Human Activity Prediction in Smart Home Environments with LSTM Neural Networks

Conference Paper · June 2018

DOI: 10.1109/IE.2018.00014

1 author:

Niek Tax
Booking.com
46 PUBLICATIONS **469** CITATIONS

SEE PROFILE

Human Activity Prediction in Smart Home Environments with LSTM Neural Networks

Niek Tax
Architecture of Information Systems Group
Eindhoven University of Technology, Eindhoven, the Netherlands
Email: n.tax@tue.nl

***Abstract*—In this paper, we investigate the performance of several sequence prediction techniques on the prediction of future events of human behavior in a smart home, as well as the timestamps of those next events. Prediction techniques in smart home environments have several use cases, such as the real-time identification of abnormal behavior, identifying coachable moments for e-coaching, and a plethora of applications in the area of home automation. We give an overview of several sequence prediction techniques, including techniques that originate from the areas of data mining, process mining, and data compression, and we evaluate the predictive accuracy of those techniques on a collection of publicly available real-life datasets from the smart home environments domain. This contrast our work with existing work on prediction in smart homes, which often evaluate their techniques on a single smart home instead of a larger collection of logs. We found that LSTM neural networks outperform the other prediction methods on the task of predicting the next activity as well as on the task of predicting the timestamp of the next event. However, surprisingly, we found that it is very dependent on the dataset which technique works best for the task of predicting a window of multiple next activities.**

I. Introduction

In recent years, technological advancements in the areas of sensor and communication technology have led to a rapid proliferation of smart home environments and ambient assisted living (AAL) systems. Smart home environments are envisioned to exhibit various forms of intelligence with the aim of home automation for increased comfort, quality of life, and safety. Smart home environments consists of sensors that log the behavior that they observe. Table I shows an example of a log generated by smart home sensors, containing sensor events from a real-life smart home experiment performed by MIT. An important aspect of smart home environments is to be able to accurately predict future behavior of the smart home inhabitant based on historical data. Logs from smart home environments enable the training such prediction models.

Activity prediction in smart home environments has several important use cases. For example, a common problem with people suffering from memory diseases (such as Alzheimer's disease) is that they forget to eat. An accurate prediction that a person is unlikely to be going to eat in the remainder of the day could be used in a system that automatically sends a message to relatives or caretakers who could then intervene and prevent that the person misses a meal. A second use case of activity predictions can be found in the area of e-coaching, where identification of *coachable moments*, i.e., the moments in time at which the subject is susceptible to coaching messages, forms an important problem. Activity predictions could help to identify such coachable moment. For example, when an inhabitant is predicted to soon be taking a shower, he is currently unlikely to read a coaching message that is being sent, and therefore a coaching system instead should wait for a more suitable moment. Finally, activity predictions have an application in the area of home automation, such as automatically pre-heating the oven when it is predicted to be likely that it will soon be needed for cooking.

Several papers have addressed the topic of prediction in smart home environments. Early work on the topic includes MavHome [1], which is a smart home system in which several techniques for *discrete sequence prediction* have been applied to predict the next activity, including LZ78 [2], Active LeZi [3] and Prediction by Partial Matching (PPM) [4]. As an alternative to the formulation of prediction in smart homes as a discrete sequence prediction problem, Mahmoud et al. [5], [6] formulated it as a *time series prediction problem*, where each sensor in the smart home is represented by a binary-valued time series. Echo State Networks (ESN) (in [5]) and Nonlinear Autoregressive netwoRk with eXogenous inputs (NARX) networks (in [6]) were applied to the time-series-based formulation of the problem of prediction in smart home environments in order to predict future occupancy of a room. In later work, Mahmoud et al. [7] generalized this work by applying NARX networks and Elman networks to predict the future time series values of each sensor in the smart home.

In this work, we take the *discrete sequence prediction* formulation of the smart home prediction problem. To the best of our knowledge, no work in this area evaluates their sequence prediction techniques on a *collection of multiple*

TABLE I
A fragment of the sensor events in the data collected in household A of an MIT smart home environment study [8].

Event ID	Activity (Sensor)	Timestamp
...	...	...
101	Cabinet 1	02-05-2003 00:32:26
102	Light switch	02-05-2003 00:56:10
103	Refrigerator open/close	02-05-2003 05:51:01
104	Refrigerator open/close	02-05-2003 05:57:44
105	Microwave	02-05-2003 05:58:59
106	Sink faucet - cold water	02-05-2003 06:19:31
...	...	...

smart home datasets. Instead, existing studies often evaluate prediction techniques for smart homes by focusing on a single particular smart home, without validating whether their findings generalize to other smart homes with different sensor configurations, which we consider to be of importance given the diversity of smart homes. In this work, we evaluate a range of sequence prediction techniques on a collection of publicly available real-life datasets from the smart home environment domain. Additionally, where existing work focuses on *what* activity is going to take place next, we additionally focus on the task of predicting *when* this next activity will take place. Finally, we explore the predictive performance of sequence prediction techniques when we do not just predict the *direct next activity*, but instead predict a window of multiple activities into the future.

This paper is structured as follows. In Section II we describe basic concepts and notation. In Section III we address the task of predicting the next activity in smart home datasets. In Section IV we address the task of predicting *when* the next event will take place. In Section V we evaluate prediction techniques for the prediction of a window of multiple next activities. In Section VI we discuss related work, and Section VII concludes this paper.

II. Background

In this section, we introduce concepts used in later sections of this paper.

A. Event logs, traces and sequences

For a given set A, A^* denotes the set of all sequences over A and $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ denotes a sequence of length n with $\sigma(i) = a_i$. $\langle\rangle$ is the empty sequence and $\sigma_1 \cdot \sigma_2$ is the concatenation of sequences σ_1 and σ_2. $hd^k(\sigma) = \langle a_1, a_2, \ldots, a_k \rangle$ is the prefix of length k (with $0<k<n$) of sequence σ and $tl^k(\sigma) = \langle a_{k+1}, \ldots, a_n \rangle$ is its suffix. For example, for a sequence $\sigma_1 = \langle a, b, c, d, e \rangle$, $hd^2(\sigma_1) = \langle a, b \rangle$ and $tl^2(\sigma_1) = \langle c, d, e \rangle$. The task of *sequence prediction* consists of predicting the suffix $tl^k(\sigma)$ of some sequence σ given its prefix $hd^k(\sigma)$.

Let $\mathcal{E}$ be the event universe, i.e., the set of all possible event identifiers, and $\mathcal{T}$ the time domain. We assume that events are characterized by various properties, e.g., an event has a timestamp, corresponds to an activity, etc. We do not impose a specific set of properties, however, given the focus of this paper we assume that two of these properties are the timestamp and the activity of an event, i.e., there is a function $\pi_{\mathcal{T}} \in \mathcal{E} \to \mathcal{T}$ that assigns timestamps to events, and a function $\pi_{\mathcal{A}} \in \mathcal{E} \to \mathcal{A}$ that assigns to each event an activity from a finite set of activities $\mathcal{A}$. The set of activities $\mathcal{A}$ for a smart homes setting can be the set of sensors in the smart home, where each event represents a sensor trigger. In this case, sequence prediction addresses the challenge of predicting which sensors are likely to be triggered next. Alternatively, activity recognition techniques (see [9] for an overview of activity recognition in smart homes) can be used to transform the data into a human activity dataset where $\mathcal{A}$ consists of activities of human behavior. In this case, sequence prediction addresses the challenge of predicting which human activities are going to occur next.

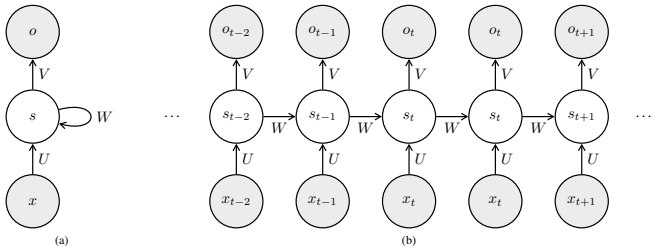

Fig. 1. *(a)* A simple recurrent neural network consisting of a single hidden layer, and *(b)* the recurrent neural network unfolded over time.

A *trace* is a finite non-empty sequence of events $\sigma \in \mathcal{E}^*$ such that each event appears only once and time is non-decreasing, i.e., for $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$ and $\pi_{\mathcal{T}}(\sigma(i)) \leq \pi_{\mathcal{T}}(\sigma(j))$. $\mathcal{C}$ is the set of all possible traces. In the context of smart home environments, we consider each day of sensor log to be one trace. An *event log* is a set of traces $L \subseteq \mathcal{C}$ such that each event appears at most once in the entire log. Table I shows a small sample of an event log on the sensor level.

Often we need to compute a sequence consisting of the values of a single property p for each event in a trace. To this end, we lift the function π_p mapping each event to the value of its property p in such a way that we can apply it to sequences of events (traces).

A function $f \in X \to Y$ can be lifted to sequences over X using the following recursive definition: (1) $f(\langle\rangle) = \langle\rangle$; (2) for any $\sigma \in X^*$ and $x \in X$: $f(\sigma \cdot \langle x \rangle) = f(\sigma) \cdot \langle f(x) \rangle$. Using this definition of lifting functions to sequences, $\pi_{\mathcal{A}}(\sigma)$ transforms a trace σ to a sequence of its activities. For example, for trace $\sigma = \langle e_1, e_2 \rangle$, with $\pi_{\mathcal{A}}(e_1) = a$ and $\pi_{\mathcal{A}}(e_2) = b$, $\pi_{\mathcal{A}}(\sigma) = \langle a, b \rangle$.

B. Neural Networks & Recurrent Neural Networks

A neural network consists of one layer of *inputs units*, one layer of outputs units, and in-between are multiple layers that are referred to as *hidden units*. The outputs of the input units form the inputs of the units of the first *hidden layer* (i.e., the first layer of hidden units), and the outputs of the units of each hidden layer form the input for each subsequent hidden layer. The outputs of the last hidden layer form the input for the output layer. The output of each unit is a function over the weighted sum of its inputs. The weights of this weighted sum performed in each unit are learned through gradient-based optimization from training data that consists of example inputs and desired outputs for those example inputs. Recurrent Neural Networks (RNNs) are a special type of neural networks where the connections between neurons form a directed cycle.

RNNs can be unfolded, as shown in Figure 1. Each step in the unfolding is referred to as a time step, where x_t is the input at time step t. RNNs can take an arbitrary length sequence as input, by providing the RNN a feature representation of one element of the sequence at each time step. s_t is the hidden state at time step t and contains information extracted from all time steps up to t. The hidden state s is updated with information of the new input x_t after each time step: $s_t = f(Ux_t + Ws_{t-1})$,

where U and W are vectors of weights over the new inputs and the hidden state respectively. In practice, either the hyperbolic tangent or the logistic function is generally used for function f, which is referred to as the activation function. The logistic function is defined as: $sigmoid(x) = \frac{1}{1+exp(-x)}$. In neural network literature, the sigmoid function is often represented by σ, however, to avoid confusion with traces, we fully write $sigmoid$. o_t is the output at step t.

C. Long Short-Term Memory for Sequence Modeling

A Long Short-Term Memory (LSTM) model [10] is a special Recurrent Neural Network architecture that has powerful modeling capabilities for long-term dependencies. The main distinction between a regular RNN and an LSTM is that the latter has a more complex memory cell C_t replacing s_t. Where the value of state s_t in an RNN is the result of a function over the weighted average over s_{t-1} and x_t, the LSTM state C_t is accessed, written, and cleared through controlling gates, respectively o_t, i_t, and f_t. Information on a new input will be accumulated to the memory cell if i_t is activated. Additionally, the previous memory cell value C_{t-1} can be "forgotten" if f_t is activated. The information of C_t will be propagated to the output h_t based on the activation of output gate o_t. Combined, the LSTM model can be described by the following formulas:

$$
\begin{aligned}
f_t &= sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_i * \tilde{C}_t \\
o_t &= sigmoid(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * tanh(C_t)
\end{aligned}
$$

In these formulas all W variables are weights and b variables are biases and both are learned during the training phase.

D. Gated Recurrent Units

Gated Recurrent Units (GRU) were proposed by Cho et al. [11] as a simpler alternative to the LSTM architecture. In comparison to LSTMs, GRUs do not keep a separate memory cell and instead merges the cell state C_t and hidden state h_t. Furthermore, it combines the input gate i_t and the forget gate f_t into a single *update gate*. While the LSTMs and GRUs are identical in the class of functions that they can learn, GRUs are simpler in the sense that they have fewer model parameters. Empirically, GRUs have been found to outperform LSTMs on several sequence prediction tasks [12].

III. Next Activity Prediction

In this section, we present and evaluate techniques to predict the activity of the next event in smart home data. The aim is to learn an activity prediction function f_a^1 such that $f_a^1(hd^k(\sigma)) = hd^1(tl^k(\pi_{\mathcal{A}}(\sigma)))$ for any trace σ from a smart dataset and for any prefix length k. We start with introducing an LSTM-based approach in Section III-A and describe alternative discrete sequence prediction approaches in Section III-B. In Section III-C we discuss the experimental setup, and in Section III-D we discuss results.

A. Next Activity Prediction with LSTMs

We transform each event $e \in hd^k(\sigma)$ into a feature vector and use these vectors as LSTM inputs $x_1, \ldots, x_k$. We build the feature vector as follows. We start with $|\mathcal{A}|$ features that represent the type of activity of event e in a so-called *one-hot encoding*. We take an arbitrary but consistent ordering over the set of activities $\mathcal{A}$, and use $index \in \mathcal{A} \rightarrow \{1, \ldots, |\mathcal{A}|\}$ to indicate the position of an activity in it. The one-hot encoding assigns the value 1 to feature number $index(\pi_{\mathcal{A}}(e))$ and a 0 value to others. We concatenate the features obtained from the one-hot encoding with three time-based features: fv_{t1}, fv_{t2}, and fv_{t3}. The first time-based feature for event $e = \sigma(i)$ is defined as the time between the previous event in the trace and the current event, i.e., $fv_{t1}(e) = \begin{cases} 0 & \text{if } i = 1, \\ \pi_{\mathcal{T}}(e) - \pi_{\mathcal{T}}(\sigma(i-1)) & \text{otherwise.} \end{cases}$

This feature allows the LSTM to learn dependencies between the time differences at different points (indexes) within the day. Feature fv_{t1}, for example, allows the model to learn a relation such as that the time difference between *waking up* and *showering* is shorter on working days compared to weekend days, thereby making the occurrence as a *leave house* event as next activity more likely to be the next event.

The behavior of people follows a daily rhythm, i.e., activities are correlated to the point in time at which they occur. To capture this daily rhythm, we add a feature fv_{t2} that contains the time within the day (since midnight). Likewise, we add a feature fv_{t3} that contains the time within the week (since midnight on Sunday) to capture the weekly rhythm, allowing the model to learn that some activities are more likely to occur on certain moments in the week.

There are several different LSTM architectures to model the next activity prediction function f_a^1 and to model a function time f_t^1 that predicts the timestamp of the next event (addressed in detail in Section IV). Firstly, we can train two separate models, one for f_a^1 and one for f_t^1, both using the same input features at each time step, as represented in Figure 2 (a). Secondly, f_a^1 and f_t^1 can be learned jointly in a single LSTM model that generates two outputs, in a multi-task learning setting [13] (Figure 2 (b)). The usage of LSTMs in a multi-task learning setting has shown to improve performance on all individual tasks when jointly learning multiple natural language processing tasks, including part-of-speech tagging, named entity recognition, and sentence classification [14]. Furthermore, multi-task learning has shown to improve the accuracy on predicting the next activity and the timestamp of the next event in a study focusing on prediction in the context of business processes [15]. A hybrid option in-between the architectures of Figures 2 (a) and (b) is an architecture consisting of a number of shared LSTM layers for both tasks, followed by a number of layers that specialize in either prediction of the next activity or prediction of the time until the next event, as shown in Figure 2 (c).

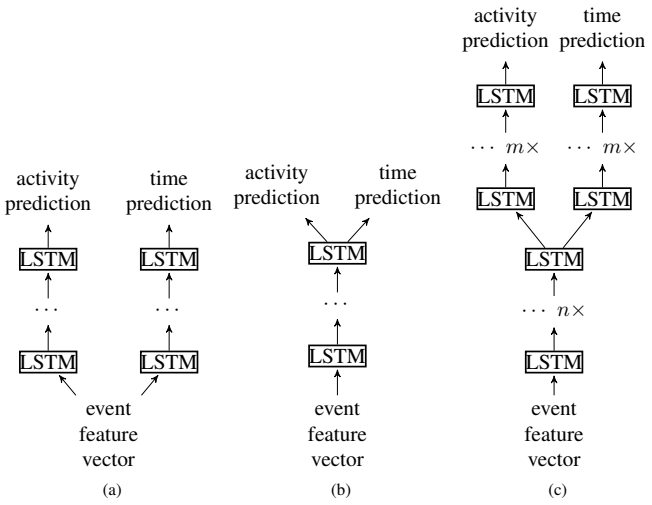

Fig. 2. Neural Network architectures with single-task layers *(a)*, with shared multi-tasks layer *(b)*, and with $n + m$ layers of which n are shared *(c)* (obtained from Tax et al. [15]).

At training time, we set the target output o_a^k of time step k to the one-hot encoding of the activity of the event one time step later. However, it can be the case that the case ends at time k, in which case there is no new event to predict. Therefore, we add an extra element to the output one-hot-encoding vector, which has value 1 when the case ends after k. We set a second target output o_t^k equal to the fv_{t1} feature of the next time step, i.e., the target is the time difference between the next and the current event. However, knowing the timestamp of the current event, we can calculate the timestamp of the following event. We optimize the weights of the neural network with the Adam learning algorithm [16] such that the cross entropy between the ground truth one-hot encoding of the next event and the predicted one-hot encoding of the next event as well as the mean absolute error (MAE) between the ground truth time until the next event and the predicted time until the next event are minimized.

It should be noted that the LSTM-based activity prediction function f_a^1 outputs the full probability distribution over the set of possible next activities $\mathcal{A}$. For evaluation purposes, we will only use the most likely continuation. Furthermore, note that the architectures shown in Figure 2 are not specific to LSTMs and GRUs can also be used instead.

B. Other Discrete Sequence Prediction Techniques

Prediction by Partial Matching (PPM) [4] is one of the early methods for sequence prediction. PPM is a technique that uses a *partial sequence matching* strategy that uses a high-order Markov model, but uses a lower order Markov model in cases when high-order ones are not yet available.

The tasks of *compressing* a sequence of discrete symbols is closely linked to the sequence prediction task, because an algorithm that can predict the next event in a sequence accurately can also be used to compress that sequence data. **LZ78** [2] is a lossless sequence compression algorithm that compresses a sequence by replacing frequent subsequences with references to a dictionary. Additionally, LZ78 can be used as a sequence prediction algorithm.

Padmanabhan and Mogul [17] developed the **Dependency Graph (DG)** sequence prediction algorithm for the purpose of improving web latency by prefetching web pages that are likely to be visited next, based on the pages that are visited so far. DG builds a model from the training data that is conceptually close to a first-order Markov model, in which each state represents an activity and there is an arc from one state to another if the corresponding activities follow each other in the training data in at most w steps. The arc is weighted by the ratio of the events of the originating state that fulfill this criteria. At inference time, the activity is predicted that can be reached which the arc with the highest weight from the state that corresponds to the last activity in the trace, i.e., $\pi_{\mathcal{A}}(\sigma(|\sigma|))$.

All k-order Markov Chains (AKOM) [18] is a sequence prediction algorithm that fits all semi-Markov models of up to order k on the training set. At inference time, the Markov model that has the highest order that also contains a state that matches the prefix for which the prediction is made is used to make the prediction.

Laird and Saul [19] developed the **Transition Directed Acyclic Graph (TDAG)** algorithm for discrete sequence prediction, and applied it to a number of prediction problems, including predictive caching. The TDAG algorithm builds a Markov tree where paths through the tree represent observed prefixes of the training sequences. The Markov tree is limited in size by pruning away less likely prediction contexts, thereby preventing an explosion of the size of the Markov tree.

Compact Prediction Tree (CPT) [20] is a sequence prediction algorithm that builds a tree representation from the sequences in the training data and uses this data structure to efficiently find sequences in the training set that are similar to the sequence σ for which the prediction is being made. In a later work, **CPT+** [21] adds a filtering step that filters out infrequent subsequences from the event log before applying CPT, which empirically outperforms CPT on a collection of datasets.

In addition to these techniques that originate from the fields of *sequence compression* and *data mining*, several sequence prediction approaches have been developed in the area of *process mining*. Process mining is a research area that focuses on the analysis of event logs from the domain of *business process management*. Van der Aalst et al. [22] proposed three methods to predict the remaining time in a business process, which we will refer to as **Set**, **Bag**, and **List**. Those approaches all build a *labeled transition system* from the training data, but differ in the way in which they construct this transition system. In the *set* approach each state in the transition system corresponds to the *set of observed activities in the prefix*, while each state in the bag approach corresponds to the *multiset* of observed activities. The *list* approach takes the ordering of activities in the prefix into account: each state represents the sequence of activities seen in the prefix. When predicting the remaining time for a given prefix from the test set, the approach matches the prefix to the corresponding state of the transition system, and then predicts the *average*

TABLE II
AN OVERVIEW OF THE DATASETS USED IN THE EXPERIMENTS.

Dataset	# of days	# of events	# of activities
Sensor-level Datasets			
MIT B [8]	17	1962	68
CASAS hh104 [24]	62	413142	112
van Kasteren [23]	23	1285	14
Activity-level Datasets			
CASAS hh102 [24]	25	1572	18
CASAS hh104 [24]	62	5928	19
CASAS hh110 [24]	27	1758	17
van Kasteren [23]	23	220	7

remaining time of all the prefixes from the training set that also correspond to that state. To predict the activity of the next event, we adapt the technique to a discrete sequence prediction approach by predicting the activity that *most frequently* follows the prefixes in the training data that correspond the transition system state for which the prediction is being made.

C. Experimental setup

We evaluate the accuracy of methods for predicting the next activity on a collection of smart home datasets. For each of the datasets we use the chronologically ordered first 2/3 traces as training data, and evaluate the activity predictions on the remaining 1/3 traces (i.e., the test set). For each trace σ in the test set, we evaluate the next activity prediction on all prefixes $hd^k(\sigma)$ of all trace σ in the set of test traces. We evaluate the prediction quality in terms of *accuracy*, i.e., the ratio of correct predictions w.r.t. the total number of prediction made.

We apply the architectures of Figure 2 to each dataset using LSTMs and as well as using GRUs. We use the LSTM and GRU implementations that are available in the recurrent neural network library `Keras`[1]. The experiments are performed on a single NVidia Tesla K80 GPU, on which the experiments took about one second per training iteration depending on the neural network architecture and the dataset. The inference time, i.e, the time needed to make a prediction, is in the order of milliseconds. For the discrete sequence prediction techniques LZ78, PPM, DG, AKOM, TDAG and CPT+ we use the implementations that are provided in the SPMF toolkit [25] for frequent pattern mining and sequence mining. For the set, bag, and list sequence prediction methods from the process mining field we use their implementations in the process mining toolkit ProM [26].

Table II provides an overview of the datasets that we use in the evaluation and lists their properties in terms of number of days of logging, the number of events, and the number of activities. All the datasets used in the experiment are generally used to benchmark methods for *activity recognition*, where the research challenge is to infer the *human activities* from the *sensor readings*. Instead, here we use those datasets to benchmark activity prediction techniques. These datasets therefore consist of two parallel datasets: one contains the *sensor-level events*, and the other contains the *activity-level events*. We are interested both in *how well we can predict which sensor is going to trigger next* and in *how well we can predict what type of human activity is going to take place next*. Therefore, we evaluate the next activity prediction techniques on both sensor-level and activity-level datasets. Note that we use the ground truth information from those datasets for activity recognition as the activity-level data. To predict the next activity in an online smart home environment one would first need to apply activity recognition techniques to infer the behavior on the human behavior level. In this study we consider this activity recognition step to be out-of-scope.

[1]https://keras.io/

D. Results

Table III shows the obtained accuracy values for each of the neural network architectures on each dataset. The results show that an architecture that consists of two neural network layers out of which one shared layer that focuses on both the activity and the time prediction outperforms other architectures on all datasets. This indicates that there is gain in using multi-task learning [13], i.e., jointly predicting the activity and the timestamp of the next event with one model leads to higher accuracy compared to predicting both with separate models. Surprisingly, the results show that an LSTM architecture outperforms a GRU architecture, which contrasts the finding that GRU architectures outperform LSTM architectures on many sequence modeling tasks [12].

Table IV shows the accuracy that is obtained using the alternative sequence prediction techniques that we introduced in Section III-B. On datasets other than CASAS hh104 and van Kasteren on the sensor-level, the LSTM leads to more accurate predictions than these alternative sequence prediction approaches. Note that the dependency graph approach outperforms all other approaches, including those based on neural networks, On the sensor-level events of the van Kasteren [23] dataset. Overall, the dependency graph approach seems outperform the other approaches that are not neural-network-based. Note that the accuracy of process mining based approaches is highly dependent on the dataset, as they are the least accurate predictor on some datasets, while the set approach is the most accurate method on the sensor-level CASAS hh104 [24].

IV. TIME OF NEXT EVENT PREDICTION

In this section we focus on predicting the timestamp of the next event. Formally, we aim to learn a function f_t^1 such that $f_t^1(hd^k(\sigma)) = hd^1(tl^k(\pi_{\mathcal{T}}(\sigma)))$ for any sequence σ from a smart home dataset and for any prefix length k. In Section IV-A we introduce techniques for the prediction of the timestamp of the next event. In Section IV-B we describe the experimental setup to evaluate techniques for the prediction of the timestamp of the next event. Finally, in Section IV-C we present and discuss the results.

A. Techniques

The neural network architectures of Figure 2 address the task of predicting the time until the next event as well as

TABLE III
THE ACCURACY OF THE ACTIVITY OF THE *predicted* NEXT EVENT WITH RESPECT TO THE ACTIVITY OF THE *actual* NEXT EVENT OBTAINED WITH DIFFERENT NEURAL NETWORK ARCHITECTURES.

Dataset	LSTM									GRU								
	Number of Layers (Number of shared Layers)																	
	1 (0)	1 (1)	2 (0)	2 (1)	2 (2)	3 (0)	3 (1)	3 (2)	3 (3)	1 (0)	1 (1)	2 (0)	2 (1)	2 (2)	3 (0)	3 (1)	3 (2)	3 (3)
Sensor-level Datasets																		
MIT B	0.135	0.144	0.139	0.146	***0.147***	0.137	0.131	0.132	0.129	0.128	0.134	0.128	0.139	0.135	0.129	0.117	0.115	0.114
CASAS hh104	0.048	0.098	0.118	0.129	0.096	0.108	0.112	0.113	0.111	0.052	0.076	0.103	0.116	0.083	0.095	0.102	0.105	0.109
van Kasteren	0.125	0.143	0.142	0.146	0.142	0.139	0.129	0.131	0.128	0.118	0.133	0.131	0.129	0.135	0.118	0.121	0.121	0.119
Activity-level Datasets																		
CASAS hh102	0.262	0.261	0.265	***0.269***	0.264	0.263	0.259	0.261	0.256	0.251	0.258	0.259	0.258	0.257	0.252	0.251	0.252	0.255
CASAS hh104	0.316	0.318	0.315	***0.321***	0.312	0.316	0.311	0.314	0.305	0.307	0.307	0.304	0.313	0.304	0.309	0.306	0.306	0.301
CASAS hh110	0.399	0.455	0.445	***0.466***	0.458	0.441	0.402	0.406	0.401	0.401	0.444	0.436	0.451	0.441	0.430	0.390	0.391	0.388
van Kasteren	0.506	0.521	0.505	***0.529***	0.521	0.515	0.498	0.501	0.495	0.498	0.492	0.501	0.512	0.506	0.496	0.482	0.482	0.478

TABLE IV
THE ACCURACY OF THE ACTIVITY OF THE *predicted* NEXT EVENT WITH RESPECT TO THE ACTIVITY OF THE *actual* NEXT EVENT OBTAINED WITH SEVERAL PREDICTION TECHNIQUES.

Dataset	Set	Bag	List	DG	TDAG	CPT+	PPM	AKOM	LZ48
Sensor-level Datasets									
MIT B	0.004	0.002	0.000	0.123	0.057	0.027	0.123	0.057	0.082
CASAS hh104	***0.175***	0.004	0.001	0.059	0.037	0.011	0.059	0.057	0.057
van Kasteren	0.097	0.020	0.014	***0.188***	0.164	0.075	0.107	0.164	0.160
Activity-level Datasets									
CASAS hh102	0.154	0.087	0.079	0.174	0.117	0.008	0.117	0.117	0.117
CASAS hh104	0.136	0.103	0.120	0.195	0.186	0.059	0.182	0.186	0.182
CASAS hh110	0.248	0.120	0.120	0.132	0.081	0.111	0.085	0.090	0.090
van Kasteren	0.250	0.250	0.200	0.383	0.133	0.383	0.183	0.183	0.183

predicting the next activity, and can therefore be used as an implementation for function f_t^1. The task of predicting the time until the next event is a sequence prediction task with a continuous output, in contrast to the task of predicting the next activity, which has a discrete output. Therefore, the LZ78, PPM, DG, AKOM, TDAG, and CPT+ algorithms cannot be used as baseline techniques.

In the original definition, the *set*, *bag*, and *list* approaches [22] from the process mining field focused on the prediction of a continuous outcome (i.e., remaining time). However, it can be naturally adjusted to predict the time until the next event, as shown in [15]. To do so, the transition system states are annotated with the average time until the next event instead of the average time until the end of the process. We will use this approach as a baseline to predict the timestamp of next event.

B. Experimental Setup

We use the same datasets and the same train/test splits that we used in Section III. To each dataset, we apply the architectures of Figure 2 both using LSTMs and using GRUs, which predict the time until the next event in addition to predicting the next activity.

Two well-known error metrics for prediction tasks with a continuous output are *Mean Absolute Error (MAE)* and *Root Mean Square Error (RMSE)*. The time gaps between consecutive events tend to be highly varying, with values sometimes differing multiple orders of magnitude. Therefore, we evaluate the predictions of the time until the next event with MAE, which is more robust to outliers than RMSE. We measure the MAE between the *predicted* time until the next event and the *actual* time until the next event in seconds and average the MAE over all prefixes in the test traces of the test set.

C. Results

Table V shows the results in terms of MAE for each of the neural network architectures and on each dataset. The results show that LSTM architectures outperform GRU architectures and that an LSTM architecture consisting of two layers and one shared layer outperforms the other LSTM architectures. These findings are in line with our findings for the prediction of the activity of the next event, which indicates that an LSTM with two layers and a single shared layer can be used for both tasks.

Table VI shows the results in terms of MAE of the *set*, *bag*, and *list* approaches and compares them to the *best performing LSTM* model. The list approach outperforms the other methods from the process mining field on the prediction of the time until the next event on smart home environment data. However, the LSTM outperforms the list approach on all datasets except CASAS hh104 on the sensor level. On some datasets, such as the activity-level data from the van Kasteren dataset, the LSTM outperforms the process mining techniques by close to a factor two in terms of in MAE.

V. ACTIVITY WINDOW PREDICTION

In this section we focus on the prediction of the activities of multiple future events. This contrast Section III, where we focused on the prediction of the activity of the *direct next event*. In Section V-A we discuss techniques to predict a window of next activities and discuss the experimental setup. In Section V-B we present and discuss the results.

A. Techniques & Experimental Setup

We can make longer-term predictions that allow us predict multiple events ahead by applying function f_a^1 repeatedly. For example, we can predict the next two activities by first predicting $f_a^1(\sigma)$ and then using this prediction to also predict the second next activity, i.e., $f_a^2(\sigma) = f_a^1(\sigma \cdot \langle f_a^1(\sigma) \rangle)$. We use $f_a^\perp$ to refer to the activity prediction function that predict the

TABLE V
THE RESULTS OBTAINED WITH DIFFERENT NEURAL NETWORK ARCHITECTURES IN TERMS OF MEAN ABSOLUTE ERROR (MAE) IN SECONDS BETWEEN THE TIMESTAMP OF THE *predicted* AND OF THE *actual* NEXT EVENT.

Dataset	LSTM									GRU								
	Number of Layers (Number of shared Layers)																	
	1 (0)	1 (1)	2 (0)	2 (1)	2 (2)	3 (0)	3 (1)	3 (2)	3 (3)	1 (0)	1 (1)	2 (0)	2 (1)	2 (2)	3 (0)	3 (1)	3 (2)	3 (3)
Sensor-level Datasets																		
MIT B	736.12	685.96	698.25	***660.54***	675.36	683.23	679.45	678.36	681.65	748.85	696.32	706.23	682.52	685.15	691.56	691.23	683.61	685.23
CASAS hh104	16.12	16.07	16.08	16.05	16.12	16.15	16.13	16.12	16.12	16.14	16.06	16.08	16.06	16.14	16.17	16.15	16.14	16.14
van Kasteren	1202.30	1172.41	1185.36	***1130.59***	1161.96	1189.32	1176.35	1175.45	1182.12	1212.05	1193.36	1198.12	1161.98	1185.44	1200.01	1191.71	1184.93	1191.49
Activity-level Datasets																		
CASAS hh102	2178.80	2002.22	1989.13	***1906.86***	1966.16	2012.03	1989.85	1981.99	2020.59	2201.73	2197.16	2032.03	2008.13	2012.89	2103.13	2065.58	2011.02	2078.74
CASAS hh104	1699.32	1511.07	1487.46	***1210.32***	1297.09	1414.78	1395.50	1312.85	1436.00	1612.64	1544.84	1472.61	1301.30	1401.98	1503.85	1485.33	1411.54	1525.34
CASAS hh110	4045.91	3875.63	3964.35	***3367.01***	3572.84	3891.36	3368.43	3599.66	3774.62	4221.26	4085.13	4114.35	3500.44	3783.91	4054.56	3539.93	3771.12	3997.06
van Kasteren	5906.08	5187.91	5284.44	***3757.70***	4896.31	5344.48	4981.17	4523.61	4774.22	5890.78	5513.95	5348.56	4118.12	4648.27	4931.42	4836.02	4732.97	4655.98

TABLE VI
THE MEAN ABSOLUTE ERROR (MAE) IN SECONDS BETWEEN THE TIMESTAMP OF THE *predicted* AND OF THE *actual* NEXT EVENT.

Dataset	Set	Bag	List	LSTM
Sensor-level Datasets				
MIT B	729.91	785.35	771.46	***660.54***
CASAS hh104	22.87	16.03	**15.84**	16.05
van Kasteren	1525.09	1236.48	1193.77	***1130.59***
Activity-level Datasets				
CASAS hh102	2611.42	2383.23	2370.34	***1906.86***
CASAS hh104	1663.69	1594.17	1527.57	***1210.32***
CASAS hh110	4541.99	4467.21	4476.94	***3367.01***
van Kasteren	8182.16	7698.02	6801.39	***3757.70***

whole remainder of the day of smart home behavior, and aim to find a function $f_a^{\perp}$ such that $f_a^{\perp}(hd^k(\sigma)) = tl^k(\pi_{\mathcal{A}}(\sigma))$. Function $f_a^{\perp}$ iteratively predicts the next activity using f_a^1, until function f_a^1 predicts the end of sequence (denote with $\perp$). Formally, we predict the remainder of the day in terms of activities as follows:

$$f_a^{\perp}(\sigma) = \begin{cases} \sigma & \text{if } f_a^1(\sigma) = \perp, \\ f_a^{\perp}(\sigma \cdot \langle e \rangle), \text{with } e \in \mathcal{E}, \pi_{\mathcal{A}}(e) = f_a^1(\sigma) & \text{otherwise.} \end{cases}$$

We evaluate the activity window prediction techniques on the same datasets and train/test splits that we used in Section III. For a given trace prefix $hd^k(\sigma)$ we evaluate $f_a^{\perp}$ by calculating the similarity between the predicted continuation $f_a^{\perp}(hd^k(\sigma))$ and the actual continuation $\pi_{\mathcal{A}}(tl^k(\sigma))$. For the sake of evaluation, we specifically focus on predicting the next three activities, i.e., we compare $hd^3(f_a^{\perp}(hd^k(\sigma)))$ and the $hd^3(\pi_{\mathcal{A}}(tl^k(\sigma)))$.

To compare the sequence of predicted activities to the sequence of actual activities we use Levenshtein distance [27], which is a well-known distance measure for sequences. Levenshtein distance is defined as the minimum number of insertion, deletion, and substitution operations needed to transform one sequence into the other. This value is normalized by the maximum of the lengths of the two sequences to yield a value on a $[0, 1]$ interval. We report the *Levenshtein similarity*, defined as one minus the Levenshtein distance, where 1 indicates that the sequences are identical and 0 indicates a maximal difference between the sequences. Note that Levenshtein similarity is equivalent to accuracy when comparing sequences of length one.

TABLE VII
THE LEVENSHTEIN SIMILARITY BETWEEN THE *predicted* AND THE *actual* SEQUENCE OF THE NEXT THREE ACTIVITIES.

Dataset	DG	TDAG	CPT+	PPM	AKOM	LZ48	LSTM
Sensor-level Datasets							
MIT B	0.132	0.071	0.032	0.127	0.075	0.090	***0.145***
CASAS hh104	***0.060***	0.036	0.012	0.059	0.059	0.059	0.56
van Kasteren	0.192	0.196	0.080	0.133	***0.222***	0.198	0.141
Activity-level Datasets							
CASAS hh102	***0.285***	0.167	0.014	0.176	0.176	0.144	0.158
CASAS hh104	0.222	0.317	0.072	***0.321***	0.317	0.305	0.306
CASAS hh110	0.229	0.154	0.123	0.178	0.192	0.205	***0.382***
van Kasteren	0.417	0.239	0.417	0.367	0.361	0.317	***0.467***

B. Results

Table VII shows the average Levenshtein similarity values over all prefixes on all datasets. Interestingly, the LSTM is the best performing method on only half of the datasets, even though it clearly outperforms the other techniques in predicting the direct next activity. It turns out to be dependent on the dataset which of the sequence prediction approaches makes the most accurate estimate of the following three activities.

VI. RELATED WORK

Wu et al. [28] provide a survey on prediction techniques in smart home environments. One of their main observations was that the majority of existing work focuses on prediction of location rather than prediction of activities of daily living.

Moon and Hamm [29] use LSTMs to predict the next activities in the data from the large-scale American Time Use Survey (ATUS). Unlike this study, Moon and Hamm focus on manually logged/reported activity data rather than data from smart home environments.

Vintan et al. [30] focused on the prediction of movement of employees between rooms in an office setting using a feed-forward neural network. However, by using a feed-forward neural network architecture instead of a recurrent one, a separate model needs to be trained for each prefix length.

Liouane et al. [31] developed a recurrent version of an Extreme Learning Machine (ELM) and applied it to predict the future values of the ability to live independently of elderly inhabitants of a smart home. The ability to live independently is quantified using the so-called *Functional Autonomy Mea-*

surement System, and the inputs are sensor-level events logged by a smart home.

Hao et al. [32] applied *Formal Concept Analysis* to recognize and predict activities of daily living based on smart home sensor data, focusing specifically on kitchen activities.

None of these related works address the prediction of both *what* a smart home occupant will do next as well as *when* he will do that.

VII. Conclusions & Future Work

In this paper, we have evaluated several sequence prediction techniques on a collection of smart home datasets for the prediction of the next activity, the time until the next event, and the prediction of a window of next activities. We have found that LSTM neural networks outperform the other approaches for the prediction of the direct next event, and have also found that applying multi-task learning by jointly predicting the next activity and the timestamp of the next event outperforms separate LSTM models for both tasks separately. Additionally, the multi-task LSTM also outperforms other prediction tasks for the prediction of the timestamp of the next event. However, the LSTM does not outperform other approaches when we aim to predict a window of three next activities instead of only the direct next event. Instead, depending on the dataset, either an LSTM, AKOM, DG, or PPM is the best performing sequence prediction technique for predicting the next three activities.

References

[1] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An agent-based smart home," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2003, pp. 521–524.

[2] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[3] K. Gopalratnam and D. J. Cook, "Active LeZi: An incremental parsing algorithm for sequential prediction," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 04, pp. 917–929, 2004.

[4] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[5] S. M. Mahmoud, A. Lotfi, N. Sherkat, C. Langensiepen, and T. Osman, "Echo state network for occupancy prediction and pattern mining in intelligent environment," in *Proceedings of the International Conference on Intelligent Environments (IE)*, vol. 2. IOS Press, 2009, pp. 474–481.

[6] S. M. Mahmoud, A. Lotfi, and C. Langensiepen, "Occupancy pattern extraction and prediction in an inhabited intelligent environment using NARX networks," in *Proceedings of the International Conference on Intelligent Environments (IE)*. IEEE, 2010, pp. 58–63.

[7] S. Mahmoud, A. Lotfi, and C. Langensiepen, "Behavioural pattern identification and prediction in intelligent environments," *Applied Soft Computing*, vol. 13, no. 4, pp. 1813–1822, 2013.

[8] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *Proceedings of the International Conference on Pervasive Computing (PERVASIVE)*. Springer, 2004, pp. 158–175.

[9] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, "Sensor-based activity recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 790–808, 2012.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 2014.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proceedings of the Neural Information Processing Systems (NIPS) Deep Learning and Representation Learning Workshop*, 2014.

[13] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[14] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 2008, pp. 160–167.

[15] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer International Publishing, 2017, pp. 477–492.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2015.

[17] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 22–36, 1996.

[18] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict worldwide web surfing," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, 1999, pp. 13–26.

[19] P. Laird and R. Saul, "Discrete sequence prediction and its applications," *Machine learning*, vol. 15, no. 1, pp. 43–68, 1994.

[20] T. Gueniche, P. Fournier-Viger, and V. S. Tseng, "Compact prediction tree: A lossless model for accurate sequence prediction," in *International Conference on Advanced Data Mining and Applications (ADMA)*. Springer, 2013, pp. 177–188.

[21] T. Gueniche, P. Fournier-Viger, R. Raman, and V. S. Tseng, "CPT+: Decreasing the time/space complexity of the compact prediction tree," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer, 2015, pp. 625–636.

[22] W. M. P. van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[23] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the International Conference on Ubiquitous Computing*. ACM, 2008, pp. 1–9.

[24] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, 2013.

[25] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: a Java open-source pattern mining library," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[26] B. F. van Dongen, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM framework: A new era in process mining tool support," in *Proceedings of the International Conference on Applications and Theory of Petri Nets (PETRI NETS)*. Springer, 2005, pp. 444–454.

[27] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[28] S. Wu, J. Rendall, M. Smith, S. Zhu, J. Xu, Q. Yang, H. Wang, and P. Qin, "Survey on prediction algorithms in smart homes," *IEEE Internet of Things Journal*, 2017.

[29] G. E. Moon and J. Hamm, "A large-scale study in predictability of daily activities and places," in *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services (MobiCASE)*. EAI, 2016, pp. 86–97.

[30] L. Vintan, A. Gellert, J. Petzold, and T. Ungerer, "Person movement prediction using neural networks," in *Proceedings of the Workshop on Modeling and Retrieval of Context*, 2006.

[31] Z. Liouane, T. Lemlouma, P. Roose, F. Weis, and H. Messaoud, "An improved extreme learning machine model for the prediction of human scenarios in smart homes," *Applied Intelligence*, pp. 1–14, 2017.

[32] J. Hao, B. Bouchard, A. Bouzouane, and S. Gaboury, "Real-time activity prediction and recognition in smart homes by formal concept analysis," in *Proceedings of the International Conference on Intelligent Environments (IE)*. IEEE, 2016, pp. 103–110.