Dynamic Streaming Sensor Data Segmentation for Smart Environment Applications

Hela Sfar(✉) and Amel Bouzeghoub

CNRS Paris Saclay, Telecom SudParis, SAMOVAR, Évry, Essonne, France
{hela.sfar,amel.bouzeghoub}@telecom-sudparis.eu

Abstract. With the increasing availability of unobtrusive, and inexpensive sensors in smart environments, online sensor data segmentation becomes an important topic in reconstructing and understanding sensor data. Usually, in the literature, the segmentation is either performed by following a fixed or a dynamic time-window length. As stated in several works, static time-window length has several drawbacks while adjusting dynamically the window length is more appropriate. However, each of previous methods for dynamic data segmentation targets only a particular type of application. Hence, there is a need for a general method independent of applications providing high degree of usability. To achieve this aim, in this paper, we propose a novel method that dynamically adapts the time-window size. The proposal is designed to be applied in a wide range of applications (activity recognition, decision making, etc.) by combining statistical learning and semantic interpretation. This hybridization allows to analyze the incoming sensor data and choose the better time-window size. The presented approach has been implemented and evaluated in several experiments using the real dataset Aruba from the CASAS project.

Keywords: Clustering · Ontology · Segmentation
Smart environment

1 Introduction

Due to the rapid advances in sensing technology, we are witnessing a growing interest in smart environments, in which a variety of sensors are continuously sending data for processing and analysis in order to be used for different domain applications. To gain a meaningful data understanding from sensor data, one of the major tasks in this area is to divide the long sequence of sensing records into a set of individual segments. Each segment corresponds to a “specific” concept which can be interpreted differently according to the target application.

In the literature, the dynamic data segmentation has proven better results than the static one [4] since it identifies the time points in a more flexible way.

However, dynamic streaming sensor data segmentation is still a challenging problem. Generally, previous researches in this field [1–3], either assume that a pre-segmented dataset is available for learning the suitable time-window size or being conceived to be applied to a particular application. Accordingly, most related work suffer from generality issue. Hence, the segmentation should be done in a general manner. Afterwards, it can be adapted according to the application.

In this paper, in order to overcome the aforementioned issue, we propose a novel method which combines a clustering method, that is to say, a statistical learning method, and a logic-based method, that relies on high-level "symbolic" representation, in order to dynamically choose the best time-window size. Initially, to tackle the cold start problem that usually concerns statistical learning approaches, an ontology with a default classification is created. This task can be done by the designer after the setup of the environment. Afterwards, once a training dataset is acquired related to the target application[1], a clustering based classification of the dataset is done offline according to a defined features. The proposal dynamically updates offline the ontology whenever change occurs in the obtained clusters. During the online process, the method uses the ontology to decide which better size for the current time-window. The proposed method thanks to this hybridization neither requires a pre-segmented dataset nor being limited only for a particular type of application's use. In order to prove the accuracy of the proposal we apply it for an activity recognition application. In addition, we test it with Support Vector Machine (SVM) [12], a machine learning method, as an activity classifier module, with the real Aruba dataset from the CASAS project [11][2], and synthetic datasets. In the following, we summarize the main contributions of the paper:

- A new method for dynamic streaming sensor data segmentation. The proposal is flexible to be used in a wide range of applications.
- A novel marriage between knowledge oriented method and machine learning to provide a flexibility for the time-window size choice.
- The method is fully implemented and tested in the specific case of activity recognition application. SVM is used as an activity recognizer module with different datasets. The output is compared with SVM using fixed time-window size as well as state of the art work. The results are promising proving the high accuracy level.

2 Related Work

In this paper we classify previous works of dynamic streaming sensor data segmentation into three main classes: (1) metric-based methods, (2) learning-based methods, and (3) knowledge-based methods.

[1] Dataset of user activities for activity recognition application, for anomalies in case of anomaly detection application, etc.

[2] http://ailab.wsu.edu/casas/datasets/.

Metrics-Based methods: the general principle of these methods is that the window size is chosen according to the result of computed metrics such as mutual information [2,5]. For activity recognition process, the authors in [2] propose a segmentation method that consists of dividing the sensor events into chunks according to the incidence of activity. To achieve this aim, they use the Pearson Product Moment Correlation (PMC) metric to compute the correlation between any pair of sensor events in the window. The authors in [5] propose to compute the mutual information between each couple of sensor events within a window and calculate a feature's weight. Only highly related sensor events are supposed to be in the same activity. Another work [13] proposes a similarity measure to segment a motion stream. The proposed measure is highly related to the body motion attributes. For accelerometer streaming data segmentation, the authors in [6] propose to classify the data according to a set of features which are used as metrics such the variance, mean, etc. The method is specific for accelerometer data.

Knowledge-Based methods. This class applies mainly semantic representation (e.g ontology) and semantic reasoning (e.g logic inference) to derive dynamically the window size. For example, the work in [3] proposes an ontology representing the activities that can be realized by the residents, the types of installed sensors in the smart home, etc. The dynamic segmentation consists of either expanding or shrinking the time-window by queering the ontology at the same time as recognizing the activity. The work in [7] proposes a semantic based approach for segmenting sensor data series using ontologies to perform a terminology box (TBOX) and a assertion box (ABOX) reasoning, along with logical rules to infer whether the incoming sensor event is related to a given sequences of the activity. As in the previous work, the segmentation method is also integrated into the activity recognition method.

Learning-Based methods. Under this class, methods use machine learning in order to learn which suitable window size regarding the coming sensor data. Generally, a pre-segmented dataset is required for the training phase (i.e a dataset containing the ground truth of time window sizes for the segments). The authors in [1] propose to use a probabilistic data driven approach to identify, as a first step, the possible window size using a pre-segmented dataset. As a second step, they propose to learn the most likely window size for an activity based on the computed probabilities of the possible window sizes.

To the best of our knowledge, all the previous methods are designed to be used for a particular kind of application domain such as activity or gesture recognition. However, it is more useful to conceive a flexible method that can be applied in different applications.

3 Method

Through this section, we explain in details the whole process of the method for the dynamic streaming sensor data segmentation.

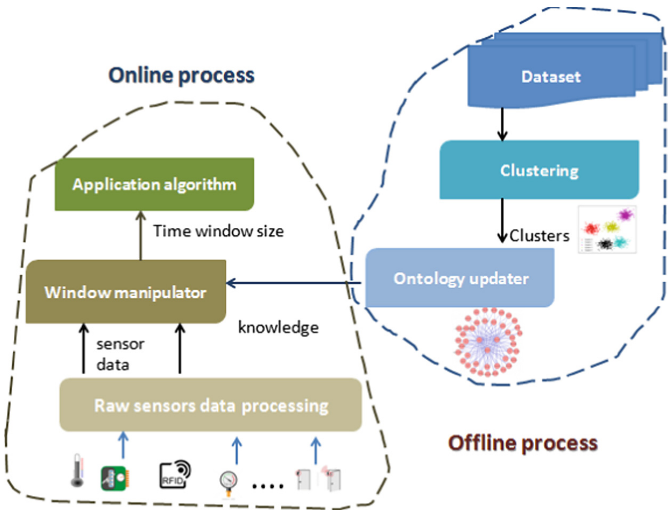

Fig. 1. Method architecture

As we can see in Fig. 1, the method holds mainly two processes: Offline and Online executions. The former starts by applying a `Clustering` algorithm in order to classify the given dataset into a set of classes based on given features. The dataset must concern one application and the features are defined accordingly. As mentioned before, the proposed method could be used with different application domains. The clustering is executed offline once a dataset is given or updated. Afterwards, the `Ontology updater` dynamically updates the ontology when the clusters are updated or new ones appear. The ontology is designed to contain the information that must be provided by the built clusters for each application. Next, when the dataset is ready and the set of clusters are obtained or changed, the ontology is dynamically updated. The online process dynamically segments the streaming sensor data into time-windows based on the knowledge provided by the ontology.

3.1 Clustering

The clustering aims, as a first step, to divide a given dataset into a set of clusters based on the given features. Each target application has its own features. For example, *activity durations* and *used sensors* are features corresponding to the activity recognition application while the *anomaly start time* can be considered as a feature for anomaly detection application.

3.2 Ontology Updater

A general ontology is designed to semantically represent the information provided by the clusters for each dataset. In order to allow a quick start of the

Algorithm 1. Ontology Updater

```
Input: Ontology O, Features set sf, Clusters C, Name of the dataset name, Application app
Output: Updated ontology O
1  if (ExecuteRule(search, O, name) $\neq \emptyset$) then
2      ExecuteRule(insertInstance, Dataset, name);
3  end
4  appC $\leftarrow$ GetClass(O, app);
5  for (each cluster $c_i \in C$) do
6      for (each element $el \in c_i$ ) do
7          ExecuteRule(insertInstance, el, hasElement(O, appC);
8      end
9      for (each feature $f_j \in sf$) do
10         fC $\leftarrow$ GetClass(O,$f_j$);
11         val $\leftarrow$ GetValue($c_i$, $f_j$);
12         ExecuteRule(insertInstance, val, fC);
13         setObj $\leftarrow$ GetObjectProperties(hasElement(appC), fC);
14         for (each objectProperty obj $\in$ setObj) do
15             ExecuteRule(insertRelation, obj, $c_i$, val);
16         end
17     end
18 end
```

system even if a dataset is not yet ready, the ontology initially contains similar information that should be provided by the clusters for each application. When the dataset is ready and the clusters are obtained, the ontology is dynamically updated with the content of the given clusters. The dynamic update is insured by using program and SPARQL queries. Algorithm 1 shows the operations performed by the ontology updater. Initially the ontology contains the three classes *Application*, *Dataset*, and *Element*. Moreover, it contains the applications that will be held by the method represented as subclasses of *Application* class. The class *Element* represents the main information provided by the application. For instance, for the activity recognition application the ontology must contain the class *Activity*, a subclass of *Element*, related to the class *activity recognition*, a subclass of *Application*, and related to the classes corresponding to the application's features (i.e Duration). After receiving the application name, the set of features and clusters, and the dataset name, Algorithm 1 starts by executing a rule named *search*, by calling the function *ExecuteRule*, which searches whether the given dataset is already in the ontology (Line 1 and 2). It is important to mention that the algorithm stores the corresponding classes in the ontology for each possible application and their corresponding features. Hence, the function *GetClass* (Line 4) returns the corresponding ontological class for the input application *app*. Afterwards, by applying a logic rule, named *insertInstance*, all the different elements belonging to the given clusters become instances of the corresponding subclass of *Element* related to the given application (Lines 5–8). For example, if the application is *Activity recognition* the ontology must hold the class *Activity* subclass of *Element*. In this case the activities in the clusters become instances of the *Activity* class. For a better insight see Fig. 2. Next, each feature value in the cluster becomes an instance of the corresponding feature's class in the ontology using a logic rule (Line 9–12). For example, 5 min is a value in the cluster of the feature Duration. Accordingly, 5 min becomes an instance

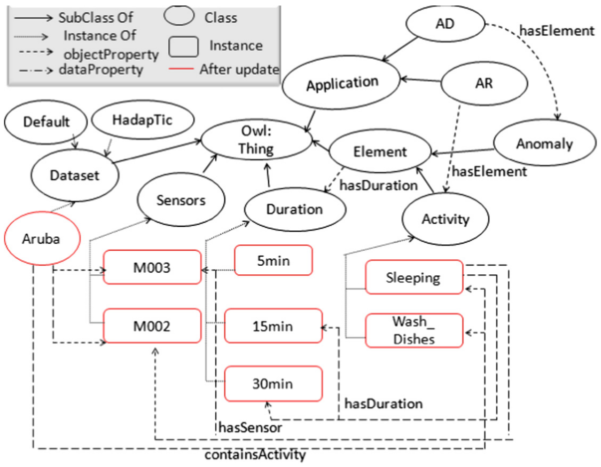

Fig. 2. An example of the ontology used in the experiments after being updated with Aruba dataset's clusters. Added elements are in red (Color figure online)

of the class *Duration* in the ontology (see Fig. 2). Finally, the algorithm extracts the set of objectProperties having the feature class as a range (Line 13). Then, it relates the instances corresponding to elements in the clusters with theirs features values using the adequate objectProperty. For example, in the ontology (Fig. 2), the feature's class *Duration* is the range of the objectProperty *hasDuration* where the Class *Activity* is its domain. Therefore, Algorithm 1 links the added activity's instances for each cluster with their corresponding durations.

3.3 Data Processing and Window Manipulator

Generally, a sensor sequence can be represented as $\{S1, S2...Sn\}$ where Si refers to the itth sensor event, and each sensor event is encoded in the template of $\{date, time, sensorID, sensorValue\}$. Once the coming sensor data are represented in this template, the main aim of this step is to divide this streaming sensor data into a time windows with size "sw". Each chunk "sw" is chosen dynamically using the Window manipulator Algorithm 2.

Initially, the time window size is the minimum duration in the ontology regarding the given application and dataset. It is important to note here that Duration is defined as a features for all applications. For instance, if we target the activity recognition application and we set up the method for aruba dataset, then the algorithm sets the initial time window size as the minimum activity's durations belonging to Aruba. This duration should not be used before in the same session (Line 7). Afterwards, the algorithm keeps extracting the set of sensor data (*sensorsData*), that are in the streaming SD, occurred during the defined time window size (*actualSize*). This is achieved using the function *Read-Online()* (Line 8). The algorithm stops the loop when *sensorsData* are included, in at least one of the feature vectors extracted from the ontology (Lines 12,18)

Algorithm 2. Window Manipulator

```
   Input: ontology O, streaming sensor data SD, application app, application's features feats,
          Dataset's name dName
   Output: set of possible results possRes, time window size sw
1  usedsizes ← $\emptyset$;
2  sensorsData ← $\emptyset$;
3  keep ← true;
4  possRes ← Null ;
5  previousSize ← 0 ;
6  while (keep) do
7      actualSize ← getMinimumDuration(O, usedSizes, previousSize, app, dName);
8      sensorsData ← ReadOnline(SD, actualSize-previousSize, sensorsData);
9      EleFeatVects ← getElementsVectors(O, app, feat);
10     for each appEl $\in$ EleFeatVects do
11         if (testAppart(sensorData, appEl) then
12             add(possRes, appEl);
13         end
14     end
15     previousSize ← actualSize;
16     Add(usedSizes, actualSize);
17     if (possRes!= Null) then
18         keep=false;
19         sw← actualSize;
20     end
21 end
```

using the *testAppart* function. In other words, based on the given application, the algorithm extracts the features vector for each given application's element. For example, in case of anomaly detection, it extracts for each anomaly its features vector holding the features' values. Then, it checks whether the data in the streaming are belonging to this vector. In the positive case, this element is added to the set of possible results *possRes* (Line 13). Finally, if the algorithm detects at least one possible element then it stops reading (Lines 19, 20). Otherwise, it extends the actual time-window size (*actualSize*) with the difference between it and the next minimum element's duration extracted from the ontology (Line (7)), the actual size becomes the previous size, and this process is repeated until possible results are detected (Line 17, 18).

4 Experiments and Discussion

To evaluate the usefulness of our proposal for dynamic streaming sensor data segmentation, we have applied it in case of activity recognition application. We tested it with 6 weeks over Aruba dataset and a synthetic dataset. On the one hand, Aruba was collected by the Center for Advanced Studies in Adaptive Systems (CASAS) [11]. The Aruba dataset contains ground-truthed activities of a home-bound person in a small apartment for 16 weeks. On the other hand, we have generated automatically two datasets that can be obtained from the GIS MADONAH[3]. For the activity recognizer method we applied the Support Vector Machine (SVM) algorithm. More details about the dataset and the experiments are given in the next subsections.

[3] http://www.bourges.univ-orleans.fr/madonah/.

4.1 Datasets

Data collected from Aruba dataset was obtained using 31 motion sensors, three door sensors, five temperature sensors, and three light sensors. 11 activities were performed for 220 days (7 months). The dataset is imbalanced, as some of the activities occur more frequently than others. Table 1 presents the statistics of the sensor events and activities performed in the 6 weeks over Aruba dataset. "Other activity" class contains events with missing labels. It covers 54% of the entire sensors events sequence.

The synthetic dataset contains sensor data values similar to that provided in GIS MADONAH. It is generated separately for half day (90 events) and one day (185 events) of an elderly like routine. It contains a set of presence, motions sensors and light detectors. The dataset holds principally four activities: Sleeping, Watching TV, Discomfort, and Eating.

These datasets were used in our method, on the one hand, as training dataset for SVM and, on the other hand, for the clustering.

Table 1. Statistics of six weeks over Aruba dataset

Id	Activity	Number of events
1	Bed_to_Toilet	266
2	Eating	3207
3	Enter_Home	404
4	Housekeeping	2117
5	Leave_Home	384
6	Meal_Preparation	57029
7	Relax	70917
8	Resperate	108
9	Sleeping	6536
10	Wash_Dishes	2092
11	Work	3264
12	Other activity	174 264

4.2 Evaluation Result

F-Score measures was used to evaluate our proposal with 6 weeks over Aruba dataset. Moreover, in order to discuss its performance regarding static time-window size, we firstly tested SVM with fixed time-window size (i.e. 5 min).[4] Afterwards we used SVM as an activity recognizer in the method. Furthermore, the results are compared with the dynamic data segmentation process, SWMIex [5]. SWMIex is a metrics-based method which uses the Mutual Information measure to compute sensor correlation. SWMIex, SVM with static time window, and our proposal are tested in same conditions (i.e dataset). For the process, we used three quarters of the dataset for the offline training and one quarter for the online testing. Figure 3 shows the different F-score values of the three methods for each activity belonging to Aruba dataset.

[4] After doing a set of experiments with different time window sizes, five minutes has shown best results for SVM using 6 weeks over Aruba dataset.

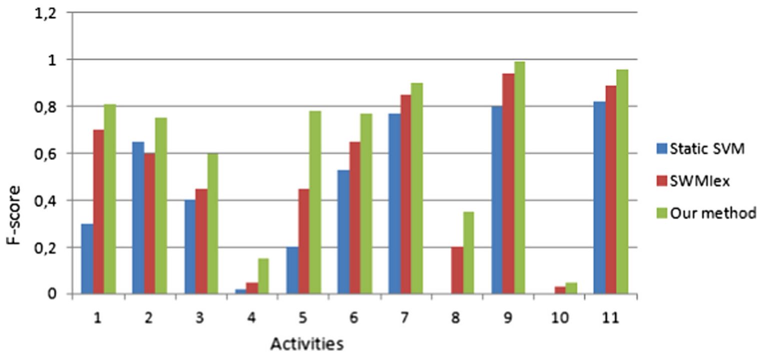

Fig. 3. F-score value for the different activities in Aruba dataset using only SVM with static time-window length (5 min), our and SWMIex methods

Table 2. Average F-score for all activities

Dataset /method	Static SVM	Our method
Half day synthetic dataset	0.39	0.42
One day synthetic dataset	0.42	0.53
Six weeks over Aruba	0.40	0.64

As it is shown in Fig. 3, our proposal outperforms static SVM and SWMIex in terms of F-score. From Fig. 3, we conclude that the three methods have unbalanced performance. In other words, they have higher efficiency to recognize some activities (i.e. 9. Sleeping, 11. Work) than to detect some others (4. HouseKeeping, 10. Wash_Dishes, 8. Resperate). This result, has two possible explanations. On the one hand, since also the dataset is unbalanced then infrequent activities have low chance to be recognized such as the *Resperate* activity. On the other hand, the *used sensor* is a feature for the three methods and some activities in the dataset can have different used sensors for each occurrence. The methods are then sometimes confused (i.e *HouseKeeping*).

The SVM with 5 min as time-window length has the lowest performance (F-score $\in$ [0..0.9]) since it has a static segmentation. In fact, the duration of each activity varies according to the resident routine. For some activities five minutes is sufficient for the recognition and for others this length is either too long or too short. The proposal has the best performance (F-score $\in$ [0.38..0.99]) thanks to its dynamic segmentation.

Table 2 shows the average F-score values of our method compared to Static SVM using the synthetic datasets and Aruba. Obviously, as we can see in the table, our method is more performing using Aruba than the two synthetic datasets and outperforms the Static SVM using the three datasets. The better F-score value obtained by our method using Aruba can be explained by the high sensors number used for Aruba. Indeed, the *used sensor* is considered as a feature in our method for classifying activities. Therefore, when the sensor number increased, our method differentiates better the activities and then provides better results. For One day synthetic dataset, our proposal has better F-score value than for that of half day because a richer training model is built from the former since it holds double data.

5 Conclusion

In this paper we proposed an hybrid method for dynamic streaming sensor data segmentation. The proposal combines clustering and ontology techniques to provide high flexibility in order to be used in different types of applications. As a matter of fact, the main advantage of our method regarding previous works are its ability to be adapted with different applications. Moreover, it is able to dynamically update the ontology whenever any update in the clusters occurs. In order to prove the efficiency of the proposal, we tested it in case of activity recognition application using a six weeks over Aruba dataset and two synthetic datasets and SVM as an activity recognizer. The evaluation result proves the proposal high efficiency level which is better than static method and a previous metric-based method.

References

1. Narayanan, C.K., Diane, J.C.: Activity recognition on streaming sensor data. Pervasive Mob. Comput. **10** (2014)
2. Jie, W., Michael, J.O., Gregory, M.P.O.: Dynamic sensor event segmentation for real-time activity recognition in a smart home context. Pers. Ubiquitous Comput. **19**, 287–301 (2015)
3. Georgo, O., Liming, C., Hui, W.: Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. Pervasive Mob. Comput. **10**, 155–172 (2014)
4. Tak-chung, F.: A review on time series data mining. Eng. Appl. Artif. Intell. **24**, 164–181 (2011)
5. Nawala, Y., Belkacem, F., Anthony, F.: Towards improving feature extraction and classification for activity recognition on streaming data. J. Ambient. Intell. Hum. Comput. **8**, 177–189 (2016)
6. Tian, G., Zhixian, Y.I., Karl, A.: An adaptive approach for online segmentation. In: International Workshop on Data Engineering for Wireless and Mobile Access (2012)
7. Darpan, T., Liming, C., ChenZumin, W.: Semantic segmentation of real-time sensor data stream for complex activity recognition. Pers. Ubiquitous Comput. **21**, 411–425 (2017)
8. Warren, T.L.: Clustering of time series data-a survey. Pattern Recogn. **38**, 1857–1874 (2005)
9. Wilpon, J.G., Rabiner, L.R.: Modified k-means clustering algorithm for use in isolated word recognition. IEEE Trans. Acoust. Speech Signal Process. **33** (1985)
10. Tran, D., Wagner, M.: Fuzzy C-means clustering-based speaker verification. In: Pal, N.R., Sugeno, M. (eds.) AFSS 2002. LNCS (LNAI), vol. 2275, pp. 318–324. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45631-7_42
11. Diane, J. C.: Learning setting-generalized activity models for smart spaces. IEEE Intell. Syst. **99** (2010)

12. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intell. Syst. Their Appl. **13** (1998)
13. Chuanjun, L., Prabhakaran, B.: A similarity measure for motion stream segmentation and recognition. In: International Workshop on Multimedia Data Mining: Mining Integrated Media and Complex Data (2005)