Pers Ubiquit Comput (2017) 21:411–425
DOI 10.1007/s00779-017-1005-5

ORIGINAL ARTICLE

Semantic segmentation of real-time sensor data stream for complex activity recognition

Darpan Triboan[1] · Liming Chen[1] · Feng Chen[1] · Zumin Wang[2]

Received: 8 October 2016 / Accepted: 30 December 2016 / Published online: 18 February 2017

Abstract Data segmentation plays a critical role in performing human activity recognition in the ambient assistant living systems. It is particularly important for complex activity recognition when the events occur in short bursts with attributes of multiple sub-tasks. Although substantial efforts have been made in segmenting the real-time sensor data stream such as static/dynamic window sizing approaches, little has been explored to exploit object semantic for discerning sensor data into multiple threads of activity of daily living. This paper proposes a semantic-based approach for segmenting sensor data series using ontologies to perform terminology box and assertion box reasoning, along with logical rules to infer whether the incoming sensor event is related to a given sequences of the activity. The proposed approach is illustrated using a use-case scenario which conducts semantic segmentation of a real-time sensor data stream to recognise an elderly persons complex activities.

✉ Zumin Wang
wangzumin@163.com

Darpan Triboan
darpan.triboan@my365.dmu.ac.uk

Liming Chen
liming.chen@dmu.ac.uk

Feng Chen
fengchen@dmu.ac.uk

[1] Context, Intelligence and Interaction Research Group (CIIRG), De Montfort University, Leicester, UK

[2] Department of Information Engineering, Dalian University, Dalian, China

Keywords Smart home · Semantic object modelling · Ontology-based segmentation and separation · Complex activity recognition · Activities of daily living (ADL)

1 Introduction

Ambient Assisted Living (AAL) systems are being continually developed to support the growing ageing population. The main goal of building an AAL system is to provide assistance to the inhabitants in a Smart Home (SH) environment to carry out their Activities of Daily Living (ADL). The stages of building an AAL system can be categorised in three simple Ps: Preparing, Processing and Presenting, as shown in Fig. 1. The preparing stage involves developing activity models, data collection and monitoring. The processing stage comprises of segmenting the raw data stream, inferencing and recognising mixed user (also referred to inhabitant) activities, providing assistance when required and learning new activities. The resource-intensive processing tasks are generally delegated from resource-constrained devices to more powerful devices such as servers with a web service interface. The presenting stage is responsible for tailoring the system to specific application types and providing intuitive human-computer interface (HCI). Figure 1 illustrates these phases as the building blocks of an AAL system.

Each of the components in the three **P**s are not only important but have their own challenges, are also interrelated, which can determine the order in which these tasks are executed. For instance, in the data-driven approach, the activity modelling task is performed after creating a training model from pre-collected datasets contrary to the knowledge-driven approach where the modelling task is performed by domain experts first. This paper focuses

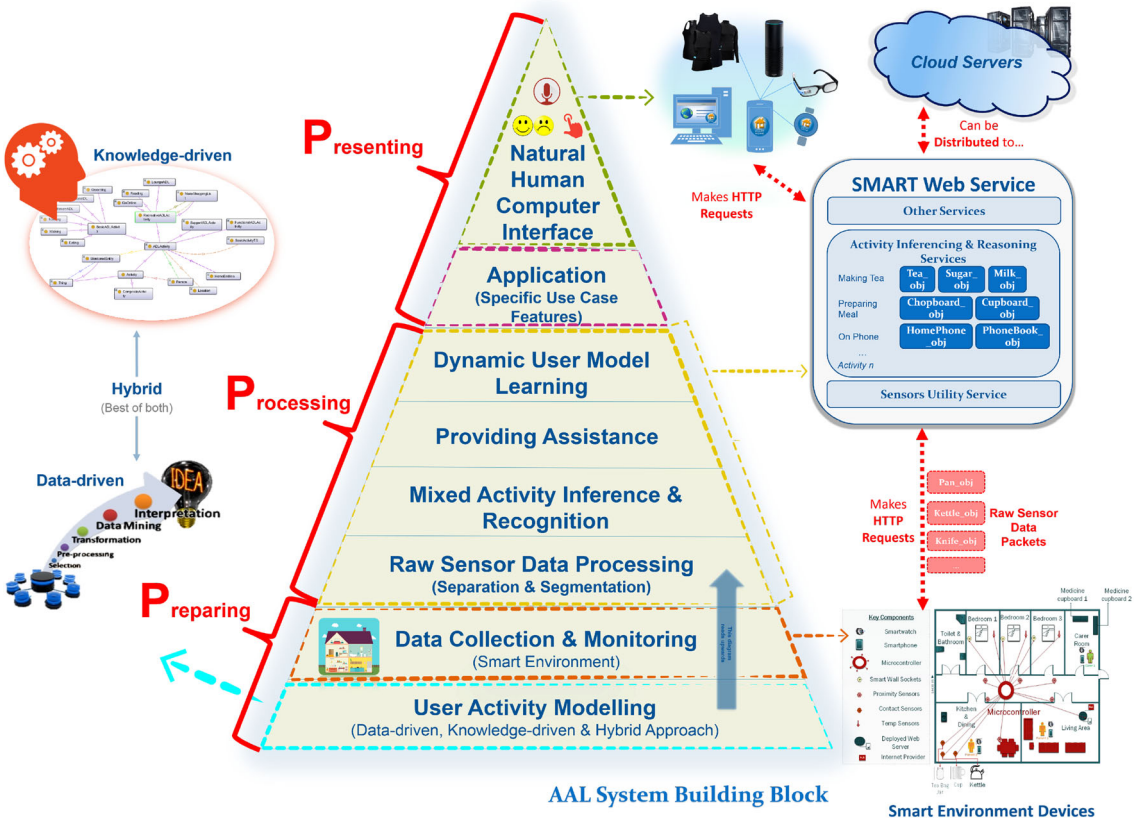
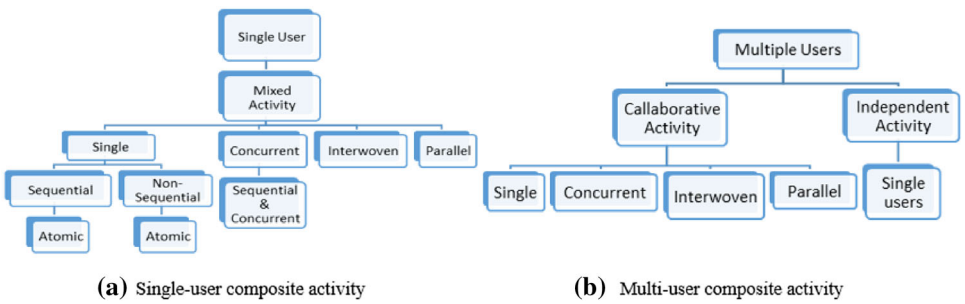


Fig. 1 AAL building block with preparing, processing, and presenting stages

particularly in the processing stage of an AAL system where challenges in segmenting real-time sensor data stream are defined below. Due to human nature, performing a simple daily task with other tasks can create a complex activity routine. Figure 2 shows the complex activities in a hierarchical manner: an inhabitant (Fig. 2a) can perform ADLs in a (non) sequential, interwoven, concurrent, parallel, manner personally and collaboratively with multiple users (Fig. 2b). Therefore, the complexity of the inhabitants performing these tasks can create many challenges to perform AR efficiently, from modelling user activities, collecting and segmenting the contextual data to recognising, and learning new user activities. Achieving accurate human activity recognition (HAR) is one of the active research topics in the AAL system.

Besides acknowledging various types of complex activity, other real-world knowledge such as ADLs, user environmental and contextual information is required to be modelled and classified. These modelling and classification approaches are generally categorised as data-driven, knowledge-driven and hybrid approaches [2, 8, 23, 28]. The data-driven approaches were developed to process large amounts of pre-existing datasets using generative and discriminative classifiers to generate user-specific activity model. The data-driven approach proved to be sensitive to unseen data, however, suffered from performance ("cold start" problem) and ability to reuse the learned model on other users. Therefore, the knowledge-driven approach gained popularity by using the domain experts knowledge to formally define real-world concepts and axioms into



(a) Single-user composite activity **(b)** Multi-user composite activity

Fig. 2 Types of complex user activities

expressive ontologies and logical rules. Although the knowledge-driven approaches resolved the performance and reusability problem of a model, both of the techniques were still far from achieving completeness and high-quality description. Hence, the hybrid approach has been adopted which combines both the data-driven and knowledge-driven approaches to discover and learn new activities.

Another challenge is to process the raw sensor data in real-time from a given smart environment or from pre-collected publically available datasets. The smart environment is created by spatially distributing a variety of sensors to monitor physical and environmental conditions that are interconnected using different communication protocols to form wireless sensor networks (WSNs). The sensing approaches are generally categorised as ambient sensing (environmental monitoring i.e. vision or sensor based), dense sensing (un-obstructively embedded into everyday objects) and wearable sensing (in/direct or implanted). The multimodal sensing approach is now common amongst researchers to extract and correlate the data to reach a finer granularity to understand inhabitants context. Further problem arises when semantically interpreting unstructured raw data to obtain meaningful information in order to perform AR. Fortunately with the knowledge-driven approach, one can use the semantic sensor network (SSN) ontology [20, 32], as a vocabulary to describe and add metadata to the sensors events. This information can then be used in conjunction with domain ontology in order to perform application-specific semantic reasoning. The vocabulary and semantical data represented in the resource description framework (RDF) format and bespoke querying languages such as SPARQLstream and STARQL which can then be used to process the raw sensor stream.

Discovering new patterns and activities poses further challenges once static training models or knowledge graphs are developed. The data-driven approach is considered to be well suited due to its strength of processing unseen data and extract general or parametric features. The learning algorithm is generally executed on the incoming data dynamically (online) for real-time application (i.e. healthcare monitoring and surveillance) or outside the system (offline) such as commercial and educational uses. The offline approach can potentially reduce the run-time load on the system and increase quality of learned features from the stored data. However, the offline approach can also suffer time delays in activity learning and migrating new activity models efficiently. This paper recognises that the static activity model needs to be enriched, and the new learnt knowledge is incorporated into the segmentation stage. It should be noted here that proposing a new activity learning algorithm is out of the scope for this paper.

The remainder of the paper is organised as follows. Section 2 presents related work in stream data segmentation. Section 3 describes complex activity characteristics and the proposed semantical segmentation approach and Sect. 4 elaborates the methods and algorithms for semantic segmentation. Section 5 conducts the evaluation of the approach using the case study and discusses the benefits and limitations of the proposed approach. The paper concludes by highlighting the key contributions and future research directions in Sect. 6.

2 Related work

This section presents state-of-the-art studies of real-time sensor data segmentation (i.e. in knowledge-driven [7, 24], data-driven [1, 17] or hybrid [9, 37] approaches) and highlights key work in activity learning so that the inferred knowledge can also be considered during the segmentation stage. The common criteria that have been used to separate a data stream is time, location, sensor type (i.e. temperature, humidity, touch) and value (binary, float, decimal, string) [5]. Other features used includes temporal, spatial and thematic (related subjects/theme) [36]. One of the popular approaches is to detect the start and end time of the activity is a sliding window protocol, where the window size is either fixed or dynamic [7, 11, 13, 14, 21, 24, 27]. In particular, work in [24] presents a novel segmentation approach to segment the continuous data stream by varying the window size using the temporal information of the sensor data and activity models described using ontology. It further describes the working mechanism and relevant algorithms of the model in the context of ontology-based AR. The experiments on the real-time prototype system show the average recognition accuracy to be above 83% for AR.

Work in [35] presents a sensor and time correlation-based approach to dynamically segment the sensor observations in real time. The data-driven approach was employed to illustrate that classifiers (Naive Bayes, Bayesian network, C4.5 decision tree, Naive Bayes tree, and HMM) achieve higher prediction of the activity. The use of Person product correlation (PMC) coefficient between sensor events and estimating activity label is central to their approach. One of the limitations of this approach is the assumption that there is a correlation between multiple sensor events and activities. For instance, sensors activating in the bedroom, hallway and bathroom are assumed to be performing personal hygiene activity. Alternatively, the approach could utilise the descriptions of the sensor and ADL in an ontology model to identify the relationship between the sensor event and ADLs.

Attempts have been made using knowledge-driven approach for concurrent AR (KCAR) [38]. KCAR uses the conceptual descriptions of the everyday entities within the smart environment in the domain ontology to perform similarity comparison using distance measure [least common subsumer (LCS)] between two nodes. For instance, the distance similarity would be closer between Stove and Computer class in the hierarchy than Stove and Computer. Furthermore, the pyramid matching kernel (PMK) approach was adapted to deal with infrequent sensor noise. The PMK technique is centred on image-based object detection where the images are compared by segmenting the image into grids at different levels to detect key features and perform statistical, which also has the capability to perform matching on hierarchical concepts. In addition, the study claims to recognise multi-user activities.

Work in [3] presents a cloud-based mobile AR system, namely MyActivity. It exploits the hybrid model using both machine learning techniques (data-driven) and stream reasoning with an ontological representation of activities (knowledge-driven). The system uses continuous SPARQL (C- SPARQL) query language in order to infer human activities with accelerometer and GPS data from mobile devices. Likewise, work in [9] presents an ontology-based hybrid approach by modelling initial domain knowledge as a "seed" ontology model and using data-driven approach to incrementally discover new activities and update the "seed" model.

In [20], the quality of the sensor observation is investigated using different quality dimensions in a probabilistic data stream management system (PDSMS). The work leverage with SSN as a vocabulary to describe relationship between sensors and their observations values to estimate prevailing conditions and propagate current quality information. The work in [15] presents a two-level probabilistic framework for concurrent and interleaving goal and activity recognition (CIGAR). It leverages on the skip-chain conditional random fields (SCCRF) approach for modelling interleaving goals and concurrent goals by adjusting inferred probabilities through a correlation graph. The reason about goal interactions is done explicitly through the correlation graph. Work in [10] provides a survey on transfer-based learning approaches. It categorises transfer-based learning approach in four ways, sensors modality, by the differences in source and target environments, data availability, and the type of information being transferred. It further highlights researches carried by the types of knowledge being transferred in relation to sensor modality and the data labelling process. From the grouping of the different studies in a table, it is clear that limited studies have been carried out in informed unsupervised (IU) and uninformed unsupervised (UU) data labelling/learning process and the relational knowledge transfer types.

In recent studies, a wide range of common ontologies (vocabularies), querying languages, software libraries and technologies have been used. SSN has been used as a vocabulary in many studies [3, 6, 20] to describe relationships between multiple sensing devices and their observation values. The SSN was first released during 2010 by the W3C Semantic Sensor Network Incubator Group. The SSN was developed with the goal to allow syntactical interoperability to provide an additional layer for the semantic compatibility, just like the Sensor Model Language (SensorML) and the Observations and Measurements (O&M) [6, 20].

C-SPARQL [3, 30, 34], SPARQLStream and STARQL [41] are query languages developed to support the continuous RDF-based sensor data stream. These languages originate from SPARQL Protocol and RDF Query Language (SPARQL). For instance, work in [19] proposes an ontology, STARQL to SQL mapping, and query language for Ontology Based Data Access (OBDA) to aggregate and perform analytical tasks over static and streaming. Likewise, work in [4] proposes a streaming linked data framework (SLD) to support for the event organisers to visualise crowd movements in real time from the social networking sites i.e. geotags from Twitter posts. Central to their approach is the C-SPARQL querying for analysing two city-scale-wide social events (London Olympic Games 2012, and Milano Design Week 2013). Work in [26] proposes a knowledge acquisition method that processes real-world sensor data to automatically generate and evolve topical ontologies based on rules which are also automatically extracted from external sources. To do this, the k-means clustering method and statistic model are used to extract and link relevant concepts from the raw sensor data and represent them in the form of a topical ontology. In the later stage, rule-based approach is adapted to label the concepts.

Apache Jena is one of the popular application programming interfaces (API) used for manipulating RDF data, inferencing and reasoning. It has been applied to studies such as the Tsunami early warning system [25], Taiyuan hospital information retrieval system [40] and Yunnan tourism [39]. Similarly, the work in [16] proposes a semantic web architecture for sensor networks (SWASN) using Jena API for inferencing and reasoning. The prototype is developed based on SWASN and has been applied on building fire emergency scenarios, and claims to understand the sensor information and process sensor information on the semantic level. SWASN defines their own sensor ontology and uses Jena API to query the sensor data and extract meaningful information through inferencing. The study in [12] presents the performance between Drools and Jena API-based systems that were used for event processing. The result on their study concluded that while Drools is about 40% faster, Jena consumes about 15% less memory.

This paper proposes a semantical approach for segmenting sensor data stream into multiple activity threads with the property description of a given sensor that it is attached with. Central to the approach is the expressive ontological model to perform terminology box (T-Box) and assertion box (A-Box) reasoning [28, 29], generic and user-specific logical rules, dynamic window size analyses [23] and continuous RDF querying language. In addition, different from other approaches, the proposed segmentation approach is sensitive to changes made to the ontological modal by a given activity learning algorithm, rules (non/specific to user) and user-defined preferences.

3 A semantic-based approach to multi-activity data segmentation

The semantic-enabled sensor data processing is proposed to automatically separate and segment the data stream into multiple dynamic threads. The proposed approach encodes the domain knowledge into an ontology model (ADL Activity adapted from [22]), allows user to define preferences (A-Box) and rules, and uses time series analysis defined in [23] to segment the activity sequences and perform inferencing. In addition, the approach allows new learnt activities to be incorporated into the initial activity model and will be automatically taken into consideration during the segmentation phase.

3.1 Characterisation of complex activity

The characteristic of the single-user activity is when the sub-activities are performed after one another in a (non) sequential order. In the complex interleaving activity, two sub-activities can be performed in between them, whereas, in concurrent activity, these two sub-activities can overlap each other. The parallel activities, on the other hand, have two activities being executed independently of each other. Table 1 illustrates how a single-user can perform three activities (a, b and c).

Table 1 An example of a complex user activity

Single User Activity	Sequential	Non-sequential
Single	aaabbbccc	aaacccbbb
Interwoven (interdependent or interrupted)	aa a bb b ccc	aa a cc c bbb
Concurrent (overlapping)	aa a bb b ccc	aa a cc c bbb
Parallel (independent activities)	aa a bb b c cc	aaa ccc bb b

3.2 Ontological activity modelling

A generic ADL ontology model can be created with interrelated class, properties and instances to express the sensors, environmental objects, activities, complex activity characteristics and user preferences. For a more comprehensive guide on working with ontologies see [18, 22]. An example of how the MakeTea activity class and its properties are described in the Fig. 3 below. These class descriptions can then be used to infer the type of the individual instance, see Sect. 3.3. The *MakeTea* activity class can be defined as a sub-class of *MakeHotDrink* and use *owl:Restrictions* and *onProperty* to describe the activity. For example, *hasAdding Milk; Sugar, hasContainer Cup*, and *hasHotDrinkType Tea*. Each of these object properties have domain and range associating to *MakeHotDrink* and *HotDrinkType*.

3.3 Separation and segmentation approach

The incoming sensor data from the WSNs are received by the web service via gateway from the smart environment, and the web service appended the events into the broadcasting queue. The web service has active session thread which mainly infers whether an incoming sensor event is a start of a new activity or associated to the active sub-activity threads that are already running. The logistics of threads management is further described in Sect. 4.2. This section focuses on answering the question "is this sensor object event, X_i, linked with activity thread Z_j?" Fig. 4 presents a generic overview of the proposed segmentation approach and the green diamond indicating where the above question is being asked in the whole process.

To find the relationship between the sensor event X_i and an activity thread Z_j in the session, few input data are required by the reasoner to perform generic or user-specific activity segmentation. The generic T-Box and logical rules reasoning is done automatically by using Jena API with internal reasoners or connecting one of the wide varieties of external reasoners available such as Hermit, Pellet and FaCT++. On the other hand, user-specific inferencing is performed by using A-Box approach to infer from loaded

```
:MakeTea rdf:type owl:Class ;   rdfs:subClassOf :MakeHotDrink ,
    [ rdf:type owl:Restriction ; :onProperty :hasAdding ; :someValuesFrom :Milk] ,
    [ rdf:type owl:Restriction ; :onProperty :hasAdding ; :allValuesFrom [ rdf:type owl:Class ; :unionOf ( :Milk :Sugar)]] ,
    [ rdf:type owl:Restriction ; owl:onProperty :hasContainer ; :someValuesFrom :Cup] ,
    [ rdf:type owl:Restriction ; :onProperty :hasAdding ; :someValuesFrom :Sugar] ,
    [ rdf:type owl:Restriction ; :onProperty :hasHotDrinkType ;:someValuesFrom :Tea] .
```

Fig. 3 MakeTea activity class description in turtle format

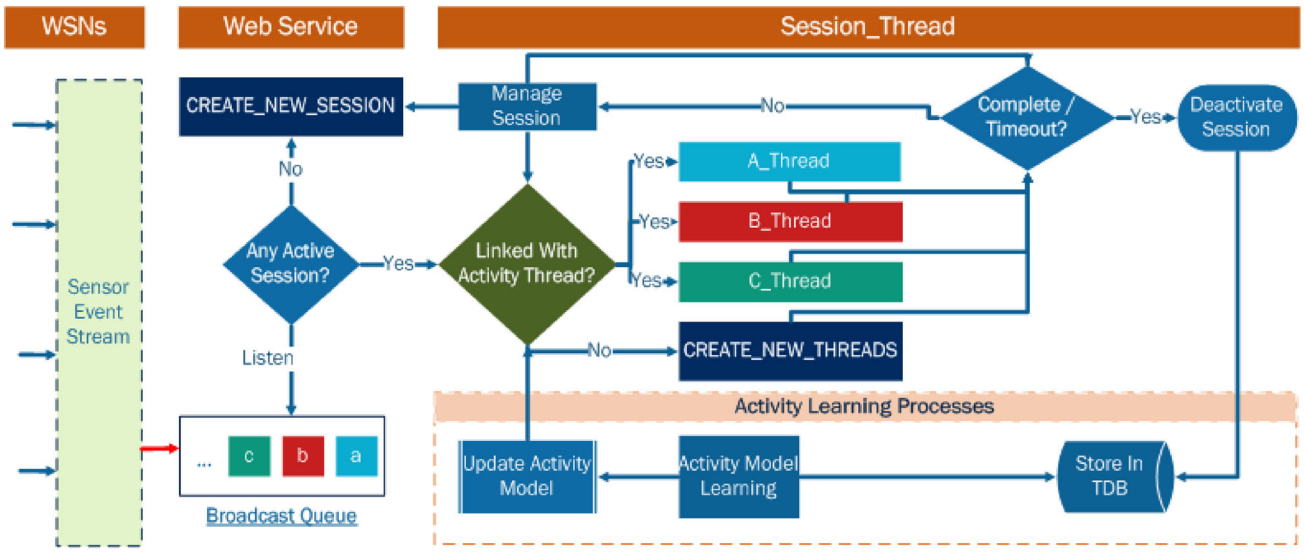


Fig. 4 Generic overview of the proposed approach to manage sensor data stream

knowledge model with user-specific rules [i.e. Semantic Web Rule Language (SWRL)] [31] or using SPARQL Inferencing Notation (SPIN))[33]) or directly querying the triplestore containing pre-defined user preferences. Therefore, the reasoner would have access to the domain ontology, semantical data store accessed via SPARQL-based query languages and the sensor data streamed mapped in RDF format. Hence, the reasoner has the ability to support T-box and A-box reasoning to extract the features such as activity type, context, total duration and the complexity in which the activities are performed within the session. Figure 5 describes the overall inferencing and reasoning approach for a given activity thread.

When setting up the sensor objects initially, they are generally encoded with limited information such as location and what object it is attached and not how it can be used. Therefore, making it difficult to add relationship/property type between sensor observations and the instance of an activity, i.e. *Thread_makeTea hasUtensil kettle*. The individual *Thread_makeTea* with a list of sensors with appropriate object property type can be used to allow automatic T-Box inferencing for the individual type of the ADL class. For instance, does the *Thread_makeTea* individual with the given sensor observations satisfy activity class description to be a *KitchenADL, MakeDrink or MakeTea*? One of the approaches is to use the description of the sensor object and the type of object it is attached to for determining the relevant object property of a sensor object to aid in performing the T-box reasoning on the sensor stream. For example, the *BritishTeaObj* sensor object can be defined with the individual type and location of the sensor and the *BritishTea* object defined as type of an *Tea* and individual, and links the *BritishTeaObj* sensor object with *hasSensor*; see the relationship between with individuals and class definitions in Table 2. A simple query can be performed to find which individual object the sensor is attached to, its type and what this is used for using domain and range of the properties. Figure 6 shows the SPARQL query and the result

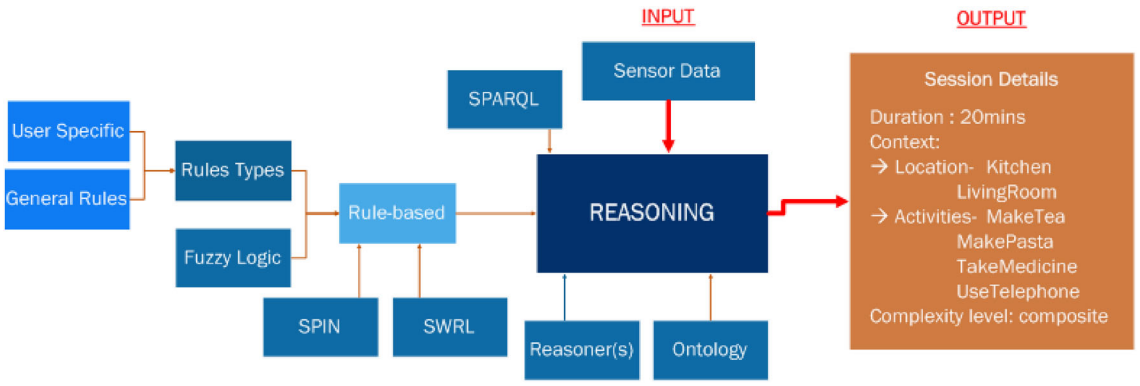


Fig. 5 Overview of the reasoning approach

Table 2 Illustrating the relationship between BritishTeaObj sensor individual, property and classes

Sensor: BritishTeaObj which is attached to BritishTea object.		
Individuals	**# 1) Sensor Object** :BritishTeaObj rdf:type owl:NamedIndividual , :ContactSensor ; :hasLocation :Kitchen.	**# 2) Object and the attached Sensor** :BritishTea rdf:type owl:NamedIndividual , :Tea; :hasSensor :BritishTeaObj.
Class and Objects	**# 3) Tea class description** :Tea rdf:type owl:Class ; rdfs:subClassOf :HotDrinkType.	**# 4) Object property of HotDrinkType** :hasHotDrinkType rdf:type owl:FunctionalProperty , owl:ObjectProperty; rdfs:range :HotDrinkType;rdfs:domain :MakeHotDrink ; rdfs:subPropertyOf :hasDrinkType .

```
PREFIX : <https://my.tech.dmu.ac.uk/~p10537478/MScProject/MobileSmartHomeOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT DISTINCT ?class ?type ?property ?domain ?range
WHERE {
  # 1) Get the class of the object that the BritistTeaObj sensor is attached
  ?subject :hasSensor :BritishTeaObj.
  ?subject rdf:type ?type. #Tea

  # 2) Get suitable property to represent the BritistTeaObj sensor
  ?class a owl:Class.
  ?class rdfs:subClassOf ?restriction.
  ?restriction owl:someValuesFrom ?type.
  ?restriction owl:onProperty ?property.
  OPTIONAL {?property rdfs:domain ?domain.}
  OPTIONAL {?property rdfs:range ?range.}
}
LIMIT 25
```

Showing 1 to 1 of 1 entries — Search: — Show

	class	type	property	domain	range
1	:MakeTea	:Tea	:hasHotDrinkType	:MakeHotDrink	:HotDrinkType

Showing 1 to 1 of 1 entries

Fig. 6 Finding relevant object property for the sensor "BritishTeaObj" that the object is attached to

table identifying the property to be used as *hasHotDrinkType* for the *BritishTeaObj* in the *Thread_makeTea* individual.

Let us now assume another sensor object, *KettleObj* activated. The *KettleObj* would have been defined as an individual with a location and another everyday object defined as kettle, individual which *hasSensor* or *isSensorOf KettleObj* and the instance class type of *Kettle*. Therefore, *Thread_makeTea* individual will record the incoming sensor events, *KettleObj*, with the associate everyday object, *kettle*, with appropriate property in order to automatically inferring the class type of the individual. The same method can be applied as above by finding the parent class of the *Kettle* class and find the object property with a *rdfs:range* of *CookingUtensil*, which is *hasUtensil*. Similarly, as other sensors objects activate and the ADL unfolds, the relevant predicate and object property can be generated and populated as shown on the right-hand side of Fig. 7. When the reasoner is activated, it can actively infer the type of the *Thread_makeTea* individual is *MakeTea* from the given activity sequence presented in see Fig. 7 .

However, a simple T-box reasoning on a generic domain model is still not enough as user may have specific items or ways to make a tea, i.e. by adding ginger and using a pot instead of a kettle to make the tea. Therefore, to support the user-specific ADL, A-box (a) and rule-based (b) approach is employed (for example see Fig. 8). The A-box reasoning approach defined here [28, 29] and rule-based generated in SWRL [31] or SPIN [33] is employed. Furthermore, dynamic time series analysis [24] is required to calculate the window size of the overall session and sub-activity threads. This method requires user to create a static pre-defined ADLs initially; however, in future, with the help of different activity learning approaches, the initial generic

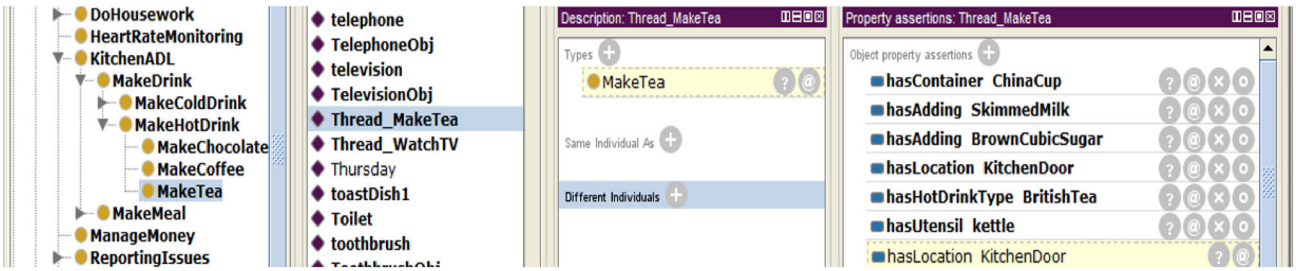
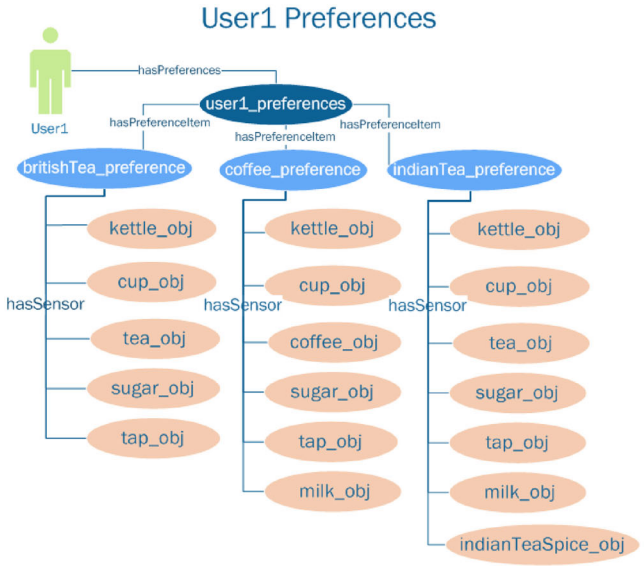


Fig. 7 T-Box reasoning for MakeTea activity from the given object properties in Thread_MakeTea



(a) User pre-defined instances for A-box reasoning

Generic rules

SweetTurmericMilk ← Milk(?x) ∧ hasTurmericPowder(?x, ?y) ∧ hasSugar(?x, ?z) ∧ swrlb:greaterThan (?z, 3)

TurmericMilkForColdORFlu(?x) ← Milk(?x, ?b) ∧ hasTurmericPowder(?x, ?a) ∧ hasSugar(?x, ?c) ∧ hasGinger(?x, ?d) ∧ hasHoney(?x, ?e) ∧ hasMint(?x, ?f)

User specific rule

CokeFloatIceCreamDrink(?x) ← hasCoke(?x, ?a) ∧ hasCup(?x, ?b) ∧ hasIceCream(?x, ?c)

(b) Generic and user specific SWRL rules example

Fig. 8 An example of with pre-defined user preferences for A-Box reasoning (**a**) and SWRL rules (**b**)

model for T-Box reasoning and user-specific preferences/ rules can be enriched overtime.

4 Real-time semantic segmentation algorithm

There are two main components involved in processing the stream data from the WSNs: web service and triplestore. The triplestore such as Jena Fuseki server can be deployed externally as an endpoint or embed it within the web service to store and manipulate the RDF-based data. The web service is responsible to manage the incoming sensor data stream queue, separate the data into multiple sub-threads in a given active session, perform activity inferencing and reasoning with a given reasoning engine and communicate with the triplestore to store/manipulate session data to allow online/ offline activity modelling learning. There are three main types of threads: session, activity and reasoning engine thread.

A session thread has an activity manager which performs three main tasks: detecting new activity unfolding from the message queue and list of sub-activities already running, managing list of sub-activity threads, and checking if the session is complete or timed out. Furthermore, upon completion of the session, the assistance results (if any) are sent to the client(s), and relevant data of the session are stored into the triplestore to enable future activity model learning algorithm to extract more axioms. The task of detecting new activity unfolding and creating the new activity thread is achieved by assessing the relations of each sensor events in the message queue against all the existing active activity threads. The relation of a sensor event in a given activity thread is true when the representative element/type of the class returned by the reasoner is equivalent or is within a sub-class of the present representative element/type. For instance, if the present representative element/type of a given activity thread is *KitchenADL* and the new reasoned result is *MakeDrink*, then the association of the sensor event with the given activity is true. However, when the scenario is inverse, then the sensor event is most likely to be part of other active activity threads in the list or a start of new activity. The

second and third tasks of the activity manager in the session thread involves removing/closing the completed activity threads from the list of active activity threads and managing timed out activity by assessing previous other active activities or the incomplete activities from the previous sessions.

On the other hand, the activity thread captures the associated sensor events from the message queue until the activity is complete or timeout. Furthermore, the activity threads can be subdivided to perform two types of inferencing, T-Box and A-Box. The T-box activity thread use reasoning engine thread mainly to perform inferencing using domain knowledge model and using generic rules. The A-Box activity thread can either use reasoning engine to run queries with user-specific rules on the inferred model or directly executing queries on the triplestore to find the user-specific preference(s) on a given set of sensor observations. The A-Box activity thread can then potentially retrieve rest of the sensor observations from the user preferences identified from the previous step and temporarily store in the memory instead of making a query request for each sensor events. Although this approach would reduce the processing tasks but also taking up more random access memory (RAM) which is already limited on a standard server/computer.

The reasoning engine thread supports T-Box, A-Box and rule-based (generic/user-specific) inferencing as described above and in Sect. 3.3, to identify the activity class type on a given activity instance with a set of sensor observations of an activity thread. As mentioned previously, there are various reasoning implementation; however, Jena API is considered to be more suitable due to its performance in the previous studies, supports for external reasoners and the Java programming based implementation. This reasoning thread could also be a singleton class with synchronised access to all the activity threads to perform inferencing on the loaded knowledge model and rules. However, the synchronised approach could cause delays for multiple clients running many sub-activity threads. Therefore, asynchronised mechanism will need to be implemented to enhance the performance of the reasoning engine and the scalability.

Figure 9 provides an overview of the steps proposed to separate and segment the data stream into as a flowchart.

4.1 Stream processing algorithm

The web service will ensure to have an active session thread to listen to the incoming sensor data stream. In addition to the three tasks performed by the session, it dynamically updates the maximum timeout depended on the maximum duration required by the sub-activity threads to complete. The sub-activity threads are responsible for listening to the sensor events queue and infer whether the sensor observation belongs to the activity thread. If the match is found, sensor event data will be appended to the activity thread, the result is broadcasted back to the clients and the timeout window size is revaluated. There are two types of threads being created, T-Box and A-Box. A T-Box thread is initially created to identify the unfolding events and no other T-Box or A-Box threads created unless the representative inferred class has no sub-classes in the model and no user preferences using A-Box could be found with current set of sensors.

The pseudo algorithm defined in Table 3 describes the core part of the segmentation process where the beginning

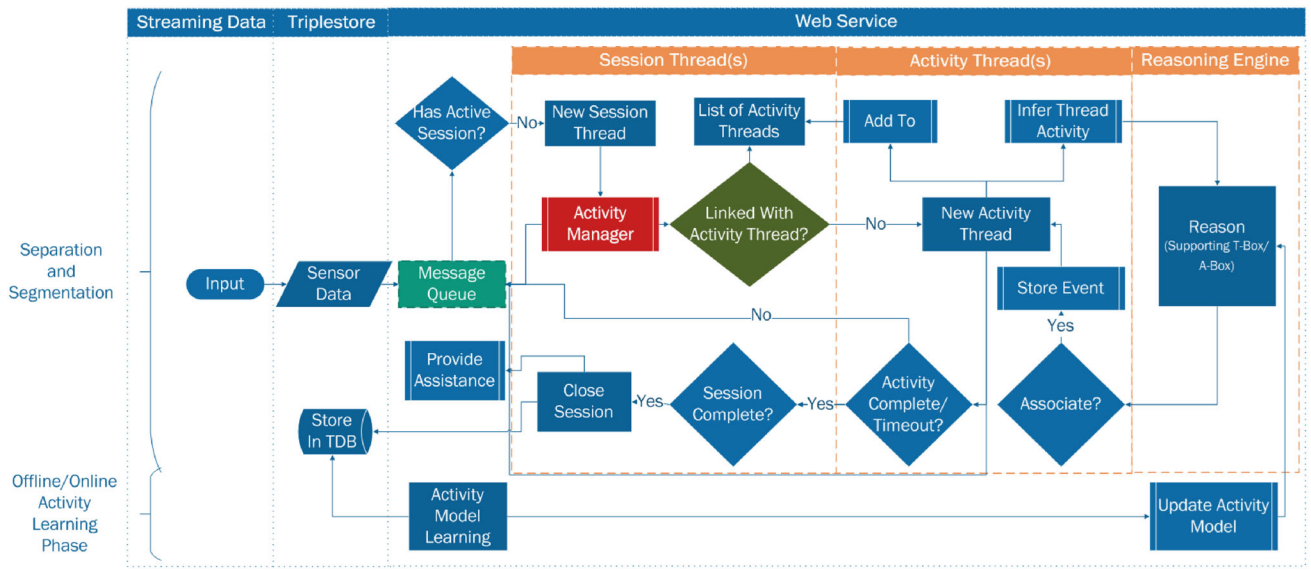


Fig. 9 Overview of segmentation steps on the sensor data stream

Table 3 Pseudo algorithm to semantically segment incoming data stream into multiple threads

```
Input: [e1, ei …, en] // raw sensor data sequences passed by the run function in the session thread.
Output: void
//1) check if e is a start of new activity or associated with existing sub activity T-Box threads
for each TBoxThread t : tboxThreads
        tempList = t.getCurrentSensorsList();   tempList.add(e);
        OWLClass c = Utills.runTBoxInferencing(tempList);   // returns representative class for a given list of sensors
        if !t.isSubClassOREqualToExisitingTBoxThreads(c)
                boolean found = false;
                //2) are there matching user preferences in the list of active A-box threads?
                for each ABoxThread at : aboxThreads (at.checkLinks(tempList)? found = true:) end for each
                //3) if there are no active A-Box threads with relevant user preferences but has some in the triplestore
                if !found && opentialAboxPreferences(tempList).size() > 0
                        aboxThreads.add(createNewABoxThreads(opentialAboxPreferences(tempList)));
                // otherwise new T-Box thread is created and recored with the inspecting sensor.
                else if !found && !opentialAboxPreferences(tempList).size()>0
                        tboxThreads.add(createNewTBoxThreads(e)); end else if
        end if
end for each
//4) maintain current list of A-box threads which are completed or timed.
manageCompeletedAndTimeoutCases(tboxThreads, aboxThreads);
```

of a new activity is detected and the new threads are created and recycled and the maximum duration window size are re-/evaluated. This process is performed by the activity manager in the session thread, and they are broken down into four stages.

The first stage is to iterate over all the active T-Box threads and use the current list of sensors observations in each thread along with the sensor event being investigated to execute a new T-Box inferencing result. This new result will return a representative class of an ADL, and it is then compared against the current activity class to decide if the sensor event is part of the ongoing activity. To do this, checks are made with the result class and current class if they are equal or if the new result is within the sub-classes/hierarchy of the current ADL class. If the result is true, no thread is created and waits for the next sensor event. In the second stage, where the result is false, similar checking is made within all the active A-Box threads. A binary flag is used to indicate if A-Box thread has already processed the sensor or not. The third stage is where the decision is made whether to create a new A-Box thread or T-Box. The A-Box thread is only created if the new sensor event is a part of an ongoing activity and has some user personal preference(s) stored in the triplestore, for which, multiple A-Box threads are executed. Otherwise, it is determined that the new sensor event is a start of a new activity, hence, starting a new T-Box thread. The final stage is where all the housekeeping for the sub-threads and the process of revaluating the maximum session timeout window takes place. The housekeeping of the threads is further discussed in the Sect. 4.2.

4.2 Session and activity threads management logistics

The notion of multithreading is adapted whereby each active session creates sub-activity threads and inspects individual sensor events from the message queue. Figure 10 illustrates how activity manager in the session thread performs housekeeping tasks described in previous section as a flowchart. The sub-activity threads, A-Box or T-Box threads, have two internal flags for the activity completion and timeout. The activity manager in its fourth phase of the algorithm checks these two flags statuses and performs specific set of tasks. In the case where all sub-activity threads are complete, the session encodes all the relevant data, store it in the triple and set its status to deactivate/complete to indicate to the web service to start a new session for the client. This will enable the virtual machine to recycle the threads and reduce the memory consumptions over period of time.

The timeout case for a given thread is handled by investigating existing shared events in the session and

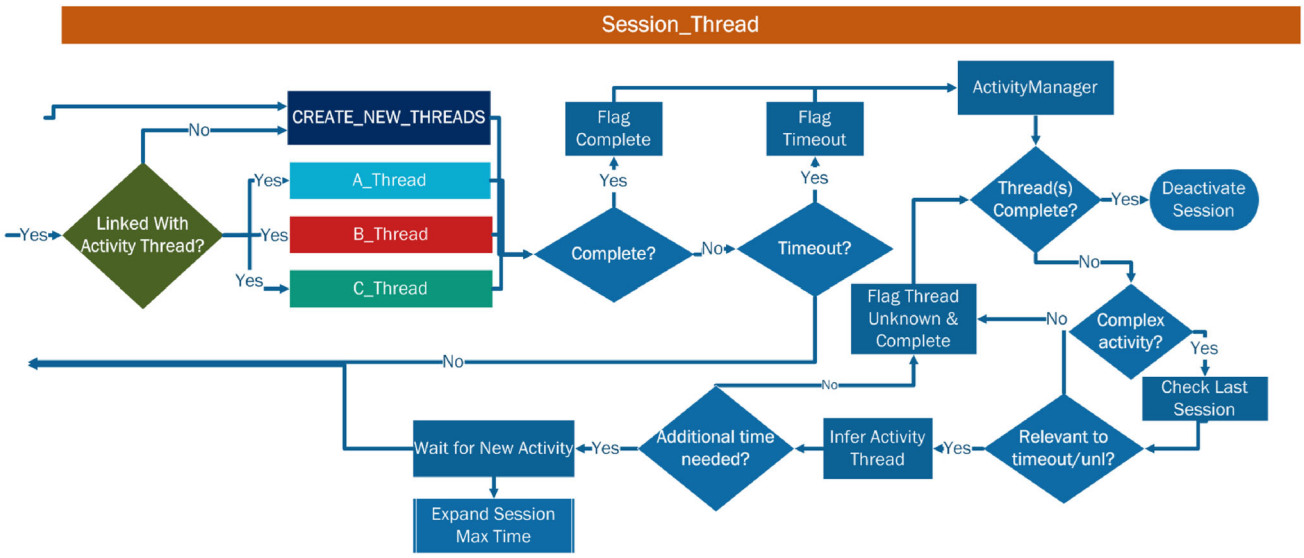


Fig. 10 Overview of the session and activity threads creation with interlinking mechanism

previous session stored in the triplestore. A given activity thread may share some sensor events, for instance, opening kitchen cupboard (*KitchenCupboard1Obj*) to get a *TeaCupObj* and *PastaSauceJarObj*. However, due to the way the threads being created at different interval, they miss some shared activities in the message queue. In this case, a copy of these shared activities are mark as shared and kept within the session with the references to the threads for further analysis. Similarly, the current session threads are also interlinked with previous session threads details and stored in the triplestore. The current and previous session details are further compared to find any relevance between with unfinished, unknown, or has timed out activity threads. This will allow session thread to evaluate whether previous sessions prematurely completed activities and current sessions incomplete activities are related. The session and sub-activity threads manage their own timeout window size, listen to the sensor message queue and perform incremental reasoning to add the related sensor events to the activity sequence.

5 Evaluation and discussion

5.1 Use-case application scenario

The following user case study example is presented to convey how the aforementioned semantic segmentation approach will operate in a given scenario. Robert is retired, 70-year-old man who lives by himself and has a mild form of Dementia. Roberts case has affected his quality of living in several ways. He now has trouble remembering how to carry out ADLs, and finds it difficult to navigate to familiar places. All of the information related to Roberts health condition in addition to personal information are modelled and stored within the modelling and management layer of the system within his unique user profile. The scenario is that Robert frequently makes pasta (*MakingPasta*) and tea (*MakingTea*) for his dinner. He starts preparing his meal around 18:20 and takes his medicine (*TakeMedicine*) before having dinner. During this process, Robert regularly gets a phone call (*TakePhoneCall*) from his son and/or daughter-in-law after they get home from work. Due to the dependent, interwoven and concurrent complexity of these activities being performed by Robert, he frequently forgets to keep track of the tasks he has already performed for the activity. Thus, Robert has not always been able to enjoy his dinner due to various reasons, such as missing ingredients, and over cooking. Figure 11 provides a snapshot of how Robert may go about performing his complex activity described above and how they will be segmented using the proposed semantical approach. In the given scenario above, Robert is performing four different activities with some activities be depended on another, interwoven and concurrently. The activity readings are initiated from when Robert enters kitchen (*KitchenDoorObj*) and then starts taking items from the kitchen cupboard (*KitchenCupboardObj*). The items that he takes out consist of *MedicineObj, PastaSauceJarObj, TeaObj and SugarObj*. At this stage, T-Box reasoning is performed incrementally and detects that three different types of ADL are being performed and creates new A-Box threads: *MakingTea* (*TeaObj* and *SugarObj*), *MakeMeal/MakingPasta (PastaSauceJarObj)* and *TakeMedicine (MedicineObj)*. These sequences of activities illustrate how opening kitchen cupboard is seen as a shared/depended across multiple sub-

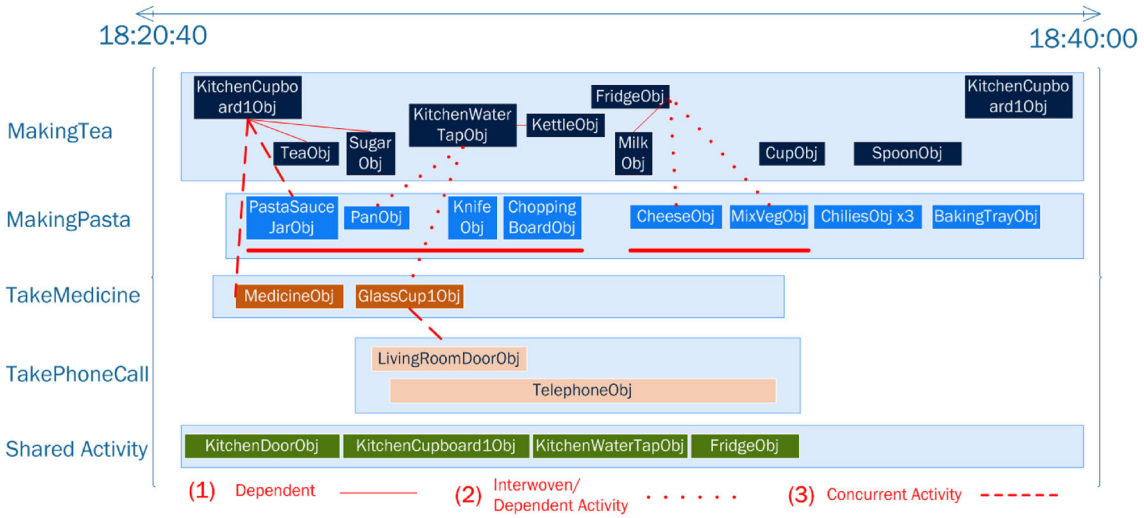


Fig. 11 An example of a session thread consisting of multiple activity threads with *1* dependent, *2* interwoven/dependent activity and *3* concurrent activities

activity. Moreover, as the activity unfolds, it appears that the kitchen cupboard is left open and more objects are retrieved from it or they are already outside: *PanObj, GlassCup1Obj, KnifeObj* and *ChoppingBoardObj*. These four sensor events would be filtered and stored in A-Box, respectively. During this phase, Robert may also receive/-make a phone call and makes his way through the opened kitchen door to the *LivingRoomDoorObj* and picks up the *TelephoneObj*. This two sensor events will make the activity manager in the session create a new T-Box reasoning thread and infer that the user might be making/taking a phone call; for illustration purposes, *TakePhoneCall* class is returned. The *TelephoneObj* sensor continues to send the message for interactions while performing the next tasks. Next, the *KitchenWaterTapObj* is activated which multiple T-Box activity threads share i.e. *MakingTea, MakingPasta* and *TakeMedicine*. Next, the *KettleObj* event is received which is seen as a utensil item for *MakingTea* thread. The same process is repeated for the next sensor observations which are mainly added to *MakingTea* and *MakingPasta* T-Box threads: *FridgeObj, MilkObj, CheeseObj, MixVegObj, CupObj, ChilliesObj, SpoonObj, BakingTrayObj and KitchenCupboard1Obj*.

Furthermore, to illustrate how a given activity thread perform the reasoning and what type of result it output, let us extend the *MakingPasta* activity thread (which performs T-Box reasoning) with a new incoming sensor event named *PastaObj*. Table 4 defines these two input parameters and outputs a Result object containing three levels of information from ADL class type, any matched rules and any specific user preference. The reasoning engine thread performs T-Box reasoning using domain ontology would return the ADL class description to be *MakePasta*. However, with the help of activity manager, another thread running A-Box reasoning due to *chiliObj* not defined as a standard adding to make pasta may return the user-specific rule *SpicyMixedVegPasta* and the preference named *spicyVegPasta_ preference*. The results are then mapped within a class and sent back to the activity thread to update the activity sequence and window size. However, currently, these two threads running A-Box and T-Box reasoning output disjointed results and it can come together by mapping a user-specific preference with an ADL class in the triplestore and the activity manager in the session thread map their outputs results together to output useful information as described in Table 4.

In general, as the activity unfolds, new sensor events are continuously monitored by the activity threads and the activity manager in the session. The activity threads incrementally run the reasoning engine and recalculate the activity window size. The result of the semantical segmentation approach creates a set of sensor event sequences within a given activity thread. These sequences are then further analysed by AR algorithm to be developed on top of the segmentation phase to decipher other inexplicit attributes such as contextual location, composite or simple activity, and generic or user-specific activity.

5.2 Discussion

The approach currently supports the semantical segmentation of a single-user complex activity. Furthermore, work in automatically detecting and segmenting the multi-user complex activity is actively being investigated. The detection parameters would be the key enabler to recognise multi-user activity. In addition, the activity learning algorithm is required to automatically enrich the domain ontology, user preferences and logical rules after inferring

Table 4 Reasoning engine to finding associate links with a given thread sensor sequences and sensor event

Reasoning Engine – linked with activity thread(s)?

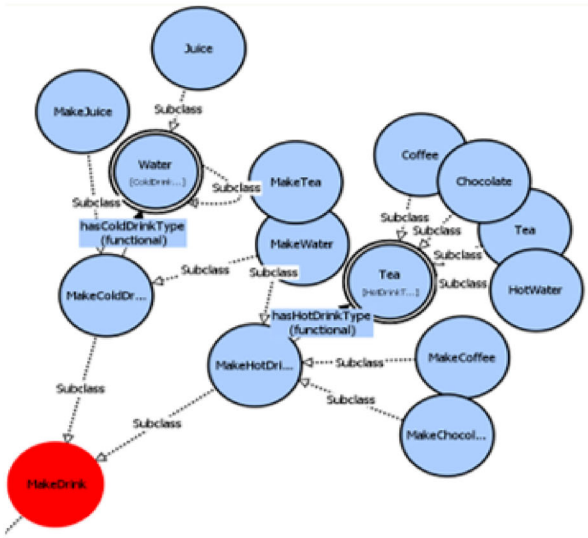
Input: *(SensorEvent, ThreadSensorSequences)*

- *Sensor Event: PastaObj,*
- *ThreadSensorSequences:* KitchenCupboard, PastaSauceJarObj, PanObj, KnifeObj, ChopppingBoardObj, CheeseObj, MixVegObj, ChiliObj x3, BakingTrayObj. **(MakingPasta)**

Output: *Result r* <MakePasta, spicyVegPasta_preference, SpicyMixedVegPasta>

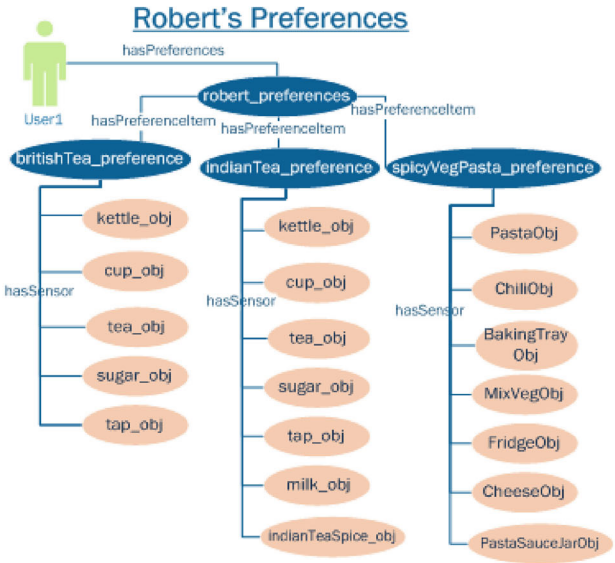
Level 1 – T-Box

→ *Ontology: domain, SSN, other common vocabularies.*



Level 2 – A-Box reasoning

→ *User Preferences(individual)*



Level 3 – Rules

→ *SWRL rules*

SpicyMixedVegPasta(?x) ← Pasta(?x, ?a) ^ hasMixVeg(?x, ?b) ^ hasPastaSauce(?x, ?c) ^ HasChili(?x, ?e) ^ swrl:greaterthan(?e, 3)

new activity patterns from the unknown session data stored in the triplestore. Although this approach enables the updating generic models and user-specific preferences and rules to be easily enriched, managing conflicts and consistency issues between general and user-specific knowledge representation could create further challenges.

One of the limitations of the proposed thread management system is that every activity thread within a session listening to the broadcast will perform inferencing upon a single-sensor event, which means $N-1$ number of activity threads may request to perform inferencing unnecessarily and creating excess computation overheads, delays to process next events and energy. One way to reduce inferencing request is by allowing active session thread to incrementally check against the list of missing/expected activity sensor sequences. These sensor sequences can be preconfigured by retrieving either from user preferences (for A-Box threads) or individuals with the type class described for a given ADL (for T-Box threads). Nevertheless, the activity manager in session thread would still iterate over the individual activity threads but do not need to use reasoning engine as the list of missing/potential sensor would be available within each activity thread. Another approach could be defined with an analogy of a lost child (sensor event) and a policeman (thread with predefined procedures) trying to figure out where the child

lives. The policeman can take the child safely to the parents house (linked activity sequences) or report as found and wait for the correct authority (new sensor event). This approach assumes child (sensor event) has limited metadata about itself. However, this approach also has its limitation because a child (sensor event with metadata) may have multiple parents claiming for custody (i.e. sensor event belonging to multiple ADLs).

6 Conclusion

This paper presents a semantical-based approach to separating and segmenting the real-time sensor stream into multithreads. The notion of interlinked session, sub-activity and broadcast queue are used to not only calculate dynamic time window but also infer the ongoing activity. The ontology and rules are used to infer and segment a given complex activity to support AR phase. In addition to terminology (T-Box) and generic rules, user preferences-based assertion (A-Box) and rules are applied to find any association of a sensor event within a given activity thread. This approach further enables new learnt activity models and rules to be more easily incorporated into the existing model at run-time and automatically taken into consideration during the separation and segmentation process. The future direction of this work is to implement and evaluate the proposed approach within a real-time sensing environment that was developed in the previous work [28, 29]. Moreover, efficient HAR and activity learning algorithms will be investigated to enrich and expand the initial domain model incrementally over period of time to provide impersonal and personal service to the user(s). After achieving desired accuracy and performance of a single-user complex AR, the challenge of identifying and tracking multi-user complex activities will be investigated.

References

1. Awan MA, Guangbin Z, Kim S-D (2012) Activity recognition in WSN: A data-driven approach. Computing and Convergence Technology (ICCCT), 2012 7th International Conference on IEEE
2. Azkune G, Almeida A, Lpez-de-Ipia D, Chen L (2015) Extending knowledge-driven activity models through data-driven learning techniques. Expert Syst Appl 42(6):3115–3128. doi:10.1016/j.eswa.2014.11.063
3. BakhshandehAbkenar A, Loke SW (2014) MyActivity: cloud-hosted continuous activity recognition using ontology-based stream reasoning. In: MOBILECLOUD '14 Proceedings of the 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp 117–126. doi:10.1109/MobileCloud.2014.27
4. Balduini M, Della Valle E, DellAglio D, Tsytsarau M, Palpanas T, Confalonieri C (2013) Social listening of city scale events using the streaming linked data framework. In: ISWC '13 Proceedings of the 12th International Semantic Web Conference - Part II 8219:1–16. doi:10.1007/978-3-642-41338-4_1
5. Barnaghi P, Wang W, Dong L, Wang C (2013) A linked-data model for semantic sensor streams. In: GREENCOM-ITHINGS-CPSCOM '13 Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp 468–475. doi:10.1109/GreenCom-iThings-CPSCom.95
6. Calbimonte J, Jeung H, Corcho O, Aberer K (2011) Semantic sensor data search in a large-scale federated sensor network. Semant Sens Netw 839:23–38
7. Chen L, Hoey J, Nugent CD, Cook DJ, Yu Z (2012) Sensor-based activity recognition. IEEE Trans Syst Man Cybern Part C Appl Rev 42:790–808. doi:10.1109/TSMCC.2012.2198883
8. Chen L, Nugent CD, Wang H (2012) A knowledge-driven approach to activity recognition in smart homes. IEEE Trans Knowl Data Eng 24:961–974. doi:10.1109/TKDE.2011.51
9. Chen L, Nugent C, Okeyo G (2014) An ontology-based hybrid approach to activity modeling for smart homes. IEEE Trans Hum Mach Syst 44:92–105. doi:10.1109/THMS.2013.2293714
10. Cook D, Feuz K, Krishnan N (2013) Transfer learning for activity recognition: a survey. Knowl Inf Syst 36:537–556. doi:10.1007/s10115-013-0665-3
11. Ferroni G, Bonfigli R, Principi E, Squartini S, Piazza F (2015) A deep neural network approach for voice activity detection in multi-room domestic scenarios. Neural Netw. doi:10.1109/IJCNN.2015.7280510
12. Fobel A, Subramanian N (2016) Comparison of the performance of Drools and Jena rule-based systems for event processing on the semantic web. In: 2016 IEEE/ACIS 14th international conference on software engineering, research, management and applications SERA 2016, pp 24–30. doi:10.1109/SERA.2016.7516153
13. Ganz F, Barnaghi P, Carrez F (2014) Acquis From Sens Data 10:1–12
14. Gu T, Wang L, Wu Z, Tao X, Lu J (2011) A pattern mining approach to sensor-based human activity recognition. IEEE Trans Knowl Data Eng 23:1359–1372. doi:10.1109/TKDE.2010.184
15. Hu DH, Yang Q (2008) CIGAR: concurrent and interleaving goal and activity recognition. In: AAAI conference on artificial intelligence, pp 1363–1368
16. Huang V, Javed MK (2008) Semantic sensor information description and processing. In: Proc-2nd Int Conf Sens Technol Appl, SENSORCOMM 2008, Incl MESH 2008 Conf Mesh Networks; ENOPT 2008 Energy Optim Wirel Sensors Networks, UNWAT 2008 Under Water Sensors Syst, pp 456–461. doi:10.1109/SENSORCOMM.2008.23
17. Kang H, Hebert M, Efros AA, Kanade T (2015) Data-driven objectness input image database of object regions high objectness regions. IEEE Trans Pattern Anal Mach Intell 37:189–195
18. Khan JA, Kumar S (2015) OWL, RDF, RDFS inference derivation using Jena semantic framework and pellet reasoner. In: 2014 international conference on advances in engineering and technology research ICAETR 2014, p 07. doi:10.1109/ICAETR.2014.7012871
19. Kharlamov E, Kotidis Y, Mailis T, Neuenstadt C, Nikolaou C, zcep,Svingos C, Zheleznyakov D, Lamparter S, Horrocks I, Ioannidis Y, Mller R (2016) Towards analytics aware ontology based access to static and streaming data (extended version), p 122. doi:10.1007/978-3-319-46547-0
20. Kuka C, Nicklas D (2014) Enriching sensor data processing with quality semantics. In: PERCOM Work 2014 IEEE international

conference on pervasive computing and communication workshops, pp 437–442. doi:10.1109/PerComW.2014.6815246
21. Liu Y, Nie L, Liu L, Rosenblum DS (2015) From action to activity: sensor-based activity recognition. Neurocomputing. doi:10.1016/j.neucom.2015.08.096
22. Okeyo G, Chen L, Wang H (2014) Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. Futur Gener Comput Syst 39:29–43. doi:10.1016/j.future.2014.02.014
23. Okeyo G, Chen L, Wang H, Sterritt R (2014) Dynamic sensor data segmentation for real time activity recognition. Pervasive Mob Comput 10(1):155–172
24. Okeyo G, Chen L, Wang H, Sterritt R (2014) Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. Pervasive Mob Comput 10:155–172. doi:10.1016/j.pmcj.2012.11.004
25. Ramar K, Mirnalinee TT (2012) An ontological representation for Tsunami early warning system. In: Advances in engineering, science and management (ICAESM), 2012 international conference on IEEE, pp 93–98
26. Riboni D, Bettini C (2011) OWL 2 modeling and reasoning with complex human activities. Pervasive Mob Comput 7:379–395. doi:10.1016/j.pmcj.2011.02.001
27. Snchez D, Tentori M, Favela J (2008) Activity recognition for the smart hospital. IEEE Intell Syst 23:50–57. doi:10.1109/MIS.2008.18
28. Triboan D, Chen L, Chen F (2016) Towards a mobile assistive system using service- oriented architecture. In (2016) IEEE Symp. Syst. Eng. Towar. IEEE, Oxford, Serv, pp 187–196
29. Triboan D, Chen L, Chen F, Wang Z (2016) Towards a service-oriented architecture for a mobile assistive system with real-time environmental sensing. Tsinghua Sci Technol 21:581–597
30. Valle ED, Grossniklaus M (2010) C-SPARQL: a continuous query language for rdf data streams. Int J Semant Comput 4:3–25
31. W3C (2004) SWRL: a semantic web rule language combining OWL and RuleML. https://www.w3.org/Submission/SWRL/. Accessed 20 Aug 2016
32. W3C (2005) Semantic sensor network ontology. https://www.w3.org/2005/Incubator/ssn/ssnx/ssn. Accessed 1 Oct 2016
33. W3C (2011) SPIN-overview and motivation. https://www.w3.org/Submission/spin-overview/. Accessed 22 Aug 2016
34. W3C (2014) RDF stream processors implementation-rdf stream processing community group. https://www.w3.org/community/rsp/wiki/RDF_Stream_Processors_Implementation. Accessed 5 Oct 2016
35. Wan J, OGrady MJ, OHare GMP (2015) Dynamic sensor event segmentation for real-time activity recognition in a smart home context. Pers Ubiquitous Comput 19:287301. doi:10.1007/s00779-014-0824-x
36. Wei W, Barnaghi P (2009) Semantic annotation and reasoning for sensor data. In: Barnaghi P, Moessner K, Presser M, Meissner S (eds) Lect. Notes Comput. Sci. Springer, Berlin, pp 66–76
37. Ye J, Stevenson G, Dobson S (2014) USMART: an unsupervised semantic mining activity recognition technique. In: ACM transactions on intelligent systems, pp 4:16–116:27. doi:10.1145/2662870
38. Ye J, Stevenson G, Dobson S (2014) KCAR: a knowledge-driven approach for concurrent activity recognition. Pervasive Mob Comput 19:4770. doi:10.1016/j.pmcj.2014.02.003
39. Zhang S, Guo J, Yu Z, Lei C, Mao C, Wang H (2010) An approach of domain ontology construction based on resource model and Jena. In: 2010 third international symposium on information processing, pp 311–315. doi:10.1109/ISIP.2010.44
40. Zhang W, Duan L, Chen J (2010) Reasoning and realization based on ontology model and Jena. In: proceedings of the 2010 IEEE 5th international conference on bio-inspired computing: theories and applications BIC-TA 2010, pp 1057–1060. doi:10.1109/BICTA.2010.5645115
41. Zep L, Mller R, Neuenstadt C (2014) A stream-temporal query language for ontology based data access. KI 2014 Adv Artif Intell, pp 183–194. doi:10.1007/978-3-319-11206-0_18