem by learning one regression function for each output variable independently. Therefore, we learn T regressors and employ them for making predictions. We have a hard learning problem at hand, which means linear functions won't suffice. We experimented with standard regression tree learners (constant outputs at leaves), but they couldn't handle the high variance for some activity times. Hence, we employed a variant of regression trees called model trees, where predictions are made by a learned linear function over all the features at each leaf node.

4. EVALUATION METHODOLOGY

In this section, we will present several evaluation metrics to evaluate activity prediction algorithms and discuss their pros and cons in the context of real-world applications. To compare the effectiveness of different solution approaches for a given problem, the evaluation metrics must be carefully chosen. The quality and usefulness of a particular metric will vary based on the application and specific evaluation criteria. Many metrics tend to emphasize particular aspects of the results, so choosing multiple metrics can be necessary to completely understand the effectiveness of an approach.

Challenges. Selecting performance metrics for activity prediction is challenging because there are multiple parameters that influence the desirability of the algorithm's per-

formance. Activity predictors can be evaluated in multiple ways, depending upon the type of performance that is desired. First, activity prediction can be viewed as a type of classification task in which any prediction that has non-zero error (or error greater than a threshold) is considered a mislabeled data point. In this case, traditional classifier-based performance measures can be utilized. Second, activity prediction can be considered as a type of forecasting algorithm. Viewed in this light, error is proportionate to the numeric distance between the predicted and actual values. In addition, activity prediction relies on the effectiveness of an online activity recognition algorithm. The performance of the activity predictor is not anticipated to exceed the reliability of the activity recognizer that is being used to train the predictor. The activity recognizer, in turn, is trained using hand-annotated data which may be inconsistently labeled.

Evaluation Metrics. We introduce several evaluation metrics and employ them to validate our prediction algorithms. Using our previous notation, $\hat{\mathbf{y}}$ represents a vector of predicted outputs for each sensor event in the evaluation dataset with elements $\hat{y}_i$. $\mathbf{y}^*$ is the vector of true values for the same event with elements y_i^*. Note that, we have T activities in total. Each evaluation metric takes a predicted output $\hat{\mathbf{y}}$ and ground truth output $\mathbf{y}^*$ as input, and returns a real-value indicating the quality of the prediction. One could perform macro-averaging of metric values over different testing instances and datasets to compute aggregate values.

Mean absolute error (MAE), as defined in Equation 1, provides a measure of the average absolute error between the predicted output and ground-truth output. It is similar to another well-known metric, **root mean squared error (RMSE)**, defined in Equation 2. Both of these measures provide the average error in real units and quantify the overall error rate, with a value of zero indicating a perfect predictor and no upper limit. Because RMSE squares each term, it does bring a disadvantage in effectively weighting large errors more heavily than small ones.

$$\text{MAE} = \frac{\sum |\hat{y}_i - y_i^*|}{T} \tag{1}$$

$$\text{RMSE} = \sqrt{\frac{\sum (\hat{y}_i - y_i^*)^2}{T}} \tag{2}$$

If the activities have varying levels of importance, we may want to use an error measure that places more emphasis on some activities than others (e.g., weighted RMSE). This might be the case if a particular activity needs to be predicted very accurately, for example. Additionally, we may need to compare results across activities or datasets where the time spacing between activity occurrences may be different. In these cases, measures such as MAE and RMSE do not give an indication of the *relative* error. For example, an error of 60 minutes in predicting a time-critical activity (e.g., taking medicine) may be unacceptable, but may be acceptable for other activities that do not need to happen at a certain time (e.g., housekeeping). In such situations, we may want to use a normalized error, such as **range-normalized RMSE (NRMSE)**, defined in Equation 3. Here, the minimum and maximum functions are computed over all ground-truth values of the test instances we are evaluating. This metric would usually be applied on each activity or dataset that we wish to separate. While NRMSE is convenient for comparing results from different sets, it does not have a well-defined normalization factor with which we can evaluate the actual magnitude of the errors.

$$\text{NRMSE} = \frac{RMSE}{\max(y_i^*) - \min(y_i^*)} \tag{3}$$

Another useful normalized metric is **mean absolute percentage error (MAPE)**, defined in Equation 4. MAPE normalizes each error value for each prediction by the true value y_i^* we are trying to predict. This metric allows us to normalize the error individually for each prediction. We can also quickly determine approximately how large the error is since it is a percentage of the true activity time. However, as y_i^* approaches zero (i.e., the activity is about to occur), an error of any insignificant amount can cause element in the summation to become large. This leads to inflation of the MAPE value due to a few outlier cases where the error is small but the true activity time is even smaller.

$$\text{MAPE} = \frac{\sum \frac{|\hat{y}_i - y_i^*|}{y_i^*}}{T} \tag{4}$$

Since the metrics we have listed so far are based on finding the averages of all errors, they are sensitive to possible distortion by outliers. As a result, the metrics can often have large values. In order to analyze the effects of outliers, other evaluation metrics can be used. One metric we introduce for this purpose is the **error threshold fraction (ETF)**, defined in Equation 5. $I(\hat{y}_i, y_i^*) = 1$ if $|\hat{y}_i - y_i^*| \leq v$ and 0 otherwise. Note that the numerator of the fraction is a count of the number of events with error below the threshold v. This metric indicates the fraction of the errors that are below the time threshold v. v should be non-negative, and $\lim_{v\to\infty} \text{ETF}(v) = 1$. By varying v we can ascertain how the errors are distributed; if we find that the ETF does not approach 1 until v is large, this may indicate that there are a significant number of large-error outliers. ETF(0) indicates the number of predictions which had zero error.

$$\text{ETF}(v) = \frac{\sum I(\hat{y}_i, y_i^*)}{T} \tag{5}$$

Yet another metric to consider is **Pearson's** r, i.e., the correlation coefficient between the predicted and actual activity occurrence times. This measure, shown in Equation 6, does not quantify the amount of error but does indicate the relationship between the predicted and actual values.

$$\text{r} = \frac{\sum(\hat{\text{y}}_\text{i} - \overline{\hat{\text{y}}_\text{i}})(\text{y}_\text{i}^* - \overline{\text{y}_\text{i}^*})}{\sqrt{\sum(\hat{\text{y}}_\text{i} - \overline{\hat{\text{y}}_\text{i}})}\sqrt{\sum(\text{y}_\text{i}^* - \overline{\text{y}_\text{i}^*})}} \tag{6}$$

Usually there is no single best evaluation metric for any particular application. We may use multiple metrics in order to evaluate multiple aspects of the performance. In fact, the nature of the data mining we present here is further complicated because error and imprecision occurs in several places:

1) Ground truth labels. Inaccurate class labels represent a source of error that exists in many datasets. We estimate the amount of error in ground truth activity labels by measuring inter-annotator agreement, or the degree of agreement of the activity labels between multiple annotators. This is typically represented using Cohen's kappa [10].

2) Activity recognition accuracy. In general, activity recognition model is trained using a limited set of training data.

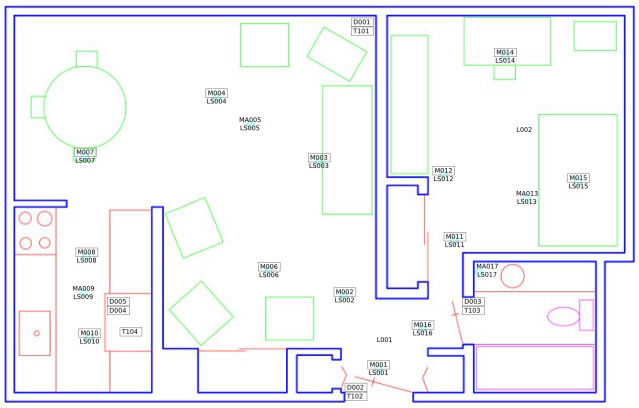

Figure 2: Floor plan of one CASAS smart home testbed. The location of each sensor is indicated with the corresponding motion (M), light (LS), door (D), or temperature (T) sensor number.

The trained model will then be used to generate activity labels for previously-unseen data. The model itself may be subject to error due to representational limitations, small training set, or shortcomings of the employed learning algorithm.

3) Predictor error. In the same way that the activity recognition algorithm will likely experience some error, so also an imperfect activity prediction algorithm will generate erroneous predictions.

Given that there are multiple sources of error, we need to reconsider the standard procedure employed for evaluating predictors. Unlike classification algorithms, where an accuracy of 100% is expected, in this case the expected accuracy will be limited to the quality of the labels. As a result, the evaluation metrics that are discussed here can be κ-*normalized* to reflect the same accuracy range that would be considered for a perfect dataset, while being sensitive to label noise that is known to be present in the data.

In our evaluation of the prediction algorithms, we employ MAE, RMSE, and ETF. MAE and RMSE values are provided in seconds, which are the same units used for the predictions. In order to detect outliers in the errors that may negatively affect the RMSE, we also report ETF values. We vary the ETF threshold from one second up to 24 hours to observe the corresponding distribution of errors for each method. Finally, in the prompting application, where the ground truth labels are provided by the actual participants at the time prompts are delivered, we will also consider κ-*normalization* of the prediction results.

5. EXPERIMENTS AND RESULTS

In this section we empirically investigate our proposed approach on real-world data using several evaluation metrics and compare it with baseline approaches.

5.1 Experimental Setup

Datasets. We evaluate our activity prediction algorithm using sensor and activity data collected from 24 CASAS smart homes[1]. Descriptions of the datasets are provided in Table 1. Each CASAS smart home test bed used in this evaluation includes at least one bedroom, a kitchen, a dining area, and at least one bathroom. While the sizes and layouts of the apartments vary, each home is equipped with combination motion/light sensors on the ceilings, combination door/temperature sensors on cabinets, and external doors. Sensors unobtrusively and continuously collect data while residents perform their normal daily routines. Figure 2 shows a sample layout and sensor placement for one of the smart home test beds.

[1]These datasets are available at http://casas.wsu.edu.

Table 1: Description of CASAS smart home testbed datasets used to evaluate activity predictors.

ID	*Residents*	*Time Span*	*Sensors*	*Sensor Events*
1	1	2 months	36	219,784
2	1	2 months	54	280,318
3	1	2 months	26	112,169
4	1	2 months	66	344,160
5	1	2 months	60	146,395
6	1	2 months	60	201,735
7	2	1 month	54	199,383
8	1	2 months	54	284,677
9	1	2 months	44	399,135
10	1	1 month	38	98,358
11	1	2 months	54	219,477
12	1	4 months	40	468,477
13	1	12 months	58	1,643,113
14	1	1 month	32	133,874
15	1	10 months	40	1,591,442
16	1	2 months	38	386,887
17	1	12 months	32	767,050
18	1	1 month	46	178,493
19	1	1 month	36	92,000
20	1	2 months	40	217,829
21	2	10 months	62	3,361,406
22	1	2 months	56	247,434
23	1	1 month	32	106,836
24	1	2 months	34	216,245

Human annotators label the events in each dataset with corresponding activities based upon interviews with the residents, photographs of the environment, and a sensor map. Each sensor event was labeled with the activity that was determined to be occurring in the home at that time. The datasets contain 118 activity classes in total, but many of them appear infrequently. For our experiments, we have focused on 11 core activities that happen on an average once per day in most of the datasets. These activities consist of many complex functions of daily living and are listed in Table 2. These activities are reflective of the inhabitant's daily health and functioning [29]. Note that the `Leave Home` activity includes times when the resident may open the door to feel connected to the outside world, a frequent behavior among older adults. Sensor events that do not fit into one of the core activity classes are labeled as `Other Activity`, and serve to provide context for the learned predictor. The activity labels on each event are collectively used to determine the ground-truth activity prediction times $\mathbf{y}^*$ at any given time. Multiple annotators label the sensor data and demonstrate interannotator agreement of $\kappa = .85$ for the activities we evaluate in this paper.

Table 2: Activity classes.

Activity	*Sensor Events*
Bathe	208,119
Bed-Toilet Transition	174,047
Cook	2,614,836
Eat	585,377
Enter Home	174,486
Leave Home	311,164
Personal Hygiene	1,916,646
Relax	2,031,609
Sleep	732,785
Wash Dishes	1,139,057
Work	2,028,419

Once ground truth labels are provided for a minimum of one month of sensor data for each dataset, we train the AR activity recognition algorithm [20] to learn a generalized model of the activity classes using data from all of the testbeds as input. AR achieves 96% classification accuracy with 10-fold cross validation on the annotated sensor data for these classes. The AR-provided labels are then used to learn the prediction models. The predictors were trained and tested separately for each dataset.

Activity Prediction Algorithms. We evaluate our **Recurrent Activity Predictor (RAP)** and the **Independent Predictor (IP)** as a informed baseline. For both algorithms, we employ a set of local features Ψ_{local} (listed in Table 7) generated from a variable-length window of recent sensor events. These features are generated from the sensor events directly and provide the regression learner with information about the context of recent sensor events. For RAP, we also generate a set of context features $\Psi_{context}$ (listed in Table 7). These features consist of the prediction from the previous event (lag) for each of the activities $\hat{\mathbf{y}}$ and provide contextual information about the activities to RAP. To account for different time spacing between events, the lag values are adjusted by the time elapsed since the previous event.

In order to determine the best-performance limit for RAP, we also test the **Oracle** recurrent predictor. The oracle predictor employs the same features as RAP, except that the features $\Psi_{context}$ are the true activity times drawn from the labeled data instead of the predicted values. This represents the upper bound of performance enhancement that could be achieved with RAP using the DAGGER algorithm by learning to correct from erroneous lag values.

We also create a second baseline called **Exponential**, which is uninformed. This method does not learn a complex model of activity times. Instead, it models the relative times of each occurrence for each activity as an exponential distribution. The Exponential method then samples from the distribution in order to generate activity predictions.

5.2 Evaluation Procedure

To evaluate the performance of our predictors on these temporal datasets, we employ a *sliding window* validation procedure. This method is similar to k-fold cross-validation, but allows us to maintain the temporal ordering of the sensor event data. We select a window of w=2000 events which we use along with the corresponding ground-truth values as the training examples $\{x_i, y_i^*\}_{i=1}^{N}$. We learn a predictor from this training data and employ it to make predictions for the next 5000 events after the window. The window is then shifted forward by 1000 events and the process is repeated. For exact-imitation training, the lag (context) values are provided using the ground-truth values from the training data, while the predicted values are employed during testing.

Table 3: Overall MAE and RMSE results for the different predictors (in seconds). These values were found by averaging the individual metrics across all the datasets. A one-way ANOVA indicates that the differences in performance are significant ($p < .05$).

Method	*MAE*	*RMSE*
Exponential	13,709.05	27,772.89
IP	19,050.85	173,501.54
RAP	8,433.38	22,337.32
Oracle	2,686.47	12,758.97

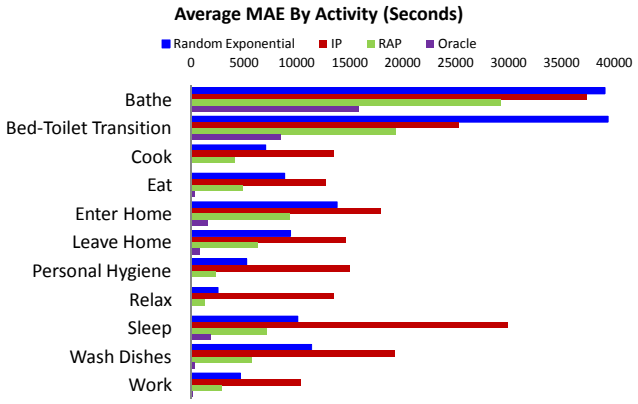

Figure 3: Average MAE for each activity. These values were averaged for each activity across all datasets.

5.3 Results and Analysis

IP vs. RAP. The average MAE and RMSE results for each of the methods are shown in Table 3. IP had an average RMSE of over 150,000 seconds. The RAP method greatly improves on this, reducing the RMSE nearly eightfold to 22,337 seconds. While the improvement over IP is less dramatic for the MAE results, it is still significant. We also note that the RMSE values can be dramatically influenced by the outliers. There are a few datapoints in which the predicted activity time is off by almost a day. RMSE squares each error there by the average performance measure can be biased by these few outliers. We conclude that to examine the overall performance of the predictors, MAE is a better measure.

The average MAE results for each activity are shown in Figure 3. Again, RAP has a lower average error than IP for all activities. In fact, even the Exponential method generally outperforms IP. The graph reflects some of the volatility with the IP learner, which may sometimes generate very erroneous predictions because it does not benefit from having access to context provided by the other activities. For ac-

Table 4: Area under the ETF curve (AUETF) values for each predictor.

Method	*AUETF*
Exponential	0.8673
IP	0.8757
RAP	0.9091
Oracle	0.9972

tivities such as Cook, Eat, Wash Dishes the error for RAP is much lower than that for IP. This may be due to the relationship these activities have with other activities (e.g., cooking, eating, and washing dishes tend to happen sequentially). RAP is able to account for this context through the lag features. RAP also performs well when compared to IP for the Personal Hygiene and Relax activities, which can occur in multiple contexts throughout the day and may not be easily related to information in the sensor events alone. RAP can provide improved performance for these activities by discovering useful relationships in the activity context. Overall, these MAE values indicate that the RAP algorithm is able to provide a significant improvement over the baseline Exponential and IP learners.

Figure 4 shows the ETF values for varying thresholds. The independent predictor has about 5% of its errors below one second. RAP has an improved performance with about 18% of errors less than a second. About 55% of RAP errors are below 15 minutes, compared to about 40% of errors for the independent predictor. Both methods converge to about 99% of errors being below 24 hours. These results indicate that RAP is able to predict more often with smaller error when compared to the independent case, while also having a majority of its predictions be within one hour of the ground-truth time, which is sufficient for many applications.

We also note that the Exponential baseline has less than 3% of its errors below 30 seconds and is outperformed in this regard by both IP and RAP. While it has lower average error than IP in many cases, this is due to the actual activity times being similar to the mean of the random distribution. This allowed it to form many predictions that were, on average, closer to the actual values. However, this overall average hides the fact that the Exponential baseline does not actively adjust its predictions to smaller differences, as demonstrated in the lower ETF at smaller thresholds.

RAP vs. Oracle. We also compare RAP against the Oracle predictor. Overall, the oracle has an average MAE of about 5,750 seconds (about 1.5 hours) lower than RAP (Table 3). Examining the error for each activity in Figure 3, the oracle performs better than both RAP and the independent case for all activities. In fact, for some activities such as Cook, Personal Hygiene, and Relax, the oracle has almost no error. This indicates that by using DAgger, we may be able to improve RAP by learning to recover from errors.

From the ETF plot in Figure 4, it is apparent that the oracle shows significant improvement over the other methods. Over 90% of the errors for the oracle method are less than one second. Thus, the greatest improvement in performance with DAgger may lie with increasing the overall fraction of perfect predictions. While there are still some large outlier errors, nearly all predictions are very accurate.

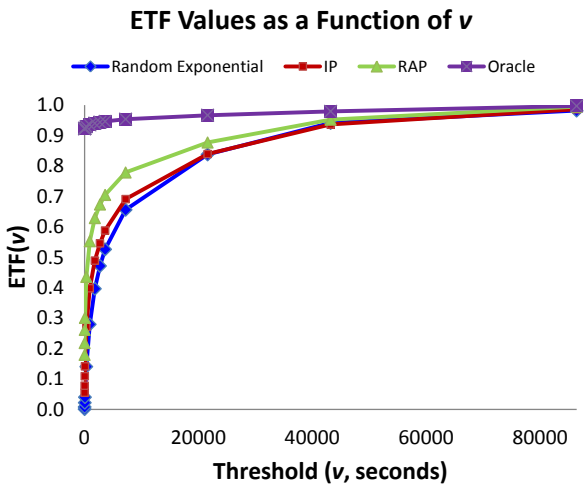

Figure 4: ETF plotted for each predictor. Threshold values range from one second up to one day.

We note the similarity between the ETF curves in Figure 4 and a standard ROC curve. In this case, the discrimination threshold is based on the time-based threshold for prediction error. As with the Area Under a ROC Curve, a perfect predictor will have an Area Under the ETF Curve (AUETF) of 1.0. The AUETF values for our three predictors are given in Table 4. Consistent with the ETF plots, the RAP method outperforms IP and both informed predictors outperform the Exponential method.

Behavior Over Time. It is also of interest to examine how the error for each method changes as we move further from the training window. Figure 5 shows the average MAE at each test horizon against how far that horizon was from the training window. For all methods except the Exponential, the average error is relatively low just after the training window (around 8 minutes for the independent case and 30 seconds for RAP and the oracle). The error generally increases as the test event gets further from the training window. However, the error for IP is much more variable than for RAP or the oracle, sometimes changing by 10000 seconds or more. RAP and the oracle have much smoother curves and produce relatively close errors even at far horizons. These differences indicate the smoothing effect provided by the activity context for RAP, which can utilize the information of lag values to more accurately transition between events and avoid the volatility exhibited by IP. This indicates that RAP is useful in providing more predictable results, while also resulting in overall lower error at increasing test horizons.

For all predictors, the error tends to increase as the event horizon moves away from the training window. However, both RAP and the oracle stabilize after about 2,000 events at error values of about 2.5 and 1 hour, respectively. We suspect that the error rates are partially related to the size of the training window used. While the event frequency is different for each dataset, 5000 events is approximately a day or two in length. At 2000 events, the training window is relatively small compared to the size of the datasets, but this window size was chosen to provide a sufficient number of test windows for computing the evaluation metrics. It is

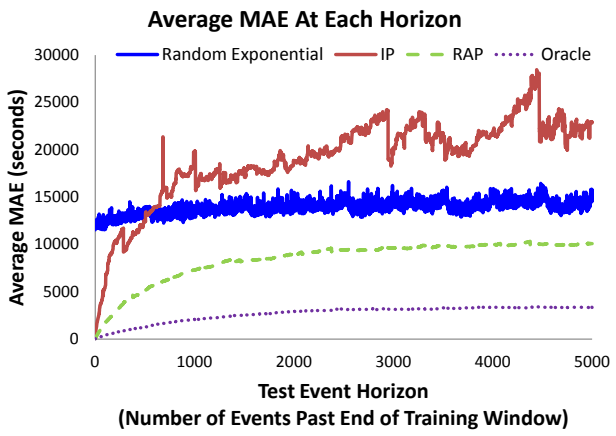

Figure 5: MAE plotted for each predictor against the test horizon. The test horizon indicates how far (in number of events) the test event is from the end of the training window. MAE values are averaged over all activities and datasets at each test horizon.

likely that increasing the training window size (and thus allowing more of the residents' activities to be observed) may reduce the error rates for the predictors. This hypothesis is supported by the results from the prompting app evaluation, shown in the next section. The predictors used for the app were trained with more than a month of data, yet still had error rates below an hour even at more than two weeks beyond the training window.

6. DIGITAL PROMPTING APPLICATION

The ability to predict, or forecast, future occurrences of activities can play a central role in activity prompting. Activity prompting can be used to remind a memory-impaired individual of an activity they typically perform or to encourage integration of a new healthy behavior into a normal routine. Prompting technologies have been shown to increase adherence to medical interventions and increase independence for individuals with cognitive impairment [8, 27].

We evaluated our IP activity predictor in the context of an activity prompting app called CAFE (CASAS Activity Forecasting Environment). Rather than relying on manual setting of reminder times or hand construction of reminder rules [1, 15], CAFE prompts individuals based on the predicted times that the activities will occur. The iOS-based app periodically queries a server for the predicted times of selected activities. An activity recognition algorithm [20] and our activity predictor both reside on the server and generate real-time labels and predictions as sensor data arrive from the smart homes. When the predicted occurrence time is reached, CAFE issues a notification, as shown in Figure 6.

We evaluate CAFE over a period of two weeks for two individuals who were living in smart homes described in Table 5. These homes are instrumented with sensors for motion, temperature, light, and door usage. Sensor data is automatically labeled using the AR activity recognition algorithm. Participant 1's apartment (referred to as "kyoto") houses two residents. Participant 2's apartment (referred to as "navan") houses a single resident. For both apartments, we utilize the generalized AR model that was trained from

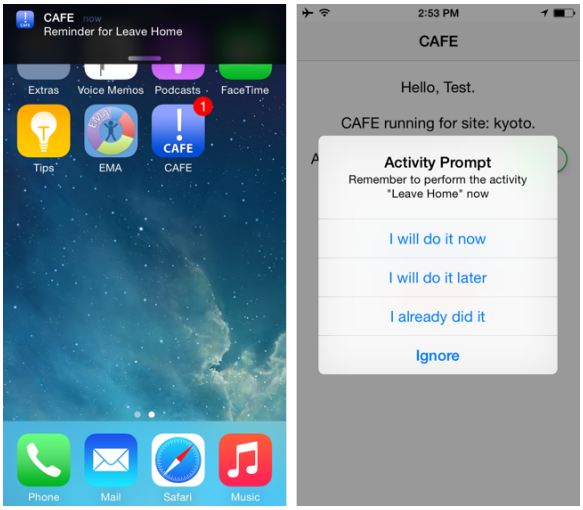

Figure 6: Interface for the EMA and CAFE apps.

Table 5: Description of CAFE testbeds.

Testbed	*Residents*	*Time span*	*Sensors*	*#Events*
kyoto	2	2 weeks	81	147,919
navan	1	2 weeks	28	57,241

the datasets described in Table 1 to generate training data. Neither apartment was part of the training set, so the training labels rely on the generalization power of the learned activity recognizer. The predictor model for kyoto was trained on two months of labeled data from that apartment; and four months of data were used for navan.

The two participants responded to CAFE activity prompts over a period of two weeks. The participants were prompted for seven activities: Bathe, Cook, Eat, Leave Home, Relax, Sleep, and Work. The participants provided a total of 112 responses, which were evenly distributed between "I will do it now", "I already did it", and "I will do it later". We note that delays may occur between the activity occurring and the prompt being generated. This is partly due to the fact that the database is updated every 15 minutes, after which AR provides labels and the prompts are generated. Once the prompt is generated, notification is scheduled for delivery but can be delayed due to the iOS behavior for obtaining updates from the server. As a result, the participants observed that occasionally they would receive a prompt to start an activity while they are in fact currently performing the activity. Therefore, they respond with "I already did it" or "I will do it later".

Given the nature of the current notification generation, we also evaluated the prompt timings based on MAE and range-normalized MAE, as summarized in Table 6. Each activity occurred at least once a day and MAE values were normalized based on a maximum error of 43,200 seconds, or half of a day. As shown in Table 6, the average MAE value is 2,925 seconds (about 48 minutes). The average prediction error is approximately 15 minutes longer than the infrastructure-created delays on average. To further assess the error we calculate the ETF value using 30 minutes, the maximum infrastructure-initiated delay, as our threshold value.

Because all of the sensor data is labeled by an activity

Table 6: Evaluation of CAFE prompts (in seconds).

MAE	*Normalized MAE*	*ETF*	*κ-Normalized ETF*
2,925	0.07	0.64	0.72

recognition algorithm, we also analyzed participant responsiveness to the prompt. During this pilot study, we observed that each time the participants responded "I will do it now", they did initiate the activity within the next 20 minutes.

Finally, we obtained ground truth activity labels during the same two-week period that participants received activity prompts. This was accomplished through a separate **EMA** app. EMA here stands for *ecological momentary assessment*. This is an established method of obtaining participant information "in the moment", when the information is likely to be most accurate [13]. Using the EMA app, participants are queried every 15 minutes about the activity they are currently performing. The responses are stored in the database and used to validate our activity recognition algorithms. From the collected responses, we report AR accuracy of 92% for these seven activities. We use this information to generate the κ-normalized values summarized in Table 6.

Interestingly, the participants noted that the app sometimes actually created a modification in their behavior. One resident pointed out that he was debating between leaving home to get groceries or watching television. Upon receiving the CAFE prompt, he left immediately to perform his errands. On another occasion, a participant started working earlier than originally planned due to the prompt notification. Integrating activity prompts into daily behavioral routines thus raises interesting challenges for intervention design that need to be carefully considered in future work.

7. RELATED WORK

Activity recognition algorithms have been investigated over the last decade for a plethora of sensor platforms, including ambient sensors, wearable sensors, phone sensors, and audio/video data [2, 3, 16, 24, 32]. Some existing work also addresses complex scenarios including real-time recognition and recognition with multiple residents [28, 31]. These algorithms map a sequence of sensor readings onto an activity class value. They can be used to track occurrences of well-known activities or partnered with activity discovery algorithms to model all of a person's routine behaviors [4]. A number of data mining approaches to this problem have been tested including generative, discriminative, and ensemble methods.

While activity prediction is not as heavily investigated as activity modeling or recognition, there are some representative first efforts in this area. Most of these techniques focus on sequence prediction to generate a label for the activity that will occur next. This work includes the Active LeZi algorithm by Gopalratnam and Cook [9] to predict the next event generated by sensors in an instrumented home. Other researchers [12, 18, 19] have investigated the use of probabilistic graphical models for sequential prediction in video data. In the work by Koppula and Saxena [19], the anticipated event was supplied to robots in order to provide better assistance. The authors report prediction F1 scores of 37.9 for 10 activities monitored in a scripted environment.

On the other hand, automated prompting systems have been developed and studied for some time. Most of these systems are rule-driven or require knowledge of a user's daily schedule [1, 15]. While these systems are able to adjust prompts based on user activities, they also require input of a user's daily schedule or predefined activity steps. In contrast, the methods we describe in this paper are data-driven. They utilize activity-labeled sensor events to learn an individual's normal routine and generate predictions based solely upon this data.

8. SUMMARY AND FUTURE WORK

We studied a data-driven approach for predicting future occurrences of activities from sensor data. We showed how powerful regression learners can be leveraged to learn an effective activity predictor that can reason about relational and temporal structure among the activities in an efficient manner. Our extensive experiments on twenty-four smart home datasets validate that our recurrent activity predictor is not only effective at forecasting upcoming activity occurrences. Additionally, we illustrated the use of the predictor as part of our CAFE activity prompting app.

In the future, we will enhance our approach by incorporating iterative prediction refinement as well as smoothing, and perform a large-scale user study. In this paper, we limited our evaluation to consider only activity initiation, but an exciting direction will be to predict the length of the activity as well as individual activity steps. We will study the theoretical properties of our approach and apply it to more general prediction problems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants 0900781 and 1262814 and by the National Institute of Biomedical Imaging and Bioengineering under Grant R01EB015853.

9. REFERENCES

[1] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *International Joint Conference on Artificial Intelligence*, pages 1293–1299, 2005.

[2] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46:107–140, 2015.

[3] D. J. Cook. Learning setting-generalized activity models for smart spaces. *IEEE Intelligent Systems*, 27(1):32–38, 2012.

[4] D. J. Cook, N. Krishnan, and P. Rashidi. Activity discovery and activity recognition: A new partnership. *IEEE Transactions on Systems, Man, and Cybernstics, Part B*, 43(3):820–828, 2013.

[5] J. R. Doppa, A. Fern, and P. Tadepalli. HC-Search: A learning framework for search-based structured prediction. *JAIR*, 50:369–407, 2014.

[6] J. R. Doppa, A. Fern, and P. Tadepalli. Structured prediction via output space search. *JMLR*, 15:1317–1350, 2014.

[7] J. R. Doppa, J. Yu, C. Ma, A. Fern, and P. Tadepalli. HC-Search for Multi-Label Prediction: An Empirical Study. In *AAAI*, 2014.

[8] N. Epstein, M. G. Willis, C. K. Conners, and D. E. Johnson. Use of technological prompting device to aid a student with attention deficit hyperactivity disorder to initiate and complete daily activities: An exploratory study. *Journal of Special Education Technology*, 16:19–28, 2001.

[9] K. Gopalratnam and D. J. Cook. Online sequential prediction via incremental parsing: The active lezi algorithm. *IEEE Intelligent Systems*, 22:52–58, 2007.

[10] K. L. Gwet. *Handbook of Inter-Rater Reliability*. Advanced Analytics, LLC, 2014.

[11] Hal Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *MLJ*, 75(3):297–325, 2009.

[12] K. P. Hawkins, N. Vo, S. Bansal, and A. Bobick. Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In *IEEE-RAS International Conference on Humanoid Robots*, pages 499–506, 2013.

[13] K. E. Heron and J. M. Smyth. Ecological momentary interventions: Incorporating mobile technology into psychosocial and health behavior treatment. *Journal of Health Psychology*, 15:1–39, 2010.

[14] M. Kääriäinen. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

[15] P. Kaushik, S. S. Intille, and K. Larson. User-adaptive reminders for home-based medical tasks: A case study. *Methods of Information in Medicine*, 47:203–207, 2008.

[16] S. Ke, H. Thuc, Y. Lee, J. Hwang, J. Yoo, and K. Choi. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013.

[17] R. Khardon. Learning to take actions. *MLJ*, 35(1):57–90, 1999.

[18] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Proceedings of the European Conference on Computer Vision*, 2012.

[19] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Robotics: Sciences and Systems*, 2013.

[20] N. Krishnan and D. J. Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 10:138–154, 2014.

[21] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[22] M. Lam, J. R. Doppa, S. Todorovic, and T. Dietterich. Learning to detect basal tubules of nematocysts in sem images. In *ICCV Workshop on Computer Vision for Accelerated Biosciences*, 2013.

[23] M. Lam, J. R. Doppa, S. Todorovic, and T. Dietterich. HC-Search for structured prediction in computer vision. In *CVPR*, 2015.

[24] O. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communication Survey Tutorials*, 15:1195–1209, 2013.

[25] C. Ma, J. R. Doppa, W. Orr, P. Mannem, X. Fern, T. Dietterich, and P. Tadepalli. Prune-and-Score: Learning for greedy coreference resolution. In *EMNLP*, 2014.

[26] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[27] M. Schmitter-Edgecome, S. Pavawalla, J. T. Howard, L. Howell, and A. Rueda. *Dyadic interventions for Persons with Early-Stage Dementia: A Cognitive Rehabilitative Focus*, chapter 3, pages 39–56. Nova Science Publishers, 2009.

[28] D. Stowell and M. D. Plumbley. Segregating event streams and noise with a Markov renewal process model. *Journal of Machine Learning Research*, 14:2213–2238, 2013.

[29] V. G. Wadley, O. Okonkwo, M. Crowe, and L. A. Ross-Meadows. Mild cognitive impairment and everyday function: Evidence of reduced speed in performing instrumental activities of daily living. *The American Journal of Geriatric Psychiatry*, 15:416–424, 2008.

[30] J. Xie, C. Ma, J. R. Doppa, P. Mannem, X. Fern, T. Dietterich, and P. Tadepalli. Learning greedy policies for the easy-first framework. In *AAAI*, 2015.

[31] J. Ye, G. Stevenson, and S. Dobson. KCAR: A knowledge-driven appraoch for concurrent activity recognition. *Pervasive and Mobile Computing*, 19:47–70, 2015.

[32] Y. Zheng, W.-K. Wong, X. Guan, and S. Trost. Physical activity recognition from accelerometer data using a multi-scale ensemble method. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference*, pages 1575–1581, 2013.

APPENDIX

Table 7: Activity prediction features.

Feature	*Description*
lastSensorEventHours*	Hour of day for current event
lastSensorEventSeconds*	Seconds since the beginning of the day for the current event
windowDuration*	Window duration (sec)
timeSinceLastSensorEvent*	Seconds since previous event
prevDominantSensor1*	Most frequent sensor in the previous window
prevDominantSensor2*	Most frequent sensor in the window before that
lastSensorID*	Current event sensor
lastLocation*	Most recent location sensor
sensorCount**	Number of events in the window for each sensor
sensorElTime**	Time since each sensor fired
timeStamp*	Normalized time since beginning of the day
laggedTimestamp*	Previous event timeStamps
laggedPredictions***	Previous event predictions
maximumValue#	Maximum value of sensor
minimumValue #	Minimum value of sensor
sum#	Sum of sensor values
mean#	Mean of sensor values
meanAbsoluteDeviation#	Average difference from mean
medianAbsoluteDeviation#	Avg. difference from median
standardDeviation#	Value standard deviation
coeffVariation#	Coefficient of value variation
numZeroCrossings#	Number of median crossings
percentiles#	Number below which a percentage of values fall
sqSumPercentile#	Sq. sum values < percentile
interQuartileRange#	Difference between 25th and 75th percentiles
binCount#	Values binned into 10 bins
skewness#	Symmetry of values
kurtosis#	Measure of value "peakedness"
signalEnergy#	Sum of squares of values
logSignalEnergy#	Sum of logs of squares
signalPower#	SignalEnergy average
peakToPeak#	Maximum - minimum
avgTimeBetweenPeaks#	Time between local maxima
numPeaks#	Number of peaks

*Used for IP and RAP experiments. **Used for IP and RAP, one sensorCount and one sensorElTime for each sensor used. ***Used for RAP, one per activity. #Based on window of recent values for each sensor.