Recognising lifestyle activities of diabetic patients with a smartphone

Mitja Luštrek, Božidara Cvetković, Violeta Mirchevska
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
Email: {mitja.lustrek, boza.cvetkovic}@ijs.si

Özgür Kafalı, Alfonso E. Romero, Kostas Stathis
Department of Computer Science
Royal Holloway, University of London
Egham TW20 0EX UK
Email: {ozgur.kafali, aeromero, kostas.stathis}@rhul.ac.uk

***Abstract*—Diabetes is both heavily affected by the patients' lifestyle, and it affects their lifestyle. Most diabetic patients can manage the disease without technological assistance, so we should not burden them with technology unnecessarily, but lifestyle-monitoring technology can still be beneficial both for patients and their physicians. Because of that we developed an approach to lifestyle monitoring that uses the smartphone, which most patients already have. The approach consists of three steps. First, a number of features are extracted from the data acquired by smartphone sensors, such as the user's location from GPS coordinates and visible wi-fi access points, and the physical activity from accelerometer data. Second, several classifiers trained by machine learning are used to recognise the user's activity, such as work, exercise or eating. And third, these activities are refined by symbolic reasoning encoded in Event Calculus. The approach was trained and tested on five people who recorded their activities for two weeks each. Its classification accuracy was 0.88.**

***Keywords*—diabetes, lifestyle, activity recognition, smartphone, sensors**

I. Introduction

According to International Diabetes Federation, 5.6 % of the global population suffer from diabetes, and this figure is increasing [1]. In diabetic patients, the pancreas does not produce enough insulin to absorb glucose from the blood, or the cells in the body do not respond appropriately to the insulin produced. Since diabetes cannot be cured, it has to be managed through appropriate lifestyle and medication, often involving injections of insulin. Key activities for diabetic patients are eating and exercise – the former because it puts glucose in the blood and the latter because it speeds up its absorption. Therefore they have to monitor and manage these activities very carefully.

Lifestyle management can be assisted by technology – either to help the patients track their activities and offer advice, or by providing their physicians a better insight into the patients' lifestyle through telemonitoring. However, since most patients can manage diabetes reasonably well on their own, any technology used must not place undue burden on them, otherwise it will do more harm than good. In the COMMODITY$_{12}$ project [2], we are developing a personal health system for continuous multi-parametric analysis of diabetes intended both for the patients and their physicians. In order not to burden the patients with unnecessary devices, we are primarily relying on a smartphone for lifestyle monitoring.

In this paper we describe an approach to recognise basic lifestyle activities with the sensors built into the smartphone. The phone can be augmented with an optional ECG monitor, which was introduced for the management of cardiovascular co-morbidities [3]. The key features extracted from sensor data are the user's location, physical activity and ambient sound. These are fed into a number of classifiers trained with machine learning that output the user's activity. The recognised activities are refined by rules encoded in Event Calculus [4]. The approach was trained and tested on five people who recorded their activities for two weeks each.

The rest of the paper is organised as follows. Section II provides an overview of related work. Section III presents first the machine-learning and then the rule-based symbolic approach to activity recognition. Section IV describes the experimental evaluation and its results. Section V concludes the paper.

II. Related Work

Activity recognition from sensor data is common in telemonitoring and personal health systems, because the patients' lifestyle plays an important role in many diseases, and because it provides the context for other health-related observations.

A. Machine-learning Approaches

Machine learning is often used for activity recognition from sensor data because it can easily deal with large amounts of data that are difficult to comprehend for humans. Wearable sensing devices are commonly utilised, but mostly to recognise elementary activities such as walking, sitting and lying. We have done the same in our previous work [5], and such elementary activities are among the features used as inputs to the approach described in this paper. The recognition of high-level lifestyle activities is less common. For example, Huynh et al. [6] proposed the use of two accelerometers to recognise daily activities such as eating, sleep and washing dishes, as well as to provide higher-level activity descriptions, such as morning activities, housework and shopping. Automatic activity recognition can also be based on ambient sensors. Helal et al. [7] developed a platform for monitoring diabetic patients using ambient sensors in a smart home. They used a Hidden Markov Model to recognise five daily activities: (1) look up a number in the phone book, dial the number, and write down the cooking instructions heard on the recording, (2) wash hands in

the kitchen sink, (3) cook a pot of oatmeal as specified from the phone directions, (4) eat the oatmeal while taking medicine, and (5) clean the dishes. In addition, they proposed automatic chewing detection by analysing the mouth region of video sequences. The recognition of eating is of particular interest to this paper due to its importance for diabetic patients. Several authors tackled automatic food-intake monitoring [8]. Their approaches involved monitoring the movement of the body while eating (with inertial, EMG and other sensors), recording chewing sounds and body temperature. These approaches can be fairly reliable, but they require rather specific sensors and could be physically uncomfortable.

B. Symbolic Approaches

An advantage of symbolic approaches for activity recognition is being able to specify activities in an intuitive way, only by using logical constraints, abstracting from the very low-level and irrelevant events, and therefore being more interpretable. There are several rule-based approaches for activity recognition [9], mainly dealing with problems related to video surveillance. Examples include an approach based on CSP (Constraint Satisfaction Problems) for vandalism monitoring [10], and an approach based on hierarchical scenarios (defined recursively as properties of the mobile objects shown in the scenario) for vehicle-tracking applications [11]. In order to allow for uncertain representations, approaches based on Markov-logic networks were presented by Hongeng et al. [12], where events were considered as compositions of different threads later recognised with the networks. Finally, Shet et al. [13] proposed an approach similar to the one used in this paper, with a low-level set of features and high-level Prolog rules to recognise the activities. It should be noted that several of the previously-mentioned approaches were not necessarily designed to work in real time. For instance, a video-surveillance method could be used to detect potentially interesting activities in a long video tape, and extract relevant segments to send to a human. In the case of health monitoring, the processing should be done in real time, as activity recognition can be crucial for the detection of a health problem while it is happening, allowing for immediate action. We thus used a framework built on Event Calculus [4], and optimized for speed and real-time purposes.

III. Activity-Recognition Approach

Our activity-recognition approach has three main steps described in the following three subsections: feature extraction, machine learning and symbolic reasoning. Feature extraction uses a range of methods to transforms raw sensor data into features that can be used for machine learning and symbolic reasoning about activities. The extraction of some features already requires machine learning, such as the recognition of elementary activities, while others are extracted using simple aggregation functions or heuristics. We still have a lot of data after feature extraction, not all of which is readily understandable by humans, so the next step is machine learning to recognise high-level lifestyle activities. Finally, we apply symbolic reasoning to the outputs of the previous two steps to refine the activities recognised using machine learning.

A. Features for Activity Recognition

Features are computed over one-minute windows and belong to six groups: sound (from the phone's microphone), wi-fi (from the phone's wi-fi module), GPS (from the phone's GPS receiver), acceleration (from the accelerometers in the phone and optionally in the ECG monitor), heart-rate (from the optional ECG monitor) and respiration-rate (from a sensor in the optional ECG monitor).

Sound features: To preserve the user's privacy, the sound from the smartphone microphone is recorded for 100 ms out of every second. To reduce the amount of data, after each minute of sound recording there is a five-minute pause. Sound features are computed in windows of size 20 ms within each one-minute window. We compute the spectral-centroid, zero-crossing, mel-frequency-cepstral-coefficient (MFCC), linear-predictive-coding (LPC) and method-of-moments values for each window within a minute (where recording is available). The sound features are averages of these values.

Wi-fi features: The wi-fi signal is used for extracting location identifiers. The location identifiers are obtained through hierarchical clustering on a training subset. The clustering uses the similarity with respect to visible wi-fi access points as the distance between pairs of one-minute windows – if they are similar, we consider that they are at the same location. As input, the function takes the signal strength (in dBm) of each basic service set identifier (BSSID, access-point identifier) for two time intervals – *instance1* and *instance2*. The length of these instances equals the number of BSSIDs scanned in the training data. The strength of a BSSID that was not scanned during a particular interval is set to the minimum value of the float data type in Java. The computation of the distance between two instances is performed as follows:

```
1:  Create vector visible1 of length(instance1);
2:  Create vector visible2 of length(instance2);
3:  for el from 1 to length(instance1) increment 1 do
4:      if instance1[el] < −500 then
5:          visible1[el] = false;
6:          instance1[el] = −500;
7:      else
8:          visible1[el] = true;
9:      end if
10:     if instance2[el] < −500 then
11:         visible2[el] = false;
12:         instance2[el] = −500;
13:     else
14:         visible2[el] = true;
15:     end if
16: end for
17: if no_common_element(visible1, visible2) then
18:     return max_value;
19: else
20:     return euclidean_distance(instance1, instance2);
21: end if
```

Lines 1–16: The set of visible BSSIDs in both instances is determined and stored in the vectors *visible1* and *visible2*. Typically, the wi-fi signal strength is in the range between -100 dBm and -20 dBm, where the first value indicates a weak signal and the second a strong signal. Considering the

typical signal strength range and our convention for annotating BSSIDs that were not scanned during a time interval, we consider a BSSID visible if its signal strength is above -500 dBm. In addition to this, the signal strength of the BSSIDs which are not visible is set to -500 dBm. With this we assign appropriate weights to differences in BSSID visibility and differences in signal strength in the distance function.

LINES 17–20: If the instances do not have a common BSSID, the maximum value of the float data type is returned. Otherwise the Euclidean distance between the vectors *instance1* and *instance2* is returned.

With the distance defined, we can proceed to the actual clustering. As input, the hierarchical clusterer obtains a stream of BSSID signal strengths in the training data annotated with a timestamp. It outputs a set of wi-fi location identifiers, as follows:

```
1: Create a vector training_bssids with all BSSIDs scanned
   in the training data;
2: Create an empty set dataset;
3: for all minute m in the training data do
4:    Create a vector instance of length(training_bssids);
5:    for all bssid in training_bssids do
6:       if bssid scanned in minute m then
7:          instance[bssid] = average_signal_strength(bssid);
8:       else
9:          instance[bssid] = min_value;
10:      end if
11:   end for
12:   add instance to dataset;
13: end for
14:
15: Create a hierarchical clusterer clusterer;
16: wifi_clusters = clusterer.cluster(dataset);
17:
18: Create an empty set locations;
19: Create an empty set loc_unknown;
20: for all cluster in wifi_clusters do
21:    if num_instances(cluster) > 20 then
22:       add cluster to locations;
23:    else
24:       add all instances in cluster to loc_unknown;
25:    end if
26: end for
27: add loc_unknown to locations;
28:
29: return locations;
```

LINE 1: The set of all distinct BSSIDs scanned in the training period is extracted – *training_bssids*.

LINES 2–13: The clustering dataset is created. One instance is created for each one-minute window in the training data. The instance length equals the size of *training_bssids*. The elements in each instance represent the average signal strength of each BSSID. If a BSSID was not scanned in a particular time interval, the minimum value of the float data type in Java is assigned.

LINES 15–16: The hierarchical clusterer using the previously defined distance function is created. The *dataset* is clustered, outputting a set of clusters *wifi_clusters*.

LINES 18–27: The set of wi-fi location identifiers *locations* is determined. All clusters whose size is over 20 instances are kept as location identifiers, while the clusters with size less than or equal to 20 instances are merged into a single unknown location.

LINE 33: The set of wi-fi location identifiers is returned.

GPS features: Three features are extracted from the GPS signal: (1) GPS location identifier, (2) velocity and (3) category of the nearest place.

GPS location identifiers are again obtained through hierarchical clustering on a training subset. The clustering uses the Euclidean distance between GPS coordinates between pairs of one-minute windows – if they are small, we consider that they are at the same location. As input, the hierarchical clusterer obtains a stream of GPS coordinates sensed in the training data annotated with a timestamp. It outputs a set of GPS location identifiers, as follows:

```
1: Create an empty set dataset;
2: for all minute m in the training data do
3:    Create a vector instance of length 2;
4:    instance[0] = average(latitude);
5:    instance[1] = average(longitude);
6:    add instance to dataset;
7: end for
8:
9: Create a hierarchical clusterer clusterer;
10: gps_clusters = clusterer.cluster(dataset);
11:
12: Create an empty set locations;
13: Create an empty set loc_unknown;
14: for all cluster in gps_clusters do
15:    if num_instances(cluster) > 20 then
16:       add cluster to locations;
17:    else
18:       add all instances in cluster to loc_unknown;
19:    end if
20: end for
21: add loc_unknown to locations;
22:
23: return locations;
```

LINES 1–7: The clustering dataset is created. One instance is created for each one-minute window in the training data. The instance length equals 2 – the first element represents the average latitude, while the second the average longitude.

LINES 9–10: The hierarchical clusterer is created using the Euclidean distance between GPS coordinates. The *dataset* is clustered, outputting a set of clusters *gps_clusters*.

LINES 12–21: The set of GPS locations *locations* is determined. All clusters whose size is over 20 instances are kept, while the rest are merged into a single unknown location.

LINE 23: The set of GPS location identifiers is returned.

The velocity is computed simply as the distance between the last and first GPS coordinates in a one-minute window, divided by difference in the times these two coordinates were obtained. The category of the nearest place is obtained from the Foursquare geolocation database using their API [14].

Acceleration features: Two features are extracted from the acceleration signal: elementary activity and energy-expenditure estimate. The methods for elementary activity recognition and energy-expenditure estimation are presented in our earlier work [5].

Heart-rate features: Three features are extracted: (1) minimum, (2) maximum and (3) average heart-rate within each one-minute window.

Respiration-rate features: Three features are extracted: (1) minimum, (2) maximum and (3) average respiration-rate within each one-minute window.

B. Machine Learning

Figure 1 shows the procedure for recognising lifestyle activities using machine learning. The following activities were intended to be recognised: sleep, home–chores, home–leisure, food preparation, eating, work, exercise, out–leisure, out–errands and transport. However, since some proved almost impossible to distinguish, the set was reduced to: sleep, home, eating, work, exercise, out and transport.

The procedure first determines when the user is at home and at work by comparing the wi-fi location to a set of wi-fi location values classified as home or work using the training data. A wi-fi location is added to the home set if at least 25 % of instances at that wi-fi location are during the weekend or at least 25 % of them are in the period 0:00–10:00 or 16:00–24:00 during work days. A wi-fi location is added to the work set if at least 25 % of instances at that wi-fi location are 10:00–16:00 during work days.

If not otherwise specified, the procedure uses majority-vote classifiers consisting of eight base classifiers: Naive Bayes, Logistic Regression, Support Vector Machine, C4.5, Random Forest, RIPPER, AdaBoost and Bagging, as implemented in the Weka machine-learning suite [15].

The procedure for the recognition of daily activities consists of 11 steps:

1) If only time features are available, return unknown as the final output, since these features alone are not enough for accurate activity recognition.
2) If only wi-fi location and time features are available, return the majority class for the current wi-fi location during the day (i.e., 8:00–24:00) or during the night (i.e., 0:00–8:00). For this purpose we use a C4.5 classifiers with only one node that separates the instances based on the wi-fi location.
3) If the velocity is greater than 10 km/h, check if the activity is transport. For this purpose we use a binary classifier created on a training dataset with only sound features and velocity containing only two classes: transport and other (other includes all classes except transport). If this classifier outputs transport, return transport as the final output.
4) If the estimated energy expenditure is greater than 2 metabolic equivalents of a task (MET) or the velocity is between 5 km/h and 25 km/h, check if the activity is exercise. This step aims at recognising running and cycling outdoors. For this purpose we use a binary classifier created on a training dataset with heart-rate, respiration-rate and acceleration features, as well as with wi-fi and GPS location, which contains only two classes: exercise and other. If this classifier outputs exercise, return exercise as the final output.
5) Check eating at home. For this purpose we use a binary classifier created on a subset of the training instances whose wi-fi locations belong in the home group. The instances contain only sound features and wi-fi location, and belong to two classes: eating and other. Resampling is performed in order to bring the eating–other class ratio to 1:2 (fewer than 10 % of training instances are annotated as eating in the original training data). If the classifier outputs eating, return eating as the final output.
6) Check eating at work. For this purpose we again use a binary classifier created on a subset of the training instances whose wi-fi locations belong in the work group. The instances contain only sound features and wi-fi location and belong to two classes: eating and other. Resampling is performed in order to bring the eating–other class ratio to 1:2. If the classifier outputs eating, return eating as the final output.
7) Check eating. For this purpose we use a binary classifier created on a training dataset with sound features, wi-fi and GPS location containing only two classes: eating or other. If this classifier outputs eating, return eating as the final output.
8) Classify home activities. A classifier is built on a subset of the training dataset whose wi-fi locations belong in the home group and which do not have a GPS signal. The features are: sound, heart rate, respiration rate, wi-fi location, acceleration and time. Return the output of this classifier as the final one.
9) Classify work activities. A classifier is again built on a subset of the training dataset whose wi-fi locations belong in the work group and which do not have a GPS signal. The features are: sound, heart rate, respiration rate, wi-fi location, acceleration and time. Return the output of this classifier as the final one.
10) Classify outdoor activities. A classifier is built on a subset of the training dataset containing only instances that have a GPS signal. The features are sound, heart rate, respiration rate, wi-fi and GPS location, acceleration and time. If the classifier predicts out–leisure or out–errands, return the output of this classifier as the final one.
11) Return unknown.

C. Symbolic Reasoning

Symbolic reasoning can take advantage of human knowledge and understanding of a domain, which makes it complementary to machine learning. While machine learning is very good at finding patterns in data, it is limited to what is contained in the data. Furthermore, symbolic reasoning is also suitable for interpreting recognised activities and acting on them. In the COMMODITY$_{12}$ activity recognition approach, we address both of these points.

Activities recognised from sensors are not always of high quality because sensor data can be noisy or missing, because sensors needed to recognise them reliably are not available,

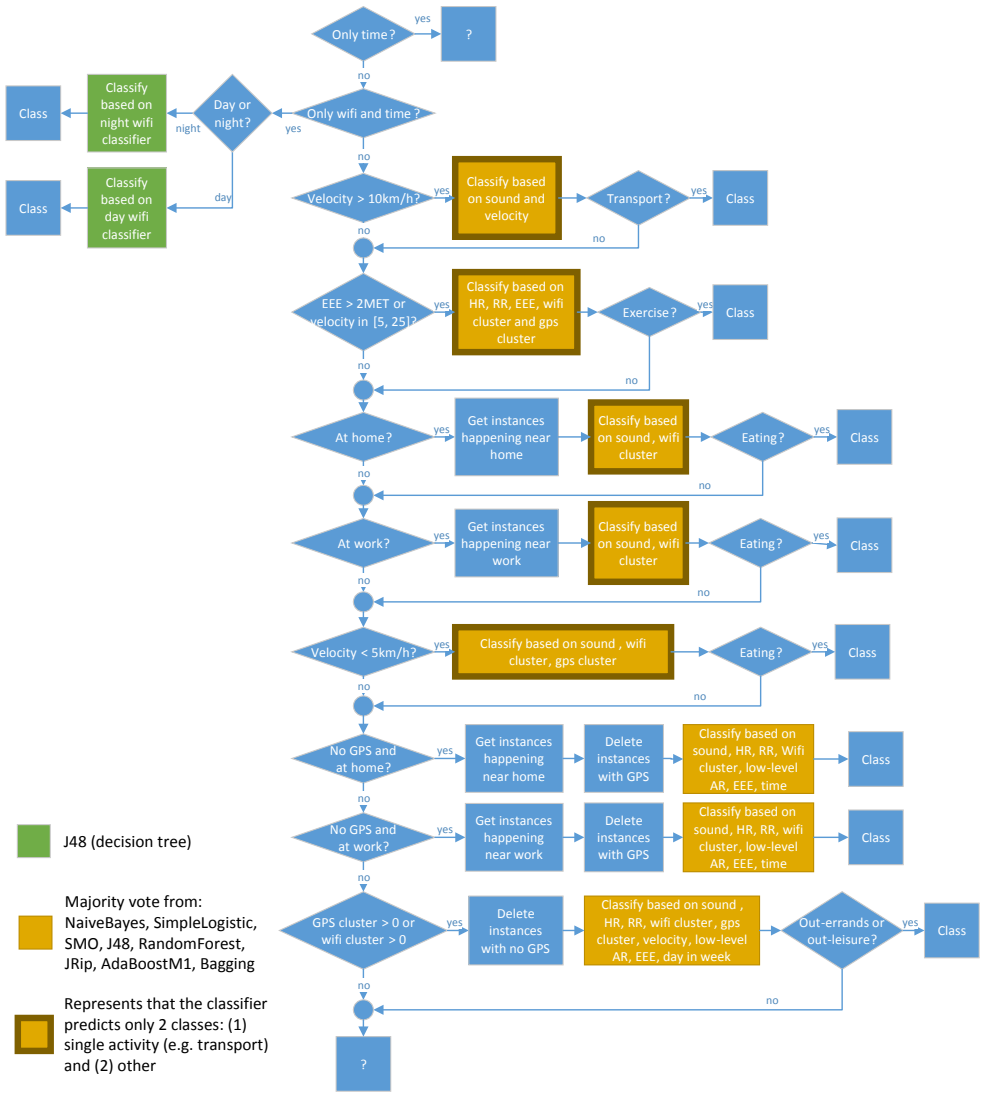


Fig. 1: The procedure for recognizing lifestyle activities using machine learning.

and simply because activity recognition is not always an easy task. This can be addressed to a degree by symbolic reasoning. Event Calculus [4] can do reasoning considering ongoing activities and the previously recognised activities. Here, we will focus on the eating activity. Detecting when a person is eating is a crucial task for diabetes management and is not necessarily an easy problem. While some persons could have a regular habit of eating at a distinctive location, other could instead eat at various locations, or even in at the workplace, making the distinction between the activities eating and work difficult.

Example: Suppose we have a set of consecutive recognitions for an eating activity, followed by a gap in the recognitions. We apply Event Calculus to extend those recognitions assuming a certain continuity in the activities of the user. In other words, if eating is true for times $T = t_s, t_s + 1 \ldots, t$, and we do not have any recognised activities for times $T = t + 1, t + 2, \ldots t_e$, we can also assume that the user was eating from $T = t+1$ to $T = t_e$ (assuming $t_e - t_s$ is a reasonable amount of time). If $t_s + n > t_e$, where n is the maximum time in which we want to propagate existing recognitions, the method can be conservative, therefore leaving an unknown for the activities ranging $T = n + 1$ onwards.

```
possible_at(activity(eating, P, _)=active, T):-
        started_at(activity(eating, P, _), Ts),
        max_time(eating, Ts, Te),
        T < Te.
```

We also use our symbolic approach to detect unexpected transitions in the recognised activities. For example, it is very unlikely that a person is at home at time $T = t - 1$, then at work for time $T = t$, and then go back to work in $T = t + 1$. Following the same principle of continuity, we change "orphan" recognitions, that is, those lasting a single time event, into their neighbouring predictions. Similarly, it is very unlikely for a person to be eating for just a single unit of time. It is reasonable that any isolated eating activity in time $T = t$ surrounded by another activity a predicted for times

$T = t - 1$ and $T = t + 1$ should be changed into the activity a. In other words, at a very high level, symbolic approaches can correct past recognitions if they do not show a reasonable behavior.

We develop these ideas on top of our knowledge-representation framework that allows an agent situated in an environment to recognise complex activities, reason about their progress and take action to avoid or support their successful completion [16]. Activities are understood as parameterised templates whose parameters consist of a unique name labelling the activity to be recognised, a set of participants co-involved in carrying out of the activity and a goal revealing the desired outcome the participants seek to bring about. The novelty of the work is the identification of an activity lifecycle where activities are temporal fluents that can be started, interrupted, suspended, resumed, or completed over time. The framework also specifies activity goals and their associated lifecycle, as with activities, and shows how the state of such goals aids the recognition of significant transitions within and between activities. We have the following components of the framework (for more details of this logic framework, see [16]):

- an activity theory that regulates the activity lifecycle,
- a goal theory that regulates the goal lifecycle,
- a domain model that describes the recognition domain,
- an event narrative that contains the events that happened in the system (dataset).

Figure 2 describes the lifecycle of an activity (a) and a goal (b). The recognition of activities is driven by the goals of the user, which we represent as a modification of the goal lifecycle presented in [17] for our purposes. An activity is first activated due to a goal being adopted by the user and a low-level event happening to start the activity. While the activity is being performed, if the user's goal changes, then the activity is no longer required (e.g., the goal is *dropped*), then the activity is *interrupted*. If the goal remains, but another goal supersedes it temporarily (e.g., the goal is *deactivated*), then the activity is *suspended*. When the user reactivates the goal again, the activity is resumed. The activity completes successfully when the user achieves the goal, in which case the activity is *completed*.

IV. Experimental Evaluation

Our activity-recognition approach was trained and evaluated on the recordings of five volunteers (four male and one female), who wore the smartphone and the ECG monitor for two weeks. The volunteers manually labeled the 10 lifestyle activities of interest through a phone application. On average, we have 7.5 hours of recordings per day with the ECG monitor and 11 hours with the phone. The experiments were performed in a user-dependent manner, which means that each user's first recording week was used for training and the second week for testing.

A. Results of Machine Learning

We evaluated the accuracy of nine machine-learning algorithms: eight standard algorithms as implemented in the Weka machine-learning suite [15], and Vote, which outputs the class voted for by the majority of these eight. The results using the full ten activity classes are shown in the second column of Table I. The Vote classifier performed best, but after examining its confusion matrix (Table II), we realised that the activities home–chores, home–leisure and food preparation were mutually misclassified so often that we merged them into the activity home. Likewise, out–leisure and out–errands were merged into out, resulting in seven activity classes. The results for the reduced number of classes are shown in the third column of Table I.

TABLE I: Classification accuracy of recognizing lifestyle activities using simple machine learning

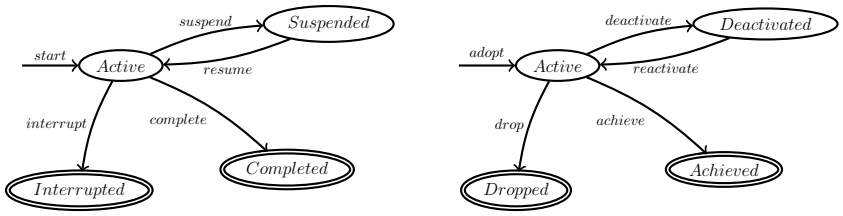
Algorithm	Ten classes	Seven classes
Naive Bayes	0.57	0.68
C4.5	0.56	0.66
RIPPER	0.62	0.72
SVM	0.64	0.72
Random Forest	0.61	0.71
Bagging	0.60	0.69
AdaBoost	0.44	0.56
Vote	0.65	0.77

TABLE II: Confusion matrix of the Vote classifier on the ten lifestyle activities

	Sleep	Home–chores	Home–leisure	Food preparation	Eating	Work	Exercise	Out–leisure	Out–errands	Transport
Sleep	**678**	0	0	0	0	0	0	0	0	0
Home–chores	92	**1064**	612	22	86	120	12	480	0	4
Home–leisure	146	527	**708**	0	37	187	4	148	0	140
Food preparation	14	88	47	**12**	23	4	0	0	0	0
Eating	20	187	94	8	**213**	137	21	35	0	102
Work	2	1	1	0	45	**6891**	5	51	22	280
Exercise	1	65	4	0	1	118	**233**	90	11	20
Out–leisure	3	20	6	0	23	113	80	**452**	10	1023
Out–errands	0	52	7	0	95	221	100	268	**77**	54
Transport	0	7	4	0	26	84	43	82	12	**1134**

The results in the third column of Table I show that with the reduced number of classes, the Vote classifier still performed best. We then examined the instances where it made mistakes, and found ways that could be used to improve the classification accuracy. For example, when the velocity is above 10 km/h, the user is probably running, riding bicycle or in a vehicle. Therefore, if the velocity is above 10 km/h, we should check if user's activity is transport. If the estimated energy expenditure is above 2 MET (moderate and vigorous activities are above 3 MET) or the velocity is between 5 km/h and 25 km/h (velocity range encompassing running and riding a bicycle), the user may be exercising. Therefore, we should check for exercise in these circumstances. In this way we created the final machine-learning approach shown in Figure 1.

The classification accuracy of the final machine-learning approach was 0.88. However, it should be noted that it did not classify 23 % of the instances, mainly because of missing sensor data, which precluded reliable recognition. Table III presents the confusion matrix of the final approach. The



(a) Activity lifecycle (b) Goal lifecycle

Fig. 2: Lifecycle of an activity and a goal. Double ellipses represent terminal states.

TABLE III: Confusion matrix of the final machine-learning approach on the seven daily activities

	Sleep	Home	Eating	Work	Exercise	Out	Transport
Sleeping	**670**	8	0	0	0	0	0
Home	48	**3203**	23	9	70	24	1
Eating	5	338	**118**	54	7	30	2
Work	1	9	30	**6468**	8	137	5
Exercise	0	54	1	3	**240**	49	51
Out	0	60	18	124	157	**304**	70
Transport	3	27	1	16	8	30	**872**

activities sleep, home, work and transport are recognised quite accurately. Out is not recognised as well, probably because the location varied during this activity and was thus not a very useful clue. Exercise, a particularly important activity for diabetic patients, was recognised correctly in 60 % of the cases, or misclassified as home (in approximately 14 % of the cases), out (12 %) and transport (13 %). The instances where it was misclassified as transport represent cycling faster than 20 km/h. 70 instances of the activity home were misclassified as exercise, mainly those for which energy-expenditure estimation was missing. 157 instances of the activity out were also misclassified as exercise, mainly those with the energy expenditure estimated over 2.5 MET. Eating, the other important activity for diabetic patients, was unfortunately recognised correctly only in 21 % of the cases. These were mainly in the canteen at work or in a restaurant, while eating at home proved too difficult to recognise.

B. Improvements with Symbolic Reasoning

We briefly describe several preliminary results we obtained by applying the symbolic approach to the aforementioned dataset. To show the validity of our approach, we tried to improve the eating activity recognised by machine learning, as it is important for diabetics. A quick analysis of the dataset gives the results shown in Table IV. A first look at the table shows two facts: *i)* The level of false negatives is rather high (harming recall), and *ii)* The level of true positives is average (which makes precision behave in a relatively decent manner).

Improving recall by extending predictions: We decided to improve the recall by trying to cover "null" activities (i.e. those instances which have no recognised activity). Our improvement makes use of the following rule: if at a given time $T = t0$ the recognition was eating and there are null activities

TABLE IV: Detailed evaluation of the eating activity

Person	TP	FP	FN	TN	Precision	Recall
Person 1	34	39	147	3911	0.4657	0.1870
Person 2	57	15	82	1888	0.7917	0.4101
Person 3	21	15	55	4697	0.5833	0.2763
Person 4	5	3	226	4039	0.6250	0.0216
Person 5	1	1	189	2484	0.5000	0.0055

at times $t0 + 1, t0 + 2, \ldots$, recognise eating for those times if the elementary activity is not walking, running or cycling, and the maximum number of filled gaps is 2. After applying this rule we obtained the following results:

- Person 1: precision dropped from 0.4657 to 0.4415, recall unchanged (we introduced 4 new FP).
- Person 2: precision increased from 0.7917 to 0.8023, recall increased from 0.4101 to 0.4964 (12 new TP, 2 new FP).
- Person 3: precision dropped from 0.5833 to 0.575, recall increased from 0.2763 to 0.3026 (2 new TP, 2 new FP).
- Person 4: precision unchanged, recall unchanged.
- Person 5: precision increased from 0.5000 to 0.6667, recall increased from 0.0055 to 0.011 (1 new TP).

Overall, we obtained 15 new true positives predictions, but we also added 8 false positives. We argue that this result is encouraging as we are interested in increasing recall, which happened in three out of five cases.

Removing false positives by smoothing predictions: False positives can be removed if we find orphan eating recognitions, i.e. an isolated eating surrounded by two other different activities. By applying this simple rule we obtained the following changes:

- Person 1: precision increased from 0.4657 to 0.4722, recall unchanged (1 FP removed).
- Person 2: precision increased from 0.7916 to 0.8260, recall unchanged (2 FP removed).
- Person 3: precision unchanged, recall unchanged (no changes at all).
- Person 4: precision dropped from 0.6250 to 0.5715, recall dropped from 0.0216 to 0.0173 (1 TP removed).

- Person 5: precision increased from 0.5000 to 1.000 (1 FP removed).

Except for person 4, the method seems to work well, increasing the quality of the recognition of eating. Note that this method does not depend on the activity and could be easily applied on the rest of activities as well.

V. Conclusion

In this paper we presented an approach that combines machine learning and symbolic reasoning to recognise lifestyle activities of diabetic patients using sensor data obtained primarily from the patients' smartphone. Machine learning was used to deal with the large quantity of difficult-to-interpret sensor data. Domain knowledge was used to structure the machine learning procedure so that multiple classifiers were invoked in the appropriate order, as well as to craft symbolic rules to refine the activities recognised by machine learning. The use of domain knowledge in the machine-learning approach increased the classification accuracy from 0.77 to 0.88 at the expense of not classifying 23 % of the instances. The symbolic approach further improved the recall for the eating activity by applying several heuristics.

As the overall accuracy indicates, most of the activities were recognised quite well. Of the two activities that are particularly important for diabetic patients – eating, which puts glucose in the blood, and exercise, which speeds up its absorption – the latter was also recognised reasonably well. Many instances of exercise that were misclassified as other activities were probably when the user had a break or was exercising less vigorously, while many instances of other activities that were misclassified as exercise involved relatively vigorous movement. Eating, however, was not classified well: while it was possible to recognise some instances based on the location (canteen, restaurant), recognising eating at home proved virtually impossible. This is probably because it cannot be recognised based on the patient's physical activity or location (sitting at the dining table – if this location can even be distinguished from others based on the wi-fi signal – is quite normal even when the patient does not eat), so the only possible clue is the ambient sound. The sound can be distinctive in some cases, but is nevertheless difficult to recognise, particularly with the privacy-preserving measures and the sound being muffled by the phone being in the user's pocket. The most effective solution to improve the recognition of eating would probably be to put a sensing device on the patient's wrist, but unfortunately this is out of scope of our work in the COMMODITY$_{12}$ project.

We plan to pursue the following directions for future work:

- We will attempt to improve the recognition of eating, primarily by employing better sound processing.
- We will develop a machine-learning approach that does not have to be trained on recordings of the patient for whom it is used, or that at least does not require each patient to label all the activities. We may require him/her to label only those activities that cannot be recognised in a patient-independent fashion (perhaps eating will prove to be such an activity).
- We will enhance the symbolic reasoning by applying more heuristics depending on the activity type.
- We will use the proposed activity-recognition approach as an input to the diagnostic reasoning in the personal health system developed in COMMODITY$_{12}$.

Acknowledgments

This work has been partially supported by the EU project COMMODITY$_{12}$ (`www.commodity12.eu`).

References

[1] Idf diabetes atlas. [Online]. Available: http://www.idf.org/diabetesatlas

[2] Commodity12. [Online]. Available: http://www.commodity12.eu/

[3] Ö. Kafali, S. Bromuri, M. Sindlar, T. van der Weide, E. Aguilar-Pelaez, U. Schaechtle, B. Alves, D. Zufferey, E. Rodríguez-Villegas, M. I. Schumacher, and K. Stathis, "Commodity$_{12}$: A smart e-health environment for diabetes management," *JAISE*, vol. 5, no. 5, pp. 479–502, 2013.

[4] R. Kowalski and M. Sergot, "A logic-based calculus of events," *New Generation Computing*, vol. 4, no. 1, pp. 67–95, 1986.

[5] C. Bozidara, J. Vito, and L. Mitja, "Demo abstract: Activity recognition and human energy expenditure estimation with a smartphone," in *PerCom 2015, 23-27 March 2015, St. Louis, USA*, 2015.

[6] T. Huynh, U. Blanke, and B. Schiele, "Scalable recognition of daily activities with wearable sensors," in *Location- and Context-Awareness, Lecture Notes in Computer Science*, J. Hightower, B. Schiele, and T. Strang, Eds. Springer Berlin Heidelberg, 2007.

[7] A. Helal, D. J. Cook, and M. Schmalz, "Smart home-based health platform for behavioral monitoring and alteration of diabetes patients," *Journal of Diabetes Science and Technology*, vol. 3, no. 1, pp. 141–148, 2009.

[8] O. Amft, "Ambient, on-body, and implantable monitoring technologies to assess dietary behavior," in *Handbook of Behavior, Food and Nutrition*, V. R. Preedy, R. R. Watson, and C. R. Martin, Eds. Springer New York, 2011.

[9] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1473–1488, 2008.

[10] N. Rota and M. Thonnat, "Activity recognition from video sequences using declarative models," in *ECAI*, 2000, pp. 673–680.

[11] G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia, "Event detection and analysis from video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 8, pp. 873–889, 2001.

[12] S. Hongeng, R. Nevatia, and F. Bremond, "Video-based event recognition: activity representation and probabilistic recognition methods," *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 129–162, 2004.

[13] V. D. Shet, D. Harwood, and L. S. Davis, "Vidmap: video monitoring of activity with prolog," in *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*. IEEE, 2005, pp. 224–229.

[14] Foursquare. [Online]. Available: https://developer.foursquare.com/

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[16] Ö. Kafalı, A. E. Romero, and K. Stathis, "Activity recognition for an agent-oriented personal health system," in *PRIMA 2014: Principles and Practice of Multi-Agent Systems*, 2014, pp. 254–269.

[17] M. B. van Riemsdijk, M. Dastani, and M. Winikoff, "Goals in agent systems: a unifying framework," in *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008, pp. 713–720.