IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 25, 2019, accepted August 20, 2019, date of publication August 26, 2019, date of current version September 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937350

Extension of HMM-Based ADL Recognition With Markov Chains of Activities and Activity Transition Cost

GREGOR DONAJ AND MIRJAM SEPESY MAUČEC, (Member, IEEE)
Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

Corresponding author: Gregor Donaj (gregor.donaj@um.si)

This work was supported by the Slovenian Research Agency (research core funding No. P2-0069–Advanced Methods of Interaction in Telecommunications). This work was also partially funded by the Republic of Slovenia and the European Union under the European Regional Development Fund–IQ Home Program ''Intelligent home of the new generation designed on smart appliances and wood''.

ABSTRACT Ambient assisted living in smart home environments is becoming an important goal in an aging society with challenges in elderly care. A key component in such environments is the accurate recognition of activities of daily living from various sensor data. Recent research directions explored several classification methods, including hidden Markov models. This research presents a hidden Markov model-based system for activity recognition, and extends it with a second-order Markov chain model of activity sequences to achieve long-term dependency in the model. We also introduce an activity transition cost to counteract the tendency of hidden Markov models to make a large number of transitions. The proposed models are used for activity recognition, with their scores being combined using heuristically determined weights for optimal performance. We also present a modified Viterbi algorithm, which incorporates both models and the activity transition cost. We used a dataset from the CASAS project to test and evaluate the proposed models. A comparison of the results shows the potential of introducing long term dependencies and the managing the number of activity transitions. We show results regarding the modeling ability to predict activity sequences, a comparison of predicted and actual activity transitions, and final recognition accuracy results. The results show an increase of total activity recognition accuracy from 93.9 % to 94.52 % on individual activities, and from 68.89 % to 70.95 % over the combination of all concurrent activities. The results also show a reduction of predicted activity transitions from 741 to 236, whereas the number of actual activity transitions in the evaluation set is 141.

INDEX TERMS Activities of daily living, hidden Markov models, Markov chain, pattern recognition, Viterbi algorithm.

I. INTRODUCTION

Smart home technologies promise many opportunities in the area of ambient assisted living, which may become especially useful for elderly or cognitively impaired people. The goal here is to assist the elderly to live longer independently, to provide remote health monitoring for physicians, and give distant family members peace of mind. The importance of smart environments is also emphasized by the ongoing trend of population aging in most developed countries.

An essential function in many smart home environments is the ability to recognize the activities of daily living (ADL) of residents. The predicted activity data (i.e., the activity recognition algorithm's output) can be used either to assist the residents from the smart home, or for monitoring the residents for unusual activity patterns, or even a lack of activities. Further processing of recognized activity patterns can then be used to determine residents' habits and deviations from them. Unusual activity patterns may then suggest the need for checking on the residents, or for intervention.

The range of possible activities that can be recognized is quite broad. Activities can be related to a person's movement or position (e.g., walking, running, sitting, standing), to the use of house appliances (e.g., cooking, eating, watching TV), or the use of more specific items (e.g., taking medicine). Activities can also be overlapping, i.e., more than one activity can occur at the same time, especially in homes with several residents.

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Zhou.

Different machine learning methods can be used for ADL recognition, including the hidden Markov models (HMMs) [1]–[5], which are often used for modeling time series.

In this research, we propose an extension to the standard HMM using a Markov chain model (MCM) of the activity sequences, enabling us to model longer dependencies than with the standard HMM. As we found a tendency of the HMM to make a large number of transitions between activities, we also propose the use of an activity transition cost (ATC) into the model to counteract this tendency. To our knowledge, those approaches have not been used in HMM-based ADL recognition yet.

To test the proposed system, we used a dataset with spontaneous living data and overlapping activities. We present results in terms of the accuracy of individual activities, a total accuracy score across all individual activities, and an accuracy score for the combination of all concurrent activities. We also present results regarding the number of state transitions in the HMM with a graphical example.

We also address briefly the issue of probability smoothing for HMM parameter estimation.

The paper is organized as follows. In Section II, we describe modeling techniques used in ADL recognition and an overview of previous research. Section III describes the dataset, which was used to build the proposed models and perform experiments. In Section IV, we describe the design and the stepwise development of the experimental system, and in Section V we present and compare the final results with regard to the different steps in the experimental system. We also discuss the results with regard to previously published research. The conclusion follows in Section VI.

II. BACKGROUND

Sensor data must be collected in order to recognize activities. Additionally, the gathered sensor data must be labeled with activity tags. This process may result in a considerable expense when using observers to label activities [6]. The sensor data and activity annotations are used to train the models of the recognition system – estimate the model's parameters.

Then, the models are used with a new set of sensor data. The task of the system is to determine the most likely sequence of activities which would generate the observed sensor data. The accuracy of the system is evaluated by comparing the predicted activity sequence with the actual activity sequence.

As this problem involves time series, suitable pattern recognition algorithms are used.

A. SENSORS

The complexity of sensor setups can have a wide range, from simple binary sensors [7] to more complex setups with a large number and different types of sensors [8], [9]. Often the choice of sensors depends on the type of activities to be recognized.

Simple sensors can be on-off switches on doors, cabinets, or certain items. Using item sensors on selected items can be very useful for recognizing specific activities, e.g., kitchen items to recognize meal preparation. More complex data can be gathered from temperature sensors and humidity sensors, water and electricity counters, or strain sensors in the floor [10].

Cameras [2], [11] and microphones [12] can be mounted in smart homes, or they can be wearable devices, e.g., a camera on a person's glasses [13]. However, microphones or cameras can be considered too invasive to the privacy of residents. Recognizing activities from video recordings also can be computationally expensive [11]. Video recognition has also been shown to perform better on videos with depth perception rather than 2D videos [14]. However, depth perception requires more than one camera, and even more computational power.

Wearable kinematic sensors can also be used. Zhu and Sheng [5] showed results using a single wearable inertial sensor on a task with 8 activities regarding a person's position or movement. Another method is to use sensors embedded in mobile devices [15]. However, those approaches suffer from person-to-person variability. Also, wearable sensors might be uncomfortable, and the residents must not forget to wear them. Less intrusive sensors, e.g., fixed motion sensors in the household, seem more appropriate for research and usage in real-live settings.

Some research is concerned specifically with the possibility of using non-intrusive methods [16], [17]. Fogarty *et al.* [12] presented a non-intrusive system where only microphones were placed at key locations in a home's water distribution system. Based on the water usage patterns, they inferred related activities, such as showering, toilet flushing, or using the dishwasher. Other research is aimed towards optimal feature selection and feature generation from sensor data [18], [19].

B. RECOGNITION ALGORITHMS

Simple handcrafted algorithms can be used in activity recognition. Tan *et al.* [20] used two simple algorithms to classify events regarding the apartment front-door into three classes: enter, exit, and brief-return-and-exit. The experiments were performed on datasets gathered from apartments with elderly residents to detect an early sign of dementia: Brief returns home due to forgetfulness. Urwyler *et al.* [17] developed two ad-hoc classifiers, and compared them to naïve Bayes and Random Forrest classifiers.

The k-nearest neighbor classifier works by calculating the distance (in terms of feature values) between a new data-point and the data-points from the training set. The new data-point is then classified to the most frequent class in the set of k nearest data-points. Gupta and Dallas [21] compared the k-nearest neighbor classifier with the naïve Bayes classifier in recognition of movement activities with a triaxial accelerometer.

Support vector machines may be chosen with smaller datasets. This classifier works by determining a hyperplane in the vector space of features. When a new data-point is observed, its position relative to the hyperplane determines the classification. This model must then be expanded, in order to discriminate more than two states. Using SVMs, Fleury *et al.* [22] achieved accuracies between 64.3 % and 97.8 % on a task with 7 different activities.

Wu *et al.* [14] used both support vector machines and HMMs in an effort to leverage the advantages of both methods, and obtained a higher recognition accuracy compared to both methods used individually.

Conditional random fields are a graph-based model, where the nodes are the observations and random variables. The random variables conditional to the observations must obey the Markov property. Vail *et al.* [23] used conditional random fields and compared them with HMMs on a simulated robot tag domain.

Recently, artificial neural networks have been used in ADL recognition with good results [5], [15], [24]. However, care must be taken to reduce the computational cost of using neural networks [18]. Neural networks can also be used in conjunction with other methods like HMMs, e.g., as a method for modeling emission probability distributions of HMM states [25].

Other methods are rule-based approaches [26], the topic model [27], positional temporal logic [28], and decision trees [29].

C. PREVIOUS WORK IN HMM BASED RECOGNITION

In [30], the authors used Markov chain models to model each activity while using sensors as states. Such models can distinguish between isolated activities. However, as those models are not appropriate for recognizing interleaved activities, the authors further used HMMs with activities as the hidden states. The authors then compared the naïve Bayes classifier and the HMM, finding that the HMM outperforms the naïve Bayes classifier by 5 % in accuracy (71.01 % vs. 66.08 %). They also managed to improve the accuracy to 84.18 % further using a sliding window for sensor events.

In [31], the authors compared naïve Bayes, HMM, and neural network classifiers, while also having an emphasis on feature selection. They found that multilayer perceptron neural networks give an accuracy of 91.8 % when the optimal feature subset was selected, whereas naïve Bayes and HMM systems performed with accuracies of 87.5 % and 86.3 % respectively.

Cheng *et al.* [1] proposed an Adaptive Learning HMM to improve accuracy in a setting where the test data have significantly different characteristics than the training data. Karaman *et al.* [2] introduced a two-level hierarchical HMM for activity recognition from video features. Lu *et al.* [3] used Beta process HMMs to extract features which were later used in a support vector machine system.

To our knowledge, previous research into using HMMs for ADL recognition has not addressed long term dependencies of activities, or the issue of the number of activity transitions in HMMs.

D. NAÏVE BAYES CLASSIFICATION

Let X be the set of all activities, and let y be the value of a given feature vector – a vector containing all used sensor data. The naïve Bayer classifier works by classifying the datapoint to the activity $\hat{x}$, which is most likely to occur given the feature vector. This is done using the conditional probability relation:

$$\hat{x} = \arg\max_{x \in X} P(x|y) = \arg\max_{x \in X} \frac{P(y|x) \cdot P(x)}{P(y)}. \tag{1}$$

As the denominator in the above equation is independent of the activity x, only the numerator must be considered to use the naïve Bayes classifier. The conditional feature vector probability $P(y|x)$ is calculated using a probability distribution estimated from the training data. The activity probability $P(x)$ is estimated based on the frequency of this activity in the training set.

The naïve Bayes classifier considers only the value of the given feature vector and does not incorporate history into its model. Therefore, it is typically considered less appropriate for modeling time series, unless temporal data are incorporated into the feature vectors themselves.

E. MARKOV CHAIN MODEL

A Markov chain is a probabilistic model for modeling time series. It is defined by a set of states and probabilities for any state to occur in the chain. Although continuous-time and continuous-state Markov chains exist, we consider only discrete-time Markov Chains with a finite number of states.

A Markov chain is a Markov process, i.e., a process in which the probability of any state to occur is conditional on the finite history of previous states. The number of states in the considered history determines the order of the Markov chain.

Let $\mathcal{X} = X_0, \ldots, X_T$ be a series of states of length $T + 1$, where each state belongs to a set of N different states, denoted by integers:

$$X_t \in \mathcal{S} = \{1, .., N\} \quad \forall t = 0, \ldots, T. \tag{2}$$

A Markov Chain Model (MCM) models the probability of a given sequence of states to occur $P(\mathcal{X})$.

The simplest chain is the first-order chain, which is defined by the number of states N, the start probabilities for each state

$$\pi_i = P(X_0 = i) \quad \forall i \in 1, \ldots, N, \tag{3}$$

and conditional transition probabilities between states

$$\lambda_{i,j} = P(X_t = j|X_{t-1} = i) \quad \forall i, j \in 1, \ldots, N. \tag{4}$$

In an n-th order chain, the start probabilities must be defined for all possible combinations of n or less starting states, and the transition probabilities are conditional on n previous states.

MCMs are used to model the probability of an observed chain given a population of chains. Another possibility is to have several MCMs, to determine which most likely fits an observed state sequence.

A first-order MCM estimates the probability of a state sequence with

$$P(\mathcal{X}) = \pi_{X_0} \cdot \prod_{j=1}^{T} \lambda_{X_{j-1},X_j}. \tag{5}$$

This estimate can be generalized to an n-th order MCM:

$$P(\mathcal{X}) = \prod_{j=0}^{n-1} \pi_{X_0,\ldots,X_j} \cdot \prod_{j=n}^{T} \lambda_{X_{j-n},\ldots,X_j}. \tag{6}$$

MCMs are often used in language processing, although, in that area, they are called n-gram models, as a sequence of n words (states), is called an n-gram. They estimate the probability of a sequence of words to occur in a given language, which is modeled.

F. HIDDEN MARKOV MODEL

A hidden Markov model is an extension to the Markov chain. Again, a sequence of states $\mathcal{X}$ from a first-order Markov chain is considered. However, this sequence is now hidden. Additionally, a sequence of observed feature vectors $\mathcal{Y} = Y_0, \ldots, Y_T$ of the same length is given.

Like an MCM, an HMM is defined by the number of states, start probabilities, and transition probabilities for the hidden Markov chain. Additionally, a set of feature vector probability distributions

$$\Phi = \{\phi_i(y) | i = 1, \ldots, N\}, \tag{7}$$

is part of the model, where

$$\phi_i(y) = P(Y_t = y | X_t = i) \tag{8}$$

is the conditional probability distribution of the feature vector having value y given the state i. These probability distributions are time-independent.

In the context of HMMs, one says that states generate the observed feature vectors, and feature vector probabilities are called emission probabilities.

Given a sequence of observed feature vectors $\mathcal{Y} = Y_0, \ldots, Y_T$ and a sequence of hidden states $\mathcal{X} = X_0, \ldots, X_T$ the conditional emission probability is

$$P(\mathcal{Y}|\mathcal{X}) = \prod_{t=0}^{T} \phi_{X_t}(Y_t). \tag{9}$$

The HMM can be used to estimate the joint probability that the state and feature vector sequences occurred using the probability of the hidden Markov chain and the conditional emission probability as

$$P(\mathcal{Y}, \mathcal{X}) = \pi_{X_0} \cdot \prod_{j=1}^{T} \lambda_{X_{j-1},X_j} \cdot \prod_{t=0}^{T} \phi_{X_t}(Y_t). \tag{10}$$

This estimation can be useful in determining how well a model fits the observed data (features and states).

While using HMMs in any form of pattern recognition, the ultimate goal is most often decoding – determining the sequence of states $\hat{\mathcal{X}}$ that most likely resulted in the given sequence of observed feature vectors $\mathcal{Y}$:

$$\hat{\mathcal{X}} = \arg\max_{\mathcal{X}} P(\mathcal{X}|\mathcal{Y}) = \arg\max_{\mathcal{X}} \frac{P(\mathcal{Y}|\mathcal{X}) \cdot P(\mathcal{X})}{P(\mathcal{Y})}, \tag{11}$$

where $P(\mathcal{X})$ is the first-order Markov chain probability defined in (5). As this expression is independent of the denominator, one can consider only values in the numerator, thus making the cost function for the HMM decoder:

$$P(\mathcal{Y}|\mathcal{X}) \cdot P(\mathcal{X}), \tag{12}$$

The decoding problem – finding the sequence which maximizes the cost function – is solved using the Viterbi algorithm [32].

G. MODEL TRAINING AND PROBABILITY SMOOTHING

During training, all the model's parameters must be estimated. Start and transition probabilities for an MCM can be estimated based on counts. For example, the transition probability from state i to state j can be estimated by the fraction

$$\lambda_{i,j} = \frac{C_{i,j}}{C_i}, \tag{13}$$

where $C_{i,j}$ is the count of states i and j occurring consecutively, and C_i is the count of state i occurring in the training data.

While training HMMs, the state sequence can be hidden in some applications, even in the training data. Therefore, for training HMMs, the Baum-Welch algorithm is most often used [33] to determine transition and emission probabilities.

In annotated ADL datasets where the HMM states represent the activities, start and transition probabilities can be estimated in the same manner as in MCMs. Next, emission probabilities can be estimated in the same manner as in the naïve Bayes classifier.

However, using estimates based on counts has a potential issue. If a combination of 2 consecutive states or the value of a discrete-valued feature does not occur in the training data, the model will contain zero probabilities. If this state combination or feature value occurs in the test data, the model will estimate the probability of the entire activity sequence to be 0.

There are several methods for smoothing probability estimations to avoid zero probabilities. We used additive smoothing in the proposed system, as described later in this paper.

Considering (13), additive smoothing by using a new probability estimation was included:

$$\lambda_{i,j} = \frac{C_{i,j} + a}{C_i + a \cdot N}, \tag{14}$$

where a is the additive count and N is the number of states. The addition in the numerator ensures its value to be positive. The addition in the denominator is necessary so that the probabilities sum up to 1.

The same method can be applied to starting probabilities for MCMs and HMM, and the emission probabilities of discrete features.

III. DATABASE PREPARATION

A. THE CASAS DATASETS

In order to train and evaluate any pattern recognition system, we need to build appropriate datasets. Residential rooms should be equipped with sensors, and the collected data have to be annotated further with activities. This can be a time-consuming and costly process. Not many appropriate datasets are available.

Given the purpose and design of the proposed system, it was decided to use the publicly available datasets from CASAS (Center for Advanced Studies in Adaptive Systems) [8], [9], which are publicly available for download on their webpage.[1]

Several datasets were created in the CASAS project. Some contain sensor data, where residents were instructed to perform a specific task with or without errors and with or without interleaving, whereas others contain daily living data spanning several months. The latter datasets are interesting for the experiments. We used a dataset with overlapping activities to perform experiments with a sufficient amount of data (number 8). It consists of sensor and activity data from May to July 2009. During this time, the three-bedroom apartment was occupied by two residents.

The apartment from the selected dataset has different types of sensors: 51 motion sensors, 4 item sensors on selected items, 15 door sensors, 5 temperature sensors in the different rooms of the apartment, and an electricity usage meter. Within the datasets, there are 8 different activities.

The activities are: Bed to toilet transition, cleaning, cooking, grooming, shower, sleep, wakeup, and work. Either of the two residents can perform the activities. The last four activities are annotated for either resident separately, thus giving a total number of 12 activities. As both residents can be present at the same time, activities in this dataset are often overlapped.

B. REFORMATTING

The original dataset is a text file, where each line contains one event – a data-point. Events are changes in sensor values, or starts and ends of activities. Each line contains a time-stamp (date and time), the name of the sensor changing its value, and the new value itself. If, at this time, an activity has also begun or ended, this is also noted in the same line.

We reformatted the dataset into a form suitable for further computer processing. In the reformatted form each line represents one data-point, such that the time-stamp, all sensor data, and all activity data, are present in each line.

We merged activities from the residents, i.e., we no longer distinguished between the same activity being performed by either of the residents. This reduces the number of activities back to 8. However, as activities can be overlapping, the number of possible combinations of activities is again higher.

We also changed all text-described sensor values to numerical values for easier processing, e.g., the sensor value "ON" was changed to "1" and "OFF" was changed to "0".

[1] http://casas.wsu.edu/

TABLE 1. Sizes of training, development, and evaluations sets from the selected CASAS dataset.

Dataset part	Start date	End date	No. of days	No. of datapoints with activities
Train	2009-5-29	2009-7-15	48	236,369
Development	2009-7-15	2009-7-23	9	40,000
Evaluation	2009-7-23	2009-7-31	9	40,000

C. SPLITTING

The reformatted dataset contained approximately 300,000 data-points – one for each event in the dataset. The data ranges from May 29th to July 31st, 2009. The data were split into three non-overlapping subsets: The training data, the development data, and the evaluation data.

The training data were used to train both models, the HMM and the MCM. The development set was used for model weight optimization and other examination of the performance of the system. Lastly, the evaluation data were used to obtain the final performance results for all models.

Approximately 25 % of the dataset was used as the development and evaluation sets, whereas all other data were used for the training set. Table 1 shows the exact sizes of the three sets, including the number of data-points with activities present.

IV. ADL RECOGNITION SYSTEM

We decided to use activity combinations as states for the HMM. Activity combinations were used since a few activities could (and in fact do) occur concurrently in the dataset. Therefore, using one state for each activity would not be appropriate.

Using separated HMMs for all activities would also not be appropriate, as one cannot expect activities to be independent. For example, a large number of activities cannot occur in an apartment with only two residents.

The training set contains 23 different activity combinations, with frequencies from 20 to 180,322. This set of activity combinations is the set of HMM states, and, therefore, the set of possible classifications for the proposed system.

The set of all sensor data at a given data-point is the feature vector. The sequence of feature vectors represents the input data for the classifiers.

After classification, the results were evaluated in terms of accuracy. This was done for each activity individually, as a total value across all activities, and as an activity score for the combination of all concurrent activities.

Algorithm 1 The Viterbi Algorithm

```
1: for all $j$ from $1 \leq j \leq N$ do
2:     $\delta_0(j) \leftarrow \pi_j \cdot \phi_j(Y_0)$
3: end for
4: for all $t$ from $1 \leq t \leq T$ do
5:     for all $j$ from $1 \leq j \leq N$ do
6:         $\delta_t(j) \leftarrow \max_{1 \leq i \leq N} \delta_{t-1}(i) \cdot \lambda_{i,j} \cdot \phi_j(Y_t)$
7:         $\psi_t(j) \leftarrow \arg\max_{1 \leq i \leq N} \delta_{t-1}(i) \cdot \lambda_{i,j}$
8:     end for
9: end for
```

A. NAÏVE BAYES CLASSIFICATIONS

One of the main characteristics of HMMs is their ability to model time series with dependencies on previous states. To assess whether this property of HMMs is useful, the naïve Bayes classifier was used for comparison.

To train the naïve Bayes classifier, we needed frequencies of all activity combinations from the training set. This was done by counting them and then using additive smoothing on those counts.

Next, we needed distributions of all variables in the feature vector for each activity combination in the training set. This was done by estimating the probability distribution for each sensor individually, and aggregating the distributions into one multivariate distribution for the feature vector.

Given the value of the current feature vector, the classifier then classifies the data-point into one of the possible activity combinations using (1).

B. BASIC HMM RECOGNITION

We designed an HMM with 23 states – one state for each activity combination appearing in the training set. State start and transition probabilities were determined from the training set using counts and additive smoothing.

The feature vector probability distributions for all states were determined in the same manner as with the naïve Bayes classifier.

The time-steps for the HMM are determined by a change in at least one sensor value. These are the data-points in the dataset. Hence, at each new value, a transition is made to another state or the same state.

In a basic HMM-based recognition system, the Viterbi algorithm is used to find the most likely state sequence that would result in the observed feature vector sequence. The basic Viterbi algorithm given in Algorithm 1, where $\delta_t(j)$ is the best-path-probability of being in state j at time t, and $\psi_t(j)$ is the backtrack indicator, indicating from which state at the previous time-step the best-path-probability was found. All other notations in Algorithm 1 are adopted from subsections II-E and II-F.

The best-path-probability of a given state is the probability of the most likely partial sequence of states from time 0 to the current time which ends in the given state. The backtrack indicator points to the state at one time-step back in this partial sequence.

The algorithm works by assigning each state at time 0 its probability according to the start and emission probabilities. Then, from time 1 onwards, the algorithm finds the best possible probability for each state at each time using best-path-probabilities from the previous time-step, transition probabilities (to account for the transition to the current time-step), and emission probabilities of the feature vector at the current time-step. The highest probability is determined at each time-step and for each state. Also, the backtrack indicator is set to the state (one time-step back) from which this highest probability was achieved.

After the Viterbi algorithm calculates the best-path-probabilities for the last time-step, the best path is determined by finding the state with the highest best-path-probability at the last time-step, and then following the backtrack pointer back to time 0.

C. HMM RECOGNITION WITH MCM AND ATC

Since the basic HMM makes transitions at each new data-point and activities span multiple data-points, many transitions are done to the same state. Therefore, the basic HMM does not consider longer dependencies.

We add the MCM to the ADL recognition system to capture longer dependencies of activities. Therefore, we define a new sequence of activities, which is derived from the sequence of activities used in the basic HMM recognition by removing repetitions.

For example, if the activities *meal preparation*, *eating*, and *watching TV* appear in the dataset, they span several sensor events. Our basic sequence of activities $\mathcal{X}$, therefore, contains repetitions. By removing repetitions from this sequence and building an MCM, we model actual sequences of activities.

The sequence of activities from the basic HMM is denoted with $\mathcal{X} = X_1, \ldots, X_T$. Let us denote the new sequence with $\mathcal{X}' = X'_1, \ldots, X'_{T'}$. Since the new sequence has repetitions removed, it is shorter: $T' \leq T$.

We combine scores from the HMM, described in (12), and the MCM, described in (6), into a new cost function. Note that (6) is now used for the sequence $\mathcal{X}'$.

We also introduce the activity transition cost (ATC), which adds a fixed probability factor for each transition between activities in the sequence $\mathcal{X}'$. The new cost function is then

$$P^* = P_{HMM} \cdot P_{MCM} \cdot ATC^{T'}, \qquad (15)$$

where P_{MCM} and P_{HMM} are the cost functions from (6) and (12), respectively.

In the practical implementation, log-probabilities are used, since the use of standard probabilities might exceed the value range of variable types in computer systems. The cost function then becomes:

$$\log P^* = \log P_{HMM} + \log P_{MCM} + \gamma \cdot T', \qquad (16)$$

where $\gamma = \log ATC$. As one cannot expect both models to have the same effect on the recognition accuracy, weights to both probability estimates were added. Hence, the final cost function is defined as:

$$\begin{aligned}\log P^* &= \alpha \cdot \log P_{HMM} + \beta \cdot \log P_{MCM} + \gamma \cdot T' \\ &= \alpha \cdot \log \left(\pi_{X_0} \cdot \prod_{t=1}^{T} \lambda_{X_{t-1},X_t} \cdot \prod_{t=0}^{T} \phi_{X_t}(Y_t) \right) \\ &\quad + \beta \cdot \log \left(\pi'_{X'_0} \cdot \prod_{t=1}^{T'} \lambda'_{X'_{t-1},X'_t} \right) + \gamma \cdot T'. \end{aligned} \tag{17}$$

The objective of the decoder is to find the state sequence which maximizes the new cost function:

$$\hat{\mathcal{X}} = \underset{\mathcal{X}}{\arg\max} \left(\log P^* \right). \tag{18}$$

The use of an MCM and the ATC was inspired from the field of continuous speech recognition. For several years, HMMs were used as acoustical models for phonemes, n-gram models (MCMs of order $n-1$) were used as language models, and the so-called word insertion penalty was used to counteract the system's tendency to prefer a larger number of shorter words over a smaller number of longer words.

D. PARAMETER OPTIMIZATION

We can look at the cost function (17) as a function of 3 scores: The HMM probability, the MCM probability, and the number of non-repeated activities in the sequence. Those scores are combined as a weighted sum, using 3 weights: α, β, and ATC.

An optimization procedure must be used to find the optimal values of the weights. This is done during the parameter optimization steps using the development set. For optimization, one of the weights, say α, can be fixed to a constant value (normally 1), resulting in 2 free variables for optimization: the MCM weight β and the ATC value.

The MCMs can have different orders, and different additive counts for the start and transition probabilities can be used. We can determine optimal values for the additive counts by using MCMs to estimate the activity sequences in the development set.

E. THE MODIFIED VITERBI DECODER

To use the new cost function, we must modify the standard Viterbi algorithm to incorporate the MCM and the ATC into the recognition system. Fig. 1b shows a graphical representation of state transitions in the modified algorithm, and a comparison to the standard Viterbi algorithm (Fig. 1a) on an HMM with three states.

In the modified Viterbi algorithm, there are several possibilities for the best-path- probability at each state. We call these possibilities hypotheses. Let us denote the best-path-probability of hypothesis h in state j at time t with $\delta_t(j, h)$.

Hypotheses are not necessary for first-order MCMs, where the state transition probabilities are conditional only on the immediately preceding state. The MCM transition probability can be multiplied with the HMM transition probabilities.

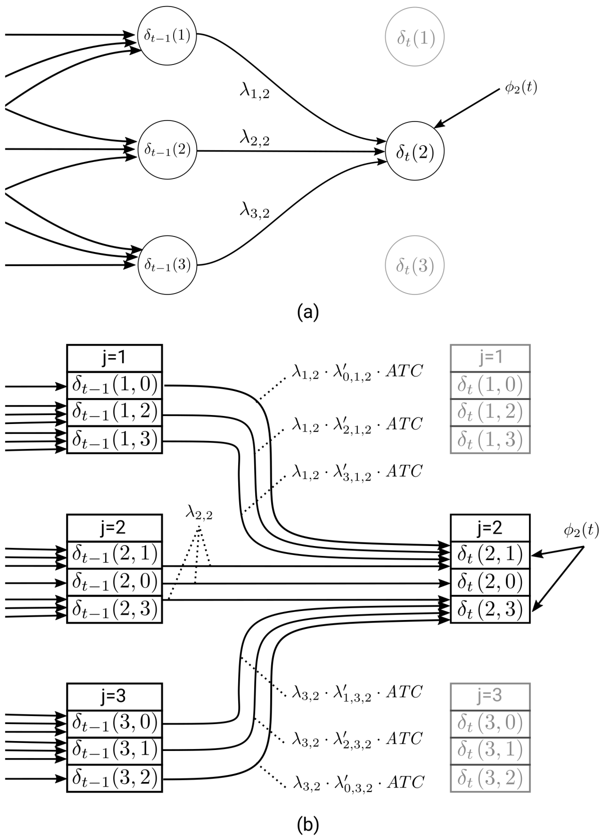

FIGURE 1. A graphical representation of the main part of (a) the standard Viterbi algorithm and (b) the modified Viterbi algorithm for an HMM with three states.

Similarly, the state start probabilities can be multiplied. Thus, the standard Viterbi decoder can be used in this case.

Hypotheses are necessary for second-order and higher-order MCMs. Let us consider a transition from state i to state j. The transition probability is conditional on more than just the immediately preceding state i. The set of different hypotheses at state i must account for all sequences without repetitions of $O-1$ states, where O is the MCM's order. If a state sequence in the hypothesis ends with the same state as the state to which this hypothesis belongs, it is also excluded, since it would generate a repetition at the current transition.

Considering repetitions in the state sequence is not necessary, since the definition of the activity sequence $\mathcal{X}'$ excludes repetitions. However, one must also consider shorter sequences of activities in the set of hypotheses to account for the first few transitions in $\mathcal{X}'$, where the state history is shorter than O. We can derive the number of necessary hypotheses as

$$H = \prod_{o=1}^{O} (N-1)^{o-1}. \tag{19}$$

In the example in Fig. 1 we have three hypotheses at each state. The hypotheses are numbered in a way that their number h represents the state from $\mathcal{X}'$ to be considered by the MCM, besides i and j. The MCM transition probability $\lambda'_{h,i,j}$ is the probability of state j occurring in the sequence $\mathcal{X}'$, conditional on the previous two states being h and i. The state-number 0 indicates shorter histories at the beginning of $\mathcal{X}'$.

Examining Fig. 1b, we can determine all possible transition into each hypothesis of a given state.

- Any transition from a state to the same state has one possible transition for each hypothesis. Since there is no transition in $\mathcal{X}'$ in this case, the transitions are always from a hypothesis to the same hypothesis, and the transitions consider only the HMM transition probability. See the middle three transitions in Fig. 1b.
- Hypothesis 0 indicates that there were no transitions in the sequence $\mathcal{X}'$ before – the path was never in any other state. There are no other possible transitions into this state, as this would contradict the hypothesis. See the very middle transition in Fig. 1b.
- Other hypotheses can have transitions into them from other states. Let us consider transitions to state j into hypothesis h. Since the hypothesis number h is used as an indicator of the previous state, transitions to this hypothesis can only come from state h. For example, transitions into hypothesis 1 come only from state 1 (see the top three transitions in Fig. 1b). In those transitions, we consider the HMM's transition probability λ, the MCM's transition probability λ', and the ATC.
- Regardless of state and hypothesis, the emission probability is taken into account the same way as in the standard Viterbi algorithm.
- No state j has a hypothesis h with the same number ($j \neq h$). This would mean a repetition in $\mathcal{X}'$ and contradict its definition.
- Additionally, at time 0, only transitions into hypothesis 0 are possible, as there are no previous states.

Considering the stated rules and restrictions for state/hypothesis transitions, we derive the modified Viterbi algorithm, presented in Algorithm 2.

The algorithm starts at time 0 and determines best-path-probabilities for hypothesis 0 of all states (line 2) using the HMM start probability (π), the HMM emission probability (ϕ), the MCM start probability (λ'), and the ATC. All other hypotheses are assigned a probability of 0 (line 4), making them inactive – any transition from them will again result in a zero probability.

Next, we determine best-path-probabilities for all other time-steps t for all states j. Determining the probability for hypothesis 0 is always done directly as a transition from the same hypothesis (line 9). Accordingly, the backtrack pointer ψ is set (line 10). The pointer in the modified algorithm must contain not only the state, but also the hypothesis at the previous time-step.

For all other hypotheses h, we can first set the best-path-probability and backtrack as a transition from the same hypothesis (lines 12 and 13). Then, we consider transitions from state h. If we find a higher best-path-probability (line 16), we correct its current value and the backtrack pointer (lines 17 and 18).

Algorithm 2 The Modified Viterbi Algorithm Using an HMM, a Second-Order MCM, and the ATC

```
 1: for all $j$ from $1 \leq j \leq N$ do
 2:     $\delta_0(j, 0) \leftarrow \pi_j \cdot \phi_j(Y_0) \cdot \pi'_j \cdot ATC$
 3:     for all $h$ from $1 \leq h \leq H$ do
 4:         $\delta_0(j, h) \leftarrow 0$
 5:     end for
 6: end for
 7: for all $t$ from $1 \leq t \leq T$ do
 8:     for all $j$ from $1 \leq j \leq N$ do
 9:         $\delta_t(j, 0) \leftarrow \delta_{t-1}(j, 0) \cdot \lambda_{j,j} \cdot \phi_j(Y_T)$
10:         $\psi_t(j, 0) \leftarrow (j, 0)$
11:         for all $h$ from $1 \leq h \leq H, h \neq j$ do
12:             $\delta_t(j, h) \leftarrow \delta_{t-1}(j, h) \cdot \lambda_{j,j} \cdot \phi_j(Y_t)$
13:             $\psi_t(j, h) \leftarrow (j, h)$
14:             for all $h'$ from $0 \leq h' \leq H, h' \neq h$ do
15:                 $\delta' \leftarrow \delta_{t-1}(h, h') \cdot \lambda_{h,j} \cdot \phi_j(Y_t) \cdot \lambda'_{h',h,j} \cdot ATC$
16:                 if $\delta' > \delta_t(j, h)$ then
17:                     $\delta_t(j, h) \leftarrow \delta'$
18:                     $\psi_t(j, h) \leftarrow (h, h')$
19:                 end if
20:             end for
21:         end for
22:     end for
23: end for
```

After the main part of the decoder completes the modified Viterbi algorithm, the highest best-path-probability at the last time-index is found, and, from there, the state sequence is determined using the backtrack pointers.

It should be noted that it would be possible to construct an equivalent HMM with $N \cdot \prod_{o=1}^{O}(N-1)^{o-1}$ states representing the hypotheses. In this case, the standard Viterbi decoder can be used. However, this would require the combination of the HMM, MCM, and the ATC into one HMM – a process we estimate to be similarly complex as the modified Viterbi algorithm.

V. RESULTS

A. ADDITIVE SMOOTHING

Before training the final HMMs and MCMs for activity recognition, we tested different additive count for the probability smoothing in these models.

We tested the performance of MCMs by estimating the probability of the activity sequence in the development set. The obtained log-probabilities are listed in Table 2. We also listed results with a zero-order model, which represents only the probabilities of activities themselves, without considering previous activities. Since zero-order models give significantly worse results, the results do show a dependency between consecutive activities.

The highest log-probability (-126.69) belongs to a model with a 2nd order MCM, which was built using an additive count of 0.1. Since higher probabilities indicate that the model fits the development set data better, we consider this

TABLE 2. Log-probabilities on the development set of MCMs with different additive counts and of different orders. The best result is noted in bold.

Additive count	MCM order					
	0	1	2	3	4	5
0.1	-192.38	-129.44	**-126.69**	-126.71	-129.55	-130.63
0.2	-191.15	-129.44	-128.25	-131.21	-135.57	-137.75
0.3	-190.42	-130.50	-131.30	-136.61	-142.02	-144.86
0.5	-189.50	-133.25	-137.46	-145.92	-152.48	-155.97
0.7	-188.92	-136.04	-142.85	-153.24	-160.33	-164.07
1.0	-188.33	-139.92	-149.59	-161.66	-169.03	-172.87
1.2	-188.06	-142.28	-153.37	-166.11	-173.51	-177.33
1.5	-187.76	-145.50	-158.24	-171.61	-178.93	-182.68
2.0	-187.44	-150.21	-164.83	-178.67	-185.72	-189.27
3.0	-187.23	-157.77	-174.40	-188.25	-194.64	-197.80
5.0	-187.53	-168.55	-186.34	-199.20	-204.48	-207.04

model to be the best performing. Therefore, it was used in all further experiments.

The results show that higher-order models perform less well on the development set. We assume that the activities do not exhibit dependencies over distances above a certain limit. Also, worse model performance can be a result of data sparsity for high-order model estimation, since the number of parameters in the model increases exponentially with model order.

Next, we can conclude that only a minimal additive smoothing constant is necessary for the model, as higher values slowly decrease the performance of the model.

We found that the same method of optimizing additive counts cannot be applied for the state and emission probabilities in the HMM. In an attempt to optimize these values, we found that increasing them towards infinity results in a better estimation score using (10). However, this results in an HMM with no discriminative ability in the decoding process – the probabilities of all possible paths would approach the same value. Still, a minimum additive count must be applied to avoid zero probabilities. We used an additive count of 1 for all counts during the estimation of the HMM's parameters. This minimum count can be higher than the count for MCMs, as there is a significantly higher number of counts for the HMM parameter estimation.

B. THE NAÏVE BAYES AND HMM CLASSIFIERS

The second column in Table 3 shows recognition accuracy results for the naïve Bayes classifier. Since no parameter optimization for this classifier was used, only final results on the evaluation set are presented.

The Table gives individual accuracies for the 8 activities present in the dataset, an average or total score across all individual activities in the dataset (*Total*), and an accuracy score of the combination of all concurrent activities (*Activity combination*). The combination accuracy is the proportion of data-points where all activities were recognized correctly at the same time. We see that the overall accuracy using the naïve Bayes classifier is 92.54 %, whereas the combination accuracy is 59.01 %.

While the overall accuracy can be considered good, further analysis of the recognition outputs shows 12,562 predicted activity transitions, whereas the actual evaluation set has only 141 actual activity transitions.

TABLE 3. Accuracy results on the evaluation set for the naïve Bayes classifier and the HMM classifier interpolated with the naïve Bayes classifier. Shown are accuracies for individual activities, the total accuracy of all individual activities, and the combined activity accuracy. The number of predicted activity transitions is 12,562 using the naïve Bayes classifier and 741 using the HMM classifier.

Activity	Accuracy (Naïve Bayes) [%]	Accuracy (HMM) [%]
Bed to Toilet Transition	99.35	99.66
Cleaning	97.79	98.37
Cooking	77.29	85.89
Grooming	97.12	97.44
Showering	95.92	95.74
Sleeping	91.07	90.83
Wakeup	96.81	98.26
Working	85.00	85.75
Total	92.54	93.99
Activity combination	59.01	68.89

First experiments with the basic HMM decoder showed a significant drop in recognition accuracy. We suspected the reason for this to be the highly unbalanced nature of the activities in the datasets. Therefore, we interpolated the HMM with the naïve Bayes classifier by including the state probability from the Bayes classifier ($P(x)$) into each state in the decoding process.

The results of the interpolated classifier are shown in the third column in Table 3. We see an increase in accuracy with most activities, most notably for *cooking* and for the combined activity score. With the interpolated model, the results showed 741 predicted activity transitions in the decoder output, which is still over five times the amount of actual activity transitions in the evaluation set.

C. INCLUSION OF THE ACTIVITY TRANSITION COST

Next, the ATC was added to the decoding algorithm. Given the fact that we want to reduce the amount of activity transition, and because of the formulation of γ in (16), the ATC value must be a negative number. To find the optimal value, we tested several values (6 values per decade from -0.1 to -100) and compared accuracy results and the number of activity transitions on the development set. The results are shown in Fig. 2.

We tested the performance on the development set regarding the total accuracy, the combined accuracy, and the number of predicted activity transitions, compared to the number of actual activity transitions in the development set, which is 161.

Examining the result, we decided to use -33 as an optimized ATC value, as a compromise regarding the three graphs in Fig. 2. The selected ATC value gives the best result of the combined accuracy, and the second-best result of the total accuracy. Regarding the number of predicted activity transitions, the results are considered to be better if they are closer to the number of actual activity transitions. The selected optimal ATC value gives a better result than selecting the ATC value, which gives the best total accuracy.

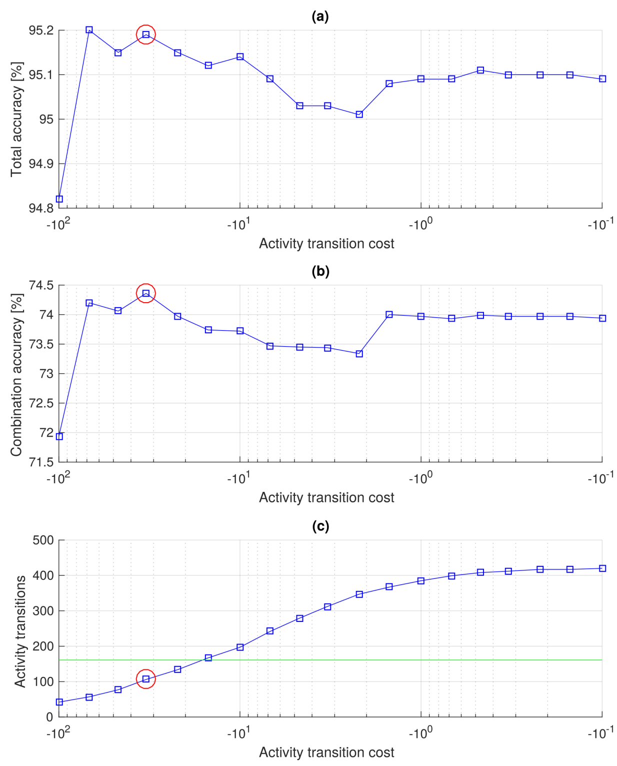

FIGURE 2. ACT optimization results on the development set in terms of (a) the total accuracy over individual activities, (b) the combination accuracy of all activities, and (c) the number of predicted activity transitions (blue) compared with the number of actual activity transitions in the development set (green). Results with the selected optimal value are marked in red.

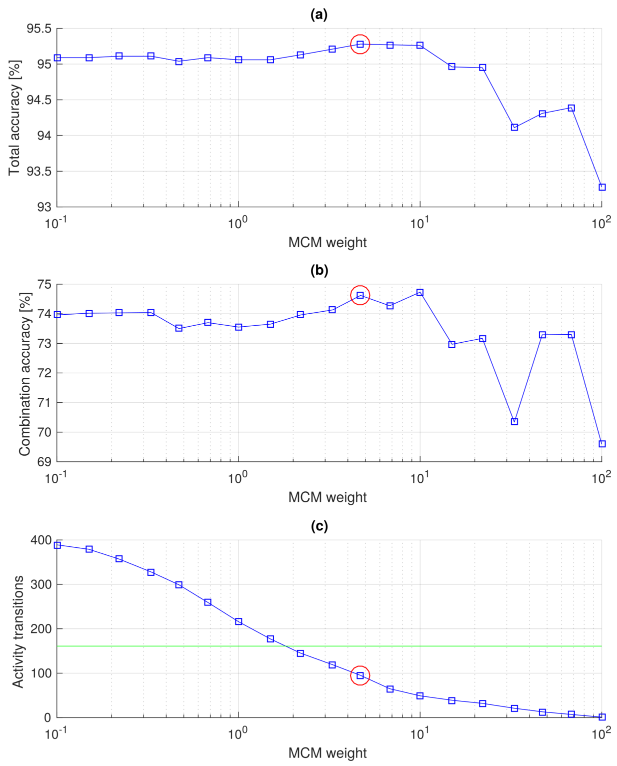

FIGURE 3. MCM weight (β) optimization results on the development set in terms of (a) the total accuracy over individual activities, (b) the combination accuracy of all activities, and (c) the number of predicted activity transitions (blue) compared with the number of actual activity transitions in the development set (green). Results with the selected optimal weight are marked in red.

TABLE 4. Accuracy results on the evaluation set for the HMM classifier extended with the ATC. The number of predicted activity transitions with these results is 136.

Activity	Accuracy [%]
Bed to Toilet Transition	99.72
Cleaning	99.09
Cooking	85.59
Grooming	97.57
Showering	96.25
Sleeping	90.68
Wakeup	98.60
Working	86.15
Total	94.20
Activity combination	69.40

This optimal ATC value was then used on the evaluation set, obtaining the results in Table 4. Again, one can see an improvement in accuracy with most activities, including the total and combined accuracy.

The most significant difference is in the number of predicted activity transitions, which is now reduced to 136. That result is much closer to the 141 actual activity transitions in the evaluation set.

D. INCLUSION OF THE MARKOV CHAIN MODEL

Next, we added the MCM model into the recognition system, firstly, without the ATC. We repeated the optimization processes from the ATC optimization, albeit with positive values. The results are shown in Fig. 3.

The MCM introduces probabilities for each transition between activities, similar to the ATC. However, transition probabilities between activities in the MCM have different values, whereas the ATC is constant. Nevertheless, there is a similar effect on the number of transitions in the recognition output.

Examining the results, we decided to use 4.7 as the optimal MCM weight. Again, this was done as a compromise regarding the three graphs in Fig. 3. The selected MCM weight gives the best result on the total accuracy and the second-best result on the combined accuracy. This values also gives a better result of predicted activity transition than selecting the best result on the combined accuracy. Using the selected MCM weight on the evaluation set, the results in Table 5 were obtained. The results, again, show an improvement over the basic HMM recognized and are very similar to the results of using the ATC.

An examination of the MCM shows that most log-probabilities in the model have values between -6 and -12. These values are added at each activity transition to the cost function. Hence, it is not surprising that the ratio between the optimal ATC value and the optimal MCM weight is in the same range ($-33/4.7 \approx -7$).

TABLE 5. Accuracy results on the evaluation set for the HMM classifier extened with the MCM. The number of predicted activity transitions with these results is 130.

Activity	Accuracy [%]
Bed to Toilet Transition	99.90
Cleaning	99.18
Cooking	85.36
Grooming	97.57
Showering	96.27
Sleeping	90.70
Wakeup	98.75
Working	86.24
Total	94.25
Activity combination	69.50

TABLE 6. Accuracy results on the evaluation set for the HMM classifier with the ATC and the MCM. The number of transitions with these results is 236.

Activity	Accuracy [%]
Bed to Toilet Transition	100.00
Cleaning	99.34
Cooking	85.06
Grooming	97.57
Showering	96.71
Sleeping	91.19
Wakeup	99.01
Working	86.35
Total	94.53
Activity combination	70.95

E. COMBINATION OF THE ACTIVITY TRANSITION COST AND THE MARKOV CHAIN MODEL

Finally, we included both the MCM and the ATC into the HMM recognition algorithm. Since both of them affect the number of predicted activity transitions in the results, we expected an interdependence between the optimal MCM weight and the optimal ATC value.

We expected optimal MCM values to be positive since the results from the previous subsection show improvement when adding the MCM. Like the negative values of the ATC, the positive values of the MCM also cause a tendency to lower the number of predicted activity transitions in the recognition output. We suspected that this would cause an increase in the optimal value of the ATC. Given high enough MCM weights, optimal ATC values might even become positive numbers. Therefore, positive and negative values have to be tested while searching for optimal weights on the development set.

We found that an MCM weight of 220 and a positive ATC of 33 give the best accuracy results on the development set, both in terms of the total accuracy (95.51 %) and the combined accuracy (76.27 %). The number of predicted activity transitions in the recognition output was 204, whereas the number of actual activity transitions in the development set is 165.

We then used these weights in the recognition system on the evaluation set. The results are shown in Table 6. We again see an increase in recognition accuracy on most activities when compared to previous results.

Table 7 shows a summarization of the results. We see that the inclusion of both the MCM and the ATC gives the best results in terms of accuracy, whereas adding only the ATC to the systems gives the best result in terms of the number of predicted activity transitions.

TABLE 7. A summary of the recognition results with different models on the evaluation set.

Model	Total accuracy [%]	Combined accuracy [%]	No. of transitions
Naïve Bayes	92.54	59.01	12,562
HMM	93.99	68.89	741
HMM + ATC	94.20	69.40	**136**
HMM + MCM	94.25	69.50	130
HMM + ATC + MCM	**94.52**	**70.95**	236

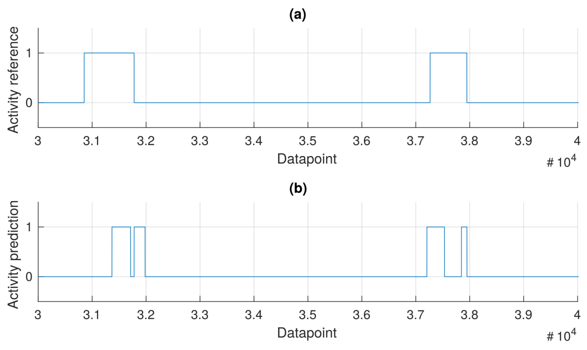

FIGURE 4. Partial timeline of the activity *sleeping* in the evaluation set as (a) the reference annotation in the dataset and (b) the predicted activity using the HMM recognizer with the ATC and the MCM.

A closer examination of the recognition results shows a plausible explanation of why the best accuracy results occur with a higher amount of predicted activity transitions. Fig. 4b shows a typical recognition example with comparison to the reference annotation in Fig. 4a. We see brief interruptions in the recognized activity as a possible reason for an excess amount of predicted activity transitions in the recognition output.

Taking an emphasis on the recognition accuracy rather than the number of predicted activities, we consider the model using the HMM, the ATC, and the MCM to be the best performing.

Comparing the obtained results with previously published research, one can see that comparable accuracy results in the same area of approximately 90 % were achieved. However, for a direct comparison, one would need a experimental setup using the same dataset, with the same data separation (into the train, development and evaluation data), and the same implementation of evaluation metric. Although research using the CASAS datasets is published, implementation details are usually not publicly available to that extent.

Therefore, we emphasized only the comparison between the basic HMM-based system and the extended system, as this was the principal intent of this paper.

VI. CONCLUSION

In this article, we presented two extensions of HMM-based ADL recognition, as well as an extension to the Viterbi algorithm to use the proposed models. The use of an MCM of activities enables us to model longer dependencies in the recognition algorithm, and the use of the ATC enables us to prevent an excess amount of predicted transitions.

Both extensions showed improvements in recognition accuracy when being used as individual extensions to the HMM-based system, whereas the use of both extensions simultaneously gave the overall best results.

Our approach considered the use of one HMM, with several states representing combinations of all possible activities. Possible future work on this approach includes the examination of whether the proposed extensions can be applied to a system with separate models for separate activities.

Also, the examinations of the timeline representation of the recognition results suggest the need for a more comprehensive evaluation of ADL recognition results, rather than using the accuracy score alone.

Another future research direction is the examination of whether HMMs with time-depended transition probabilities can be used to prevent brief interruptions of predicted activities in the results.

ACKNOWLEDGMENT

The authors would like to thank the authors of the CASAS datasets for making their data publicly available.

REFERENCES

[1] B.-C. Cheng, Y.-A. Tsai, G.-T. Liao, and E.-S. Byeon, "HMM machine learning and inference for activities of daily living recognition," *J. Supercomput.*, vol. 54, no. 1, pp. 29–42, Oct. 2010. doi: 10.1007/s11227-009-0335-0.

[2] S. Karaman, J. B.-Pineau, V. Dovgalecs, R. Mégret, J. Pinquier, R. André-Obrecht, Y. Gaëstel, and J.-F. Dartigues, "Hierarchical Hidden Markov Model in detecting activities of daily living in wearable videos for studies of dementia," *Multimedia Tools Appl.*, vol. 69, no. 3, pp. 743–771, Apr. 2014. doi: 10.1007/s11042-012-1117-x.

[3] L. Lu, C. Qing-Ling, and Z. Yi-Ju, "Activity recognition in smart homes," *Multimedia Tools Appl.*, vol. 76, pp. 24203–24220, Nov. 2017. doi: 10.1007/s11042-016-4197-1.

[4] H. Wei, J. He, and J. Tan, "Layered hidden Markov models for real-time daily activity monitoring using body sensor networks," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 479–494, Nov. 2011. doi: 10.1007/s10115-011-0423-3.

[5] C. Zhu and W. Sheng, "Recognizing human daily activity using a single inertial sensor," in *Proc. 8th World Congr. Intell. Control Automat.*, Jinan, China, Jul. 2010, pp. 282–287.

[6] S. Zhang, S. I. McClean, and B. W. Scotney, "Probabilistic learning from incomplete data for recognition of activities of daily living in smart homes," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 3, pp. 454–462, May 2012. doi: 10.1109/TITB.2012.2188534.

[7] S. S. Chawathe, "Recognizing activities of daily living using binary sensors," in *Proc. 4th Int. Conf. Universal Village (UV)*, Boston, MA, USA, Oct. 2018, pp. 1–6.

[8] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods Inf. Med.*, vol. 48, no. 5, pp. 480–485, May 2009. doi: 10.3414/ME0592.

[9] D. J. Cook, M. Schmitter-Edgecombe, A. Crandall, C. Sanders, and B. Thomas, "Collecting and disseminating smart home sensor data in the CASAS project," in *Proc. CHI Workshop Develop. Shared Home Behav. Datasets Advance HCI Ubiquitous Comput. Res.*, Boston, MA, USA, Apr. 2009, pp. 1–7.

[10] J. Rowan and E. D. Mynatt, "Digital family portrait field trial: Support for aging in place," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Portland, OR, USA, Apr. 2005, pp. 521–530.

[11] K. Avgerinakis, A. Briassouli, and I. Kompatsiaris, "Recognition of activities of daily living for smart home environments," in *Proc. 9th Int. Conf. Intell. Environ.*, Athens, Greece, Jul. 2013, pp. 173–180.

[12] J. Fogarty, C. Au, and S. E. Hudson, "Sensing from the basement: A feasibility study of unobtrusive and low-cost home activity recognition," in *Proc. 19th Annu. ACM Symp. User Interface Softw. Technol.*, Montreux, Switzerland, Oct. 2006, pp. 91–100.

[13] K. Zhan, F. Ramos, and S. Faux, "Activity recognition from a wearable camera," in *Proc. 12th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Guangzhou, China, Dec. 2012, pp. 365–370.

[14] H. Wu, W. Pan, X. Xiong, and S. Xu, "Human activity recognition based on the combined SVM&HMM," in *Proc. IEEE Int. Conf. Inf. Automat. (ICIA)*, Hailar, China, Jul. 2014, pp. 219–224.

[15] I. M. Pires, N. M. Garcia, N. Pombo, and F. Flórez-Revuelta, S. Spinsante, and M. C. Teixeira, "Identification of activities of daily living through data fusion on motion and magnetic sensors embedded on mobile devices," *Pervas. Mobile Comput.*, vol. 47, pp. 78–93, Jul. 2018. doi: 10.1016/j.pmcj.2018.05.005.

[16] B. Chikhaoui, S. Wang, and H. Pigot, "A frequent pattern mining approach for ADLs recognition in smart environments," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Singapore, Mar. 2011, pp. 248–255.

[17] P. Urwyler, L. Rampa, R. Stucki, M. Büchler, R. Müri, U. P. Mosimann, and T. Nef, "Recognition of activities of daily living in healthy subjects using two ad-hoc classifiers," *Biomed. Eng. Online*, vol. 14, Jun. 2015, Art. no. 24. doi: 10.1186/s12938-015-0050-4.

[18] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Syst. Appl.*, vol. 105, pp. 233–261, Sep. 2018. doi: 10.1016/j.eswa.2018.03.056.

[19] E. Zdravevski, P. Lameski, V. Trajkovik, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, and N. Garcia, "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering," *IEEE Access*, vol. 5, pp. 5262–5280, 2017. doi: 10.1109/ACCESS.2017.2684913.

[20] T.-H. Tan, M. Gochoo, F.-R. Jean, S.-C. Huang, and S. Kuo, "Front-door event classification algorithm for elderly people living alone in smart house using wireless binary sensors," *IEEE Access*, vol. 5, pp. 10734–10743, 2017. doi: 10.1109/ACCESS.2017.2711495.

[21] P. Gupta and T. Dallas, "Feature selection and activity recognition system using a single triaxial accelerometer," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 6, pp. 1780–1786, Jun. 2014. doi: 10.1109/TBME.2014.2307069.

[22] A. Fleury, M. Vacher, and N. Noury, "SVM-based multimodal classification of activities of daily living in health smart homes: Sensors, algorithms, and first experimental results," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 274–283, Mar. 2010. doi: 10.1109/TITB.2009.2037317.

[23] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *Proc. 6th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Honolulu, HI, USA, May 2007, pp. 1–8.

[24] H. Zhu, H. Chen, and R. Brown, "A sequence-to-sequence model-based deep learning approach for recognizing activity of daily living for senior care," *J. Biomed. Informat.*, vol. 84, pp. 148–158, Aug. 2018. doi: 10.1016/j.jbi.2018.07.006.

[25] L. Zhang, X. Wu, and D. Luo, "Human activity recognition with HMM-DNN model," in *Proc. IEEE 14th Int. Conf. Cognit. Informat. Cognit. Comput. (ICCICC)*, Beijing, China, Jul. 2015, pp. 192–197.

[26] S. Okour, A. Maeder, and J. Basilakis, "An adaptive rule-based approach to classifying activities of daily living," in *Proc. Int. Conf. Healthcare Informat.*, Dallas, TX, USA, Oct. 2015, pp. 404–407.

[27] I. K. Ihianle, U. Naeem, and A.-R. Tawil, "Recognition of activities of daily living from topic model," *Procedia Comput. Sci.*, vol. 98, pp. 24–31, Jan. 2016. doi: 10.1016/j.procs.2016.09.007.

[28] T. Magherini, A. Fantechi, C. D. Nugent, and E. Vicario, "Using temporal logic and model checking in automated recognition of human activities for ambient-assisted living," *IEEE Trans. Human-Mach. Syst.*, vol. 43, no. 6, pp. 509–521, Nov. 2013. doi: 10.1109/TSMC.2013.2283661.

[29] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2004, pp. 1–17.

[30] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies," *Int. J. Biosci., Psychiatry Technol.*, vol. 1, no. 1, pp. 25–35, Jan. 2009.

[31] H. Fang, L. He, H. Si, P. Liu, and X. Xie, "Human activity recognition based on feature selection in smart home using back-propagation algorithm," *ISA Trans.*, vol. 53, no. 5, pp. 1629–1638, Sep. 2014. doi: 10.1016/j.isatra.2014.06.008.

[32] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967. doi: 10.1109/TIT.1967.1054010.

[33] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," in *Proc. Int. Comput. Sci. Inst.*, UC Berkeley, Berkeley, CA, USA, Apr. 1997, pp. 7–11.

GREGOR DONAJ received the B.Sc. degree in electrical engineering, the B.Sc. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of Maribor, Slovenia, in 2010, 2011, and 2015 respectively.

From 2010 to 2015, he was a Young Researcher with the Faculty of Electrical Engineering and Computer Science, University of Maribor, under the supervision of mentor Dr. Z. Kačič, where he is currently a Teaching Assistant in the study programs electrical engineering and telecommunications with the Faculty of Electrical Engineering and Computer Science. His research interests include pattern recognition in various areas, including speech recognition, machine translation, and computer–brain interfaces, as well as statistical analysis of the performance of pattern recognition in those areas.

MIRJAM SEPESY MAUČEC (M'11) received the B.Sc. and Ph.D. degrees in computer science from the Faculty of Electrical Engineering and Computer Science, University of Maribor, in 1996 and 2001, respectively. From 1996 to 1997, she was a Researcher with the Center for Interdisciplinary and Multidisciplinary Research and Studies, University of Maribor, where she was a Young Researcher with the Faculty of Electrical Engineering and Computer Science, under the supervision of mentor Dr. Z. Kačič, from 1997 to 2001. In 2010, she became an Associate Professor in telecommunications. She has been involved in several projects and in national research program Advanced Methods of Interaction in Telecommunication. She participated in several international and national research projects. Her research interests include machine translation, computational linguistics, and evolutionary computing.