An LSTM Based System for Prediction of Human Activities with Durations

KUNDAN KRISHNA*, Adobe Research, India
DEEPALI JAIN*, Adobe Research, India
SANKET V. MEHTA†, Carnegie Mellon University, United States
SUNAV CHOUDHARY, Adobe Research, India

Human activity prediction is an interesting problem with a wide variety of applications like intelligent virtual assistants, contextual marketing, *etc.* One formulation of this problem is jointly predicting human activities (*viz.* eating, commuting, *etc.*) with associated durations. Herein a deep learning system is proposed for this problem. Given a sequence of past activities and durations, the system estimates the probabilities for future activities and their durations. Two distinct Long-Short Term Memory (LSTM) networks are developed that cater to different assumptions about the data and achieve different modeling complexities and prediction accuracies. The networks are trained and tested with two real-world datasets, one being publicly available while the other collected from a field experiment. Modeling on the segment level public dataset mitigates the cold-start problem. Experiments indicate that compared to traditional approaches based on sequence mining or hidden Markov modeling, LSTM networks perform significantly better. The ability of LSTM networks to detect long term correlations in activity data is also demonstrated. The trained models are each less than 500KB in size and can be deployed to run in real-time on a mobile device without any dependencies on the cloud. This can help applications like mobile personal assistants by providing predictive context.

CCS Concepts: • **Human-centered computing → User models**; • **Computing methodologies → Neural networks**;

Additional Key Words and Phrases: Human Activity Prediction, Daily Routine Generation, Long-Short Term Memory Network

ACM Reference Format:
Kundan Krishna, Deepali Jain, Sanket V. Mehta, and Sunav Choudhary. 2017. An LSTM Based System for Prediction of Human Activities with Durations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 147 (December 2017), 31 pages.
https://doi.org/10.1145/3161201

1 INTRODUCTION

As we increasingly move towards a world of automation and artificial intelligence, the already pervasive computing devices in our daily lives are becoming smarter and more powerful. Throughout the last decade, smart phone and cloud computing technologies have made rapid advances and as a result, these technologies power many of our

*Both authors contributed equally to the paper
†Work done when the author was affiliated with Adobe Research

Authors' addresses: Kundan Krishna, Adobe Research, Adobe Tower, Block A, Prestige Tech Platina, Kadubeesanahalli, Bengaluru, Karnataka, 560087, India, kunkrish@adobe.com; Deepali Jain, Adobe Research, Adobe Tower, Block A, Prestige Tech Platina, Kadubeesanahalli, Bengaluru, Karnataka, 560087, India, deepjain@adobe.com; Sanket V. Mehta, Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15213, United States, svmehta@andrew.cmu.edu; Sunav Choudhary, Adobe Research, Adobe Tower, Block A, Prestige Tech Platina, Kadubeesanahalli, Bengaluru, Karnataka, 560087, India, schoudha@adobe.com.

2474-9567/2017/12-ART147 $15.00
https://doi.org/10.1145/3161201

everyday experiences and interactions today. A remarkable consequence of these developments has been the ability of using computing to better understand and predict human behavior [34, 52]. Indeed, personal smart phones have unprecedented access to a wide diversity of data about their owners and cloud computing enables mining insights from behavioral data at a massive scale using machine learning algorithms. Understanding human behavior is a topic of broad interest across a wide range of applications [3, 15, 39, 46]. Our work focuses on predicting human activities. Human activity prediction is an important discipline in its own right and finds applications across criminal behavior detection [15], modeling and predicting human driver behavior [29].

Context recognition and inference from sensor data from mobile devices [26, 51] as well as from other smart-sensing environments [31, 41, 50] has been an area of active research recently. Further, prediction of future context has also been of interest. To this end, there has been noticeable work in the domain of next location prediction [33, 45]. We believe that daily routine activities like commuting, working, eating, *etc.* are more capable of capturing the contextual information about a user rather than just location. Predicting these activities with a reasonably high accuracy can enable various context-aware applications. For example, knowledge of daily human activities can enable virtual assistants like Siri to deliver appropriate reminders and recommendations like suggesting leisure reading materials while commuting, suppressing unwanted notifications while at work, suggesting pre-ordering meals shortly before lunch time, *etc.*

Although the uses for a human activity prediction system are many, there are multiple challenges involved in a real-world setting. Firstly, many activity prediction approaches use very limited data [29]. This may work well for instantaneous recognition of context from raw sensory data on mobile devices, but most context prediction models hitherto studied discard long-term dependencies on past behavior in the activity patterns by virtue of modeling assumptions. Secondly, limited data collection problems are further compounded on account of inherent randomness in human behavior and noise introduced due to detection or self-declaration errors in labeling activities within the available reference data. This could result in predictable patterns in activity staying hidden. Thirdly, each human activity is different in terms of when it is likely to start and how long it is likely to last. For example, *eating* is likely to take place only during certain times of the day (say around 0800, 1230 and 2000 hours) and lasts for a certain duration (say 30 minutes) and this is completely different from commuting which is likely to take place around 0730 and 1700 hours and may last for 15 minutes. It is non-trivial for many prediction approaches to be able to incorporate this variance in data to boost the achievable prediction accuracies. Furthermore, for context prediction as it applies to most use cases discussed in the previous paragraph, prediction of the future activity with its duration is far more valuable than just predicting the future activity. In the domain of activity prediction, the duration varies a lot with the activity - sleeping might be 6 hours while eating may last for only 15 minutes. We explicitly handle this issue by jointly predicting a separate duration for each activity that has a probability of occurrence. This contrasts with a non-joint prediction where there is a single duration predicted for the next activity with the duration prediction being independent of the predicted activity. We show that a joint prediction approach is superior to a non-joint prediction in Section 8.3. Finally, it is desirable if prediction approaches could utilize individual data across multiple days (for personalized prediction of activity) and/or one-day long datasets across different individuals belonging to a segment of population (for general contextual insights), whichever is available. These are disparate data sources and not easy to use simultaneously for many prediction approaches, in case both types of data are available. Our proposed prediction approach is agnostic to data source and can work in scenarios where we have disparate data sources. Further, data across multiple individuals is necessary to address the 'cold start' problem, *i.e.* to make predictions for individuals with little or no observed data.

In this paper, we formulate the human activity prediction problem as a machine learning task with the following characteristics. We approach the problem as a task of jointly predicting human activity and estimating its duration given the past behavior of the user. Thus, the prediction problem is to estimate probabilities for `(activity, duration)` pairs during the remaining part of a day, after having observed the past sequence of such pairs

that took place during the course of the day. This formulation has aspects common with problems in sequence prediction and this motivates us to leverage the LSTM based variants of deep recurrent neural networks to model our learning task, since LSTM networks have been empirically demonstrated to be able to detect long-term correlations in sequences [24]. Allowing our LSTM based formulation to learn potential long-term correlations in `(activity, duration)` pairs mitigates some of the limited data problems in other prediction approaches like hidden Markov modeling. We develop two different LSTM network architectures. One of those, called Hybrid LSTM model (see Section 4.1), simultaneously learns the probability distribution of next activity as well as estimates the duration for each possible activity. The same neural network is used for simultaneously predicting both the activity and duration. We design a custom loss function for the model to minimize that allows it to learn such distribution. The other LSTM architecture, called Cascaded LSTM model (see Section 4.2) predicts the next activity using a neural network and subsequently predicts its duration using a separate model. The two types of possible data sources, *viz.* individual data across multiple days and/or one-day long datasets across different individuals can be utilized simultaneously in the proposed approach and their mathematical treatment is identical in terms of formulating a loss function for training the LSTM networks. Experimentally, we verified that our LSTM models trained on data across different individuals from the same segment (as classified by the American Time Use Survey [37] dataset) resulted in good prediction accuracies for future activities and durations for a new individual in the same segment.

A further pleasantly surprising outcome from our experiments is that our learned models are extremely lightweight (not exceeding 500KB) and thus can be deployed on low-power or power constrained mobile devices with no dependency on the cloud. They can execute in real time to make activity predictions on mobile devices and thus significantly help mobile personal assistant software by providing activity context inputs (see Section 9 for some application scenarios).

Finally, the next activity prediction approach proposed by us, can also be naturally applied to the task of daily-routine generation. By sequentially predicting activities and their durations, we illustrate how a complete day of the user can be generated. This routine generation ability opens up avenues for applications like context-aware scheduling of actions by virtual assistants.

In summary, this paper makes the following contributions:

(1) We address the problem of jointly predicting future activities and durations using a deep learning based approach. We propose two new LSTM based architectures (in Section 4) and we compare their performance with models used previously in literature for technically similar problems (in Section 8).
(2) We show the application of our proposed model architecture in learning from two real-world datasets. One dataset consists of 7 individuals with multiple days' activity data for each individual. The other dataset is the American Time Use Survey (ATUS) [37] dataset containing data from many different individuals belonging to a demographically similar segment of population, with a single day's data for each individual. Our method achieves increased prediction accuracy compared to all baselines in both scenarios. This suggests that activity patterns learned from a segment of population can be used to tackle the *cold-start* problem often encountered in human activity prediction tasks.
(3) We quantitatively measure the effectiveness of LSTM in learning correlations between activities of a subject that occur far apart in time (more details in Section 8.1). For example, if a person dropped her kids to school in the morning, she would pick them up in the afternoon. Using its hidden cell state, LSTM is capable of propagating information forward in time indefinitely. Baseline sequence prediction models based on Hidden Markov Models (HMMs) and Sequence Matching were not able to encode long-term dependencies due to their assumption that the current activity is dependent only on a small finite history.
(4) We demonstrate how our model can be applied to generate a sequence of future activities for an entire day. Further we illustrate the robustness of our model in case of any unusual occurrence during the day.

For example, a user decides to go home early on a workday. Our model generates rest of the day which consists of activities done at home.

The rest of this paper is organized as follows. In Section 2, we describe how our work fits into existing literature. Next, we formally state our problem statement and notational conventions in Section 3. In Section 4 we describe our proposed LSTM models for activity and duration prediction. After that, in Section 5 we describe the datasets on which we applied our work. Further, in Section 6 we state comparison baselines for activity and duration prediction. Section 7 and 8 describe some of the interesting results we obtained. We discuss some practical application scenarios for our method and directions for future work in Section 9. Finally, we finish with our conclusions in Section 10.

2 RELATED WORK

2.1 Context Inference

Mobile devices carry lot of sensors and remain with the users for most part of the day. This makes them ideal for monitoring users' context like their location, physical state and social environment [14, 26, 51]. There has been considerable focus on the task of context inference from sensor data [31, 41, 50] in recent years. Castro et al. [8] looked at the problem of inferring current activity from human egocentric images. However, these papers address only the problem of instantaneous recognition of context from raw data.

2.2 Context Prediction

Context prediction enables various application scenarios like rendering virtual assistants and context-aware systems to pro-actively take actions by anticipating future context of the users. There is a series of prior work in the domain of next location prediction [45] and mobility pattern mining [33]. These papers characterize users' context in terms of their locations. However, in our work, we define context as the daily activity that a user is currently involved in, such as eating, commuting, working, *etc.* as it can capture rich contextual information of users' behavior. In this work we focus on predicting future activities of the users given their past behavior. Moreover, we also estimate the duration of predicted activities to be able to sequentially predict far-out in the future.

Prior work done in the domain of location prediction uses Markovian methods [3, 4]. However, they are limited by their inability to look back in time due to the Markovian assumption according to which the state at time t is dependent only on the state at time $t-1$. Higher order Markov Models have been known to suffer from high state-space complexity, reduced coverage, and sometimes even low prediction accuracy [11].

Sequence mining is another popular method used for technically similar problems [7]. This method helps to discover frequent sequential patterns in the data which can be exploited for predicting future elements of the sequence. These methods are computationally less complex than HMMs. But theoretically these models do not have the ability to model long-term dependencies in the data. We implement an alignment based pattern matching technique proposed by Sigg et al. [45] for our task. We also use a baseline model which uses a probabilistic sequence matching technique adapted to our task (explained in Section 6.4), and compare its performance with the other models.

Recently, recurrent neural network based models have been used for many sequence prediction tasks [5]. Notably, their Long Short-Term Memory architecture [22] has been more successful in solving various sequence analysis problems [19, 24]. Unlike HMMs which are modeled on the Markovian assumption and have a finite number of hidden states, LSTM have the advantage of having a continuous space memory which theoretically allows it to base its predictions on arbitrarily long past observations. In this paper, for the task of next activity prediction, we propose two Long Short-Term Memory network based models for joint prediction of next activity as well as its duration.

2.3 Daily-Routine Modeling

We further consider the problem of modeling the complete routine of users. This will give a more holistic view of the typical pattern of how they spend their time during the day. Eagle and Pentland [13] have given a technique for generation of a whole day of activities at once, without intermediate sequential prediction of the next activities. They rely on dimensionality reduction methods to encode a day as a linear combination of a definite number of characteristic vectors or *eigenbehaviors* representing the behavioral structure in the data. While this method is good to generate realistic-looking days, we believe that sequential prediction of activities allows the model to be more responsive to the activities that have happened so far. Instead of restricting a day to a linear combination of certain types of days, building it sequentially with activities which follow naturally from the previous observed activities allows the model to be more diverse and robust.

LSTM networks have also been applied for generation task mostly in the domain of text [20], speech [43] and music [9]. Therefore, we illustrate the application of our LSTM based models for routine generation task. We sequentially predict one activity after the other to generate a complete day. We also discuss the flexibility of this approach in anomalous scenarios (see Section 8.4).

3 PROBLEM DESCRIPTION

Let us first introduce some notation to formally describe our problem setup. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ denote the set of all unique activities in the dataset. Consider all activities that take place within a single 24-hour period. Each such activity is naturally associated with a starting time and a duration taken to perform that activity. We assume the starting time for any activity as rounded to the nearest hour, *i.e.* belonging to the set $\{0, 1, \ldots, 23\}$. Further, the duration will always be expressed in minutes, but for simplicity we will talk about the duration as if its domain is $\mathbb{R}^+$.

We define a state to be a triplet represented as (a, d, t) where $a \in \mathcal{A}$ is an activity, $t \in \{0, 1, \ldots, 23\}$ is the time within the 24-hour period when the activity a commenced (rounded to the nearest hour), and $d \in \mathbb{R}^+$ is the duration for which the activity lasted. Thus, a change in the activity being performed will create a new state and vice versa. At any given time of the 24-hour period in question, there will be certain activities that would have been performed since the start of that 24-hour period. These activities will manifest as different states according to our definition for states. We denote the i^{th} state ($i \geq 1$) by s_i and the corresponding activity creating that state by $A_i \in \mathcal{A}$. Let $D_i \in \mathbb{R}^+$ denote the duration and $T_i \in \{0, 1, \ldots, 23\}$ denote the starting time of activity A_i. Further, for any integer $m \geq 1$ and any timestamp t between 0000 and 2359 hours, let us define the ordered list $\boldsymbol{l}_{m,t}$ as

$$\boldsymbol{l}_{m,t} = \begin{cases} [s_1, s_2, \ldots, s_n], & \text{if } T_n \leq t < T_{n+1} \text{ and } n < m, \\ \left[s_{n-(m-1)}, s_{n-(m-2)}, \ldots, s_n\right], & \text{if } T_n \leq t < T_{n+1} \text{ and } n \geq m. \end{cases} \tag{1}$$

Hence, $\boldsymbol{l}_{m,t}$ consists of the ordered sequence of the last m states $s_{n-(m-1)}, \ldots, s_n$ if at least m states have occurred during the day up to timestamp t, else it consists of the ordered sequence of all states $s_1, \ldots, s_n$ that have occurred during the day up to timestamp t.

Formally our problem can be stated as follows: *Given the list $\boldsymbol{l}_{m,t}$ of past states that occurred during a given 24-hour period (with s_n being the latest state in $\boldsymbol{l}_{m,t}$), predict the next activity A_{n+1} (i.e. the $(n+1)^{th}$ activity) and its duration D_{n+1}.*

4 LSTM BASED MODELS

Owing to desirable properties exhibited by LSTM models for learning sequences [24], we develop two different LSTM based models in this section. We designate these as *hybrid LSTM model* (predicts both the next activity

and its duration jointly) and *cascaded LSTM model* (only predicts next activity and the duration is predicted by a different regression step).

4.1 Hybrid LSTM model

This model predicts both the next activity and its duration by attempting to simultaneously minimize the corresponding prediction errors. The error for activity prediction is captured by the categorical cross-entropy metric, whereas the error in duration prediction is captured by squared difference between predicted and observed durations.

- **Modeling the joint distribution of activity and duration**: Let A_{n+1} and D_{n+1} be as defined in Section 3 and let $\boldsymbol{L}_{m,t}$ denote the random variable representing the past m states if at least m states have occurred during the day up to timestamp t, else it consists of all states that have occurred during the day up to timestamp t. This definition is analogous to how $\boldsymbol{l}_{m,t}$ is defined by (1) and $\{\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}\}$ will constitute the observed realization of $\boldsymbol{L}_{m,t}$. We wish to model the joint distribution of the categorical random variable A_{n+1} and the continuous random variable D_{n+1} conditioned on the state sequence realization $\{\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}\}$. Notationally, for any $a \in \mathcal{A}$ and $d \in \mathbb{R}^+$, we assume
$$\Pr(A_{n+1} = a, D_{n+1} \in (d, d + \mathrm{d}\varepsilon] | \boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}) = p(a|\boldsymbol{l}_{m,t}) q(d|a, \boldsymbol{l}_{m,t}) \mathrm{d}\varepsilon \tag{2}$$
where $p(\cdot|\boldsymbol{l}_{m,t})$ is the conditional probability mass function of A_{n+1} given $\{\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}\}$ and $q(\cdot|a, \boldsymbol{l}_{m,t})$ is the conditional density function of D_{n+1} given $\{A_{n+1} = a\}$ and $\{\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}\}$.
We model $q(\cdot|a, \boldsymbol{l}_{m,t})$ to be normally distributed with variance σ^2 (independent of the activity A_{n+1}) and mean $\bar{d}_a$ that depends on the realization $A_{n+1} = a$ and the observed sequence of states $\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t}$. Thus, for each $1 \leq i \leq |\mathcal{A}|$, $\bar{d}_{a_i}$ is the mean for the distribution $q(\cdot|a_i, \boldsymbol{l}_{m,t})$ and we use the shorthand $\boldsymbol{d} \triangleq \left(\bar{d}_{a_1}, \bar{d}_{a_2}, \ldots, \bar{d}_{a_{|\mathcal{A}|}}\right)$ to refer to the collection of mean durations. We use the shorthand $\bar{p}_{a_i} \triangleq p(a_i|\boldsymbol{l}_{m,t})$ for each $1 \leq i \leq |\mathcal{A}|$ and refer to the collection of these probabilities by $\boldsymbol{p} \triangleq \left(\bar{p}_{a_1}, \bar{p}_{a_2}, \ldots, \bar{p}_{a_{|\mathcal{A}|}}\right)$.
- **Loss Function Derivation**: After training, we want the hybrid LSTM network to generate good estimates for the probability vector $\boldsymbol{p}$ and the mean durations vector $\boldsymbol{d}$ given an observed sequence $\boldsymbol{l}_{m,t}$ of past states. Thus, the hybrid LSTM network needs to learn the functional dependency of $\boldsymbol{p}$ and $\boldsymbol{d}$ on the state history $\boldsymbol{l}_{m,t}$. Let these functional dependencies be denoted as $\boldsymbol{p} = f(\boldsymbol{l}_{m,t}|\theta_1)$ and $\boldsymbol{d} = g(\boldsymbol{l}_{m,t}|\theta_2)$. The functions $f(\cdot|\theta_1)$ and $g(\cdot|\theta_2)$ will be realized by the hybrid LSTM network with $\Theta = (\theta_1, \theta_2)$ denoting the set of unknown weight parameters of the network that need to be learned.
To derive a loss function *w.r.t.* Θ, we compute the log-likelihood function *w.r.t.* the unknown parameters of our joint distribution model for activity and duration. Let $A_{n+1} = a$ be the $(n+1)^{\text{th}}$ observed activity and $D_{n+1} = d$ be its observed duration. Also assume that σ is a fixed parameter and $\boldsymbol{l}_{m,t}$ is the observed state sequence. Using (2), the log-likelihood function *w.r.t.* $(\boldsymbol{p}, \boldsymbol{d})$ is given by
$$\begin{aligned} \log L(\boldsymbol{p}, \boldsymbol{d}|a, d, \sigma) &= \log p(a|\boldsymbol{l}_{m,t}) + \log q(d|a, \boldsymbol{l}_{m,t}) + \log \mathrm{d}\varepsilon \\ &= \log \bar{p}_a - \frac{\left(d - \bar{d}_a\right)^2}{2\sigma^2} - \log \sigma + \log \frac{1}{\sqrt{2\pi}} + \log \mathrm{d}\varepsilon. \end{aligned} \tag{3}$$
Since log-likelihoods are functions to be optimized *w.r.t.* the unknown parameters, we can drop the terms that are independent of $(\boldsymbol{p}, \boldsymbol{d})$. Further, capturing the dependency of $\boldsymbol{p}$ and $\boldsymbol{d}$ on Θ explicitly, we have the equivalent log-likelihood expression
$$\log L'(\Theta) = \log L(\boldsymbol{p}, \boldsymbol{d}|a, d, \sigma) = \log f_a(\boldsymbol{l}_{m,t}|\theta_1) - \frac{1}{2\sigma^2}\left(d - g_a(\boldsymbol{l}_{m,t}|\theta_2)\right)^2 \tag{4}$$
where $f_a(\boldsymbol{l}_{m,t}|\theta_1) = \bar{p}_a$ and $g_a(\boldsymbol{l}_{m,t}|\theta_2) = \bar{d}_a$.

Without loss of generality, let us assume that there are K distinct days in an individual dataset and the i^{th} day had N_i states for $1 \leq i \leq K$. Further, let $\boldsymbol{l}^i_{m,t}$ denote the realization of the state sequence random variable $\boldsymbol{L}_{m,t}$ on the i^{th} day, and let A^i_n and D^i_n respectively denote the n^{th} activity on the i^{th} day and its corresponding duration. We formulate the training loss function $\mathcal{L}(\Theta)$ as the negative of the average over all activities across all days of the log-likelihood expression in (4), and is given by

$$\mathcal{L}(\Theta) = \sum_{i=1}^{K} \sum_{n=1}^{N_i} \Big[-\log f_{A^i_n}\left(\boldsymbol{l}^i_{m-1,T_m} \middle| \theta_1\right) + \alpha \Big(D^i_n - g_{A^i_n}\left(\boldsymbol{l}^i_{m-1,T_m} \middle| \theta_2\right) \Big) \Big], \text{ where, } \alpha = \frac{1}{2\sigma^2} \tag{5}$$

Now considering that we have a one-day long data per individual across different individuals, rather than a multi-day data for the same individual, we shall have a loss function formulation that is essentially same as (5). To see this, we need to assign a different semantic meaning to the variables described above. Specifically, we shall assume that K represents the number of individuals in the dataset, the variables $\boldsymbol{l}^i_{m,t}$, A^i_n and D^i_n are defined *w.r.t.* the i^{th} individual, and that the loss function is arrived at by averaging over all activities across all individuals in the dataset. This results in the same loss function expression as in (5). For the purpose of experiments, α is taken to be a hyper-parameter that is optimized to different datasets (see Section 7.1) in order to balance between errors from activity and duration predictions. The training proceeds through standard mini-batch gradient descent method [21] *w.r.t.* Θ to minimizes the loss function $\mathcal{L}(\Theta)$.

- **Architecture**: This is illustrated in Figure 1 and described below.

(1) **Input Layer:** The hybrid LSTM model takes as input two sequences:
   - The first sequence consists of `(activity, duration)` tuples. The activity is encoded as a one-hot vector. The duration is converted into a vector by multiplying it with the one-hot activity vector.
   - The second sequence is of the same length as the first and encodes the starting time of the activity from the first input. This encoding is in the form of a one-hot vector of size 24 with the corresponding hour of day set to 1. The input sequences are concatenated for feeding into the next layer.

(2) **LSTM Layer:** The encoded input then goes through an LSTM layer which has a hidden state that stores historical information and is carried forward to subsequent time-steps.

(3) **Activated Dense Layer:** The output of the LSTM is fed into a Dense layer which outputs a vector of length $2|\mathcal{A}|$ at each time step. We apply a softmax activation [12] over the first $|\mathcal{A}|$ elements which represent the probabilities assigned to each of the $|\mathcal{A}|$ activities. The next $|\mathcal{A}|$ elements have a rectified linear unit (ReLU) activation [35] applied on them. They represent the expected duration for each of the $|\mathcal{A}|$ activities respectively, if they indeed occur at the current time step.

(4) **Reshape Layer:** This layer reshapes the output to group together the predicted probability and estimated duration for each of the activities, thereby converting the output to a list of tuples.

(5) **Output Layer:** At each timestep, the layer outputs a list of `(probability, duration)` tuples - one for each of the activities in our vocabulary $\mathcal{A}$. The `probability` value represents the probability that the corresponding activity will occur next, with expected duration value given by `duration`. Since this prediction is made at each timestep, the output is a temporal sequence of predictions of the same length as the input.

For a sequence of activities observed until time T, the LSTM model returns a `(probability, duration)` tuple for each of the activities in our vocabulary $\mathcal{A}$. The `probability` value for an activity is the estimated probability of its occurrence as the next observed activity at time T. The accompanying `duration` is the expected duration of that activity conditioned on its occurrence at time T. While training, the loss is effectively computed for each prefix subsequence of the complete sequence of states within a day and thus the learning task could be interpreted as one of sequence to sequence learning.

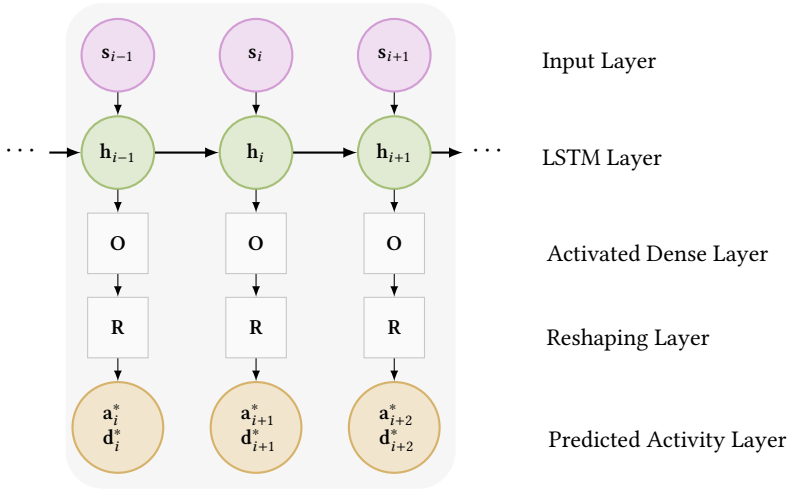


Fig. 1. Hybrid LSTM model architecture

4.2 Cascaded LSTM model

We carried out an experiment by trying to predict the duration D_{n+1} conditioned on the activity A_{n+1}, the time of day T_{n+1} and the history of past activities $\boldsymbol{L}_{m,t}$. Our results showed that no added advantage in terms of prediction accuracy was seen by considering the past activities. That is, $\Pr(D_{n+1}|A_{n+1}, T_{n+1}, \boldsymbol{L}_{m,t}) \approx \Pr(D_{n+1}|A_{n+1}, T_{n+1})$. In fact, the variation of prediction accuracy with $\boldsymbol{L}_{m,t}$ revealed a slightly negative correlation. This led us to conclude that previous activities and their durations can be ignored altogether for the duration prediction task. The complexity of the task will greatly reduce, with little compromise on the accuracy.

Cascaded Modeling: Based on this observation, we built another model that learns the probability distribution of the next activity and subsequently its duration, rather than as a joint distribution. It is a two-level cascaded model. First, the probability distribution of the next activity given the list of past states $\Pr(A_{n+1}|\boldsymbol{L}_{m,t}) = p(A_{n+1})$ is learned and then the distribution of duration given the next activity and its start-time $\Pr(D_{n+1}|A_{n+1}, T_{n+1}) = q(D_{n+1})$ is learned.

- **Activity Modeling:** Let the observed next activity be some $a_k \in \mathcal{A}$. Now, the probability of observing a_k is $p(A_{n+1} = a_k)$. Let the set of parameters of the stochastic process dictating the distribution of activities be θ_a. Now, the likelihood of observing a_k can be represented as the following function:

$$L(\theta_a|a_k) = p(a_k|\theta_a) \tag{6}$$

and the log-likelihood as:

$$\log L(\theta_a|a_k) = \log p(a_k|\theta_a) \tag{7}$$

We define an estimation function as follows to model the probability p:

$$f_{\theta_a}(\boldsymbol{l}_{m,t}) = p(A_{n+1}) \tag{8}$$

This function takes the realization $(\boldsymbol{L}_{m,t} = \boldsymbol{l}_{m,t})$ as input and outputs a vector $\bar{p} = \left(\bar{p}_{a_1}, \bar{p}_{a_2}, \ldots, \bar{p}_{a_{|\mathcal{A}|}}\right)$ where $\bar{p}_{a_k}$ denotes the estimated probability of A_{n+1} being a_k. The log-likelihood of observing a_k as A_{n+1}

according to this estimation function becomes:

$$\begin{aligned} \log L(\theta_a|a_k) &= \log \bar{p}_{a_k} \\ \log L(\theta_a|a_k) &= \langle \log \bar{p}, I_k \rangle \\ \log L(\theta_a|a_k) &= \langle \log f_{\theta_a}(\boldsymbol{l}_{m,t}), I_k \rangle \end{aligned} \tag{9}$$

where I_k is a one hot vector of size $|\mathcal{A}|$ with k^{th} element being 1 and all others being 0. Now, the objective is to find the optimal θ_a for the estimation function f (which in our case will be simulated by a neural network architecture) so as to maximize the log-likelihood of the observation a_k. Therefore we define the loss (or objective function) for our neural network training as negative of log-likelihood:

$$-\log L(\theta_a|a_k) = -\langle \log f_{\theta_a}(\boldsymbol{l}_{m,t}), I_k \rangle \tag{10}$$

As described in the previous model, summing this loss over a dataset with K days containing $N_1, N_2, ...N_K$ states respectively, we get the final loss function as:

$$\mathcal{L}_a(\theta_a) = \sum_{i=1}^{K} \sum_{n=1}^{K} -\left\langle \log f_{\theta_a}\left(\boldsymbol{l}^i_{m-1,T_m}\right), I^i_{k_n} \right\rangle \tag{11}$$

- **Duration Modeling:** Let the duration estimation function be represented as:

$$f_{\theta_d}(A_{n+1}, T_{n+1}) \tag{12}$$

Let the observed duration for an observed next state, $(A_{n+1} = a_{n+1})$ and $(T_{n+1} = t_{n+1})$ be $d \in \mathcal{R}^+$. Now the objective is to minimize a loss function defined on f_{θ_d} and d. A common choice of loss function in regression tasks is the squared loss. So we define the loss function as:

$$L(\theta_d|d) = \left(f_{\theta_d}(a_{n+1}, t_{n+1}) - d\right)^2 \tag{13}$$

Again, summing this loss over a dataset with K days containing $N_1, N_2, ...N_K$ states respectively, we get the final loss function as:

$$\mathcal{L}_d(\theta_d) = \sum_{i=1}^{K} \sum_{n=1}^{N_i} \left(f_{\theta_d}\left(a^i_n, t^i_n\right) - d^i_n\right)^2 \tag{14}$$

We model f_{θ_d} as a non-linear regression model.

Architecture:

The architecture of the activity prediction model is described in detail below:

(1) **Input Layer:** The activity model input is in the form of a temporal sequence consisting of `(activity, end_hour, duration)` tuples. Here, `activity` and `end_hour` both are encoded in one-hot vector format.
(2) **Encoding Layer:** The input first goes through a fully connected hidden layer (dense layer) which encodes the input data at each timestep into a vector of lower dimensionality than the input.
(3) **LSTM Layer:** The encoded input then goes through an LSTM layer which has a hidden state that stores historical information and is carried forward to subsequent time-steps.
(4) **Dropout Layer:** The output of the LSTM layer is fed through a dropout layer [47] which prevents the model from over-fitting on the training data by setting the activations of a certain fraction of neurons (called the dropout rate) to zero.
(5) **Output Layer:** Finally the output of the dropout layer goes through a fully connected layer with a softmax activation. The output of model is also a temporal sequence of the same length as input which consists of vectors with probability distribution of next activity at each time step.

(6) **Target:** The target vector is of the same size as the output probability vector. It consists of true `next_activity` elements in one-hot vector format. The loss, $\mathcal{L}_a$ is calculated as the categorical cross-entropy between the predicted probabilities and the target.

The duration prediction model takes the next activity predicted by the activity model along with the start time of the next activity as input. The input is fed through a stacked ensemble regressor model that outputs a vector with duration value of the activity. A stacked ensemble regressor is built by fitting multiple shallow regressor models on the data. The outputs from these regressors are then taken as input to train a meta regressor. The regressors used in the models are:

- Decision tree [42]
- Random forests [27]
- Support Vector Machine (with RBF kernel) [49]
- Gradient Boosting [17]
- Bayesian Ridge [23]

The architecture of the combined cascaded prediction model is illustrated in Figure 2 .

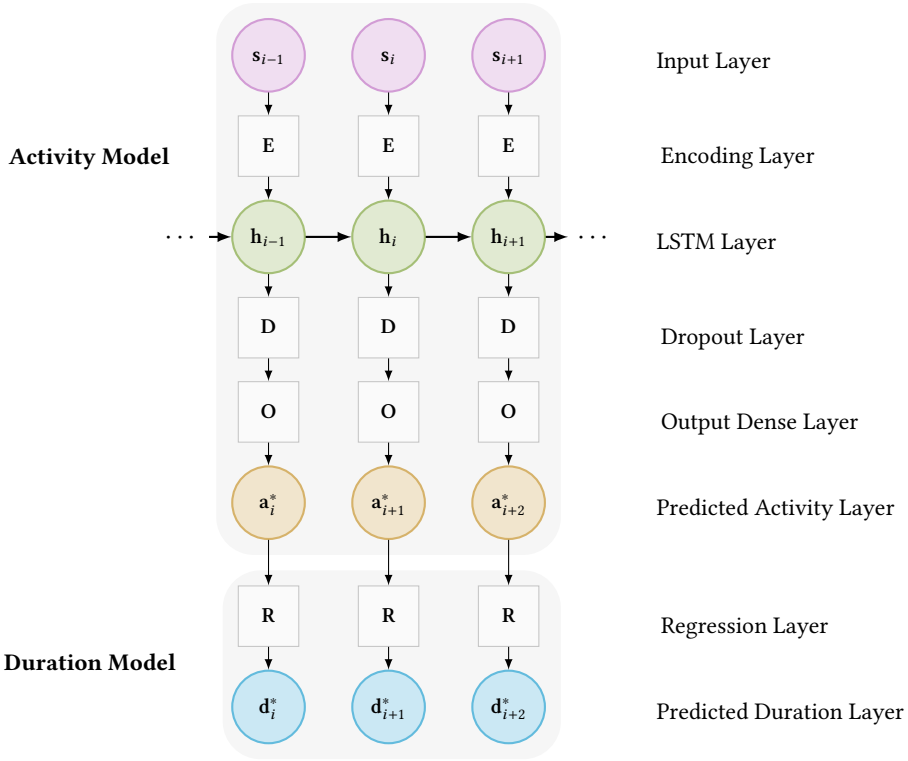


Fig. 2. Cascaded LSTM model

5 DATA DESCRIPTION

Some patterns of our daily life activities can be attributed to our demography (job, age, geography, *etc.*). While some others are unique to us (hobbies, exercise routines, *etc.*). So, we study the problem of activity prediction and

routine modeling using two datasets, one based on individual routine of seven subjects who are researchers by profession, recorded for a month each and the other based on routines recorded from specific segments of the American population.

5.1 Individual datasets

For collecting the individual datasets, we set up an experience sampling [25] field experiment. We built an activity tagging mobile application named ContextTagger[1]. The app prompts the subject to periodically record her activities. We picked a set of activities relevant to the daily lives of the subjects. There were 21 such activities as sleeping, personal care (grooming), commute, eating, work, *etc.* The app was designed to require minimal efforts from the subject to periodically tag her daily activities. The app has a time-line where the user can add activities. To add an activity, the user can tap-hold and drag to select a period, and then select the activity performed during this duration from a prompt. Figure 3 illustrates the process and the complete work-flow is described in Figure 4. The app also has the following features:

- Periodic notifications to the user to label their current activity. The notifications can be turned off and on by the user.
- Instant sharing of the activity history in CSV format through standard sharing apps like email-clients or instant messaging apps.
- Functionality to add and delete past activities in case the user forgets to log or wants to modify any activities.

The Experience Sampling Method (ESM) frequency [25] is set to 1 hour, which means that the notifications are pushed after every hour. If the participants forgot to annotate some activity, they had the option of tagging it later by going back in the embedded timeline of the app (see Figure 3a). While tagging the data, it is natural for some blank space to be left between two tagged activities because selecting an accurate time period by touching and dragging is difficult. While preprocessing the data from the app, for cases where the untagged duration between two activities is less than 15 minutes, we assume that they are contiguous. We extend the durations of the two activities so that each of them occupy half of the untagged duration in between. For cases where we encounter an untagged duration of greater than 15 minutes, we tag that entire duration with a new activity type called "Missing activity". The collective duration of all "Missing activity" occurrences is 0.23% of the entire duration of all activities in the entire collected individual dataset.

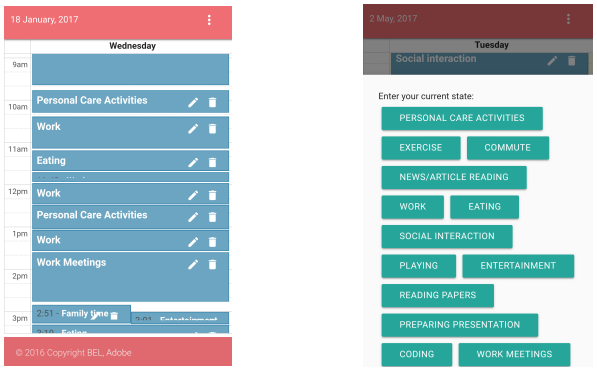
5.2 Segment level datasets

For segment level activity data we use American Time Use Survey [37] Dataset. This dataset collects information on how people living in the United States spend their time. This multi-year data set contains data on the amount of time that people spent doing various activities in the years 2003 through 2015, such as paid work, child care, religious activities, volunteering, and socializing *etc.* The dataset contains 104 such activities. The data has been collected about the sequence of activities perform on a particular day by an individual. There are approximately 170,000 unique participants in the dataset.

5.3 Statistical summary

We finally create 11 datasets in all. 7 of them come from each of the subjects from whom individual data was collected. For the remaining 4 datasets, we picked 4 segments of population from the ATUS data based on their occupation. We select these 4 segments based on two criteria. First is total number of tagged days and the other is variance in daily-activity pattern among subjects within the segment. All 4 segments contain sufficient number of tagged days (greater than 200). Two segments, *i.e.* food service managers and security guards show low variation

[1]The app can be installed from Google Play-store.



(a) Activity Timeline (b) Activity label prompt

Fig. 3. Screenshots of ContextTagger app

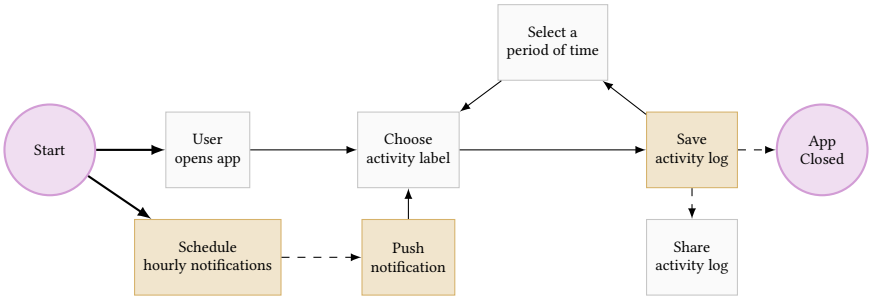


Fig. 4. ContextTagger app workflow

in daily activity pattern across subjects. While the other two segments, *i.e.* real estate agents and other teachers (teachers who don't teach at preschool to postsecondary levels, and are also not special education teachers) show high variation in their activity patterns within segments. To capture the variation in activity patterns of the segments, average standard deviation of the time spent on different activities per day ($\bar{\sigma}$) was calculated:

$$\sigma_a = \sqrt{\frac{1}{N_s - 1} \sum_{i=1}^{N_s} (t_{ai} - \mu_a)^2}$$
$$\bar{\sigma} = \frac{1}{N_a} \sum_{j=1}^{N_a} \sigma_{a_j} \tag{15}$$

where,

- N_s is the number of subjects in segment s,
- t_{ai} is the time spent by subject i on activity a in her tagged day,
- μ_a is the average time spent by all subjects of segment s on activity a,
- σ_a is the standard deviation of the time spent on activity a per day by segment s,
- N_a is the number of unique activities, and

Table 1. Aggregate statistics for different segments in the dataset

Dataset segment	# subjects	# tagged days	# unique activities	Avg. activities/day	$\bar{\sigma}$
Researchers	7	149	22	19.07	–
Real estate agents	653	653	84	20.29	0.59
Other teachers	670	670	87	21.86	0.59
Security guards	209	209	65	17.28	0.26
Food service managers	110	110	58	16.65	0.37

(a) Researchers (b) Real Estate Agents (c) Other Teachers (d) Security Guards (e) Food Service Managers

Fig. 5. Percentage of Time Spent on Activities by Different Segments

- $\bar{\sigma}$ is the variation in activity pattern of segment s.

The statistics for the dataset are presented in Table 1. The researcher segment has data from 7 people who are researchers or research interns by profession have each tagged between 14 to 41 of their days. The rest of the four segments are people with the corresponding occupation from ATUS dataset. For each of the 5 segments, the distribution of time spend on different activities is shown in Figure 5. Most of the days in the individual dataset are working days with commute to and from office. The observed working hours are approximately from 1000 to 2000 hours, with lunch between 1200 to 1400 hours, with variations across people as well as days. Social interaction and exercise usually occur more in the evening around 1800 hours.

5.4 Data pre-processing

We pre-process all datasets and encode them in a standard format. We treat every tagged data-point as a 4-tuple of the form (a, d, T_s, T_e). Here, a is the activity, d is the duration in minutes, T_s is the start time of the activity (in hours) and T_e is the end time of the activity (again, in hours). The final datasets are in the form of lists of these tuples on each day. For each dataset, activity codes belong to a predefined set of unique activities. We add an extra sentinel code to the list for denoting *End of the day*. This is added to distinguish between consecutive days. No overlaps were allowed between activities in any of the datasets. We added a placeholder activity called *Nothing* in time slots where no activity was tagged. So, any instance exactly one activity was tagged in both datasets.

6 COMPARISON BASELINES FOR ACTIVITY AND DURATION PREDICTION

Drawing from traditional modeling approaches, we state some simple baseline models (not based on deep learning) for activity and duration prediction. In particular, we have compared the proposed LSTM based approach against multinomial and hierarchical HMMs, time of day based classification, and probabilistic sequence matching. We

provide very brief descriptions below while deferring more detailed explanations to the supplementary material in Section 11.

6.1 Activity Prediction: Multinomial Hidden Markov Model (MHMM)

Considering the sequential nature of activity prediction, one model family of relevance is the HMM. Since the observables in our setup are activities that are discrete in nature, we use a Multinomial HMM [32] variant as a baseline to model the activity sequence and predict future activities. Further details on how we have used MHMM are in Section 11.1.

6.2 Activity Prediction: Hierarchical Hidden Markov Model (HHMM)

An HHMM consists of a hierarchy of multiple nested MHMMs. We implemented a HHMM with depth 2 inspired by its prior usage by Baratchi et al. [4]. Further design and implementation details are in Section 11.2.

6.3 Activity Prediction: Time Of Day (TOD) Model

Given that the activities performed by a person during a day could adhere to a routine, it makes sense to use the time of day as a feature during prediction. To test this hypothesis, we trained various standard classification models to predict activity given the time of day. While it is alright to expect that a person would do a particular activity around a certain time of day (for example, a given person might eat around noon each day), but it is too presumptuous to expect that this time would show very little variation across days. So, we decide to only use the hour of day as a predictor variable.

We tried popular multi-class classifiers mentioned below:

- Decision tree [42]
- Random forests [27]
- Support Vector Machine (with Radial Basis Function kernel) [49]
- Logistic Regression [44]

6.4 Activity Prediction: Probabilistic Sequence Matching (PSM)

This algorithm is described in detail under Section 11.3 with pseudo-code supplied as Algorithm 1. We only give a brief description below. This baseline utilizes a sequence mining approach for activity prediction and considers the time of day as an input feature in addition to the sequence of activities. Given an input seed sequence, this method performs a lookup on all sequences in the training dataset to search for occurrences of similar sequences. For any occurrence where the seed sequence matches, the next activity in that occurrence becomes a candidate for next-activity prediction. All such candidates are found and their frequency distribution is normalized to get a probability distribution. For example, consider the given seed sequence *(sleep, personal care, eating)*. The algorithm will lookup this sequence in the training days and find what was the next activity observed after this sequence. Let's say *commute* was found as the next activity in one of the matches. Then the *commute* activity becomes a candidate. If no matches are found, then the algorithm retries with smaller and smaller seeds until a match is found. The matching step is non-trivial since time of day constraints are enforced in addition to the activity sequence constraints (full details are in Section 11.3). In particular, all matches that do not lie close enough to the required prediction time of day are ignored.

6.5 Activity Prediction: Alignment Approach

In their paper, Sigg et al. [45] have outlined an alignment prediction approach to determine the next context in ubiquitous computing environments. Like the PSM idea in Section 6.4, the alignment approach is based on finding approximate matches between a seed sequence and the typical sequences seen in the training data,

followed by a determination of the next likely sequence element based on these matches. All the semi-global alignments between any two context sequences are computed using the Needleman-Wunsch algorithm [36]. We have implemented this method as a baseline for comparison with further details explained in Section 11.4 and pseudo-code provided as Algorithm 2.

6.6 Duration Prediction

Along with the next activity we are also interested in its duration. As a baseline estimate of the duration of next activity, we model it as a distribution conditioned on the activity and the time of day (at hourly granularity). The predicted duration for activity A at time of day T, denoted by $d^*(A, T)$, is taken to be the empirical mean of this conditional distribution computed from the training data samples as

$$d^*(A, T) = \frac{1}{n} \sum_{i=1}^{n} d_i(A, T') \tag{16}$$

where $d_i(A, T')$ is the duration of activity A at time of day T' for the i^{th} such instance in the training data, and there exist a total of n occurrences in the training data such that activity A occurred at time of day T' with $T - \delta \leq T' \leq T + \delta$ being satisfied.

7 EXPERIMENTAL SETUP

7.1 Model Selection and Hyper-Parameter Tuning

Each of the datasets is divided into separate training and test sets. For each individual in the individual datasets, we select random 20% of days as the test set and use the rest for training. Because the individual datasets are smaller in size, the results vary relatively more depending on the test-train partition. So, the random partition is done 5 times and the evaluation metrics are averaged for the models trained and evaluated according to those 5 partitions. Before training our models, a hold-out validation set is randomly separated out from the training dataset partition. To prevent overfitting, the models are trained until their perfomance on the validation set reaches a maximum. Similarly , for hyperparameters, we choose values which achieve the best performance on the validation set. A similar procedure is followed for the segment level datasets, except that we do the train-test split only once since the dataset is large enough.

For the PSM model (see Section 6.4), We tuned the values of parameters n and δ for each dataset such that the highest performance is observed on a hold-one-out validation sample. For all datasets, n lies between 3 and 5 and δ lies in the range of 2 to 4.

The LSTM networks (see Section 4) were implemented in Python using the TensorFlow library [1]. We used minibatches [21] while training, and RMSProp [48] algorithm for gradient descent. The duration was scaled down to range $[0, 1]$ while training to keep the input values small to facilitate stability while performing gradient descent.

For Hybrid LSTM model (see Section 4.1) α and hidden state size were hyper-parameters which were selected separately for each dataset based on performance on the validation set during training phase. Usually, the value of α chosen was 0.1 or 0.2, the number of hidden states was 20 or 40. Hybrid LSTM models for all datasets were trained for 256 epochs, although the performance on validation set reached a peak almost always within 100 epochs, before going down indicating overfitting. The model parameters with best performance on the validation set were selected, and then evaluated on the separate test set.

For the Cascaded LSTM Model (see Section 4.2), the hyper-parameters chosen (Encoding layer size, LSTM layer size and Dropout rate) for the different datasets by validation set accuracy based model selection are summarized in Table 2. Since the number of unique activities in the ATUS segment level datasets ($65 - 87$) is quite higher than individual datasets (22), the layer sizes chosen in the networks for segment level datasets is significantly higher

Table 2. Hyper-parameters of Cascaded LSTM model for different datasets

Hyper-parameter	Range (Individual Datasets)	Range (Segment Level Datasets)
Encoding Layer size	$100 - 300$	$400 - 700$
LSTM Layer size	$200 - 500$	$1000 - 3000$
Dropout Rate	$5\% - 60\%$	$50\% - 75\%$

than that for individual datasets. The batch for all datasets was chosen to be 1 day of activities. Models for all datasets were trained till the validation accuracy showed an upward trend. Models took ~ 30 epochs to train.

7.2 Evaluation metrics of next activity prediction

We have used the following metrics for evaluating our activity prediction models against the baseline models.

- **Top-K Categorical Accuracy:** In some cases, especially in the Probabilistic Sequence Matching model, we often have ties between predictions, *i.e.* more than one activities have the same predicted probability. In this case, we follow the following method to assign an accuracy score:
  Sort the predicted activities in descending order of their probabilities. Call this sequence S. Let the first K activities in this sorted order be represented by $M = S[1 \dots K]$. The smallest predicted probability amongst all activities in M is p_0 and n_1 elements in M have probability equal to p_0. Let the number of activities in S which have probability equal to p_0 be called n_2. Let the observed activity be a. We assign a score as follows:
  (1) If a has predicted probability strictly greater than p_0, the score given is 1.
  (2) If a has predicted probability exactly equal to p_0, the score given is $\frac{n_1}{n_2}$.
  (3) If a has predicted probability strictly less than p_0, the score given is 0.
- **Categorical Cross-Entropy (CC):**

$$\text{CC} = \frac{1}{n} \sum_{i=1}^{n} -\log p_{y_i} \tag{17}$$

  where, p_{y_i} is the predicted probability of observing the true value y_i. Categorical cross-entropy is a measure of how bad is the predicted probability distribution in following the actual probability distribution. It gives higher value for more uniform probability distributions with low confidence.
- **Confusion Matrix:** Figure 7 shows the confusion in activity prediction for Cascaded LSTM for one of the subjects in the individual dataset. Looking at the confusion matrix gives finer insights about the predictions made by a model than other metrics. This is because the confusion matrix shows the accuracy and confidence of prediction for every activity separately.

7.3 Evaluation metrics for duration prediction

Let d_i^* and d_i be the predicted and actual durations for the i^{th} activity, respectively. We use several different metrics for comparing the duration prediction models:

- **Mean Squared Error (MSE):** This metric forms part of the loss function on which the Hybrid and Cascaded models have been trained (see (5) and (14)). It is the average squared error between the predicted and actual durations of activities:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (d_i^* - d_i)^2. \tag{18}$$

- **Mean Absolute Error (MAE):** The MSE metric is known to overpenalize large duration errors because of the square term. This may lead to large average error even if there are only few significantly large duration prediction errors. The mean absolute error metric

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|d_i^* - d_i|, \tag{19}$$

  does not suffer from this shortcoming of the MSE metric and hence is important to better assess the overall statistical characteristics of the duration predictor.
- **Mean Absolute Percentage Error (MAPE):** Activities with longer durations tend to have larger absolute variations in duration. For example, a person might sleep for 6 to 8 hours, thereby displaying a variation of 2 hours for *sleeping*. However, it is unlikely for anyone to have a variation of 2 hours in the their eating duration (typically, *eating* lasts for less than 30 minutes). So, it may be worthwhile to study the duration prediction error as a fraction of the actual duration, and averaged over all activities. We use the MAPE metric for this purpose and it is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|d_i^* - d_i\right|}{d_i}. \tag{20}$$

8 RESULTS

8.1 Long Term Dependency Detection

Sometimes one may find dependencies between activities in a person's routine. These activities can even happen at times which are widely separated. For example, if a person has been to the gym in the morning, he would not go to the gym in the evening, and might spend his time elsewhere. In this section, we test whether an LSTM network is actually able to learn such long-term correlations, if they are present in the data. We designed a controlled experiment where we manually introduced such long-term dependencies into daily activity logs and tested if our LSTM based model can learn the correlation. Because we had more data at the segment-level, we decided to use that for this task.

We picked 2000 random people's days from the ATUS dataset and split them into a training set of size 1600 and a test set of size 400. We filtered people from both the sets so that for everyone in each of the sets, first activity of the day is sleeping. This reduced the size of the training dataset to 1467 days and that of the test dataset to 363 days. In both the training and testing set, the following modifications were made:

(1) For half of the set, the second activity of the day was set to A_{1s} and the last activity to A_{1l}.
(2) For the other half of the set, the second activity of the day was set to A_{2s} and the last activity to A_{2l}.

So, the last activity of any day is A_{1l} if and only if the second activity of the day is A_{1s}. Similarly, the last activity of any day is A_{2l} if and only if the second activity of the day is A_{2s}. Therein, we have introduced a synthetic long-term dependency. We then trained the PSM model and the LSTM model for the task of next activity prediction as explained earlier. Note that there was no change in the procedure of training to aid the model in learning these long-term dependencies.

The results are summarized in Table 3. The table presents the predicted probabilities for the last activity of the day conditioned on the second activity of the day given by PSM and Cascaded LSTM. We see that LSTM is able to learn the correct mapping from the second activity to the last activity of the day with much higher confidence than PSM which selects the last activity with almost equal probability irrespective of the second activity. As PSM decides its prediction by looking back at a chunk of the history of activities at a time, it is not expected that it would be able to pick up the dependencies between a specific individual activity in the past and an activity in

Table 3. Results of long term dependency experiment

Model	2^{nd} act.	Pr(last act. = A_{1l})	Pr(last act. = A_{2l})
PSM	A_{1s}	45.05%	54.39%
	A_{2s}	53.03%	46.96%
Cascaded LSTM	A_{1s}	77.47%	22.52%
	A_{2s}	13.81%	86.18%

future. It is also limited by its nature of being unable to look too far back in the past. LSTM on the other hand is well suited for learning such long-term relations and our experiment validates this hypothesis.

8.2 Next activity prediction

The next activity prediction results are presented in Figures 6 and 7.

- **Top-k Plot:** For Top-1 accuracy, we observe a consistent order of performance amongst the models tested. Cascaded LSTM performs the best while MHMM and HHMM are clearly the worst. Further, HHMMs tend to perform a little worse than MHMMs with very high categorical cross-entropy.
  For the TOD baseline model, the Random Forest Model works best for the individual level datasets, while decision trees perform the best for segment level datasets. TOD model outperforms MHMM in categorical accuracy metrics for all datasets. This supports our hypothesis that time of day is a strong predictor for next activity. This result is in line with previous findings in the literature. Castro et al. [8] have shown that accuracy of activity prediction increases after including feature capturing contextual meta-data such as time and day of week.
  Probabilistic Sequence Matching model outperforms both MHMM and TOD model on all categorical accuracy metrics which suggests that it learns repeating patterns in the data well and tends to pick up the most likely next activity through the frequency method. However, the probability distribution over the activity is far from that observed in the test data, as can be seen from categorical cross-entropy results.
  The alignment model performs better than the Probabilistic Sequence Matching method while also exhibiting low categorical cross-entropy. However, LSTM based models perform better than it on all datasets and metrics.
  While LSTM models do exhibit the best performance, the simple PSM method also performs quite good by just looking at the history without using any complicated functions like LSTM networks. However, their performance is relatively poor on the segment-level datasets. This suggests that most of the times people are quite predictable and you can use simple heuristics too (like PSM does) to predict a person's activities quite well, by just looking at his habits in the past. However, this benefit does not avail itself to PSM in case of segment-level data.
  The Top-2 and Top-3 accuracies seem to vary comparatively less across models for the individual datasets. However, for the segment-level datasets, the difference in performance between LSTM models and the baselines (except alignment model) increases. This might be because there are considerably more activities in the vocabulary for the segment-level dataset.
  The results on the four ATUS segments indicate that the relative results across the different algorithms is the same regardless of the segment considered. Given that two of our segments are more homogeneous and two less non-homogeneous (see Table 1), this indicates that the level of variation within a segment does not dictate the relative performance of the proposed algorithms.

Fig. 6. Next Activity Prediction Evaluation on Test Datasets

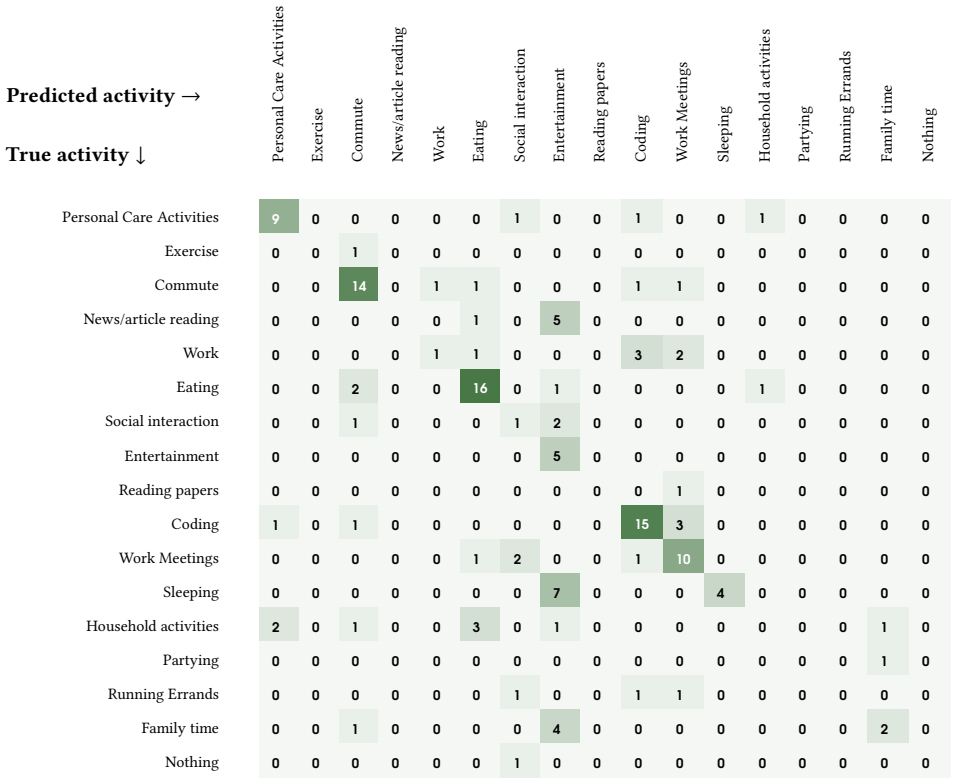
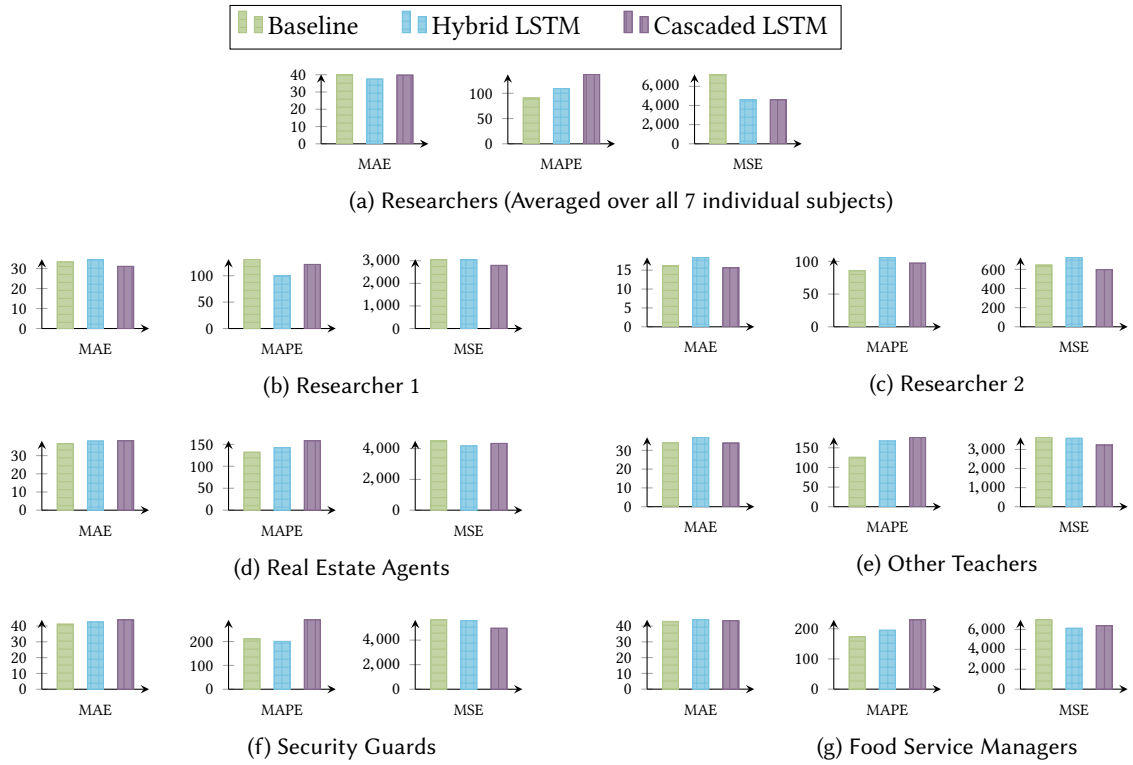


Fig. 7. Activity Prediction Confusion Matrix on Test Dataset- Cascaded LSTM Model

- **Categorical Cross Entropy Plot:** Results show that LSTM based models exhibit the lowest categorical cross-entropy. This can be because the analytical expression of categorical cross-entropy matches that of the loss function in (10) and this is explicitly minimized while training the LSTM models. While the probabilistic sequence matching does well on the top-categorical-accuracy metric, it exhibits a strikingly high categorical cross-entropy. This shows that although the topmost predicted activity often ends up being correct, the overall probability distribution over the predicted activities does not match well with the distribution exhibited by the ground truth. One would expect that the performance on Top-2 and Top-3 accuracy would be worse if the cross-entropy is high. We see that these metrics are indeed worse for PSM when compared to LSTM based models.
- **Confusion Matrix Plot:** Sometimes, it is more acceptable to be confused between certain activities than others. For example, it is difficult to say at a given time when the person is in office, whether she will code or be in a meeting. However, predicting that she would have lunch in the afternoon and commute to office in the morning is easier to predict. In PSM, the predictions are directly made by looking at the activity sequence of the training data. So, one would expect the confusions to be reasonable. However, it is not obvious that LSTM models, owing to its complex parametrized architecture, would exhibit justified confusions. We show that the confusions shown by our trained LSTM model are indeed justified. For example, often the model predicts that the subject would have some entertainment but she goes to sleep.



(a) Researchers (Averaged over all 7 individual subjects)

(b) Researcher 1

(c) Researcher 2

(d) Real Estate Agents

(e) Other Teachers

(f) Security Guards

(g) Food Service Managers

Fig. 8. Duration Prediction Evaluation on Test Datasets

This is possibly attributable to the time for entertainment at night often conflicts with time to go to sleep, so the model cannot be completely certain while choosing between these activities. Similarly, the model often predicts work meeting instead of coding, work, or reading papers, which is again reasonable.

The duration prediction results are presented in Figure 8. For the task of duration prediction, we could not find a clear trend in the performance of the models. It seems that duration prediction does not decidedly improve with more input in the form of more history. Even our model that has the complexity of LSTM networks could hardly perform better than the baseline model that predicts the average observed duration for the particular activity during that time of day. In our future work, we would further investigate the predictability of duration and experiment with different approaches for duration prediction.

8.3 Joint versus non-joint prediction

To see the difference between joint and non-joint prediction, we implemented variations of our models, to predict one duration value instead of different duration values conditioned on the activities. Mathematically, instead of modelling $p(d|a)$, we model $p(d)$. We took three models - Baseline, cascaded LSTM, and hybrid LSTM and made a non-joint variant for each of them which predicts a single value for the duration of next activity. Also, for the hybrid LSTM variants, we modified the loss function so that all the emphasis is on reducing the error on predicted durations.

We compared the quality of predicted durations for joint and non-joint variants for one individual in the researcher segment and two ATUS segments (Real Estate Agents and Food Service Managers), one with high

Table 4. MAE for duration prediction, averaged across activities for the researcher

Method	Joint Model	Non-Joint Model
Cascaded LSTM	21.75	32.58
Hybrid LSTM	21.86	52.20
Baseline	34.01	61.19

Table 5. MAE for duration prediction, averaged across activities for segment level datasets

Segment	Method	Joint Model	Non-Joint Model
Real Estate Agents	Cascaded LSTM	38.27	60.99
	Hybrid LSTM	38.49	52.73
	Baseline	36.54	104.66
Food Service Managers	Cascaded LSTM	43.30	78.89
	Hybrid LSTM	47.35	61.21
	Baseline	42.91	130.66

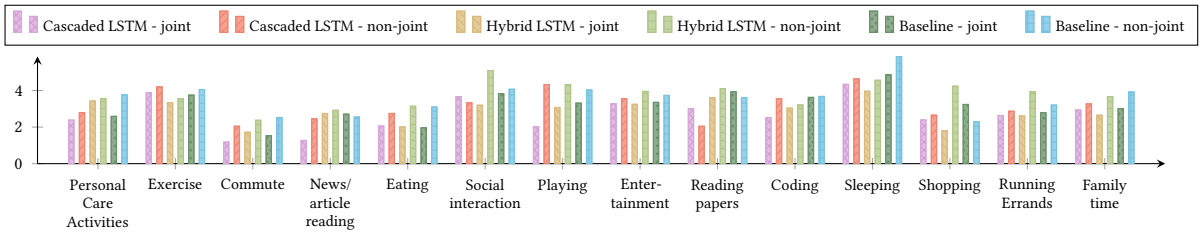


Fig. 9. log(MAE) for joint vs non-joint duration prediction

variation in activity pattern and the other low. The MAE (mean absolute error) of predicted duration averaged across all activities for the researcher is presented in Table 4 and in Table 5 for ATUS segment. We also chose to look at the errors in predicted durations activity-wise to reveal if the models are more prone to mis-predict the durations of some activities. Somewhat surprisingly, while the non-joint models performed worse than their joint counterparts, they gave accuracy of predicted durations in the same ballpark. This was most evident in the non-joint variant of cascaded LSTM, which uses a regressor on the time-of-day feature to predict the duration. We suspect that this can occur when the schedule is quite repetitive, so that conditioned on the time of day, there is very little additional information provided by the knowledge of the activity that will occur. For example, if the time is 0100 hours, the subject is most likely asleep at that time and just knowing the time can allow the regressor to predict the duration for which he typically sleeps. So there is little additional benefit added by conditioning our prediction on the activity. On the contrary, if the schedule is not very repetitive, one can expect the joint prediction to be much more effective. To test this, we took the same subject, and tested the models on 8 of his tagged weekends, while training on weekdays. Since activities on weekends are less repetitive, the dependence of the duration on time of day should be significantly weaker. For each model, the natural logarithm of MAEs on predicted durations of different activities are shown in Figure 9.

We see that the joint models are able to predict the durations much better than the non-joint ones, with significant differences for some activities like eating, commute, and sleeping.

8.4 24-hour Routine Generation

In Figure 10, we illustrate the application of our LSTM-based next activity and duration prediction model to generation of a whole at once. For generating a day, we start with a list of seed activities and then sequentially predict the next activities and their durations. For each prediction, we either select the most probable next activity (deterministic method) or sample from the predicted probability distribution (probabilistic method). As can be seen from Figure 10b, the day generated by our model given only the *Beginning of the Day* activity as a seed, is quite similar to a typical observed day of the user (Figure 10a). This suggests that the model has learned an underlying distribution for modeling the complete routine. We also carried out some initial explorations to see the responsiveness of the model to unusual occurrences in the routine. We tried generating rest of the day with a sequence of observed activities on half of a typical day as input. However, we also introduced an unusual activity sequence which contains a premature commute back home followed by *Entertainment* and *Personal Care Activities*. As can be seen from Figure 10c, the model was able to generate a day consistent with this anomaly. The model predicts that the user will carry out activities usually done at home and will not do exercise that he typically does at office only. This hints at the ability of the model to adapt to changes in routine of the user without making absurd predictions.

9 DISCUSSION AND FUTURE WORK

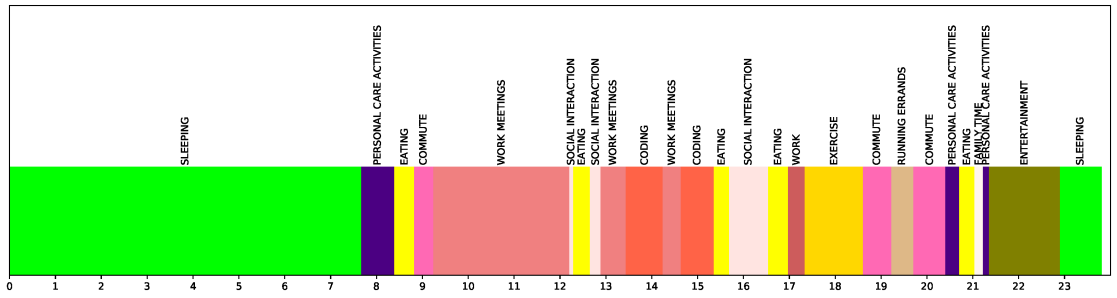
The ability to anticipate the next activities of a person can help applications and personal digital assistants become much more context-aware [40] and proactive. For example, as seen in Figure 10a, a digital assistant can remind a person to have his pre-workout meal, if our model predicts that the next activity will be Exercise. Being able to look forward in the future will help assistants initiate interactions with the user much more frequently. For example, if the assistant anticipates that the user is likely to spend some time for social interaction in a few hours, it might suggest some nearby friends so that the user can make plans.

With the increase in the number of applications pushing notifications on mobile devices, the competition for users' attention has increased and has become a challenge both for the applications and the users [10]. Being context aware helps the assistant avoid irrelevant notifications. For example, as shown in in Figure 10c, the assistant can predict that there would be no exercise in the evening since the user has already left the office premises. So continuing from the previous example, there wouldn't be any reminder to take a pre-workout meal. Although one can design a rule based system which specifically encodes that the user has a gym at the office and cannot use the gym if he left early. But learning it directly from activity patterns is more tractable compared to hand-designed rules, which can be large in number and must be coded differently for every person.

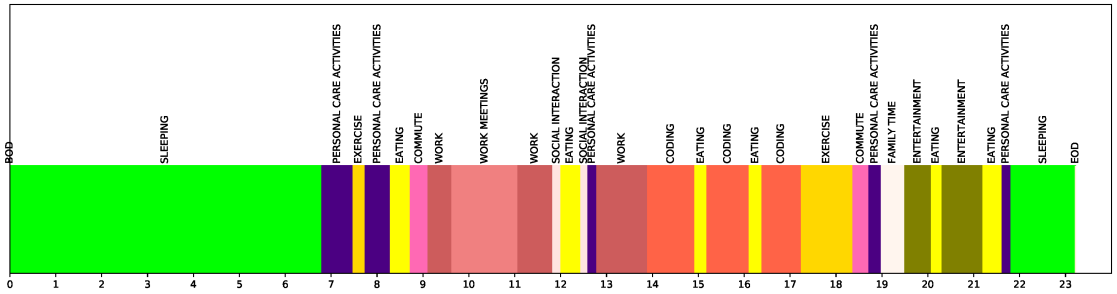
For marketers, the system can provide an opportunity to send promotions at appropriate times to better catch users' attention. For example, if the system predicts that the user will eat after some time, online food delivery companies can push notifications to the person's phone with targeted offers and recommendations. The context of the user can also be incorporated to personalize recommendations as shown by Adomavicius et al. [2] and Liu and Aberer [30].

The models show ability to predict the next activity for a person, by learning from the data of the corresponding segment. This can be leveraged in cases where we have little data about a person leading to a cold start problem [28]. Once the model has sufficient data for a person, it can start learning and basing its predictions on the individual's data only. Such an approach has been applied in the domain of location recommendation by Gao et al. [18].

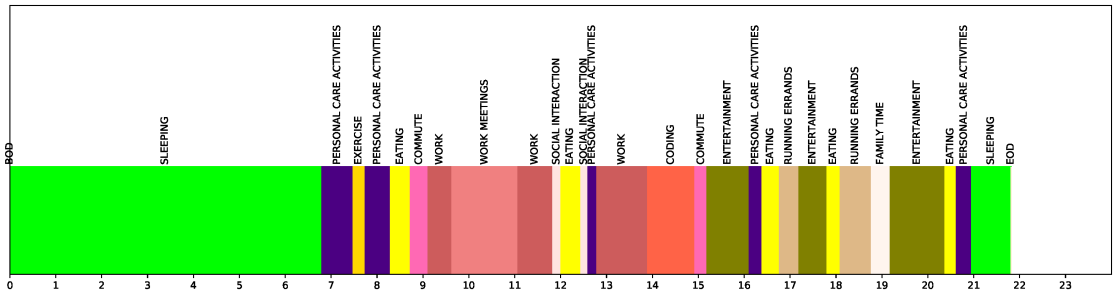
The prediction of corresponding durations along with activities helps in deciding when the next and the subsequent activities will occur. This is critical for the use cases suggested previously because the digital agent or marketer must initiate their response at the correct time. Our experiments suggest that we can predict durations reasonably well if we predict separate durations depending on the activity in a joint fashion.



(a) A typical observed day for a user



(b) A day completely generated by our model



(c) Day generated following unexpected observed behaviour

Fig. 10. Activity and duration prediction for entire day of a user

As shown by the confusion matrix in Figure 7, not all activities are equally easily predictable. In fact, even if you take the person who is the subject, and give him one of his past days, he would be more certain about a few things like leaving to the office after eating in the morning. On the other hand, he might not be very sure about what he would be doing just after having lunch in the office. These variations raise questions about the suitability of the accuracy metric for measuring the success of a model. Ideally, the model should be penalized differently for different predictions, with lower penalties for predictions that were difficult. And this could be done both while training and testing. However, figuring out from the data, which predictions to consider difficult is in itself a challenging task. Our future works will try to tackle this issue.

10 CONCLUSION

In this paper, we proposed and compared methods for prediction of next activity and its duration based on the past activities performed by a person during the day. This work builds upon previously investigated field of context inference from sensor data collected by ubiquitous computing devices such as mobile phones. We discussed the importance of storing past information for long term to improve accuracy of future activity prediction. We proposed two LSTM based models for predicting next activity as well its duration by successfully encoding long-term dependencies on historical activities. We showed that our models outperform previously used models for similar tasks, *i.e.* MHMM, HHMM and two other baseline models on predicting next activity from real-world data. We also showed that you can predict next activities for a completely new person given the observed days of other people which belong to her population segment. This suggests the existence of some uniformity in the routines of a population segment, which our models learn. Next, we show applicability of our activity prediction model in generating a sequence of future activities to model the daily-routine of a person. This opens interesting avenues for further research in daily-routine modeling. Our future work will focus on devising better evaluation metrics for sequence generation techniques and generating constrained days.

11 SUPPLEMENTARY MATERIALS

In this section, we supply details on the design and implementation of the various baseline algorithms stated in Section 6.

11.1 Activity Prediction: Multinomial HMM

HMMs are covered in detail in [32]. Essentially, HMMs rely on the assumption that the observed sequence of activities are probabilistically generated as the output of a system with a hidden state variable that evolves like a Markov process. The hidden state variable could assume multiple values with state transitions being governed by a transition probability matrix. At each timestep, the model outputs an activity by sampling from the emission probability distribution of the current state. This is a multinomial distribution in case of Multinomial HMMs. Thereafter, the state of the system probabilistically changes according to the transition matrix and the process repeats at subsequent timesteps. The optimal number of hidden states is chosen for each dataset according to the best prediction performance on the hold-out validation dataset as described in Section 7.1. For the MHMM approach, we set the size of the validation dataset to 10% of the training dataset. Once the trained MHMM model is obtained and given any sequence of activities $a_1, a_2, \ldots, a_n$, the following steps are taken to predict the next activity a_{n+1}.

(1) Use the Viterbi Algorithm [16] to infer the hidden state values $h_1, h_2, \ldots, h_n$ corresponding respectively to the activities $a_1, a_2, \ldots, a_n$.
(2) Sample the next state value h_{n+1} from the transition matrix given that the current state is h_n.
(3) Sample the activity a_{n+1} from the emission probability distribution of the state h_{n+1}.

11.2 Activity Prediction: Hierarchical HMM

For a HHMM, as a first step, a base MHMM is trained on the complete observed sequence of activities in the training set. Then, for each hidden state h of the base MHMM, a new MHMM is trained over the subsequence of observations that were attributed to the hidden state h by the Viterbi Algorithm [16]. If there were k unique hidden state values in the base MHMM, we now have k new MHMMs. This process can be repeated recursively for each of the k new MHMMs and this gives rise to a hierarchy of trained MHMMs that together constitute the HHMM.

We use a HHMM with depth 2 for constructing this baseline, *i.e.* we train a primary MHMM on the training dataset and then train one secondary MHMM for each hidden state value of the primary MHMM. The emission

probability distributions are multinomial for all MHMMs. Analogous to Section 11.1, the number of hidden state values for the primary MHMM is chosen as the number for which the best prediction performance is achieved on the hold-out validation set, where the validation set size is 10% of the training set size. We fixed the number of hidden states for the secondary MHMMs as the number of unique activities having nonzero emission probabilities in the corresponding hidden state of the primary MHMM. If a hidden state in the primary MHMM has nonzero emission probability for only one activity, then we do not train a secondary MHMM for that hidden state. We note that an explosion in the number of hyperparameters to estimate made it computationally infeasible to optimize for number of states for the secondary MHMMs.

Once the trained HHMM model is available (with primary MHMM H having k hidden states with corresponding secondary MHMMs $H_1, H_2, \ldots, H_k$) and given any sequence of activities $a_1, a_2, \ldots, a_n$, the following steps are performed to predict the next activity a_{n+1}.

(1) Use the Viterbi Algorithm [16] on H to infer the hidden state values $h_1, h_2, \ldots, h_n$ corresponding respectively to the activities $a_1, a_2, \ldots, a_n$
(2) Sample the next state value h_{n+1} from the transition matrix of H, given that the current state is h_n. .
(3) Extract the subsequence $A_q = a_{q_1}, a_{q_2}, ..., a_{q_l}$ of activities from $a_1, a_2, \ldots, a_n$ that get attributed to the hidden state value h_{n+1} by the Viterbi Algorithm above.
(4) Sample a_{n+1} using the steps described in Section 11.1 using the MHMM $H_{h_{n+1}}$ and the subsequence of activities A_q.

11.3 Activity Prediction: Probabilistic Sequence Matching (PSM)

The pseudo-code for this approach is outlined in Algorithm 1. We initialize by setting frequency = 0 for every potential next activity at time t. Let us consider a given list $\boldsymbol{l}_{m,t}$ of $m \geq 1$ activities on a test day. If no activities are given in the test seed sequence, then we take *Beginning of the Day* activity as the seed. We conduct a simple match of all occurrences of $\boldsymbol{l}_{m,t}$ against the activity sequences in the training data. Let a^* denote the next activity for one such simple match. If the starting time of a^* falls within the interval $[t - \delta, t + \delta]$, we increment the frequency of a^* by 1, else we discard this simple match from the set of valid matches. We repeat the process across all simple matches to obtain counts for every candidate a^* and normalize these counts to get a probability distribution p for the next activity.

We start with a seed with default maximum length of 3. Note that for large values of m, there may be no exact matches in the training data. In such cases, we progressively decrease the look-back period until a match is found. For example, decreasing the look-back period from m to $m - 1$ is equivalent to matching $\boldsymbol{l}_{m-1,t}$ in the training data. Thus, we shall always have non-zero counts prior to normalization which yield a valid probability distribution post-normalization.

11.4 Activity Prediction: Alignment Approach

The pseudo-code for this approach is provided in Algorithm 2 and the method basically implements the Needleman-Wunsch algorithm [36] to find alignments of a seed activity sequence with days previously seen in the training set of sequences. The Needleman-Wunsch algorithm is popular in bio-informatics for aligning short protein or DNA sequences. In their paper, Sigg et al. [45] introduce a slight variation to the basic algorithm in order to compute semi-global alignments instead of global alignments. The overall method is a dynamic programming based approach and is described below:

(1) We start with an empty matrix W whose columns represent the activities in one training sequence and rows represent the activities in the seed sequence. Also, there is an extra row and column in the beginning of the matrix. All cells are initialized with 0.

Algorithm 1 Next activity prediction from PSM model

function PREDICT_NEXT_ACTIVITY($\boldsymbol{l}_{m,t}, t$)
  $\boldsymbol{l}_{tr} \leftarrow$ complete historical sequence of activities in the training dataset
  $f[a] \leftarrow 0 \quad \forall a \in \mathcal{A}$ ▷ Where f is a dictionary
  for $i = 1$ to n **do**
    $\boldsymbol{l}_{i,t} \leftarrow \boldsymbol{l}_{m,t}[i..m]$
    $I \leftarrow$ SUBSTRING_INDICES($\boldsymbol{l}_{tr}, \boldsymbol{l}_{i,t}$) ▷ list of indices where $\boldsymbol{l}_{i,t}$ is a substring of $\boldsymbol{l}_{tr}$
    for $j \in I$ **do**
      $k = j + n - i + 1$ ▷ index of the next activity after instance of $\boldsymbol{l}_i$ in $\boldsymbol{l}_{tr}$
      $a^* \leftarrow \boldsymbol{l}_{tr}[k]$ ▷ the next activity observed in $\boldsymbol{l}_{tr}$ after $\boldsymbol{l}_{i,t}$
      if start time of $a^* \in [t - \delta, t + \delta]$ **then**
        $f[a^*] = f[a^*] + 1$ ▷ increment frequency of a^*
      end if
    end for
    if $\exists a \in \mathcal{A}, f[a] > 0$ **then**
      break ▷ Break loop if substring matches found
    end if
    $i = i + 1$
  end for
  $p \leftarrow$ NORMALIZE(f) ▷ Normalize to get the probability distribution
  return p
end function

(2) A scoring scheme for the final alignment is decided. In this case, we assign a score of 1 for every matching element in the alignment, −1 for mis-matched activities and −1 for any insertions or deletions/gaps.
(3) Now we start filling the matrix from second row and column going column by column. For a cell (i, j), the score is calculated as the highest non-negative score from top, left or top-left. Score calculated from top or left represents a gap in the alignment (hence it becomes the score in top/left cell plus the gap score) and the score from top-left represents a match/mismatch between activities in row $i - 1$ and column $j - 1$ (hence it becomes the score in top-left cell plus the match/mismatch score). If there are no positive scores from the top, left or top-left cells, then we assign a score of 0 to the current cell. After completing the matrix this way, we get semi-global alignments of the seed sequence in the last row of the matrix. Hence, for a length m seed sequence, the j^{th} cell of the last row of W contains the alignment score $W[m, j]$ of the seed sequence with the subsequence formed by the first j elements of the training sequence.
(4) For every unique activity a, we compute a confidence score by a double sum computed as below:
  - For a given training sequence, all $W[m, j]$ scores where the j^{th} element of that training sequence is a, are added to give a per-sequence confidence score.
  - Then, all per-sequence confidence scores for a are added across all training sequences to arrive at a overall confidence score. The overall confidence score is assumed to be proportional to the probability of predicting the next activity as a.
(5) Finally, we normalize the confidence scores across all unique activities to get a probability distribution p for predicting the next activity.

Algorithm 2 Next activity prediction with Needleman-Wunsch semi-global alignment algorithm

function PREDICT_NEXT_ACTIVITY($\boldsymbol{l}_{m,t}, t$)
  $m \leftarrow$ length of $\boldsymbol{l}_{m,t}$
  $\boldsymbol{l}_{tr} \leftarrow$ set of sequences of activities in the training dataset
  $f[a] \leftarrow 0, \quad \forall a \in \mathcal{A}$ ▷ Where f is a dictionary
  $d_{match} \leftarrow 1$ ▷ Reward for a match between sequences
  $d_{mismatch} \leftarrow -1$ ▷ Penalty for a mis-match between sequences
  $d_{gap} \leftarrow -1$ ▷ Penalty for a gap between sequences
  for $\boldsymbol{l} \in \boldsymbol{l}_{tr}$ **do** ▷ For every sequence in training set
    $n \leftarrow$ length of $\boldsymbol{l}$
    $W \leftarrow$ ZERO_MATRIX$(m+1, n+1)$ ▷ Matrix of size m+1, n+1 initialized with zeros
    for $i = 2$ to $n+1$ **do**
      for $j = 2$ to $m+1$ **do**
        if $\boldsymbol{l}_{m,t}[j-1] = \boldsymbol{l}_{tr}[i-1]$ **then**
          $diag \leftarrow W[j-1, i-1] + d_{match}$ ▷ Matching activities found in the sequence
        else
          $diag \leftarrow W[j-1, i-1] + d_{mismatch}$ ▷ Activities do not match
        end if
        $top \leftarrow W[j-1, i] + d_{gap}$ ▷ Gap in the seed sequence
        $left \leftarrow W[j, i-1] + d_{gap}$ ▷ Gap in the training sequence
        $W[j, i] \leftarrow$ MAX$(diag, top, left, 0)$ ▷ Pick highest (non-negative) score
      end for
    end for
    for $i = 2$ to $n+1$ **do** ▷ Sum over scores in final row of W for each next activity in $\boldsymbol{l}$
      $f[\boldsymbol{l}_{tr}[i]] \leftarrow f[\boldsymbol{l}_{tr}[i]] + W[m+1, i]$
    end for
  end for
  $p \leftarrow$ NORMALIZE(f) ▷ Normalize to get the probability distribution
  return p
end function

11.5 Implementation details

All the models and algorithms are implemented using the Python programming language. We used the `keras` library[2] with the `tensorflow` backend [1] to implement the neural network architectures. Hyperparameter optimization was carried out using the `hyperopt` [6] and `hyperas`[3] packages. MHMM and HHMM were implemented using the `hmmlearn`[4] package. The `scikit-learn` [38] package was used for the regressors in the Time of Day Model (see Section 6.3) and for duration prediction (see Sections 6.6 and 4.2).

ACKNOWLEDGMENTS

The authors would like to thank Anandhavelu N., Anshul Agrawal and Bhushan Kulkarni for conceptualization of the problem, and Dr. Shiv Saini for insightful discussions.

[2] https://github.com/fchollet/keras/tree/master/keras
[3] https://github.com/maxpumperla/hyperaspackages
[4] https://github.com/hmmlearn/hmmlearn

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)* 23, 1 (2005), 103–145.

[3] Nikola Banovic, Tofi Buzali, Fanny Chevalier, Jennifer Mankoff, and Anind K. Dey. 2016. Modeling and Understanding Human Routine Behavior. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 248–260. https://doi.org/10.1145/2858036.2858557

[4] Mitra Baratchi, Nirvana Meratnia, Paul JM Havinga, Andrew K Skidmore, and Bert AKG Toxopeus. 2014. A hierarchical hidden semi-Markov model for modeling mobility data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 401–412.

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. 1171–1179.

[6] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery* 8, 1 (2015), 014008. http://stacks.iop.org/1749-4699/8/i=1/a=014008

[7] Oliver Brdiczka, Norman Makoto Su, and James Bo Begole. 2010. Temporal Task Footprinting: Identifying Routine Tasks by Their Temporal Patterns. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 281–284. https://doi.org/10.1145/1719970.1720011

[8] Daniel Castro, Steven Hickson, Vinay Bettadapura, Edison Thomaz, Gregory Abowd, Henrik Christensen, and Irfan Essa. 2015. Predicting Daily Activities from Egocentric Images Using Deep Learning. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers (ISWC '15)*. ACM, New York, NY, USA, 75–82. https://doi.org/10.1145/2802083.2808398

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[10] Mary Czerwinski, Ran Gilad-Bachrach, Shamsi Iqbal, and Gloria Mark. 2016. Challenges for designing notifications for affective computing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 1554–1559.

[11] Mukund Deshpande and George Karypis. 2004. Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology (TOIT)* 4, 2 (2004), 163–184.

[12] Rob A Dunne and Norm A Campbell. 1997. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, 181*, Vol. 185.

[13] Nathan Eagle and Alex Sandy Pentland. 2009. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology* 63, 7 (2009), 1057–1066.

[14] Nathan Eagle and Alex (Sandy) Pentland. 2006. Reality Mining: Sensing Complex Social Systems. *Personal Ubiquitous Comput.* 10, 4 (March 2006), 255–268. https://doi.org/10.1007/s00779-005-0046-3

[15] Marcus Felson and Lawrence E. Cohen. 1980. Human ecology and crime: A routine activity approach. *Human Ecology* 8, 4 (1980), 389–406. https://doi.org/10.1007/BF01561001

[16] G David Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

[17] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[18] Huiji Gao, Jiliang Tang, and Huan Liu. 2015. Addressing the cold-start problem in location recommendation using geo-social correlations. *Data Mining and Knowledge Discovery* 29, 2 (2015), 299–323.

[19] Felix A Gers and E Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12, 6 (2001), 1333–1340.

[20] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[21] Geoffrey Hinton, NiRsh Srivastava, and Kevin Swersky. 2012. Lecture 6a Overview of mini–batch gradient descent. (2012).

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[23] Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12, 1 (1970), 55–67.

[24] Andrej Karpathy. 2015. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. (2015).

[25] Reed Larson and Mihaly Csikszentmihalyi. 1983. The experience sampling method. *New Directions for Methodology of Social & Behavioral Science* (1983).
[26] Joohyun Lee, Kyunghan Lee, Euijin Jeong, Jaemin Jo, and Ness B. Shroff. 2016. Context-aware Application Scheduling in Mobile Systems: What Will Users Do and Not Do Next?. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 1235–1246. https://doi.org/10.1145/2971648.2971680
[27] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.
[28] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.
[29] Andrew Liu and Dario Salvucci. 2001. Modeling and prediction of human driver behavior. In *Intl. Conference on HCI*.
[30] Xin Liu and Karl Aberer. 2013. SoCo: a social network aided context-aware recommender system. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 781–802.
[31] Ye Liu, Liqiang Nie, Lei Han, Luming Zhang, and David S. Rosenblum. 2016. Action2Activity: Recognizing Complex Activities from Sensor Data. *CoRR* abs/1611.01872 (2016). http://arxiv.org/abs/1611.01872
[32] Iain L MacDonald and Walter Zucchini. 1997. *Hidden Markov and other models for discrete-valued time series*. Vol. 110. CRC Press. 115–120 pages.
[33] James McInerney, Sebastian Stein, Alex Rogers, and Nicholas R. Jennings. 2013. Breaking the habit: Measuring and predicting departures from routine in individual human mobility. *Pervasive and Mobile Computing* 9, 6 (2013), 808 – 822. https://doi.org/10.1016/j.pmcj.2013.07.016 Mobile Data Challenge.
[34] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. In *Aaai/iaai*. 187–192.
[35] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.
[36] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.
[37] United States Department of Labor. Bureau of Labor Statistics. 2015. American Time Use Survey (ATUS), 2003-2015, Multi-Year Data. (2015). https://www.bls.gov/tus/datafiles_0315.htm
[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[39] Alex Pentland and Andrew Liu. 1999. Modeling and prediction of human behavior. *Neural computation* 11, 1 (1999), 229–242.
[40] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 414–454.
[41] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. 2016. Unsupervised Recognition of Interleaved Activities of Daily Living Through Ontological and Probabilistic Reasoning. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/2971648.2971691
[42] S Rasoul Safavian and David Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics* 21, 3 (1991), 660–674.
[43] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128* (2014).
[44] George AF Seber and Alan J Lee. 2012. *Linear regression analysis*. Vol. 936. John Wiley & Sons.
[45] Stephan Sigg, Sandra Haseloff, and Klaus David. 2010. An alignment approach for context prediction tasks in ubicomp environments. *IEEE Pervasive Computing* 9, 4 (2010), 90–97.
[46] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. 2006. Learning to predict driver route and destination intent. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 127–132.
[47] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[48] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012).
[49] Vladimir Vapnik. 1998. The support vector method of function estimation. In *Nonlinear Modeling*. Springer, 55–85.
[50] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 3995–4001. http://dl.acm.org/citation.cfm?id=2832747.2832806
[51] Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu. 2011. Cross-people Mobile-phone Based Activity Recognition. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three (IJCAI'11)*. AAAI Press, 2545–2550. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-423

[52] Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach.. In *AAAI*, Vol. 10. 236–241.

Received May 2017; revised August 2017; accepted October 2017