# Toward Learning and Mining from Uncertain Time-Series Data for Activity Prediction

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

Prediction of human activities is important in a wide variety of fields. However, developing activity predictors that work well with the uncertainties surrounding activity data is a challenging task. In this paper, we present activity prediction algorithms that are designed to form accurate predictions in this environment by analyzing the task in the context of time-series mining techniques. We explore factors that contribute to activity data uncertainty and discuss metrics that can be used to evaluate prediction performance. We evaluate our methods using activity data from 24 physical smart home testbeds and demonstrate that they can achieve reasonable accuracy. We also describe the results of a pilot study using our predictors in an activity prompting app and explore how data variations affect user experience.

## 1. INTRODUCTION

Predicting human activities is an important aspect of many fields of study. In order to optimize the resource usage and improve interaction with users, it is important to understand when they will perform certain activities. Often, it is important to predict not only that an activity will occur, but also the actual time when it will occur. These times for different activities at regular intervals can be seen as a multi-variate time-series data, and we are interested in predicting activity times at any given moment. This allows us to leverage the large body of work on learning and mining time-series data to build efficient and accurate activity predictors.

A major concern for activity prediction is the availability of information about users' activities for learning and mining patterns. In recent years, the growing number of ubiquitous sensors has increased the ability to collect data about human behavior. Sensors can be deployed in a variety of environments in order to observe users' actions over increasingly long time-periods. This data contains rich information about the observed behaviors that can be useful for activity prediction.

While the raw sensor data is becoming easily available, it can be difficult and time-consuming to annotate large amounts of raw data to obtain ground-truth labels to facilitate learning. It is often challenging to collect this information from users while they are performing their activities. Consequentially, the activity labels are often provided after the fact by trained human annotators or automatic activity recognizers. These activity labels can be imprecise at best, and rely on the annotator's interpretation of sensor events to be accurate. Similarly, the raw sensor data can be erroneous in some cases. Therefore, the uncertainty in the training data due to possible errors and noise makes the learning problem very challenging.

In this paper, we describe activity prediction algorithms and consider the impact of label uncertainty on their performance. These algorithms extract useful information about activities from sensor data and use it to inform simple regression learners in order to form the predictions. We consider the use of these predictors for the prediction of activities in smart homes. Smart home data faces many of the uncertainties noted above, allowing us to examine the impact of data uncertainty on the prediction performance. We want to understand how discrepancies in raw sensor data and activity labels affect the performance of the learned predictors. As an end goal, we hope to use this understanding to build predictors that are more robust to many forms of uncertainty.

In order to facilitate this analysis, it is important to be able to quantitatively measure the performance of a given predictor. To that end, we examine a number of evaluation metrics and discuss how they may be affected by the uncertainty in the data. We also consider the effect of data uncertainty in the use of a prototype prompting application. This allows us to gather user feedback and learn about issues that may not be apparent from evaluation metrics alone.

## 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 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 $\lambda_i$ 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 $\Lambda$ to the corresponding activity label, $a_\Lambda$. 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* $\Phi$ that computes a $d$-dimensional feature vector $\Phi(\lambda_i) \in \Re^d$ for any sensor event $\lambda_i$ using the context of recent sensor events

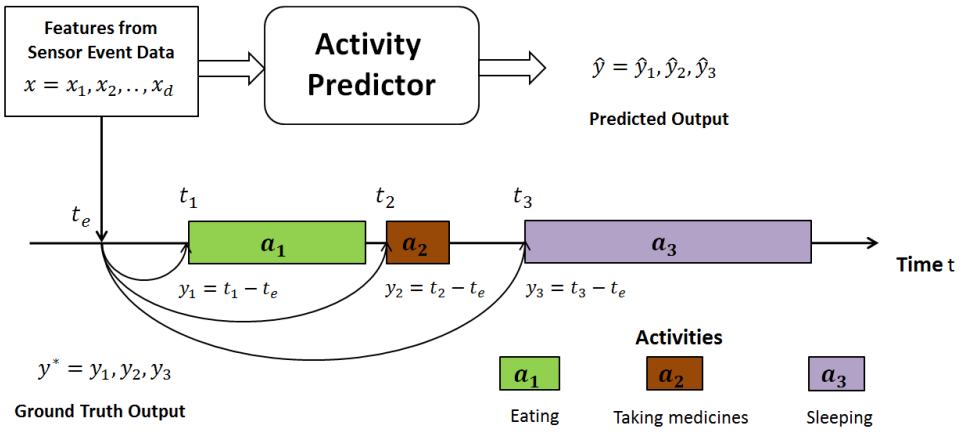


**Figure 1: A high-level overview of the activity prediction problem. Given features $\mathbf{x} \in \Re^d$ extracted from the sensor event data at time $t_e$ as input, the activity predictor needs to predict the relative occurrence time of each activity. In this example, we have three activities: $a_1$ (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 $a_i$.**

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. CHALLENGES IN LEARNING FROM UNCERTAIN DATA

The activity prediction problem is faced with a number of challenges resulting from uncertainty in the data. Uncertainty due to sensor noise and variability is an issue faced in many time series domains. However, in the activity prediction application, we can also face uncertainty in the data labels themselves. In order to properly train our activity predictors, our training data labels should accurately describe which activities were occurring when each sensor event occurred. If the activity labels are incorrect, the training data may indicate that activities occurred during the wrong sensor events. This can lead to learning an improper prediction model, which may have difficulty performing accurate predictions relative to the actual activity times.

Much of the uncertainty in activity labeling is due to the difficulty of obtaining such labels. It is often costly or impossible to query users directly about their activities. Even when labels can be obtained directly from the user, they are likely to be noisy due to known errors associated with self-report. One popular method of querying users is through a smart-phone app, where the user is asked periodically what activity they are performing. This method can capture relevant activity information from the user himself. However, users may sometimes forget to respond to the app or may respond incorrectly due to a time delay. This can lead to lowered accuracy of gathered activity information. Another concern with directly querying the user may be that such querying can interrupt a user's normal routine and habits, leading to irregularities in the sensor data and making the predictor learning more difficult.

In most cases, we must determine activity labels for sensor data after it has been collected and without direct user input. This can be performed by a human annotator or an activity recognition algorithm such as AR. Both of these methods involve indirect observation of activities, which can lead to greater uncertainty. In the case of a human annotator, the annotator must interpret which activities are occurring based on the raw sensor data. While visualization tools can aid in this task [27], there are likely to be some errors in the generated labels. This is especially the case when two activities have similar sensor patterns: the annotator may not be able to distinguish between the two and thus apply an incorrect activity label. When an activity recognition method is used, the quality of the labels will depend on the quality of the learned recognition model. If the model was trained on erroneous data or data that is dissimilar to that being labeled, the new activity labels are more likely to be erroneous as well.

The nature of the activities themselves may also lead to inaccuracies in labeling. Labels are usually tied to sensor events, with distinct start and end times. However, activities are not necessarily tied to the timing of specific sensor events. For example, some activities may occur during gaps between sensor events, such that they cannot be related to any events. Activity start and end times may also be unclear, with the transition between different activities occurring over multiple events. This is further complicated by occurrences where the user may be performing multiple activities simultaneously or in an interwoven fashion.

All the above factors introduce significant amount of uncertainty to the training data for learning activity predictors. Therefore, activity prediction algorithms must be designed to take all these different uncertainties into account. Predictors that can adapt to labeling inaccuracies are likely to be more robust when available sensor data may be inaccurate.

# 4. ACTIVITY PREDICTORS

In this section we describe our activity prediction algorithms, and observe how they can be affected by and adapt for uncertainties in the sensor data.

## 4.1 Independent Predictor

Our basic activity predictor, called the *Independent Predictor*, predicts each activity independently. For each sensor event data $\lambda_i$ in the training sequence $\Lambda$, we extract the features $\mathbf{x}_i = \Phi(\lambda_i) \in \Re^d$ listed in Table 8 (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.

This approach provides a basic predictor with low test-time complexity which is able to form real-time predictions. However, since it treats each event independently, it is subject to large fluctuations of the predicted output depending on the features generated at each event. This can be exacerbated even further by errors in the data labeling.

## 4.2 Recurrent Activity Predictor

In order to address this weakness, we consider joint models which are able to use the relationships between different activities to make the predictions more robust. The *recurrent predictor* employs both the local features $\Psi_{local}(i) = \Phi(\lambda_i)$ computed from the recent sensor event window (see Table 8) and context features $\Psi_{context}(i)$ that provide the context of recent history of activity predictions. The joint features $\Psi_{context}(i)$ consist of the predicted activity times (lags) $\hat{y} \in \Re^T$ for all $T$ activities for the last $H$ events in the history window. The joint feature vector will be of size $H \cdot T$.

In order to learn a recurrent activity predictor model, we employ the framework of imitation learning. Imitation learning allows us to train a learner to imitate the behavior of an expert when performing a sequential decision-making task. It has been successfully applied to a variety of structured prediction tasks in natural language processing and computer vision [11, 6, 5, 23, 29, 20, 21]. 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).

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 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 [16, 24].

## 4.3 Multi-Output Regression Learner

We decompose the multi-output regression learner used in our activity predictors by learning a separate regression function for each activity. That is, we learn $T$ regressors for making our predictions. This approach is inspired by the Binary Relevance classifier for multi-label classification [7]. Since activity relationships are often complex, a simple linear function does not provide sufficient prediction capability. Hence, we employ a variant of regression trees called model trees as our regression function. These trees learn a linear function at each leaf node.

**Algorithm 1** RAP Learning via Exact Imitation

**Input**: $\Lambda$ = 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
```

# 5. EVALUATION METRICS

In this section, we will discuss the challenges of evaluating activity prediction problems and present several evaluation metrics which take these factors into account. Given the uncertain data used in the activity prediction problem, it is important to carefully consider how activity predictors are evaluated. The quality of the labeled data will affect the activity prediction performance. Thus, it is important to utilize a variety of evaluation metrics that enable understanding of the interplay between observed performance and underlying data concerns.

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

**Evaluation Metrics.** We describe 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 mea-

sures provide the average error in real units and quantify the overall error rate, with a value of zero indicating a perfect predictor. Because RMSE squares each term, it effectively weights large errors more heavily than small ones.

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

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

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.

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

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.

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

Normalized metrics can be beneficial when considering predictions from uncertain data, as the normalizing factor will also be based on the same data. Thus, the normalized metric can provide information about the magnitude of the error relative to the uncertain data. However, it can be difficult to determine the error magnitude from a normalized metric.

An important factor in performance analysis, especially with uncertain data, is understanding the distribution of the error results. 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.

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

When considering uncertain data, we can also analyze the inconsistency in the data labels and use this to adjust the observed metrics. One common method of doing this is to observe interannotator agreement when using multiple annotators. For example, we can compare the agreement between two human annotators, or between a user's logged activities and an activity recognition algorithm. This is typically represented using Cohen's kappa [10], which can then be used to reduce the measured error to reflect the what could be expected from a perfect dataset. However, the kappa is typically only used with accuracy-type measurements. Thus, it could be used to adjust a metric such as the ETF, but it is not well-defined for error measurements such as MAE.

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 $\kappa$-*normalization* of the prediction results.

# 6. EXPERIMENTS

In this section we empirically investigate our activity predictors with real-world data using several evaluation metrics and compare them with baseline approaches.

## 6.1 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.

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

[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 |

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 [28]. 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 activity prediction times $\mathbf{y}^*$ associated with each event. Multiple annotators label the sensor data and demonstrate interannotator agreement of $\kappa = .85$ for the activities we evaluate in this paper.

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 [19] 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 RAP, the context features consist of the predictions for the previous event ($H = 1$). The predictions are adjusted by the time since the previous event to account for different time spacing between events.

**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 |

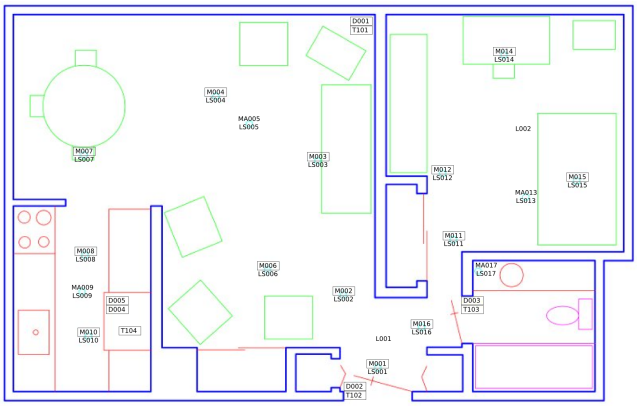


**Figure 2: Floor plan of one CASAS smart home testbed. The location of each sensor is indicated with the corresponding motion (M), light (LS), door (D), or temperature (T) sensor number.**

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 [24] by learning to correct from erroneous lag values. DAGGER is an iterative algorithm which generates a sequence of predictors that progressively learn to correct from training errors made by previous predictors. The best-performing predictor is then chosen as the final predictor, which should be most capable of correcting from erroneous predictions. Thus, Oracle represents the performance that could occur if RAP is trained to perfectly 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.

## 6.2 Evaluation Procedure

To evaluate the performance of our predictors on these temporal datasets, we employ a *sliding window* validation

**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 |

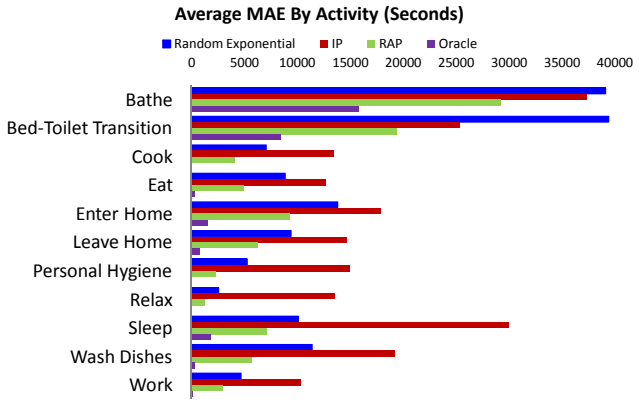


**Figure 3: Average MAE for each activity. These values were averaged for each activity across all datasets.**

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.

## 6.3 Results and Analysis

**Average Errors** The average MAE and RMSE results for each of the methods are shown in Table 3. The addition of context features for RAP allows it to greatly improve compared to IP, with nearly and eightfold reduction of RMSE. The MAE also improves, lowering by over 10,500 seconds (about 3 hours). We note that the predictions for IP tend to be highly variable, as shown in Figure 4. This is largely due to the fact that it only uses the local features. RAP's predictions are more stable, with the context features allowing it to take previous predictions into account. This indicates that RAP is more robust against individual event fluctuations, such as those caused by label errors.

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. This graph also indicates some of the

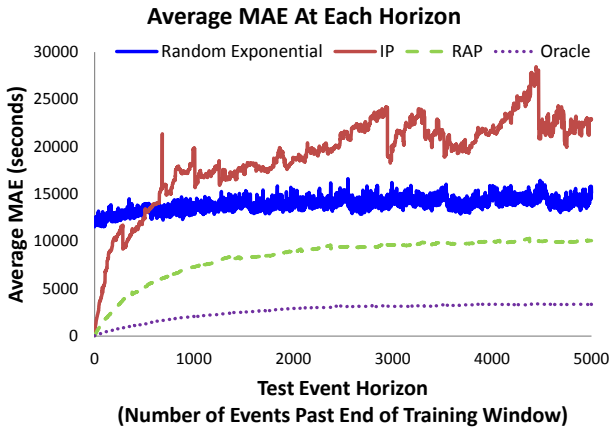


**Figure 4: MAE plotted for each predictor against the test horizon. The test horizon indicates how far (in number of events) the test event is from the end of the training window. MAE values are averaged over all activities and datasets at each test horizon.**

volatility of the IP method. For activities 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. These MAE values indicate that RAP can provide significant improvement over the baseline Exponential and IP learners.

The Oracle predictor improves even further upon the performance of RAP. It achieves an average MAE of about 1.5 hours lower, and performs better than the other predictors for all activities (Figure 3). In fact, for some activities, such as Cook, Personal Hygiene, and Relax, Oracle has almost no error. This indicates that by using DAGGER, we may be able to improve RAP by learning to recover from errors.

**ETF and Uncertainty** Figure 5 shows the ETF values for varying thresholds. The Exponential predictor has less than 1% of its errors below one second, reflecting the fact that it simply returns "average" errors, in contrast to the more active predictions of the other methods. 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. This performance is improved upon further by Oracle, which achieves over 90% of errors 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

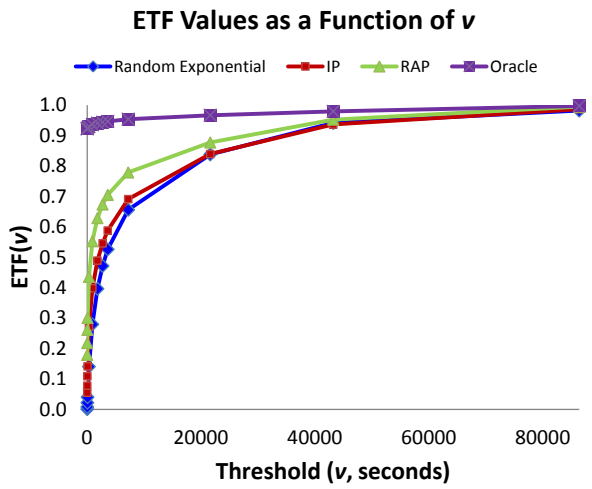


**Figure 5: ETF plotted for each predictor. Threshold values range from one second up to one day.**

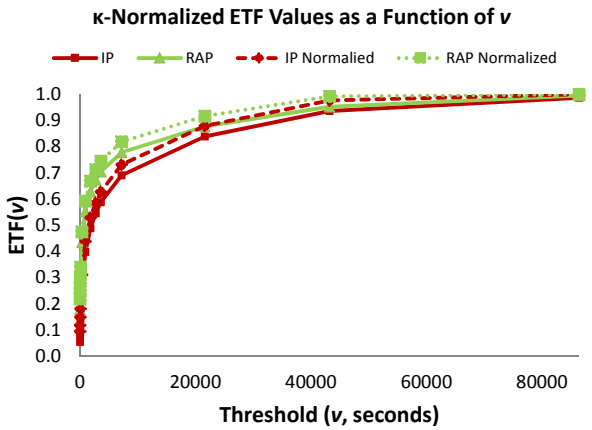


**Figure 6: Original and normalized ETF plotted for IP and RAP. Threshold values range from one secondup to one day.**

still some large outlier errors, nearly all predictions are very accurate.

We note the similarity between the ETF curves in Figure 5 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.

The ETF values share some similarities with classification accuracy values. At a given threshold $v$, the ETF equates to an accuracy measure if a "correct" prediction is considered one with error below the threshold. Using this interpretation, we can use Cohen's Kappa from the accuracy of AR labels to normalize ETF values. Since the accuracy of AR on the labeled data is 96%, we can generate $\kappa$-normalized

**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 |

**Table 5: Correlation of AR labeling classification metrics with RAP MAE performance.**

| Classification Metric | Correlation |
|---|---|
| Activity Accuracy | 0.289 |
| Sensitivity | -0.323 |
| Specificity | 0.319 |
| G-Mean | -0.197 |
| Precision | 0.063 |
| Recall | -0.323 |
| F1 Measure | -0.077 |

ETF plots, as shown in Figure 6. With the normalization factor taken into account, nearly 100% of the errors for both methods are below one day.

We are correlate AR's classification performance with the RAP MAE results to understand how label accuracy affects prediction. AR labels were compared against human labels for the datasets in Table 1. The resulting correlations, shown in Table 5, indicate that AR classification performance does not necessarily correlate with performance of the RAP algorithm. For example, RAP MAE tends to increase with increased AR accuracy, which seems counterintuitive. However, the error rate decreases when AR has higher sensitivity and recall. Further examination of the individual activity results indicates that the correlations do not wholly describe these relationships.

These results highlight the difficulty in analyzing the activity prediction problem. There are a number of factors that can influence performance, and relating this to uncertainty in the underlying data can be challenging. It appears that simply having additional training points can improve prediction performance, though even this is not always consistent. The combination of data uncertainty and complex activity relationships makes activity prediction a complicated process. We hope to further explore these complications in our future work.

## 7. PROMPTING APPLICATION

One important use of an activity predictor is generating predictions for an activity prompting application. 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, 25].

A prompting application may be affected by many of the same data uncertainty issues as in the overall prediction

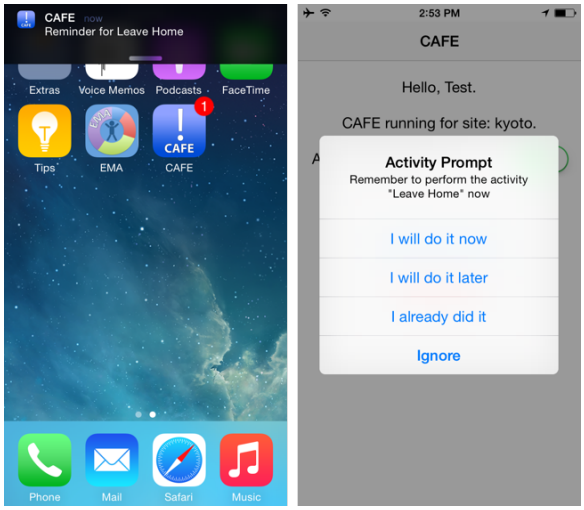


**Figure 7: Interface for the EMA and CAFE apps.**

**Table 6: Description of CAFE testbeds.**

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

problem. For instance, if an activity is mis-labeled, the learned predictor could predict its occurrence at the wrong time of day. This could lead the prompting application to prompt the user to perform the activity at the wrong time, potentially confusing or frustrating the user. We are interested in better understanding the effects of data uncertainty on prompt accuracy and users' experience.

To that end, we have evaluated an activity prompting app called CAFE (CASAS Activity Forecasting Environment). CAFE uses predicted activity times to determine when to generate activity prompts, rather than manual scheduling or rule generation [1, 14]. The app runs on an iOS mobile device and periodically queries a server for the predicted times of selected activities. These predictions are provided by our IP algorithm and are generated every 15 minutes for new sensor events arriving from smart homes. When the predicted occurrence time is reached, CAFE issues a notification, as shown in Figure 7.

We evaluate CAFE over a period of two weeks for two individuals who were living in smart homes described in Table 6. These homes are instrumented with sensors for motion, temperature, light, and door usage. Participant 1's apartment (referred to as "kyoto") houses two residents. Participant 2's apartment (referred to as "navan") houses a single resident. We utilize the generalized AR model that was trained from 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".

**Table 7: Evaluation of CAFE prompts. MAE value is in seconds. Normalized MAE is normalized 43,200 seconds. ETF values are for a threshold of 30 minutes.**

| *MAE* | *Normalized MAE* | *ETF* | *$\kappa$-Normalized ETF* |
|---|---|---|---|
| 2,925 | 0.07 | 0.64 | 0.72 |

In addition to the activity predictor, we also ran the AR algorithm to provide activity labels for the observed events. The AR model was the same as the one used to generate the training data described in the previous paragraph. Using these labels, we can compute the error of each activity prompt relative to the labeled data, as shown in Table 7. Since each activity usually occurred at least once per day, we normalize the MAE by a maximum error of 43,200 seconds, or half of a day. The MAE is approximately 48 minutes, which is about 15 minutes longer than the infrastructure delay in pushing new predictions to the app.

We also compute the ETF value for a threshold of 30 minutes, which is the approximate infrastructure-initiated delay. This ETF value, shown in Table 7, indicates that 64% of errors are below the threshold. In order to observe the effect of mislabeling by the AR algorithm, we also collected activity information directly from the participants using a separate mobile app. This app, called *Ecological Momentary Assessment* (EMA), uses an established method of obtaining participant information "in the moment", when it is likely to be most accurate [13]. The app would query the participants every 15 minutes about the activity they were currently performing and their responses were stored in a database.

Using these responses, we report AR accuracy of 92% for these seven activities. We can find the $\kappa$-normalized ETF value accordingly, as shown in Table 7. This value indicates that over 70% of the errors would be expected to be below 30 minutes if the activity labels were more accurate. It also indicates that about 8% of the prediction error may be due to errors in the activity labels. Thus, it is important to keep the inaccuracies in the activity labels in mind when assessing a prompting application.

Because all of the sensor data is labeled by an activity 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. 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.

# 8. 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, 15, 22, 31]. Some existing work also addresses complex scenarios including real-time recognition and recognition with multiple residents [26, 30]. 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 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, 17, 18] have investigated the use of probabilistic graphical models for sequential prediction in video data. In the work by Koppula and Saxena [18], 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, 14]. 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.

# 9. CONCLUSION

We studied the prediction of human activities from sensor data in the context of time series analysis. We described how difficulties in acquiring activity labels can lead to uncertainties in prediction data. We showed that regression learners can be designed to address these uncertainties to form effective activity predictors that are robust to data variability. Our experiments with twenty-four smart home datasets validate the effectiveness of these approaches and allowed us to explore evaluation metrics for uncertain data. The considerations regarding use of our predictors in the context of the CAFE prompting app were also discussed.

While the methods and evaluation presented here provide a useful basis for activity prediction in uncertain datasets, there are many aspects of this problem to be addressed. One particular concern lies in determining how to normalize error metrics based on annotator reliability. While this has been explored for accuracy-like values, the application of these methods to error measurements remains an open question. Also of interest is determining how to make activity prediction methods more robust to errors in training data. This will rely on enabling prediction methods to adapt to better correct for variations in users' habits and activity times. It is also important to develop new methods for obtaining activity labels to reduce the uncertainty in the data. We intent to continue exploring these questions and finding new methods to make activity prediction from unreliable data more accurate and robust.

## 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.

# 10. 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] 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.

[15] 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.

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

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

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

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

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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.

[27] B. L. Thomas and A. S. Crandall. A demonstration of PyViz, a flexible smart home visualization tool. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 304–306, 2011.

[28] 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.

[29] 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.

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

[31] 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 8: 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.