University of Calcutta

Department of Computer Science & Engineering

AN ONTOLOGY BASED SEGMENTATION APPROACH FOR ACTIVITY RECOGNITION

A project report is submitted for the fulfillment of the requirement for the degree in Master of Science (M.Sc.) in Computer Science from the University of Calcutta

Submitted by,

Madhurima Sen (Roll Number: C91/CSC/201010)

Registration Number: 611-1211-1232-17

Kankana Ghosh (Roll Number: C91/CSC/201008)

Registration Number: 223-1211-0004-17

Snigdhadip Banerjee (Roll Number: C91/CSC/201020)

Registration Number: 012-1111-1394-17

2 Year M.Sc. in Computer Science

Date: June 21, 2022

Supervised by,

Dr. Sankhayan Choudhury

Professor, Department of Computer Science and Engineering

University College of Science & Technology

University of Calcutta, JD-2, JD Block, Sector-III, Bidhannagar, Kolkata, West Bengal 700106

CERTIFICATE OF APPROVAL

The project titled "**AN ONTOLOGY BASED SEGMENTATION APPROACH FOR ACTIVITY RECOGNITION" was** prepared by **Madhurima Sen (Roll Number: C91/CSC/201010), Kankana Ghosh (Roll Number: C91/CSC/201008) and Snigdhadip Banerjee (Roll Number: C91/CSC/201020)** is hereby approved and certified as a creditable study in technological subject carried out and presented in a manner satisfactory to warrant its acceptance for the fulfillment of the Master of Science(M.Sc.) degree in Computer Science from the Department of Computer Science and Engineering, the University of Calcutta for which it is submitted.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed, or, conclusions drawn therein, but approved the project report for the purpose for which it is submitted.

………………………..	……..……………………………
Project Supervisor (Computer Science and Engg.)	Chairperson, PG Board of studies

…………………………

External Examiner

ACKNOWLEDGEMENT

We are thankful to the Department of Computer and Engineering, at the University of Calcutta for allowing us to carry out our project preparation with all the necessary guidance, inspiration, and support. It is a great privilege for us to express our profound and sincere gratitude to our project guide **Dr. Sankhayan Choudhury**, Professor, Department of Computer Science and Engineering, University of Calcutta, for his guidance, valuable assistance, and inspiration throughout the course without which it would have been difficult to complete our project work.

We also express our heartiest gratitude to **Moumita Ghosh** (senior research fellow) and the faculties of the Department of Computer Science and Engineering and the Department Library for their consistent support in supplying all the necessary resources.

Cordially

…………………………………. ….………………………………

Madhurima Sen Kankana Ghosh

(Roll Number: C91/CSC/201010) (Roll Number: C91/CSC/201008)

……………………………………

Snigdhadip Banerjee

(Roll Number: C91/CSC/201020)

CONTENTS

Chapter 1
Introduction

Ambient Assisted Living (AAL) systems are being developed to support the growing aging population. An AAL system is built so that it can assist the persons living in a Smart Home (SH) environment so that they can carry out their Activities of Daily Living (ADL). A smart home that is equipped with different types of sensors in different parts of the home assists inhabitants of that home in living with comfort and safety. Individuals need to be able to complete Activities of Daily Living (ADLs) such as eating, grooming, cooking, and sleeping, to lead an independent life. Thus, automating the recognition and tracking of these ADLs is an important step toward monitoring the functional health of a person who is residing in a smart home environment. The ambient sensors located at different corners of the house continuously produce data when they get triggered.

Based on sensor data, Activity Recognition is the problem of predicting the movement of a person mainly in indoor conditions. Human Activity Recognition (AR) is one of the most prominent research topics for an ambient-assisted living (AAL) in smart spaces. Activity Recognition is required to detect the tasks performed by the individual so that assistance can be provided to them. [1] In the ambient assistant living systems, to perform human activity recognition, data segmentation plays a very significant role. By using static/dynamic window approaches, substantial efforts have been made in data segmentation. It is required to appropriately partition the continuously growing sensor data stream for running the activity recognition algorithms in real-time. To recognize activity from the sensor data, we need to segment it. In simpler terms, segmentation of data means slicing the sequence of sensor data based on a segmentation approach.[2][3] The data within a window needs to be tested for the recognition of the specific activity.

Ontology is a representation of domain knowledge. The overall objective of the work is to use the domain knowledge to propose a more accurate segmentation approach. IoT devices collect sensor data. A standardized way to form an ontology based on the sensor is by using the Semantic Sensor Network (SSN) ontology. The main intention behind using SSN ontology is to obtain a standardized version of an ontology. It gives global recognition and acceptance along with smooth interoperability. By applying the ontology to the sensor data, we are proposing a novel segmentation approach, and based on the segmentation the activity is recognized.

The rest of the report is laid out as follows: Chapter 2 walks through the prerequisites of Segmentation and Ontology. Chapter 3 describes the scope of the work. Some of the previous works on Segmentation and Activity Recognition are discussed in Chapter 4. A segmentation algorithm based on ontology is proposed in Chapter 5. Chapter 6 describes in detail the process of implementation of the ontology and the corresponding findings. The conclusion and the scope of future work are mentioned in Chapter 7. Chapter 8 lists the references.

Chapter 2

Prerequisite: on Ontology and Segmentation

In this section, we want to elaborate on various concepts necessary to understand before going into the depths of the proposed work. For a software system, something that exists can be represented. It generally means that the set of concepts that can be represented in the universe of discourse is described by the ontology. It provides conceptual description models that can be understood both by humans and computers. It describes a specific domain, by describing its related concepts and their relationships. It aims at bridging as well as integrating multiple and heterogeneous digital contents on a semantic level. This is exactly the key idea of the Semantic web.

2.1 Semantic Web

The Semantic Web extends the World Wide Web through standards set by the World Wide Web Consortium (W3C). [4] Its vision is to move from a web of documents to a web of data connecting a large amount of data available to everyone. The main idea behind semantic web is to publish different data on the web so that different organizations can make use of the data and query them to get the information and thus linking the data. By creating ontology, we are publishing the data on the web for everyone to use.

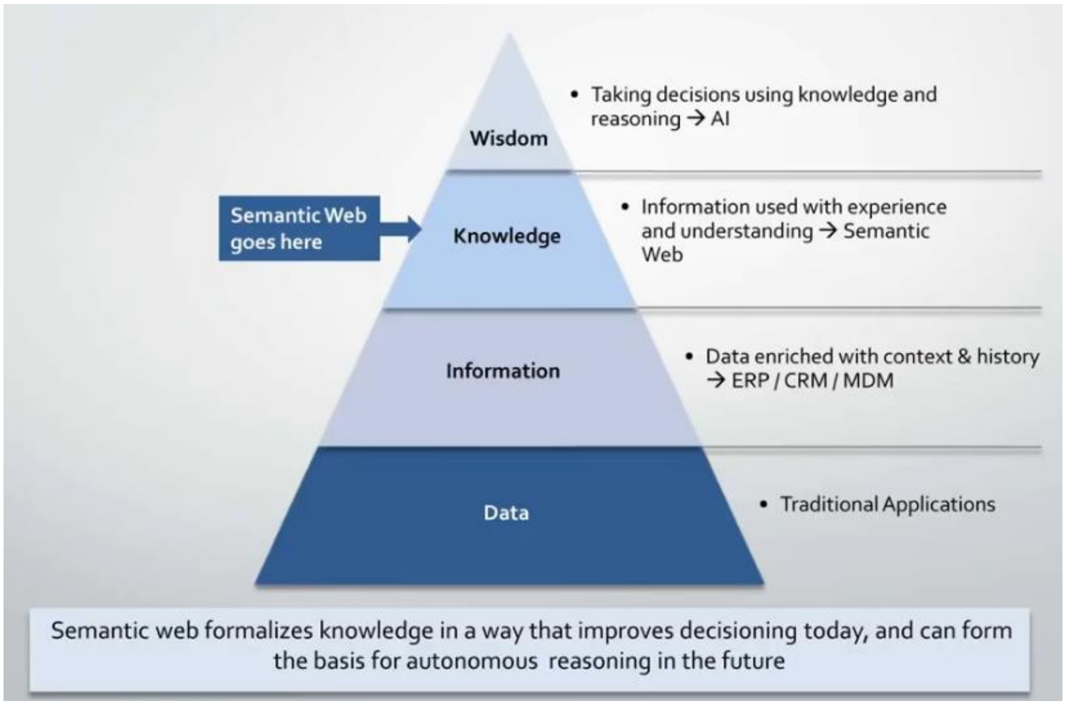

Wisdom
Taking decisions using knowledge and reasoning → AI
Semantic Web goes here
Knowledge
Information used with experience and understanding → Semantic Web
Information
Data enriched with context & history → ERP / CRM / MDM
Data
Traditional Applications
Semantic web formalizes knowledge in a way that improves decisioning today, and can form the basis for autonomous reasoning in the future

Fig. 1 The Semantic Web

Few major parts of the Semantic Web comprise RDF, RDFS, and OWL. The foundation of vocabularies and effective communication on the semantic web is Ontology. This information is given a well-defined meaning, better enabling computers and people to work in cooperation. The representation of the ontologies is made easier for machine and human understanding with the help of RDF, RDFS, and OWL.

2.2 RDF

RDF (Resource Description Framework) is a standard model for data representation and interchange on the Web. [5] It is to define data in the form of a sentence subject-predicate-object to be understood by the machines Each part of the sentence has a unique identifier later this identifier can be used to add more information about the things. It is used to represent everything that is to be uniquely identified and referenceable using URI like web pages, books, locations, or any other data.

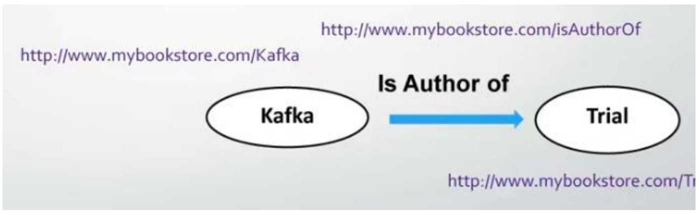

http://www.mybookstore.com/isAuthorOf
http://www.mybookstore.com/Kafka
Is Author of
Kafka
Trial
http://www.mybookstore.com/Tr

Fig. 2 RDF representation

In Fig. 2 Kafta is the author of the book “Trial”. Individual things and not just files are given unique identifiers. The below Fig. 3 shows a database of English books that is talking about a book which is called the glass palace and the glass palace is written by the author Amitabh Ghosh. Hear the book is an object. The title is the glass palace which is written in 2000 the publisher is someone who lives in the City of London and has the name HarperCollins. This data from the database record is exported as RDF in triple format. The book has a unique identifier.

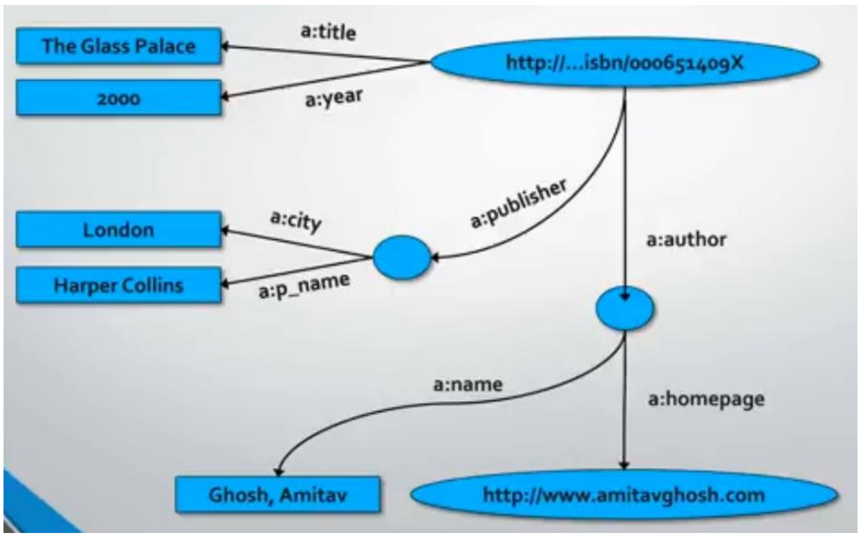


Fig. 3 Exporting data as RDF, English Book database

When having the same database but in French (Fig. 4), i.e the book has been translated to French. It has a translator, the author Amitav Ghosh, and the title of the book in French. The identifier (URI) for the book is the same for both English and French. So, these two can be very easily merged without having to move all the information from one database to another. Turtle is the Terse RDF Triple Language which is a concrete syntax for RDF. A Turtle document is a textual representation of an RDF graph. This format is easy to understand and is more user-readable.

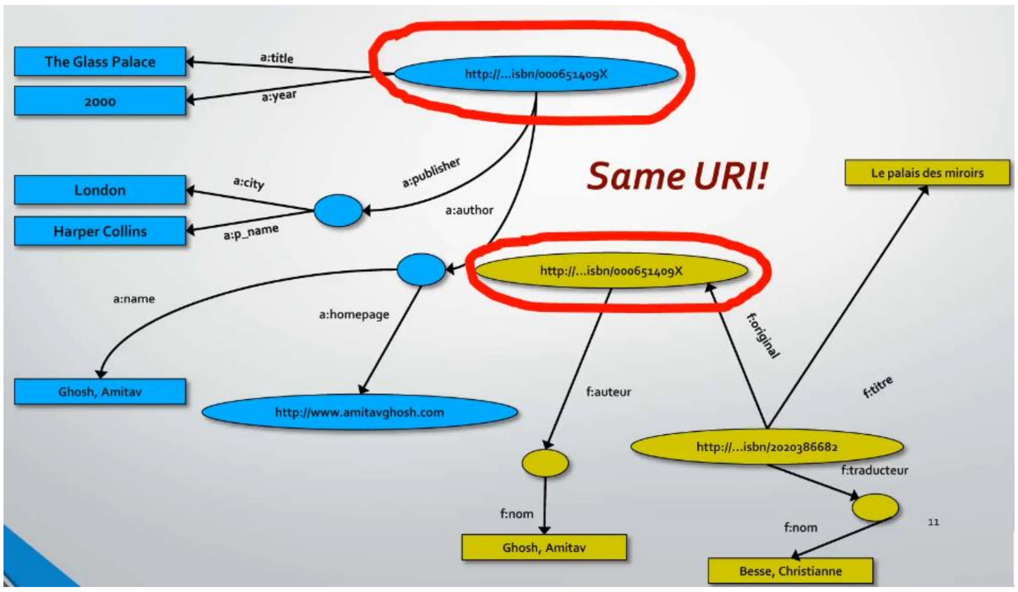


Fig. 4 Exporting data as RDF, for both the English Book database and its French translation

Both the French and English databases have the author in common. So, when talking about the author we can see their name, their homepage, their French name, and everything that is related to them.

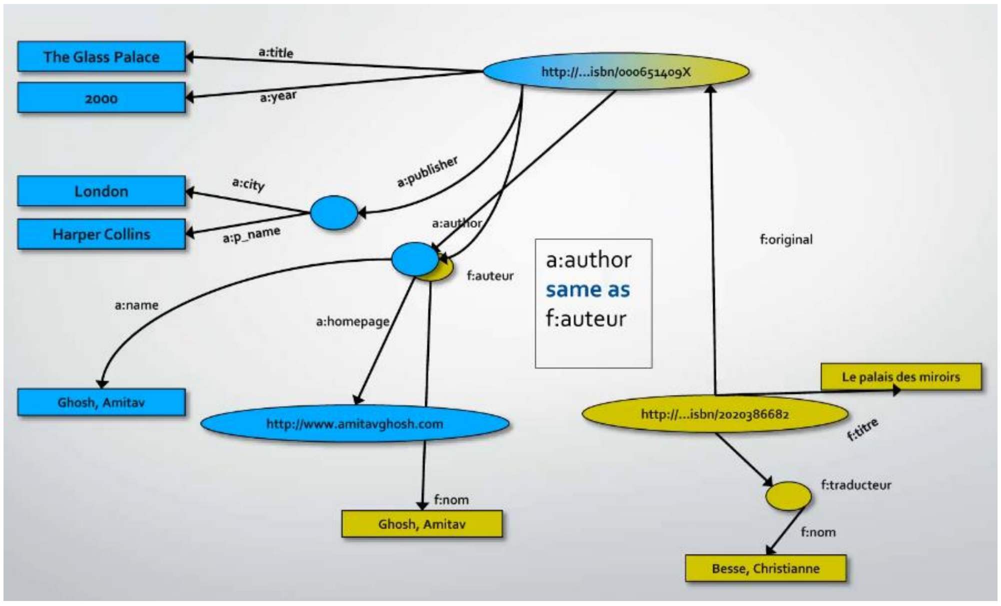


Fig. 5 RDF of the merged English Book and French Translation database

RDF Schema (RDFS) is an extension of RDF. RDFS is a general purpose language that is used for representing RDF vocabularies and data models on the web. It describes the resources with classes, properties, and values. RDFS provides a framework to describe application-specific classes and properties. RDFS is *object oriented* in its nature. It is about describing classes of objects. Turtle is the Terse RDF Triple Language which is a concrete syntax for RDF. A Turtle document is a textual representation of an RDF graph. This format is easy to understand and is more user user-readable.

2.3 OWL

OWL (web ontology language) [6] is a family of knowledge representation languages for authoring ontologies there are characterized by formal semantics. They are built upon the W3 C XML standard for the objects called RDF. The meaning of information (semantics) is made explicit by formal (structured) and standardized knowledge representation (ontologies). It is a defacto standard for ontology development based on RDF that introduces the components like class, properties, individuals, and restrictions for better representation and understanding. It also introduces the ability to automate reasoning and inference. Ontology editors are tools used for editing, developing, or modifying ontologies. They provide support to the ontological development process, as well as conceptualizing the ontology. The Web ontology languages are supported by editors such as Protégé[24], OWL-P, and OilEd.

2.4 Protégé: Ontology Creation

The Protégé meta-tool [7] built by Mark Musen in 1987 and developed by a team at Stanford University is a free, open source ontology editor and a knowledge management system. This software is the most popular and widely used ontology editor in the world. Protégé is written in Java and works on a variety of operating systems including Mac OS X, Windows, Unix, and Linux. Protégé has a layered design, including a knowledge model layer, an ontology store layer, an application programming interface (API), and a graphical user interface (GUI). The Protégé knowledge model is an object-oriented model of ontologies that are kept in a persistence layer and accessible using the Protégé API. Both the protégé GUI application, which allows users to view ontologies, and application programs use the API. Plug-ins, which access ontologies through the Protégé API and with which the user can interact by plugging into the Protégé GUI, are used to add new functionality to Protégé. In Protégé, the user may choose and install plugins from a plugin library. Protégé can be downloaded from: https://protege.stanford.edu/. It also has the functionality tool to import data from excel files stored on the system and various plugins for visualization of the entities and relationships.

2.5 VOWL: Ontology Visualization

We used a tool named VOWL to visualize the complete ontology including all the classes, subclasses, and the relationships between them in a graphical format, where classes and subclasses are denoted by blue circles and the relationships with dotted and bold arrows. WebVOWL is a web application for interactive ontology visualization. It implements the

Visual Notation for OWL Ontologies (VOWL) [8] by displaying graphical representations of Web Ontology Language (OWL) elements in a force-directed graph layout that represents the ontology. Exploration of the ontology and customization of the visualization is possible due to interaction techniques. The VOWL visuals are created automatically from JSON files into which the ontologies need to be converted. Along with WebVOWL, a Java-based OWL2VOWL converter is included. WebVOWL can be accessed from: https://service.tib.eu/webvowl/

2.6 Segmentation approaches

2.6.1 Static Windowing Approach

I. Time Windows (TW)

It divides the data stream into time segments with a regular time interval. The advantage of this approach is its simplicity. But the disadvantage of this approach is the selection of the optimal duration of the time interval. If the size of the window is small, it may not contain any relevant information. If the window size is large, then the information may be related to several activities, and the prediction of activity will not be correct. [9][10]

II. Sensor Event Windows (SEW)

It divides the stream via a window into segments containing an equal number of sensor events. The size of the window in terms of several events, as for any type of window, is also a difficult parameter to determine. The disadvantage of this approach is If the window is too small, there will be a lack of information. However, if it is too large, it will be difficult to interpret and the prediction of activity will not be accurate. [11]

2.6.2 Dynamic Windowing (DW) Approach

DW uses a non-fixed window size, unlike the previous methods. Here window size is not initially fixed. Based on incoming events and activity predicted, the window size is determined. [12] These scenarios can be divided into two main categories: *overlapping* and *non-overlapping* time windows. In the case of overlapping time windows, one sensor data can be shared by two or more than two windows. In the non-overlapping time window case, sensor data of each time window are exclusive to itself.

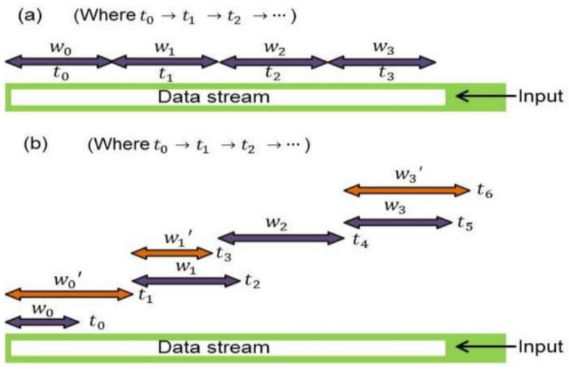


Fig. 6 Data segmentation scenarios. (a) static-fixed time window, (b) dynamic-variable time window.

Chapter 3

Scope of the Work

Center for Advanced Studies in Adaptive Systems (CASAS) smart home project includes the Aruba dataset which is based on the home of a volunteer adult. The resident in the house was a woman. The residence is equipped with different types of sensors. These sensors are deployed in different strategic locations of the house and get triggered when any motion is detected. The data generated from these triggered sensors produce a continuous flow of sensor data. To recognize the activities, the continuous flow of sensor data must be segmented. However, determining how this flow of data should be segmented, referred to as data segmentation, remains one of the most difficult challenges. The goal is to perform this segmentation and recognize the activities.

In our approach, we are creating an ontology for defining a set of representational primitives with which to model a domain of knowledge. But the challenge lies in creating a standardized version of ontology for getting global recognition and acceptance along with smooth interoperability. Keeping this in mind, we used Semantic Sensor Network (SSN) ontology for describing the sensors and their observation, the observed properties, the actuators, the studied feature of interest, and the procedures which are involved.

Protégé is a tool that is written in Java and it is a free and open-source program for creating, modifying, and managing ontologies. The SSN ontology is imported and extended in Protégé to create ADL_SSN ontology. The Aruba dataset and the house map are studied very carefully to obtain certain statistical information and knowledge about the dataset. The information along with the knowledge is loaded in the ADL_SSN ontology in the form of classes, sub-classes, and instances. The knowledge from the ontology works on the sensor data for performing successful segmentation and activity recognition. Based on this segmentation, we measure the performance of the proposed algorithm.

Chapter 4

Present State of Art: Segmentation

Nowadays, there are lots of activity-based segmentation approaches that exist for activity recognition. But that approaches can be broadly classified into two categories. Those approaches are the data-driven approach and the knowledge-driven approach. In data-driven approaches, a model is trained by the pre-recorded dataset and a Machine Learning technique is used. Data-driven approaches suffer from cold start issues for processing pre-recorded datasets.

In knowledge-driven approaches, domain experts conceptualize the factual elements and interlinked the relation among them. After that, an ontological model is prepared where different entities take into different classes and the same types of objects are assigned to that class. The classes are interlinked through their relations. That classes and relations are the formal and logical theory of well-defined knowledge. Both the approaches have a problem that is it assumes the complete description of all the objects and concepts within the activity model.

4.1 Data Driven Approaches

Real-time data segmentation using Jaro-Winkler distance measurement uses Jaro-Winkler distance (JWD) as a string edit distance indicating that calculates the similarity between two strings. The higher JWD the for two strings is, the more similar the two strings are [13]. JWD is a variant of the Jaro distance which is depicted by eq. (1):

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \qquad (1)$$

Here, |s1| = length of string 1.

|s2| = length of string 2.

m = identical characters

t = half the number of transpositions

If two characters of s2 and s1 are identical and the distance between them is less than ⌊max(|s1|,|s2|)/2⌋-1, they are considered matching. Finally, JWD, is computed as follows:

$$d_w = d_j + \left(\text{lp}\left(1 - d_j\right)\right) \qquad (2)$$

In the above formula eq. (2), dj is the Jaro distance, l is the length of common prefix having a value up to 4, and p is the given scaling factor of l. The longer a common prefix is, the higher score of dw can be acquired in proportion to p. Also, p is typically set to 0.1 and cannot exceed 0.25, while 4 is the maximum value of l to keep the value of dw between 0 and 1. They proposed an automatic segmentation methodology for the real-time activity prediction process. This approach can dynamically capture the endpoint of the predicted activity in the current data stream by using the Jaro-Winkler distance (JWD) measurement, and store the segment into the training dataset after tagging it, which can be referred to as the automatic segmentation.

The main contribution of this work is providing generalized methodologies of manipulation of windows in real-time using temporary information and automatic data segmentation and labeled training data collection while a predicted activity is determined by machine learning algorithms. The approach is based on the hypothesis that a breakpoint of a data stream has the peak value of JWD between the window and the training dataset. This approach manipulates non-overlapped, variable, and dynamic time windows using temporary information such as JWD obtained from comparing in real-time the predicted activity's data with the current data stream to support continuous, real-time activity prediction, and the usage of constant parameters is minimized to make this approach more generalized.

The Approach for Automatic Data Segmentation in Real-time in Fig. 1 shows the Architecture of real-time activity prediction and data segmentation. The Event Window continuously receives a sequence of events. The Event Window is analyzed by the Activity Inference Engine. Once an activity is predicted with a probability over a certain threshold by the Activity Inference Engine, the JWD Operator starts comparing the training data of the predicted activity stored in the Training Dataset with the current event window to segment the data stream. The event window grows one event at a time, while the JWD Operator keeps computing JWD and searches a certain length of the window that maximizes the JWD via the JWD Monitoring component. Deciding the optimal length of the current event window and finding the turning point of the JWD value is not enough since a local optimum problem can occur. To discover a true peak value and perform segmentation correctly, we need to ignore the temporary increasing or decreasing state. If a turning point is detected, further JWD computation for the following n/4 data is executed, where n is the average length of the corresponding activity's instances in the training dataset. If the turning point's JWD is the maximum among all the results from the further test, the point is determined to be a breakpoint. If not, the other point which has the maximum JWD value becomes the new candidate breakpoint and further JWD computation is executed again.

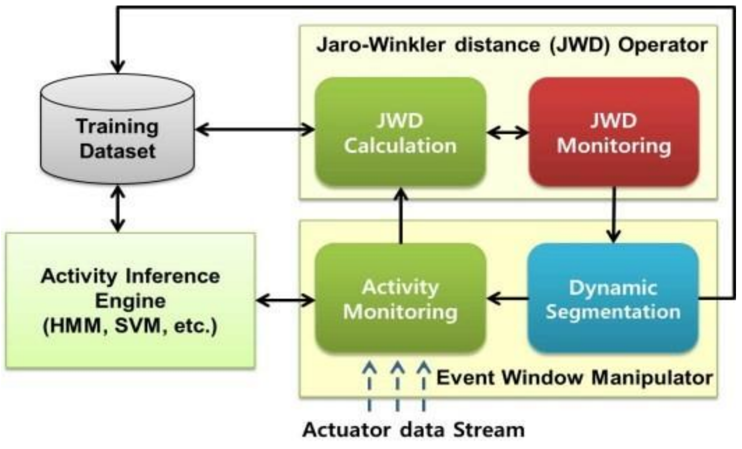


Fig. 1: Architecture of real-time activity prediction and data segmentation.

The Dynamic real-time segmentation and recognition of activities using a multi-feature windowing approach [14] proposed a framework that is divided into two phases: offline modeling and online recognition. In the off-line phase, a representation called Activity Features (AFs) is built from statistical information about the activities from annotated sensory data, and a Naive Bayesian (NB) classifier is modeled accordingly.

The activity Features include Mutual Information (MI), MI is to calculate the chance of two sensors to occur consecutively in a stream of data [10]. It reduces the influence of sensor events from very different functional areas on the feature vector defining the last sensor event within a window. Frequency of Activated Sensors for an Activity (FreSen) calculates the frequency of occurrence of each sensor in a series of N sensor events [16]. Last Two-State Sensor Method. (L2S), is an extra feature that has been proposed to assist other activity features.[14] This feature checks whether the last two sensors of an activity A_i and the streaming event are the same or not. Activity Time Interval. (Int) the feature is to calculate the time interval of an activity A_i where the formulas are used to find the mean and standard deviation for the condition that if the time interval for A_i is between mean and mean plus twice the standard deviation then the activity is recognized and can be segmented.

A dynamic multi-feature windowing strategy employing AFs and the learned NB classifier is used in the online phase to segment unlabelled sensor data and predict associated activity, as shown in Fig. 2.

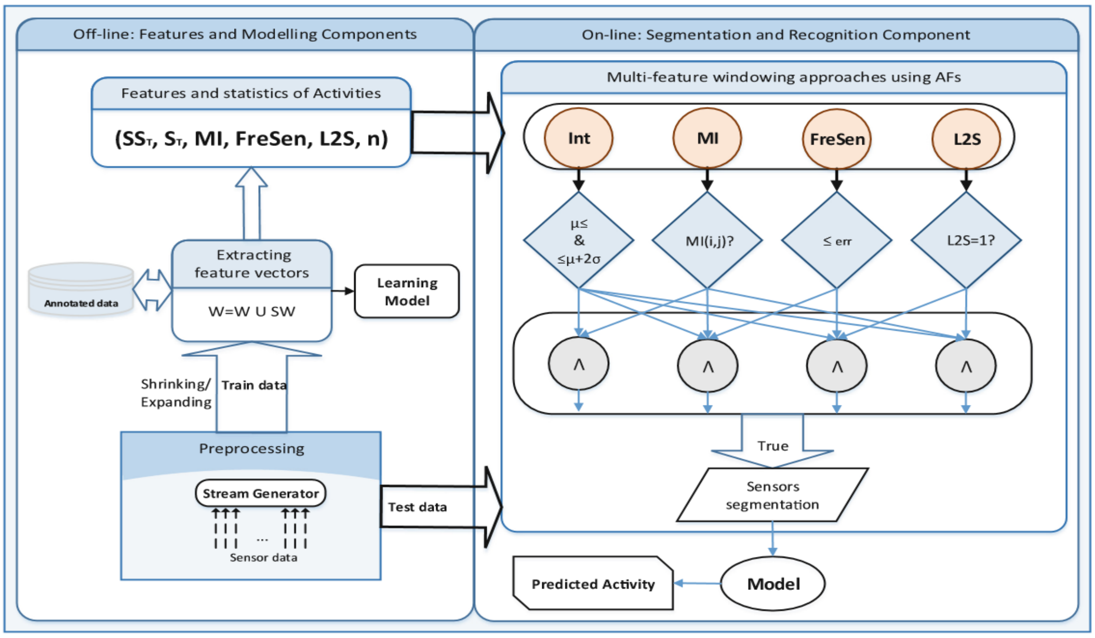

Off-line: Features and Modelling Components
On-line: Segmentation and Recognition Component
Features and statistics of Activities
(SS_T, S_T, MI, FreSen, L2S, n)
Multi-feature windowing approaches using AFs
Int
MI
FreSen
L2S
μ≤ & ≤μ+2σ
MI(i,j)?
≤ err
L2S=1?
Extracting feature vectors
W=W U SW
Learning Model
Annotated data
Shrinking/ Expanding
Train data
Preprocessing
Stream Generator
Sensor data
Test data
True
Sensors segmentation
Model
Predicted Activity

Fig. 2: Framework of activity modeling and recognition.

Preprocessing and modeling components are conducted in the offline phase. In this phase, we applied an adaptive windowing model to read the sensor data based on annotated labels. An adaptive windowing method for streaming sensor data uses three elements: window length adaptation (shrinking and/or expanding the window), a time decay function, and a scheme for accommodating past sensor information. A Sensor Window, *SW*, is stored with Sensor data, *S*, at the time, *t* ($SW \leftarrow SW \cup St$). If the length of *SW* is less than or equal to an initial size then the activity is recognized and *SW* is added to the window data matrix, *W*. Otherwise, if Size(*SW*) is exhausted, the initial size will be expanded by a predefined extension value (*ext*), but this is

expanded once only. During reading the sensors if an activity is not recognized and Size (*SW*) is exhausted after the expansion, then the sensor data is added to the Past Sensor Information pool that is used to store the useful information for the next window. The details of this method are elaborated in [17]. This phase is where the NB classifier is built from the annotated sensory data. In this paper, their main focus and a key challenge are addressing the online recognition phase. A dynamic multi-feature windowing approach using AFs in this phase is introduced to segment the unlabeled sensory data in a real-time setting.
Each Activity Feature (AF) alone is capable of contributing to classifying sensor data. However, to achieve more accurate segmentation as well as recognition, a multi-feature windowing approach is proposed which is derived from AFs online to dynamically segment unlabeled streamed data. For the windowing purposes, we considered using multi-feature of AFs to segment sensor data precisely which is described in Table 1.

Table 1. Notation of dynamic multi-feature windowing approaches for segmentation.

Notation	**Description**
MI_Int	Combining AFs of MI and Int
MI_FreSen_Int	Combining AFs of MI, FreSen, and Int
F reSen_L2S _ Int	Combining AFs of FreSen, L2S, and Int
MI L2S Int	Combining AFs of MI, L2S, and Int

Table 2. Notation of Activity Features (AFs)

Symbol	**Description**
MI	Mutual Information [4]
FreSen[20]	Frequency of activated sensors in an activity
L2S	Last two (2) sensors of an activity
Int	Time interval of an activity

Table 3. The satisfying criterion of Activity Features (AFs)

Symbol	**Check Conditions**
MI	MI(i,j)=1
FreSen[14]	*Ai*, should not exceed the error threshold, *T*, which is set at 0.05
L2S	*L2S* = 1
Int	μ <=Int< (μ + 2σ), where μ is the mean and σ is the standard deviation.

Multi-feature windowing approaches combine more features to segment the sensor data in an online fashion. Indeed, multi-feature methods are appealing mainly because they can improve on a single feature which can make more accurate recognition. On the other hand, in a smart home test bed, different activities trigger a similar set of sensors which causes overlapping activities. Nevertheless, this proposed dynamic multi-feature method can distinguish these overlapping activities. When sensors are triggered, our multi-feature method reads unlabelled sensor data and segments them based on the generated AFs.

- MI stores the Prior probability $P^i(S_{j-1}, S_j)$ of sequence sensors for activity A_i;
- FreSen sorts the prior probability $((P^i_S)$of sensors for an activity where A_i where i = 1, ..., m in descending order (list$[P^i_S]$);
- L2Si contains the list$\{S_{j-1}, S_j\}^i$ where j = 1, . . ., N;
- Int stores the mean and standard deviation of the time interval for each activity;
- Wi contains S_1Initial window of activities with the first activated sensor;
- Triggered sensor (say S_j)a is added to all available activity windows $W_{1\dots m}$
- The condition of features is checked with the current sensor data S_j for each activity $A_i$using a CheckConditions (Activity Features, S_j) function
- If the condition is satisfied with W_i for activity A_i, W_i is retained and the incoming data is stored in W_i
- All other windows that did not match the conditions are discarded and recreated with the incoming data.
- If all the conditions of activity A_i satisfy the sensor data in W_i at this point the W_i is segmented and W_i is stored for feature extraction for future recognition.

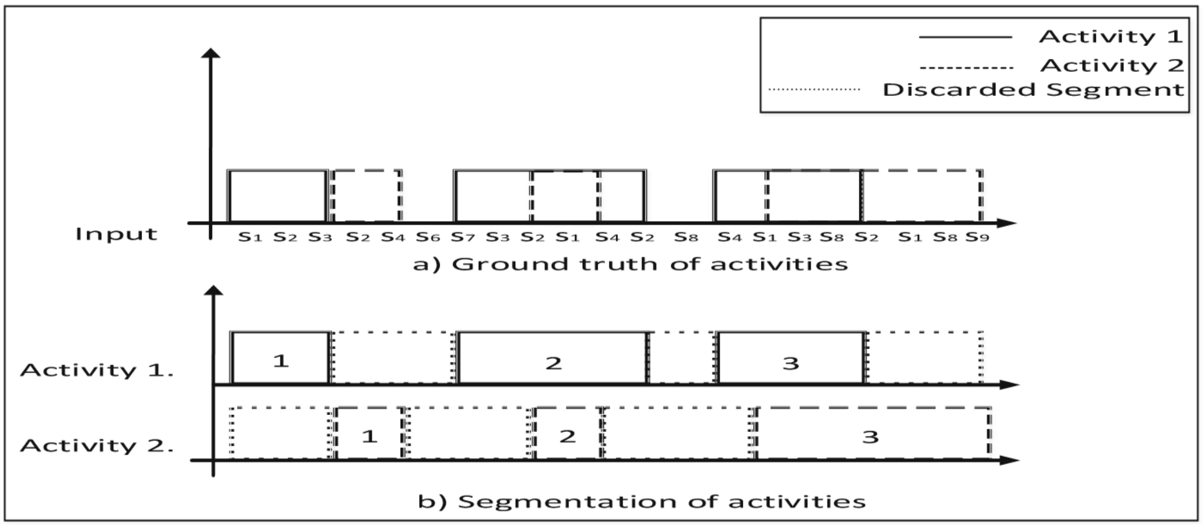

Activity 1
Activity 2
Discarded Segment
Input
S1 S2 S3 S2 S4 S6 S7 S3 S2 S1 S4 S2 S8 S4 S1 S3 S8 S2 S1 S8 S9
a) Ground truth of activities
Activity 1.
1
2
3
Activity 2.
1
2
3
b) Segmentation of activities

Fig. 3: An example of segmenting activities using a dynamic multi-feature method using AFs**.**

Fig. 3, shows how activities, particularly overlapping activities, are segmented. In Fig. 3, sensors *S*1, *S*2 and *S*3 are added to segment 1 of both *Activity 1* and *Activity 2*. However, *Activity 2* did not satisfy the conditions and has been discarded. Right after, a new window of *Activity 2* is created to read incoming activated sensors *S*2 and *S*4. The same procedure is performed for Segment 2 of *Activity 2* which is overlapping with Segment 2 of *Activity 1*.

4.2 Knowledge Driven Approach

One of the articles proposes an ontological model focusing on general knowledge and personal choice of the residents to conduct ADL to support the segmentation. A multi-threaded algorithm is developed to take its own decision. [18] A prototype is constructed. The idea is to establish a relationship between the daily used object and known ADLs (Activity Daily Living). Thus, we can also identify the action of a composite activity and we can arrange it separately to classify the activity. To do that, an ontology has been developed that conceptualizes and captures the context of the environment (i.e., environmental attributes, daily used objects, location, sensors), the general and specific preferences of residents for performing ADL, and their semantic relations. The stream of data is segmented using a semantic decision mechanism. That decision engine takes three inputs: currently observed sensor events, the ontological model, and a set of segmented sensors for a given activity.

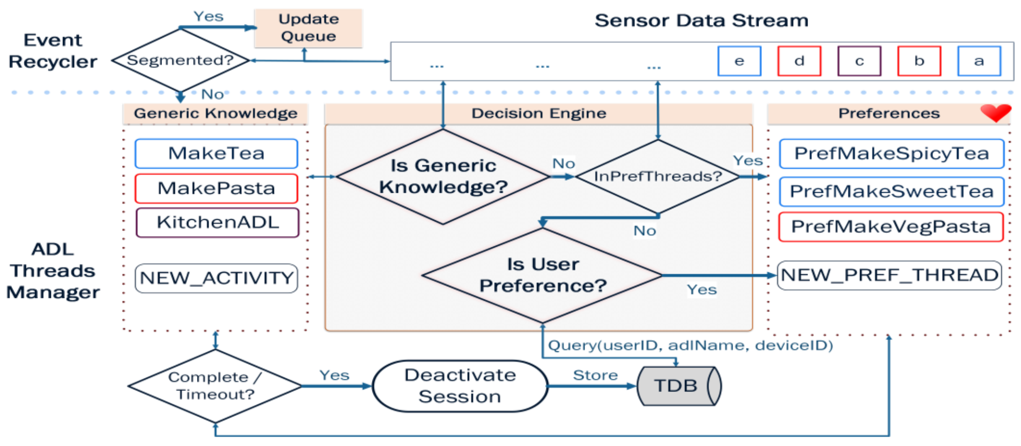


Fig. 4: Overview of the semantically enabled segmentation approach with generic (T-box) and preferences (A-box) KB for reasoning.

As shown in Fig.4 when sensor events come as a data stream, multiple ADL threads, generic and preference, probe the sensor events using a decision engine and keep the relevant events in the same group. So, there may be a case that sensor events are shared among the multiple ADLs. If this happens then the upcoming sensor event is checked along with the previous set of sensors and then determine if it complies with the general ADLs or not. If it is matched then the event handler updates the data stream. Else, a new ADL thread is created. This process is performed by semantic reasoning and invoking queries to the TDB (Triple store Database) for personalized actions. If the upcoming sensor events or set of events does not match with any existing ADLs thread then a new activity thread is created. If the upcoming events are matched with any personalized actions and no active personalized thread is there, then a new personalized thread is created.

ADL modeling takes place in three phases: (1) environmental context modeling, (2) semantical relationships modeling, and (3) personalized object interactions. In environmental context modeling, different environmental entities are introduced. Like a person, rooms and ambient characteristics, sensor characteristics, and everyday fixed/portable objects. Every day fixed portable objects are kettles, mugs, etc.

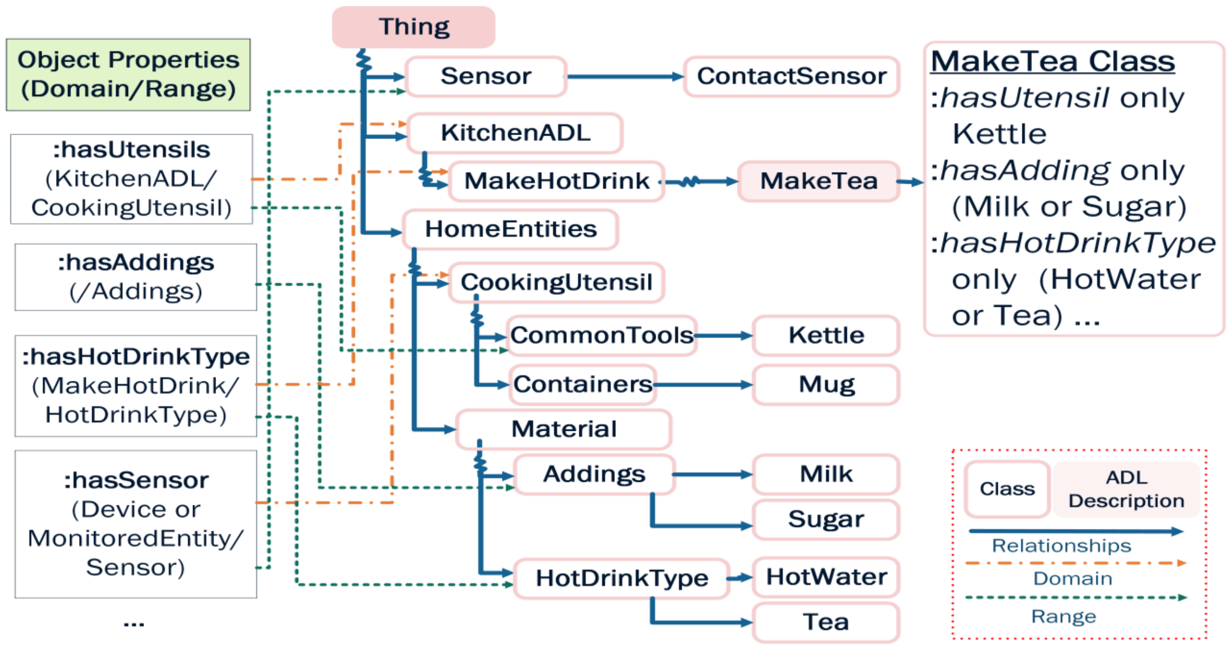


Fig. 5 Semantical relationship properties between everyday objects, set of actions for MakeTea ADL, and sensor characteristics. [18]

Different semantic relationship properties are mentioned between Environmental context classes and ADLs. For example, milk is associated with MakeTea class so there exists a relation ‘hasAdding only’ as shown in Fig. 5. Along with that, which sensor is attached to which object is also mentioned. Then the relationship is established between the upcoming sensor event i.e. observation class along with the primitive sensor reading. The relationship between the sensor and instances of the observation class is mentioned. Which person is associated with which ADLs and their preferences are also kept in a note as a relationship. Once this relationship is introduced properly, the decision engine is now capable of performing reasoning. The common ADLs are automatically identified by T-Box reasoning whereas the preferences i.e. inhabitant specific actions are identified by A-box reasoning. In this way, activity recognition and segmentation perform.

For modeling composite activity and its recognition, it requires three basic components, namely activity ontology models, temporal activity models, and association rules; each element models a specific portion of the composite activity.

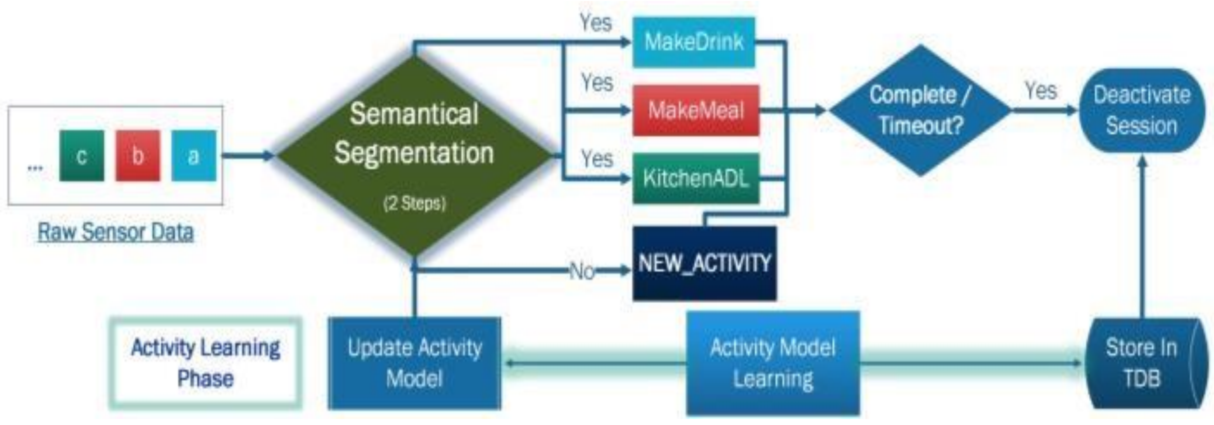


Fig. 6 Overview of Semantical Segmentation Approach.

When a sensor event comes, it is added to the queue and multiple activity threads start analyzing it. If it gets any kind of similarity then relevant events are stored separately.

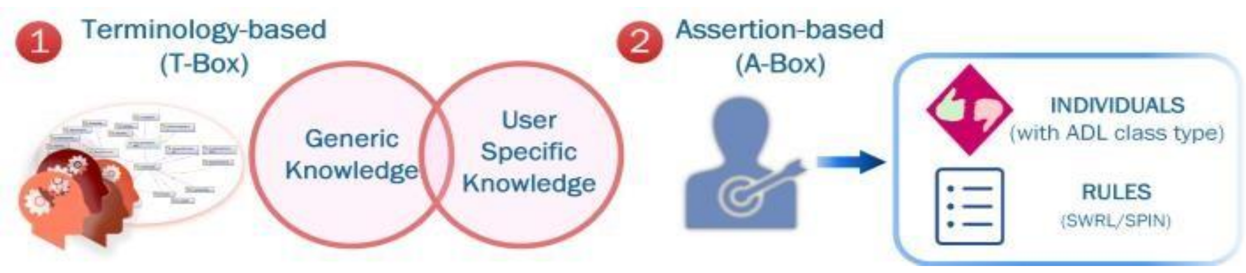

1 Terminology-based (T-Box)
Generic Knowledge
User Specific Knowledge
2 Assertion-based (A-Box)
INDIVIDUALS (with ADL class type)
RULES (SWRL/SPIN)

Fig. 7 Twofold segmentation process: (1) generic (T-box) and (2) preference (A-box) reasoning. [21]

Initially, there may be the case that one event can be shared independently by two distinct activity threads with different ADL motives. So, to differentiate properly the activity thread performs generic knowledge (T-box) reasoning after that the inhabitant's preferences (A-box) reasoning takes place to check for the relationship between the sensor event and the current ADL class of the thread as shown in Fig. 7.

The semantic web framework provides web ontology language OWL to express the complex knowledge into classes, relationships (object & data properties), and data (individuals) [19]. SPARQL Inferencing Notation (SPIN) rules [20] can be used to perform queries on the data stored in the triplestore containing the inhabitant's preferences. If a sensor event is not matched with any of the ADLs then new activity threads are created. To support inhabitant specific preferences, individuals are created. These individuals are associated with the resident and with this ADL class, which has a list of sensors attached to objects and other attributes. Generic and inhabitant-specific knowledge can be created and linked using classes, individuals, and rules. SWRL rules can also be added as a string in the individual, however, this may require an independent rule engine to be executed. Alternatively, SPARQL Inferencing Notation (SPIN) rules can be used to perform queries on the data stored in the triplestore containing the inhabitant's preferences. Identifying inhabitant-specific and generic ADL ontology descriptions along with SWRL rules can be achieved by using the OWL API and Jena API to create and manipulate the model once generic and inhabitant-specific models are combined and rules are loaded into the virtual memory. Pellet reasoner and JESS rule engine are used for reasoning [21].

In another study, they are mainly working on composite activity modeling and recognition using ontological and temporal knowledge. To do that, they use a hybrid approach for composite activity modeling and recognition. They develop generic activity models for composite activity. That models have three main components. One is ontological activity models another is temporal activity models and entailment rules. The generic model is proposed for modeling composite activities in different applications. In this paper, they create activity models for ADLs in the context of smart homes which can be reused for testing and evaluation. Then develop a system architecture for composite activity recognition and a single activity recognition algorithm. Both are integrated into one. That algorithm can run over data streams against composite activity models to perform incremental real-time activity recognition. Along with that, they have introduced a system prototype for experiments for testing and evaluation.

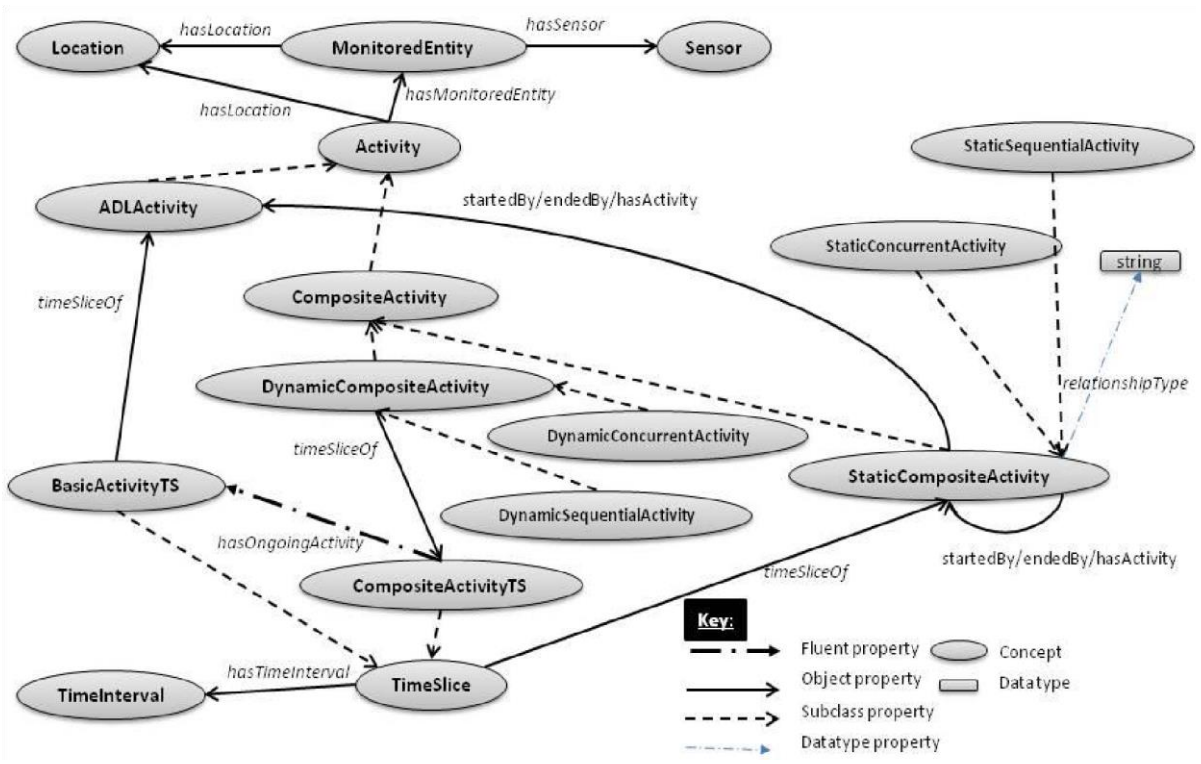


Fig. 8 Part of the activity models showing concepts and their inter-relationships activities [22].

Activities are categorized into actions, simple activities, and composite activities as shown in Fig. 8. An action is a building block of activity. A set of actions constitute simple activity. A composite activity can be defined as the collection of two or more simple activities in a certain time interval. Composite activities can be further categorized as sequential or multi-task activities. Sequential activities are those which occur sequentially. Activities are characterized by temporal information like repetitive time patterns, time sequences, time duration, and time instants or intervals. Representing temporal knowledge in OWL can be done using the 4D-fluents approach. It has two building blocks time slices and fluents. It provides a vocabulary for representing dynamic temporal parts of an individual. The time slices are used for representing the temporal parts of a particular entity at particular moments. Activity recognition takes place by following these steps. The observations are matched and tried to represent predefined actions which can be done by checking different property restrictions which are defined in the ADL Ontology by ontological reasoning and then grouping those actions into simple activities defined in an ontology. A general activity has general descriptions otherwise activity has some specific descriptions. To identify more general activity a large number of sensor data and context information are required to identify the correct sub-activity. New activity descriptions will be created as sensor data is obtained. Sometimes, the existing ones are modified. In this process, some of the partial activity descriptions become complete. Although, all the existing partial or complete activity descriptions will remain valid until sensor observations in a given segment are rejected. [22]

Some of the recent studies use the SSN ontology for expressing their ontological model. That ontological model is a hydrological sensor web ontology based on SSN ontology which is introduced to describe the resources of heterogeneous hydrological sensor web. This is brought in by the time and space ontology, creating the hydrological classes, and defining reasoning rules. In this study, the hydrological sensor web ontology is described conceptually and then the reasoning rules are defined for the identification of the flood stages. After that, an ontology is introduced with specific platforms, sensors, observations, and water bodies. This ontology is created to carry out three basic functions: (1) to create the links between the main concepts in the three reference ontologies, i.e. of GeoSPARQL, W3C Time ontology, SSN ontology (2) to extend the ontology with hydrological classes and to instantiate that ontology with hydrological sensor web information, and (3) to establish the rules for the reasoning status of the hydrological event. OGC GeoSPARQL core classes i.e. SpatialObject, Feature, and Geometry were used to describe spatial information. The object properties of GeoSPARQL i.e. covers crosses, meets, and within. These are used to determine topological relations between the request area and the observation area. Apart from that, the class Event was created and has subclasses flood, rainstorm, etc. for different kinds of disaster events. Time ontology was applied as the time property and used for searching for suitable sensor web resources within a required time. There are three classes i.e. TemporalEntity, interval, and instant.

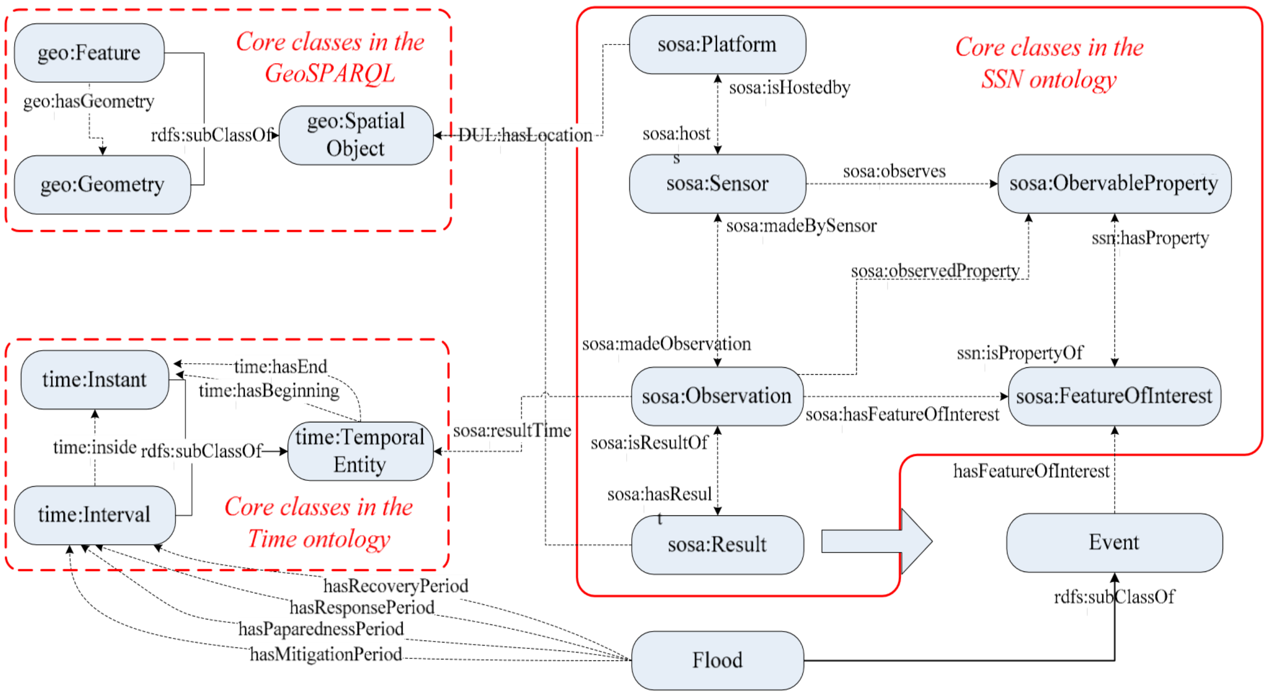


Fig. 9 The core classes and properties in the hydrological/flood ontology are based on Semantic Sensor Network (SSN) [23].

Chapter 5

A Segmentation Algorithm based on Ontology

This chapter gives a detailed explanation of our approach to segmenting the sensor data. The ontology allows us to define the domain and the relationship between them. SSN Ontology provides flexible and coherent perspectives representing the entities, the relationships between them, and the activities involved in sensing. In our work, we have used Aruba dataset from the CASAS datasets for our findings. The Centre for Advanced Studies in Adaptive Systems at Washington State University provides the research needs for testing technologies using real data through the use of a smart home environment. The Aruba data set records the activities of a single person through various sensors.

5.1 Dataset: Aruba

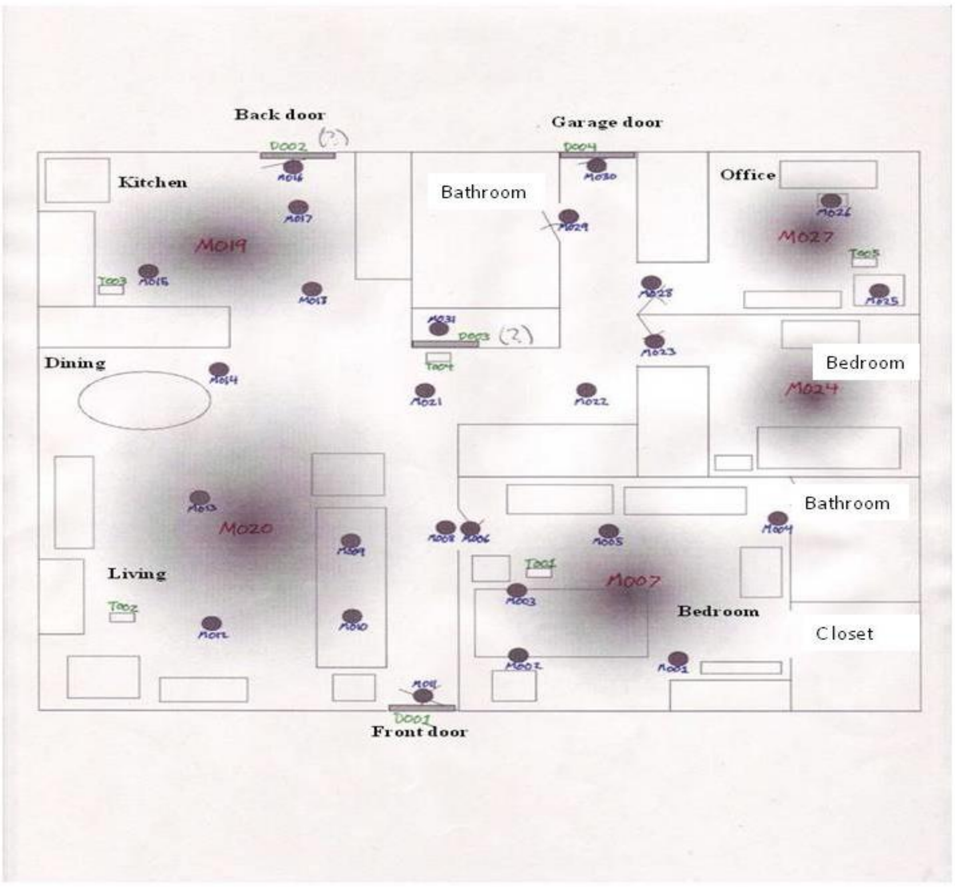


Fig. 1 The Layout of the sensors in the ARUBA dataset.

The sensor events are generated from motion sensors (these sensor IDs begin with "M"), door closure sensors (these sensor IDs begin with "D"), and temperature sensors (these sensor IDs begin with "T"). [25]

The data format for the data set is as follows:

Date	Time	Sensor	Trigger	Activity	Annotation

'Date' and 'Time' refer to the date and the time on which the sensor was triggered. 'Sensor' field denotes the name of the sensor which triggered at the corresponding time of the day. In our work, we focused on ambient sensors which have two possible values ON or OFF denoted by the 'Trigger' field. 'Activity' denotes the corresponding activity occurring at the time of the day. The start and end of the activity mark in the field 'Annotation'. The Aruba Dataset is a 9-month data that comprises 10 distinct activities namely Sleeping, Bed_to_Toilet, Meal_Preparation, Relax, Housekeeping, Eating, Wash_Dishes, Leave_Home, Enter_Home, and Work. We have split the dataset into 4:5 ratios and have used the data of the first four months as our training data and the rest five months as our test data.

5.2 Process Description

Initially, we preprocessed the dataset to obtain more information about the given data. We calculated the duration of each occurrence of an activity, the number of times each of the Motion Sensors was triggered during the course of any activity, phase of the day during which that particular occurrence of the activity is in action (i.e. whether it is Morning, evening or late night, etc.), the next activity that would follow the current one, the previous activity just before the current activity and the last field.

We further preprocessed the dataset to calculate various features of each occurring activity that would help in better segmenting the sensor events. The features used include Frequency of sensors, Average duration of Activities, Previous Activity, Shift, and Location of Activity. The extracted features are stored in the proposed ADL_SSN ontology as domain knowledge under the observable property class that is used to store various properties of the data. The various features are described below:

Frequency of sensors

This feature is to find out the frequency of the sensors in each occurrence of an activity. This would help us to find out the sensors that are highly probable to occur in a sequence of sensor events of an Activity. Frequency is calculated as the probability of the frequency of occurrence of sensor S_i in the space of N sensor events for each instance of Activity A_j and then finding the mean probability of the sensor for every instance of the activity in the entire training data set.

$$FreSen^i(k) = \frac{1}{N}\sum_{k=1}^{N}\delta_f(S_k) \qquad (1)$$

$$\delta_f(S_k) = \begin{cases} 1 & \text{if } \{S_k\} \in I_i \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

$$Mean\ (S_k) = \frac{1}{M} \sum_{i=1}^{M} FreSen^i(k) \qquad (3)$$

Where N is the number of sensors for each instance I_i of an activity A_j. $\delta_f(S_k)$ takes value 1 if the sensor S_k is activated in instance I_i and otherwise 0. M is the total number of instances of each Activity A_j. The mean probability of each sensor of Activity A_j is the sum of the frequency of sensors in each instance of the Activity divided by M. The sensors with a high probability of occurrence for an Activity are accepted. Here we consider sensors with top k highest probability where k comprises the sensors with probability greater than 0.04.

Average Duration

This feature as by its name is used to find the average duration of occurrence of each activity. This feature helps to distinguish between various activities based on the duration for which an activity may occur. Average Duration is calculated by first calculating the duration of each instance I_i of an activity A_j and then finding the mean duration for each activity.

$$Duration^j(i) = \delta_d(I_i) \qquad (4)$$

$$\delta_d(I_i) = \text{End Time} - \text{Start Time} \qquad (5)$$

$$Mean\ \left(A_j\right) = \frac{1}{M} \sum_{i=1}^{M} Duration^j(i) \qquad (6)$$

Here the duration of each instance $\delta_d(I_i)$ is the difference between the start time of the instance and the end time of that instance. Mean duration is the sum of all the durations of an Activity divided by M the total number of instances of that activity.

PreActivity

This is a special feature to find out the activities that are probable to occur before an activity. The PreActivity feature is used to calculate the probability values of the occurrence of activity before the respective activities. This is done by finding the previous activity of each instance I_i of an activity A_j. Then finding the probability of each previous activity of an activity A_j with sample space as the total number of instances of an Activity. The preActivities with a high probability of occurrence for an Activity are accepted. Here we consider the top k highest probability where k comprises of the preActivity with the probability greater than 0.04.

Shift

This is another special feature that finds out the period of a day when the activity usually occurs. The Shift class stores the probability values of the shift of the day when each activity occurs. The time of a day is initially divided into various time frames (Morning, night, early morning, noon, etc). The starting time of each instance is compared with the time frames and given a particular time frame. Then calculating the probability of each Shift of occurrence of an activity A_j with sample space as the total number of instances of an Activity, considering top k highest probability where k comprises of the Shift with probability greater than 0.04.

Location of Activity

This feature is used to store information about the frequent location of occurrence of each activity which is an efficient feature that helps in distinguishing among various Activities by monitoring the change in the location, especially for an environment with a single individual where most of the activities are performed independently at separate locations with few activities with the same location of occurrence.

5.3 Algorithm

We are testing our model using six-month data from the ARUBA dataset that includes eleven activities, and sensors including thirty-one motion sensors, and three door sensors. The knowledge about the data is loaded in our ontology including the distinct activities in Feature_of_Interest class, sensors in the Sensor class, deployment location of the sensors in the Deployment class, and properties that will help in segmentation. We take individual sensor events as input and check if the conditions for the features match with the sensor. When either of the conditions fails to match a sensor, we check if the sensor is falsely triggered, if not we consider it the start of new activity and segment it.

Algorithm to segment incoming sensor data:

```
STEP 1: Take each sensor event ($S_i$) one by one from the dataset
STEP 2: Check whether the present sensor event, $S_i$ is Start sensor of any Activity or not, using
        the ADL_SSN ontology
STEP 3
        a: From the start sensor, all the associated activities are determined from the
           ADL_SSN ontology by the relationship hasStartSensor from class
           FeatureOfInterest: class Sensor
        b: Now we check all the Features (preActivity, Frequency_of_sensors,
           Average_Duration) corresponding to the predicted activities (from STEP 3a)
STEP 4: Determine the location of the start sensor and based on the location predict the activity
        using the relationship deployedSystem from class Deployment: class Sensor
```

STEP 5: Now take sensor event S_{i+1}, and repeat STEP 4

STEP 6

a: Check if presently determined activity based on the current sensor, matches with the Activity that is determined based on the start sensor (STEP 2).

b: Check whether the current sensor event is present within the Feature list (Frequency_of_Sensor) of the predicted Activities

STEP 7

a: If the condition in STEP 6a and STEP 6b satisfy then current sensor is appended to the list where the start sensor event is present.

b: If the condition in STEP 6a and STEP 6b does not satisfy then a new segment is started and repeat from STEP 2 to STEP 6

Chapter 6

Experimentation

This section is to discuss in detail the process of implementation of our scope of work. This section mentions in detail how we created the ontology with the help of the editor and implemented it in python with the help of a special Python Library and then the result and findings of our work. We have used Protégé to implement and extend the SSN ontology.

6.1 Ontology creation using Protégé

6.1.1 Class Description

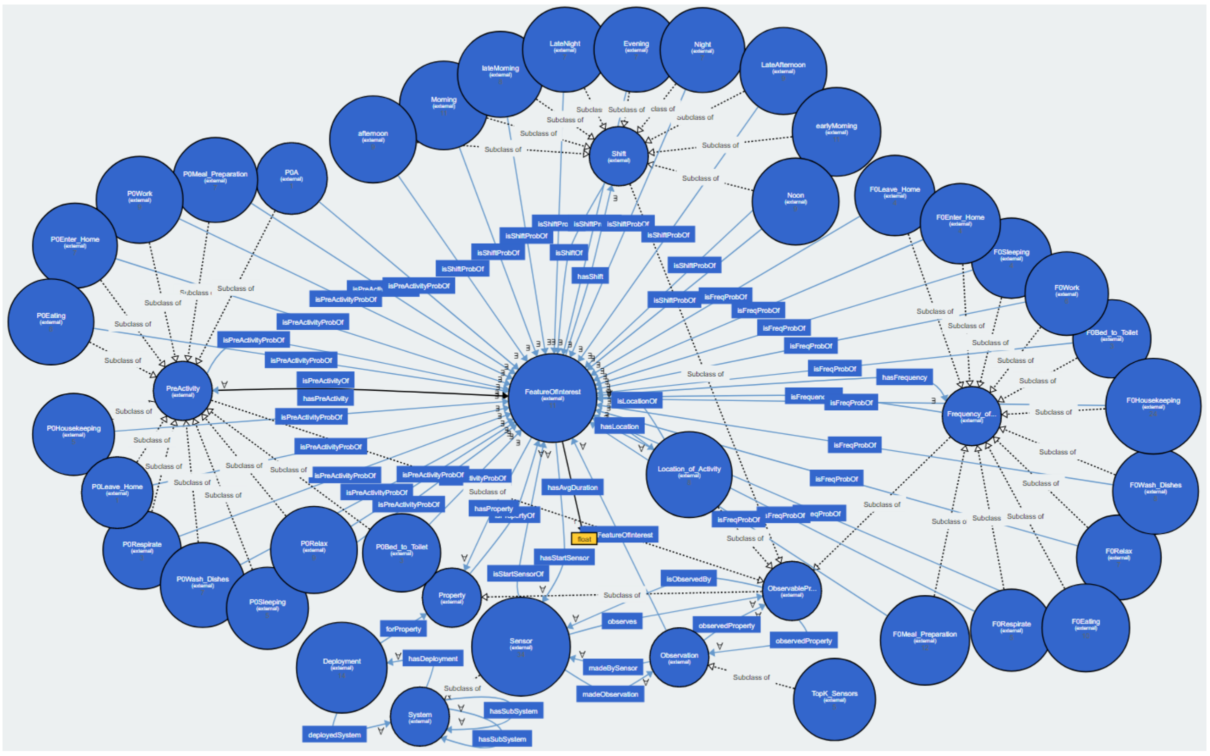
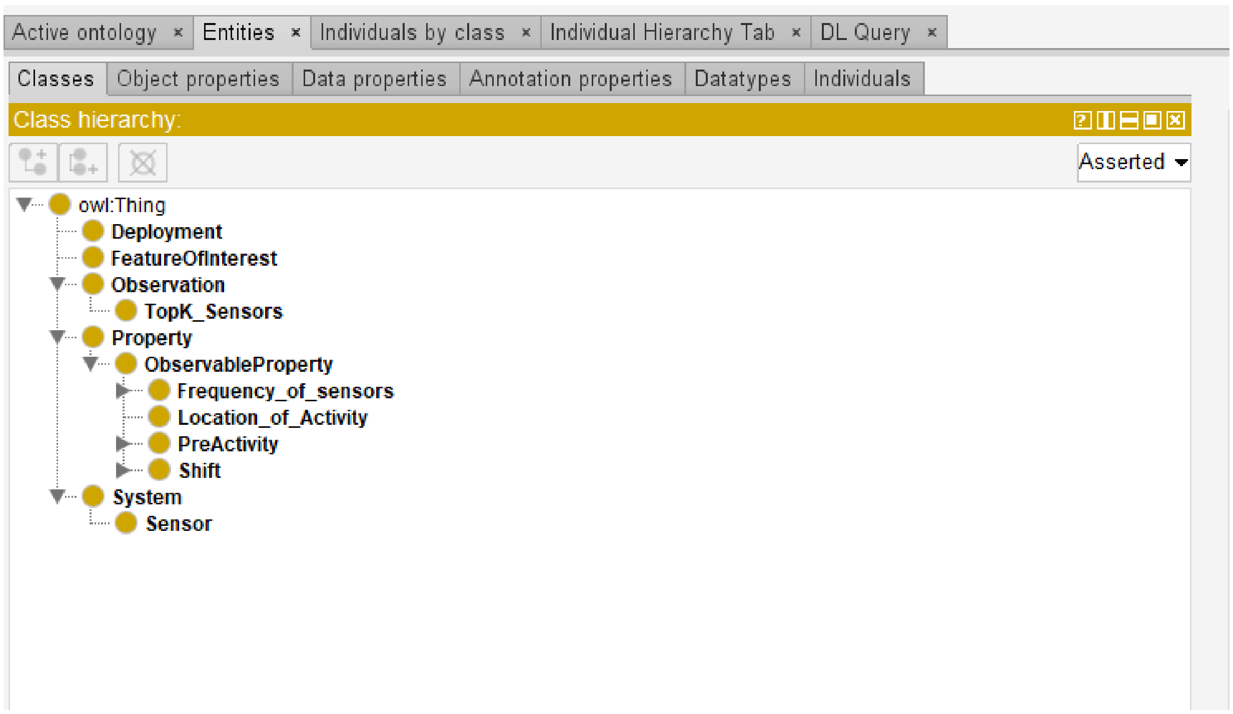


Fig.1 Overview of the ADL_SSN ontology

The ADL_SSN ontology implements the classes Deployment, FeatureOfInterest, Observation, Property, and System.



```
owl:Thing
    Deployment
    FeatureOfInterest
    Observation
        TopK_Sensors
    Property
        ObservableProperty
            Frequency_of_sensors
            Location_of_Activity
            PreActivity
            Shift
    System
        Sensor
```

Fig 2. ADL_SSN class hierarchy

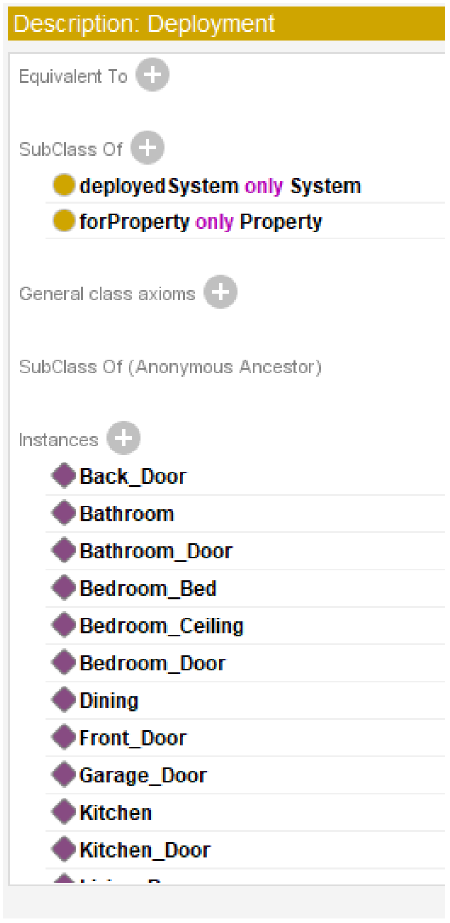
The Turtle snippet of the ontology

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn> .

<http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn> rdf:type
owl:Ontology .
```

Ssn:Deployment Class

This class describes the placement of one or more than one system for a specific intention. It is used to store the locations at which the sensors are deployed. For example, a Motion sensor M003 is deployed in the bedroom ceiling, a door sensor D002 is deployed at the location front door of the house. This class connects with the Sosa: Sensor class (subclass of ssn:system) with the relationships hasSystem and hasDeployment. The Sosa:Sensor class stores the name of the sensors deployed in the house.


Description: Deployment

Equivalent To

SubClass Of
- deployedSystem only System
- forProperty only Property

General class axioms

SubClass Of (Anonymous Ancestor)

Instances
- Back_Door
- Bathroom
- Bathroom_Door
- Bedroom_Bed
- Bedroom_Ceiling
- Bedroom_Door
- Dining
- Front_Door
- Garage_Door
- Kitchen
- Kitchen_Door

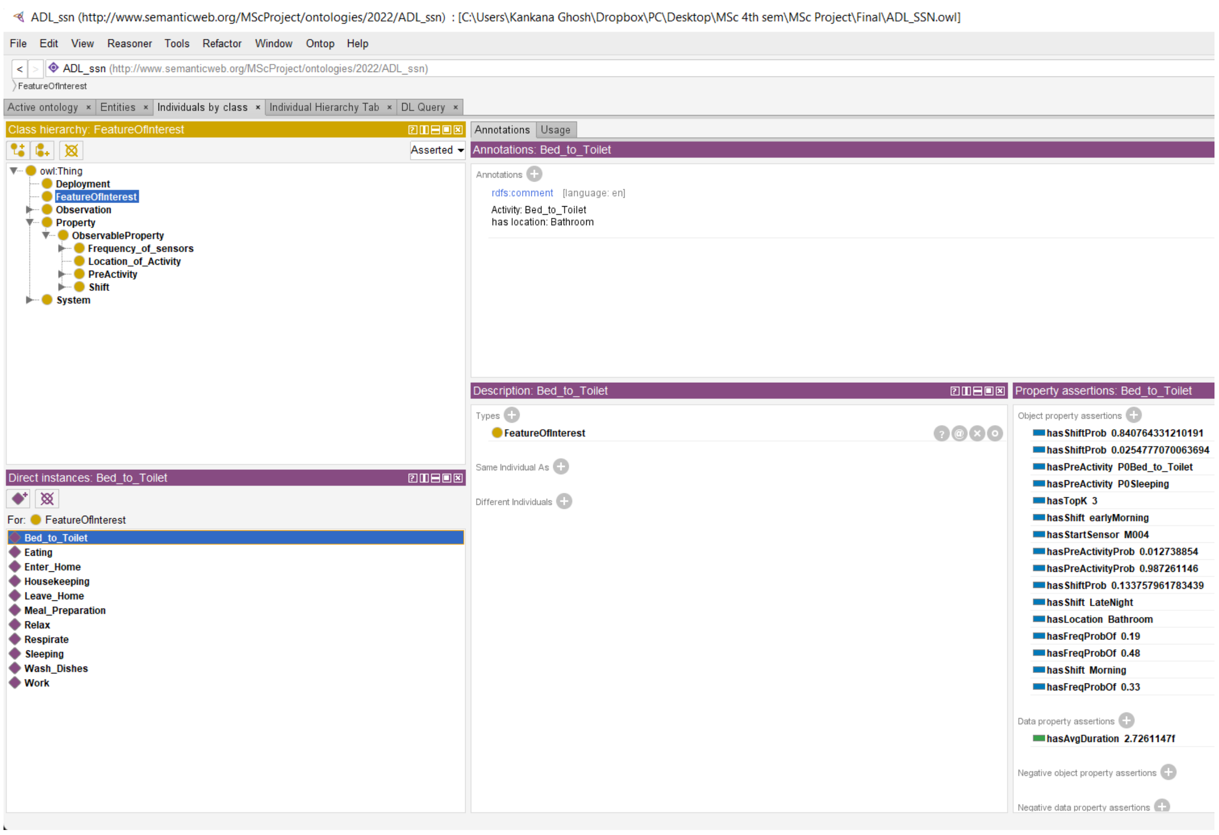
Fig. 3 Deployment Class Description

Deployment Class (Turtle)

```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#Deployment
:Deployment rdf:type owl:Class ;
            rdfs:subClassOf [ rdf:type owl:Restriction ;
                              owl:onProperty :deployedSystem ;
                              owl:allValuesFrom :System
                            ] ,
                            [ rdf:type owl:Restriction ;
                              owl:onProperty :forProperty ;
                              owl:allValuesFrom :Property
                            ] .
```

Sosa:FeatureOfInterest Class

It is to store things whose property is being computed. For example, when measuring the height of a tree, the height is the observed ObservableProperty, 20m may be the Result of the Observation, and the tree is the FeatureOfInterest. We want to compute the properties of the activities occurring in the house. Thus, the several activities occurring in each housing system are stored in FeatureOfInterest. It has relationships with various classes as in Fig. 3. Each activity in the FeatureOfInterest also has a data Property hasAvgDuration which stores a floating value of the average duration of each activity.


ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

File Edit View Reasoner Tools Refactor Window Ontop Help

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)
FeatureOfInterest

Active ontology | Entities | Individuals by class | Individual Hierarchy Tab | DL Query

Class hierarchy: FeatureOfInterest (Asserted)

```
owl:Thing
    Deployment
    FeatureOfInterest
    Observation
    Property
        ObservableProperty
            Frequency_of_sensors
            Location_of_Activity
            PreActivity
            Shift
    System
```

Direct instances: Bed_to_Toilet

For: FeatureOfInterest

- Bed_to_Toilet
- Eating
- Enter_Home
- Housekeeping
- Leave_Home
- Meal_Preparation
- Relax
- Respirate
- Sleeping
- Wash_Dishes
- Work

Annotations: Bed_to_Toilet

Annotations
rdfs:comment [language: en]
Activity: Bed_to_Toilet
has location: Bathroom

Description: Bed_to_Toilet

Types
FeatureOfInterest

Same Individual As

Different Individuals

Property assertions: Bed_to_Toilet

Object property assertions
```
hasShiftProb 0.840764331210191
hasShiftProb 0.0254777070063694
hasPreActivity P0Bed_to_Toilet
hasPreActivity P0Sleeping
hasTopK 3
hasShift earlyMorning
hasStartSensor M004
hasPreActivityProb 0.012738854
hasPreActivityProb 0.987261146
hasShiftProb 0.133757961783439
hasShift LateNight
hasLocation Bathroom
hasFreqProbOf 0.19
hasFreqProbOf 0.48
hasShift Morning
hasFreqProbOf 0.33
```

Data property assertions
```
hasAvgDuration 2.7261147f
```

Negative object property assertions

Negative data property assertions

Fig. 4 Description of FeatureOfInterest Class

FeatureOfinterest Class (Turtle)

```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#FeatureOfInte
rest
:FeatureOfInterest rdf:type owl:Class ;
                   rdfs:subClassOf [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasFrequency ;
                                     owl:someValuesFrom
                                     :Frequency_of_sensors
                                   ] ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasShift ;
                                     owl:someValuesFrom :Shift
                                   ] ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasLocation ;
                                     owl:allValuesFrom:Location_of_Activity
                                   ] ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasPreActivity ;
                                     owl:allValuesFrom :PreActivity
                                   ] ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasProperty ;
                                     owl:allValuesFrom :Property
                                   ] ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasStartSensor ;
                                     owl:allValuesFrom :Sensor
```

```
                                      ] .
```

Sosa:ObservableProperty Class

It is a subclass of Class ssn: Property. An observable property defines an observable characteristic or quality of a sosa:FeatureOfInterest. Like, the height of a tree, the depth of a water body, or the temperature of a surface are examples of observable properties, while the value of a classic car is not (directly) observable but asserted. We have used this class to store various features of the activities that are the properties of each activity and that would help to distinguish between them and in recognition. The various features are stored as subclasses of the class Sosa:ObservableProperty.

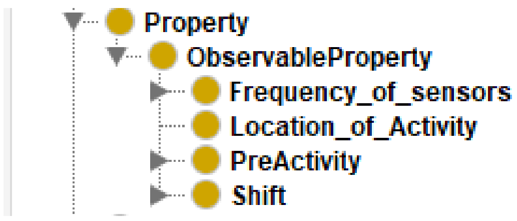


Fig. 5 The Features of the activities as subclasses of Observable Property Class

Observable Property Class (Turtle)

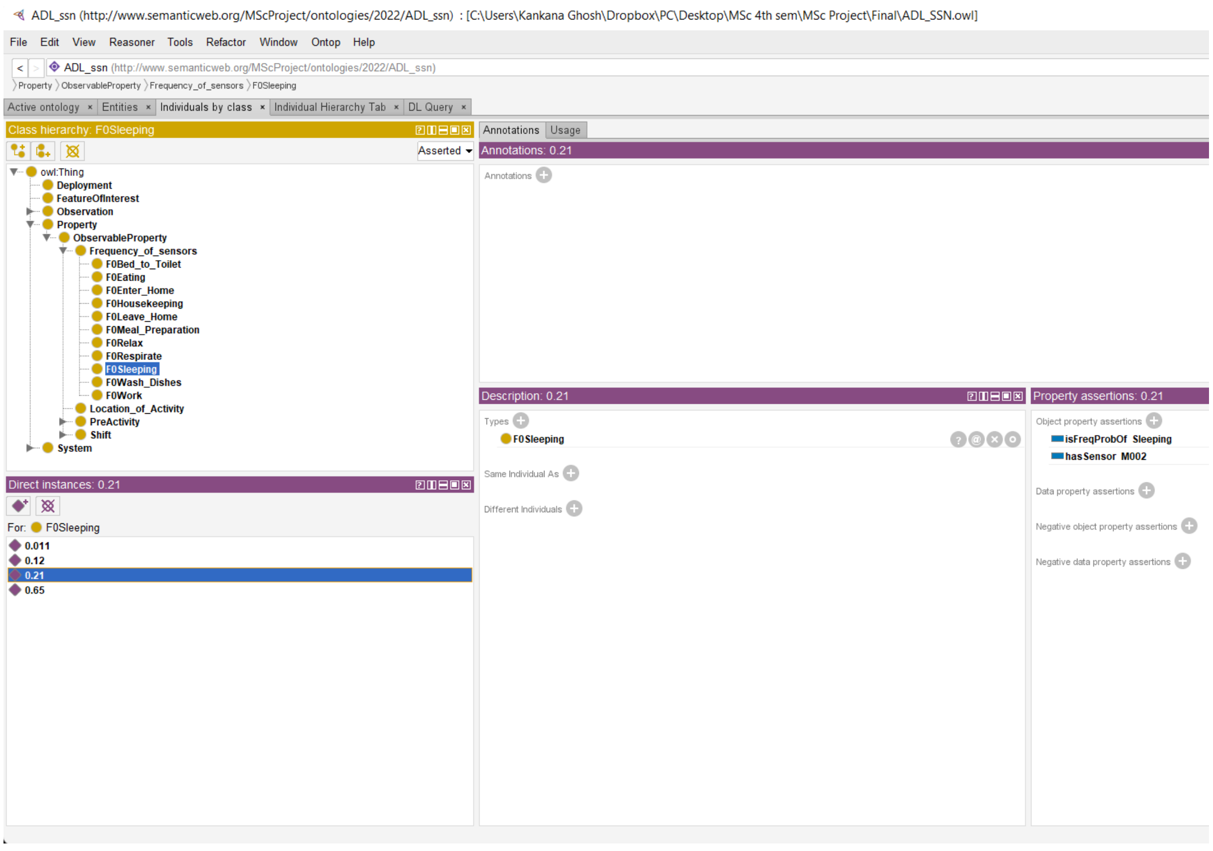
```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#ObservablePro
perty
:ObservableProperty rdf:type owl:Class ;
                    rdfs:subClassOf :Property ,
                                    [ rdf:type owl:Restriction ;
                                      owl:onProperty :isObservedBy ;
                                      owl:allValuesFrom :Sensor
                                    ] ,
                                    [ rdf:type owl:Restriction ;
                                      owl:onProperty [ owl:inverseOf
:observedProperty
                                                     ] ;
                                      owl:allValuesFrom :Observation
                                    ] .
```

These features are extracted from pre-existing data sets from CASAS Smart Home Data Sets. The features include Frequency_of_sensors, Location_of_Activity, PreActivity, and Shift.

Frequency_of_sensors Class

```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#Frequency_of_
sensors
:Frequency_of_sensors rdf:type owl:Class ;
                      rdfs:subClassOf :ObservableProperty ,
                                      [ rdf:type owl:Restriction ;
                                        owl:onProperty :isFrequencyOf ;
                                        owl:someValuesFrom
                                        :FeatureOfInterest
                                      ] .
```

This class stores the calculated frequency probability of each sensor concerning each activity as described in the previous chapter 5. The frequency probability of each activity is stored in subclasses of Frequency_of_sensors for each activity.


ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

Property 〉 ObservableProperty 〉 Frequency_of_sensors 〉 F0Sleeping

Class hierarchy: F0Sleeping

```
owl:Thing
    Deployment
    FeatureOfInterest
    Observation
    Property
        ObservableProperty
            Frequency_of_sensors
                F0Bed_to_Toilet
                F0Eating
                F0Enter_Home
                F0Housekeeping
                F0Leave_Home
                F0Meal_Preparation
                F0Relax
                F0Respirate
                F0Sleeping
                F0Wash_Dishes
                F0Work
            Location_of_Activity
            PreActivity
            Shift
    System
```

Direct instances: 0.21

For: F0Sleeping

```
0.011
0.12
0.21
0.65
```

Description: 0.21

Types: F0Sleeping

Property assertions: 0.21

Object property assertions:
isFreqProbOf Sleeping
hasSensor M002

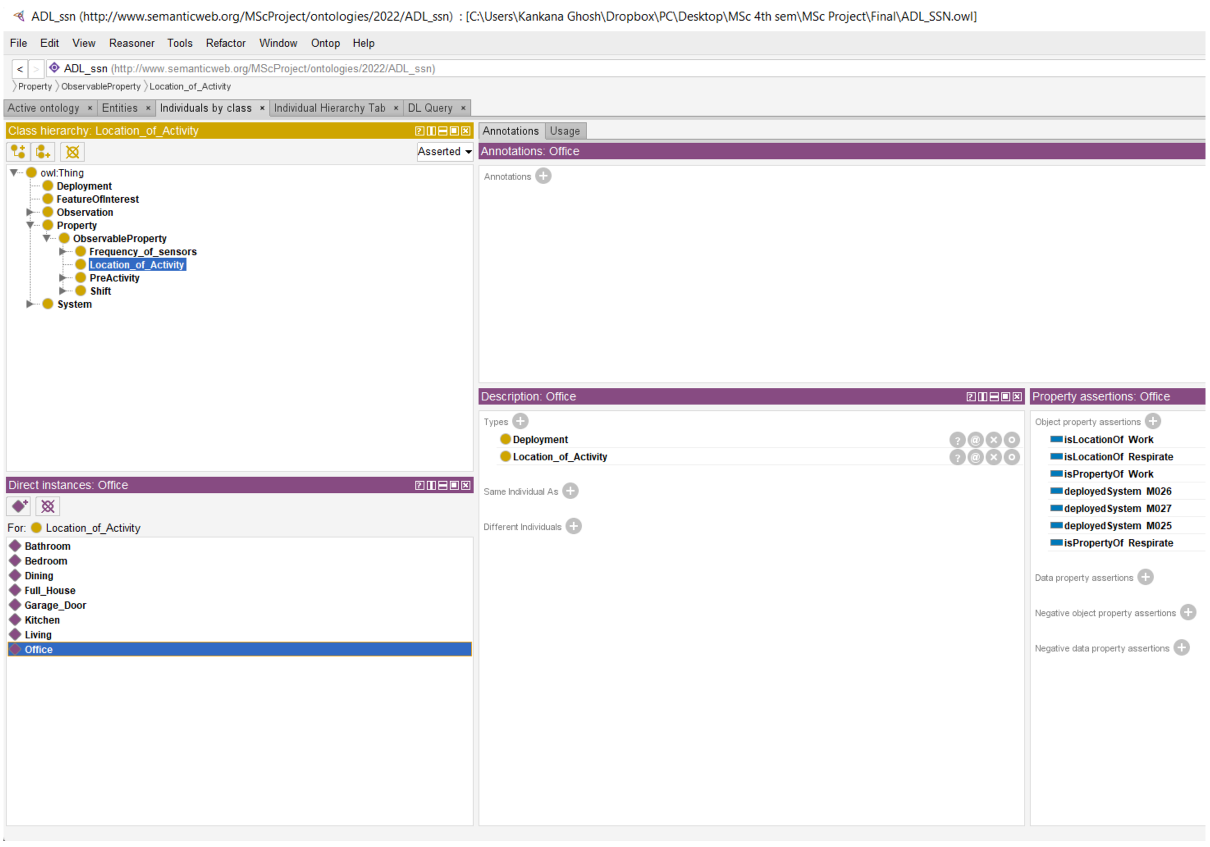
Fig. 6 Frequency_of_sensors class with its instances and properties

The probability of this class is assigned to their respective activity and sensors using the object properties hasFrequencyProbOf, isFreqProbOf, and hasSensor, isSensorOf.

Location_of_Activity Class

```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#Location_of_A
ctivity
:Location_of_Activity rdf:type owl:Class ;
                      rdfs:subClassOf :ObservableProperty ,
                                      [ rdf:type owl:Restriction ;
                                        owl:onProperty :isLocationOf ;
                                        owl:allValuesFrom:FeatureOfInterest
                                      ] .
```

This class stores the values of the Location_of_Activity feature as described in Chapter 5. As for sleeping the location is usually a bedroom concerning the datasets. Each location is connected with their corresponding activity by the hasLocation and isLocationOf object properties.


ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

File Edit View Reasoner Tools Refactor Window Ontop Help

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

Property 〉ObservableProperty 〉Location_of_Activity

Active ontology × Entities × Individuals by class × Individual Hierarchy Tab × DL Query ×

Class hierarchy: Location_of_Activity

Asserted

- owl:Thing
  - Deployment
  - FeatureOfInterest
  - Observation
  - Property
    - ObservableProperty
      - Frequency_of_sensors
      - Location_of_Activity
      - PreActivity
      - Shift
  - System

Direct instances: Office

For: Location_of_Activity

- Bathroom
- Bedroom
- Dining
- Full_House
- Garage_Door
- Kitchen
- Living
- Office

Annotations Usage

Annotations: Office

Annotations

Description: Office

Types
- Deployment
- Location_of_Activity

Same Individual As

Different Individuals

Property assertions: Office

Object property assertions
- isLocationOf Work
- isLocationOf Respirate
- isPropertyOf Work
- deployedSystem M026
- deployedSystem M027
- deployedSystem M025
- isPropertyOf Respirate

Data property assertions

Negative object property assertions

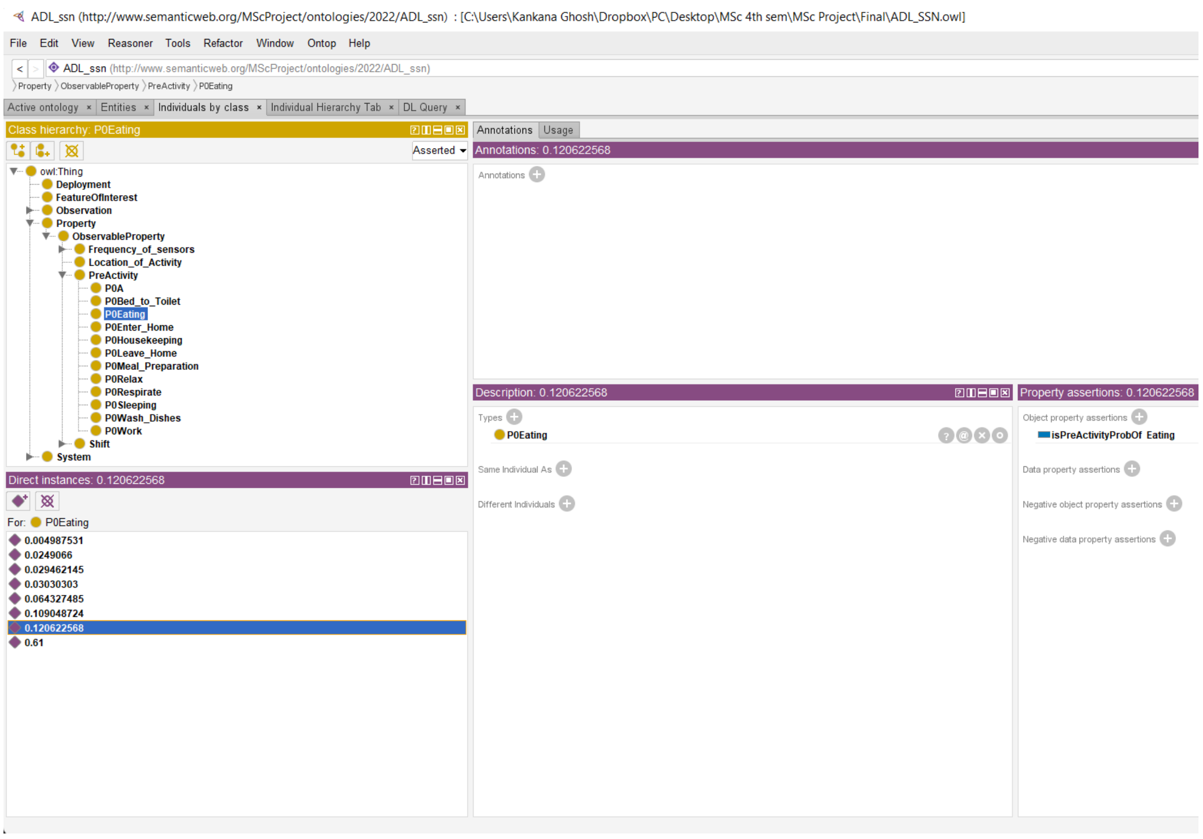
Negative data property assertions

Fig. 7 Location_of_Activity class with its instances and properties

PreActivity Class

```
###
http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#PreActivity
:PreActivity rdf:type owl:Class ;
             rdfs:subClassOf :ObservableProperty ,
                             [ rdf:type owl:Restriction ;
                               owl:onProperty :isPreActivityProbOf ;
                               owl:someValuesFrom :FeatureOfInterest
                             ] .
```

This class stores the Previous Activity probability values calculated as mentioned in Chapter 5. Each previous activity is declared as a different class that stores the probability values of each activity for which it is a previous activity. Each probability value is connected with their respective activity in the FeatureofInterest class by the isPreActivityProbOf object property and in the FeatureOfInterest class, individuals are connected with their respective previous activity class using the hasPreActivity object property.



ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

Property 〉ObservableProperty 〉PreActivity 〉P0Eating

Class hierarchy: P0Eating

```
owl:Thing
    Deployment
    FeatureOfInterest
    Observation
    Property
        ObservableProperty
            Frequency_of_sensors
            Location_of_Activity
            PreActivity
                P0A
                P0Bed_to_Toilet
                P0Eating
                P0Enter_Home
                P0Housekeeping
                P0Leave_Home
                P0Meal_Preparation
                P0Relax
                P0Respirate
                P0Sleeping
                P0Wash_Dishes
                P0Work
            Shift
    System
```

Direct instances: 0.120622568

For: P0Eating

```
0.004987531
0.0249066
0.029462145
0.03030303
0.064327485
0.109048724
0.120622568
0.61
```

Annotations: 0.120622568

Annotations

Description: 0.120622568

Types
P0Eating

Same Individual As

Different Individuals

Property assertions: 0.120622568

Object property assertions
isPreActivityProbOf Eating

Data property assertions

Negative object property assertions

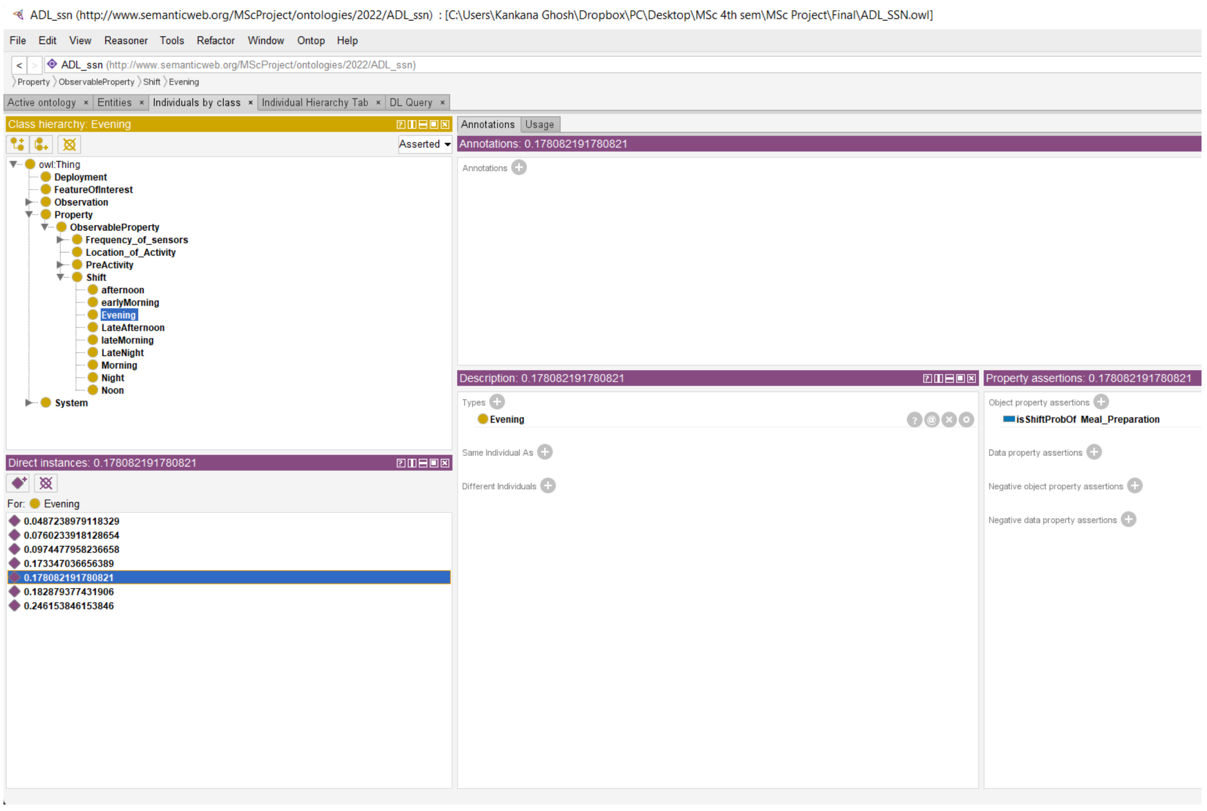
Negative data property assertions

Fig. 8 PreActivity class with its instances and properties

Shift Class

```
###  http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#Shift
:Shift rdf:type owl:Class ;
       rdfs:subClassOf :ObservableProperty ,
                       [ rdf:type owl:Restriction ;
                         owl:onProperty :isShiftOf ;
                         owl:someValuesFrom :FeatureOfInterest
                       ] .
```

This class stores the Shift probability values calculated as mentioned in Chapter 5. Each shift is declared as a different class that stores the probability values of each Activity that has that shift. Each shift probability value is connected with the respective Activity in FeatureOfInterest class by the isShiftProbOf object property and each activity in FeatureofInterest class connects with its respective shift class using the hasShift object property.



ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

Property 〉 ObservableProperty 〉 Shift 〉 Evening

Class hierarchy: Evening

```
owl:Thing
    Deployment
    FeatureOfInterest
    Observation
    Property
        ObservableProperty
            Frequency_of_sensors
            Location_of_Activity
            PreActivity
            Shift
                afternoon
                earlyMorning
                Evening
                LateAfternoon
                lateMorning
                LateNight
                Morning
                Night
                Noon
    System
```

Direct instances: 0.178082191780821

For: Evening

0.0487238979118329
0.0760233918128654
0.0974477958236658
0.173347036656389
0.178082191780821
0.182879377431906
0.246153846153846

Annotations: 0.178082191780821

Description: 0.178082191780821

Types: Evening

Property assertions: 0.178082191780821

Object property assertions: isShiftProbOf Meal_Preparation

Fig. 9 Shift class with its instances and properties

sosa:Sensor Class

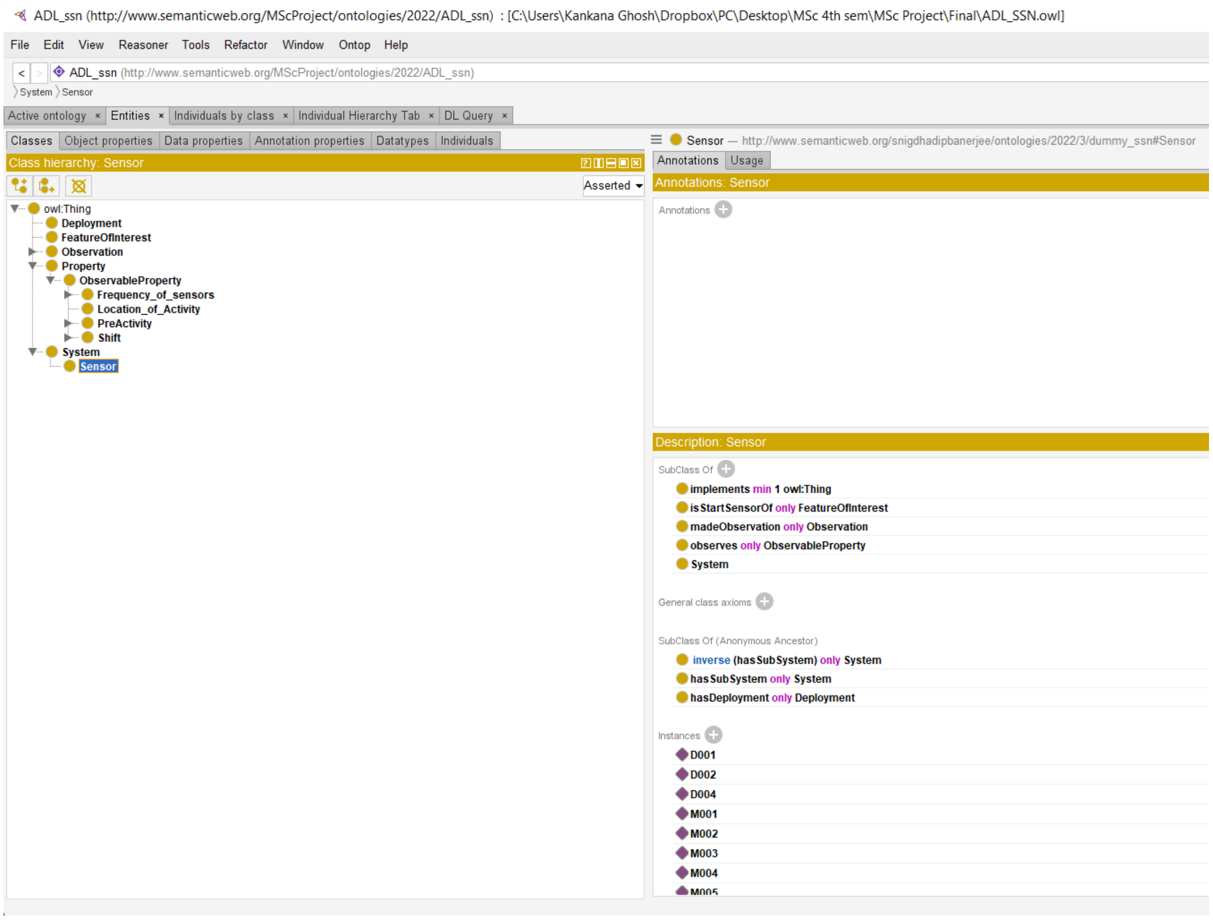
This class is a subclass of the ssn:system that is used to store the sensors deployed in the house. For example, the Ambient sensors that detect motion, the door sensors, and temperature sensors.

Sensor Class (Turtle)

```
###  http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn#Sensor
:Sensor rdf:type owl:Class ;
        rdfs:subClassOf :System ,
                        [ rdf:type owl:Restriction ;
                          owl:onProperty :isStartSensorOf ;
                          owl:allValuesFrom :FeatureOfInterest
                        ] ,
                        [ rdf:type owl:Restriction ;
                          owl:onProperty :madeObservation ;
                          owl:allValuesFrom :Observation
                        ] ,
                        [ rdf:type owl:Restriction ;
                          owl:onProperty :observes ;
                          owl:allValuesFrom :ObservableProperty
```

] .

The instances of this class connect with its deployment location by the hasDeployment object property as depicted in Fig. 10 below.



ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_SSN.owl]

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

System 〉 Sensor

Class hierarchy: Sensor (Asserted)

- owl:Thing
  - Deployment
  - FeatureOfInterest
  - Observation
  - Property
    - ObservableProperty
      - Frequency_of_sensors
      - Location_of_Activity
      - PreActivity
      - Shift
  - System
    - Sensor

Sensor — http://www.semanticweb.org/snigdhadipbanerjee/ontologies/2022/3/dummy_ssn#Sensor

Annotations: Sensor

Description: Sensor

SubClass Of
- implements min 1 owl:Thing
- isStartSensorOf only FeatureOfInterest
- madeObservation only Observation
- observes only ObservableProperty
- System

General class axioms

SubClass Of (Anonymous Ancestor)
- inverse (hasSubSystem) only System
- hasSubSystem only System
- hasDeployment only Deployment

Instances
- D001
- D002
- D004
- M001
- M002
- M003
- M004
- M005

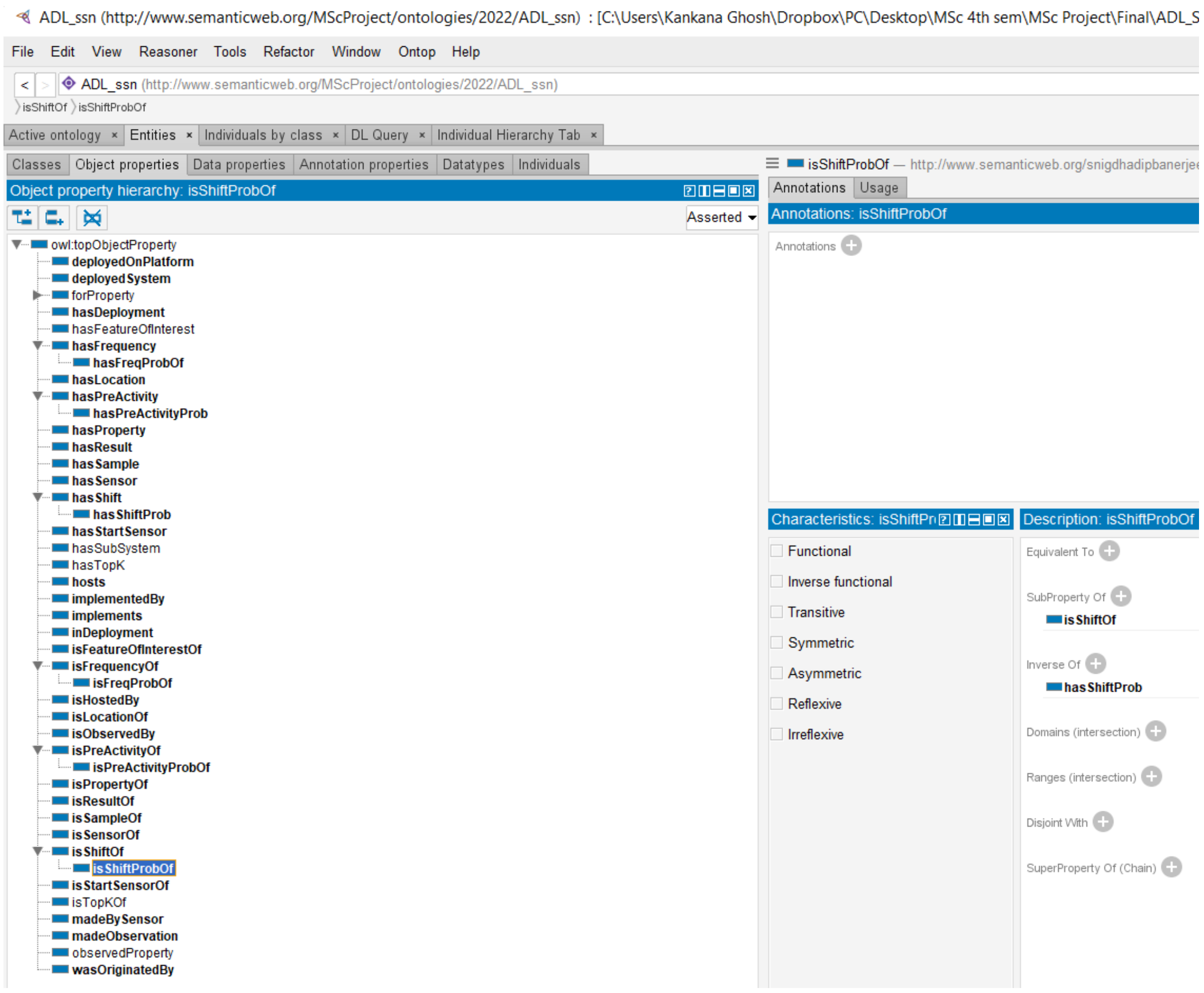
Fig. 10 sosa:Sensor Class with its instances and properties

The class hierarchy framework can be extended with more open-ended relationships, called object and data properties. In ontologies, unlike object-oriented programming, properties are defined independently of classes. OWL considers three categories of properties: data properties whose values are data (numbers, texts, dates, Booleans, etc.), object properties whose values

are entities (i.e., ontology individuals), and annotation properties which do not intervene in semantics or reasoning and can therefore mix data and entities without restriction.

6.1.2 Object Properties

Object properties connect two individuals (a subject and object) with a predicate. Like, from the English Book dataset example mentioned in Chapter 2, Amitav Ghosh(Subject) *isAuthorOf* "The Glass Palace"(object), here the author Amitav Ghosh is connected with his written book The Glass Palace by the relationship(predicate) *isAuthorOf*. In Protégé, the "Object Properties" tab allows to create object properties. We created the following object properties:


ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn) : [C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\Final\ADL_S

File Edit View Reasoner Tools Refactor Window Ontop Help

ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

isShiftOf 〉 isShiftProbOf

Active ontology × | Entities × | Individuals by class × | DL Query × | Individual Hierarchy Tab ×

Classes | Object properties | Data properties | Annotation properties | Datatypes | Individuals

Object property hierarchy: isShiftProbOf — Asserted

```
owl:topObjectProperty
    deployedOnPlatform
    deployedSystem
    forProperty
    hasDeployment
    hasFeatureOfInterest
    hasFrequency
        hasFreqProbOf
    hasLocation
    hasPreActivity
        hasPreActivityProb
    hasProperty
    hasResult
    hasSample
    hasSensor
    hasShift
        hasShiftProb
    hasStartSensor
    hasSubSystem
    hasTopK
    hosts
    implementedBy
    implements
    inDeployment
    isFeatureOfInterestOf
    isFrequencyOf
        isFreqProbOf
    isHostedBy
    isLocationOf
    isObservedBy
    isPreActivityOf
        isPreActivityProbOf
    isPropertyOf
    isResultOf
    isSampleOf
    isSensorOf
    isShiftOf
        isShiftProbOf
    isStartSensorOf
    isTopKOf
    madeBySensor
    madeObservation
    observedProperty
    wasOriginatedBy
```

isShiftProbOf — http://www.semanticweb.org/snigdhadipbanerjee

Annotations | Usage

Annotations: isShiftProbOf

Annotations

Characteristics: isShiftPr

- Functional
- Inverse functional
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

Description: isShiftProbOf

Equivalent To

SubProperty Of
- isShiftOf

Inverse Of
- hasShiftProb

Domains (intersection)

Ranges (intersection)

Disjoint With

SuperProperty Of (Chain)

Fig. 11 All Object properties

Each object property can be configured by specifying:

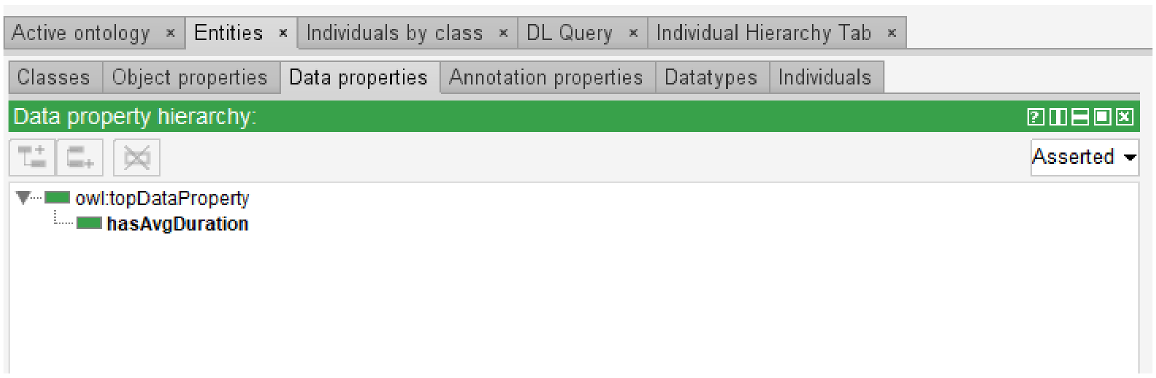
• Its *domain* ("Domains (intersection)" in Protégé): This is the class for which the property is defined.

• Its *range* ("Ranges (intersection)"): This is the class of associated objects. As before, if several domains or ranges are indicated, it is their intersection that is considered.

• Its *inverse property* ("Inverse Of"): The inverse property corresponds to existing relationships when the property is read backward; if a property exists between A and B, then its inverse property exists between B and A. For example, the property "is_shape_of" is the inverse of "has_shape".

6.1.3 Data Properties

Unlike Object Properties, with data properties, the predicate connects a single subject with some form of attribute data. In Protégé, the "Data Properties" tab allows you to create data properties. We created a new data property called "hasAvgDuration", that is to store floating values of a feature Average Duration, that calculates the average duration of each Activity by finding the mean of the durations of all instances of each activity.

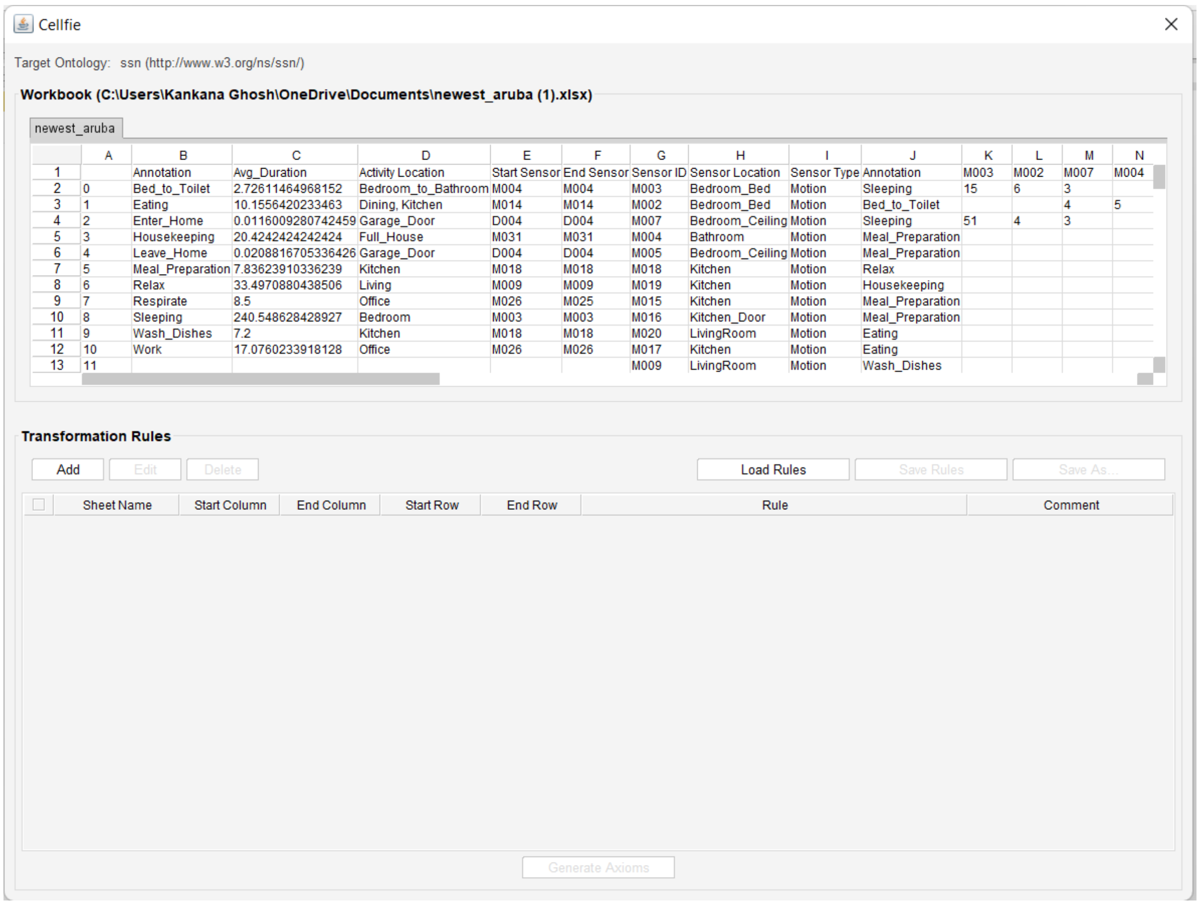


owl:topDataProperty
hasAvgDuration

Fig. 12 Data property

6.2 Making Ontologies from Spreadsheets

Cellfie; is a Protégé Desktop plugin used for importing spreadsheet data into OWL ontologies. We used Cellfie to import data from Excel files into our ADL_SSN ontology.


Target Ontology: ssn (http://www.w3.org/ns/ssn/)

Workbook (C:\Users\Kankana Ghosh\OneDrive\Documents\newest_aruba (1).xlsx)

newest_aruba

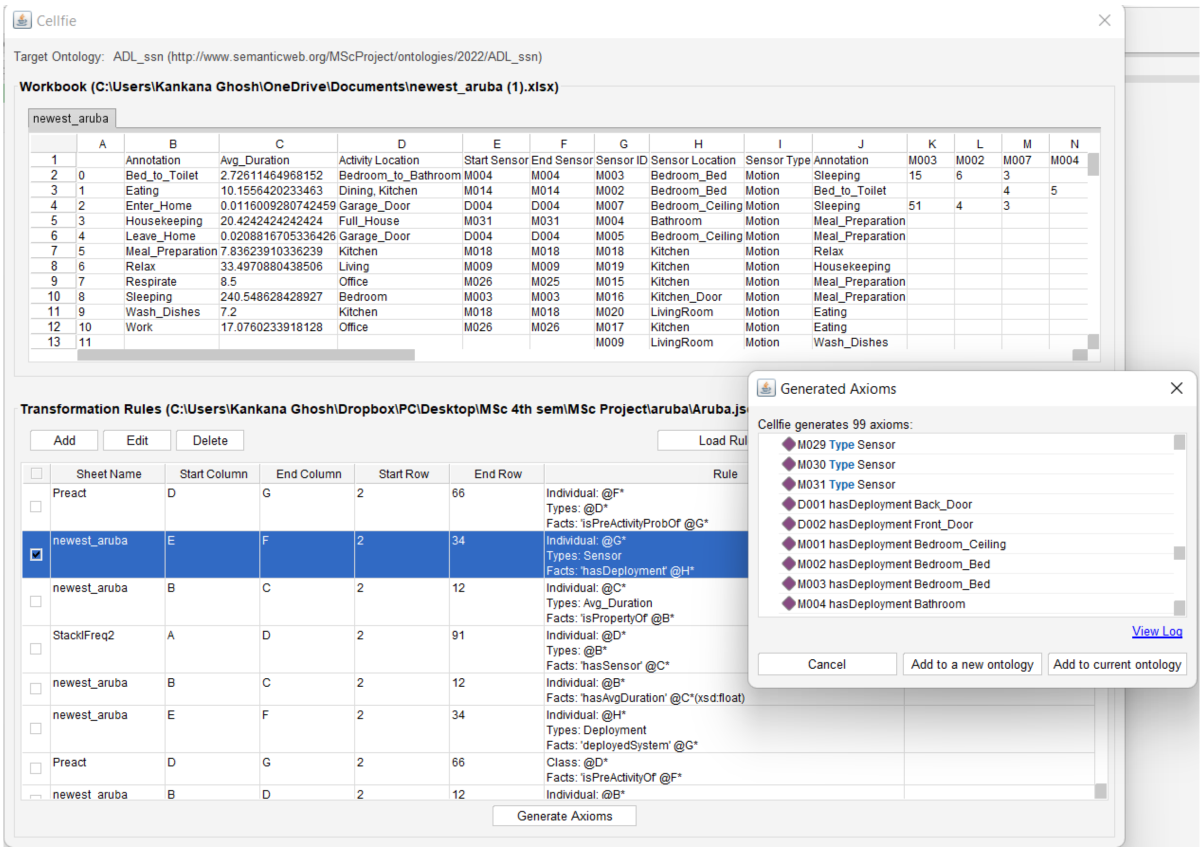
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Annotation	Avg_Duration	Activity Location	Start Sensor	End Sensor	Sensor ID	Sensor Location	Sensor Type	Annotation	M003	M002	M007	M004
2	0	Bed_to_Toilet	2.72611464968152	Bedroom_to_Bathroom	M004	M004	M003	Bedroom_Bed	Motion	Sleeping	15	6	3	
3	1	Eating	10.1556420233463	Dining, Kitchen	M014	M014	M002	Bedroom_Bed	Motion	Bed_to_Toilet			4	5
4	2	Enter_Home	0.0116009280742459	Garage_Door	D004	D004	M007	Bedroom_Ceiling	Motion	Sleeping	51	4	3	
5	3	Housekeeping	20.4242424242424	Full_House	M031	M031	M004	Bathroom	Motion	Meal_Preparation				
6	4	Leave_Home	0.0208816705336426	Garage_Door	D004	D004	M005	Bedroom_Ceiling	Motion	Meal_Preparation				
7	5	Meal_Preparation	7.83623910336239	Kitchen	M018	M018	M018	Kitchen	Motion	Relax				
8	6	Relax	33.4970880438506	Living	M009	M009	M019	Kitchen	Motion	Housekeeping				
9	7	Respirate	8.5	Office	M026	M025	M015	Kitchen	Motion	Meal_Preparation				
10	8	Sleeping	240.548628428927	Bedroom	M003	M003	M016	Kitchen_Door	Motion	Meal_Preparation				
11	9	Wash_Dishes	7.2	Kitchen	M018	M018	M020	LivingRoom	Motion	Eating				
12	10	Work	17.0760233918128	Office	M026	M026	M017	Kitchen	Motion	Eating				
13	11						M009	LivingRoom	Motion	Wash_Dishes				

Transformation Rules

	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment

Fig. 13 Importing excel file in Cellfie

We created transformation rules using the transformation rule edit panel and generated OWL axioms as shown in Fig. 14.



Target Ontology: ADL_ssn (http://www.semanticweb.org/MScProject/ontologies/2022/ADL_ssn)

Workbook (C:\Users\Kankana Ghosh\OneDrive\Documents\newest_aruba (1).xlsx)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Annotation	Avg_Duration	Activity Location	Start Sensor	End Sensor	Sensor ID	Sensor Location	Sensor Type	Annotation	M003	M002	M007	M004
2	0	Bed_to_Toilet	2.72611464968152	Bedroom_to_Bathroom	M004	M004	M003	Bedroom_Bed	Motion	Sleeping	15	6	3	
3	1	Eating	10.1556420233463	Dining, Kitchen	M014	M014	M002	Bedroom_Bed	Motion	Bed_to_Toilet			4	5
4	2	Enter_Home	0.0116009280742459	Garage_Door	D004	D004	M007	Bedroom_Ceiling	Motion	Sleeping	51	4	3	
5	3	Housekeeping	20.4242424242424	Full_House	M031	M031	M004	Bathroom	Motion	Meal_Preparation				
6	4	Leave_Home	0.0208816705336426	Garage_Door	D004	D004	M005	Bedroom_Ceiling	Motion	Meal_Preparation				
7	5	Meal_Preparation	7.83623910336239	Kitchen	M018	M018	M018	Kitchen	Motion	Relax				
8	6	Relax	33.4970880438506	Living	M009	M009	M019	Kitchen	Motion	Housekeeping				
9	7	Respirate	8.5	Office	M026	M025	M015	Kitchen	Motion	Meal_Preparation				
10	8	Sleeping	240.548628428927	Bedroom	M003	M003	M016	Kitchen_Door	Motion	Meal_Preparation				
11	9	Wash_Dishes	7.2	Kitchen	M018	M018	M020	LivingRoom	Motion	Eating				
12	10	Work	17.0760233918128	Office	M026	M026	M017	Kitchen	Motion	Eating				
13	11						M009	LivingRoom	Motion	Wash_Dishes				

Transformation Rules (C:\Users\Kankana Ghosh\Dropbox\PC\Desktop\MSc 4th sem\MSc Project\aruba\Aruba.js

	Sheet Name	Start Column	End Column	Start Row	End Row	Rule
☐	Preact	D	G	2	66	Individual: @F* Types: @D* Facts: 'isPreActivityProbOf' @G*
☑	newest_aruba	E	F	2	34	Individual: @G* Types: Sensor Facts: 'hasDeployment' @H*
☐	newest_aruba	B	C	2	12	Individual: @C* Types: Avg_Duration Facts: 'isPropertyOf' @B*
☐	StackIFreq2	A	D	2	91	Individual: @D* Types: @B* Facts: 'hasSensor' @C*
☐	newest_aruba	B	C	2	12	Individual: @B* Facts: 'hasAvgDuration' @C*(xsd:float)
☐	newest_aruba	E	F	2	34	Individual: @H* Types: Deployment Facts: 'deployedSystem' @G*
☐	Preact	D	G	2	66	Class: @D* Facts: 'isPreActivityOf' @F*
☐	newest_aruba	B	D	2	12	Individual: @B*

Generated Axioms

Cellfie generates 99 axioms:

- M029 Type Sensor
- M030 Type Sensor
- M031 Type Sensor
- D001 hasDeployment Back_Door
- D002 hasDeployment Front_Door
- M001 hasDeployment Bedroom_Ceiling
- M002 hasDeployment Bedroom_Bed
- M003 hasDeployment Bedroom_Bed
- M004 hasDeployment Bathroom

Fig. 15 Generating axioms using transformation rules

These transformation rules are written using MappingMaster which is a domain-specific language (DSL) that defines mappings from spreadsheet content to OWL ontologies. This language is based on the Manchester OWL Syntax, which is itself a DSL for describing OWL ontologies. MappingMaster introduces a new *reference* clause for referring to spreadsheet content. In this DSL, any clause in a Manchester Syntax expression that indicates an OWL named class, OWL property, OWL individual, data type, or a literal can be substituted with this reference clause. The declarations containing such references are pre-processed and the relevant spreadsheet content specified by these references is imported. As each declaration is processed, the appropriate spreadsheet content is retrieved for each reference.

6.3 Owlready

Owlready [26] is a module for ontology-oriented programming in Python that loads OWL 2.0 ontologies as Python objects, modifies them, and can be saved to OWL XML, and perform reasoning via HermiT (included). Owlready allows transparent access to OWL ontologies (contrary to the usual Java-based API). We have used the Owlready2 package in Python to access the contents of an ontology in Python to work on the segmentation of the sensor data.

6.4 Result

Dataset	Number of activities in Original Dataset	Number of activities that could be correctly identified
Aruba	10	8

Table 6.1 Showing the number of activities found

Activity	Number of Instances Identified	Number of Instances in the Original Dataset
Sleeping	167	175
Bed_to_Toilet	51	58
Meal_Preparation	918	787
Housekeeping	3	0
Relax	1108	1275
Leave_Home	167	185
Work	190	66
Eating	415	97
Wash_Dishes	NA(Could not identify)	12
Enter_Home	NA(Could not identify)	185

Table 6.2 Count of Instance

Activity	RECALL	PRECISION
Sleeping	0.954	1.0
Bed_to_Toilet	0.879	0.96
Meal_Preparation	*	0.66
Relax	0.869	0.805
Leave_Home	0.903	0.95
Work	*	1.0
Eating	*	0.191
Wash_Dishes	NA(Could not identify)	NA(Could not identify)
Enter_Home	NA(Could not identify)	NA(Could not identify)

Table 6.3 Accuracy Measure

Overall Precision Measure – 0.69575

* denotes that the number of predicted instanced was greater than the original number of instances

Chapter 7

Conclusion

In the Aruba Dataset on running the Algorithm with Sensor list, we could not identify the activities Wash_Dishes, Respirate, and Enter Home. The Sensor list of Meal_Preparation is very similar to that of Wash_Dishes as a result of which whenever the activity Wash_Dishes occurs the Algorithm maps it to the activity Meal_Preparation. The same reason holds for activity Enter_Home as well. The sensor list of activity Enter_Home matches exactly with the Sensor list of activity Leave_Home. So, the Algorithm always mapped Enter_Home to Leave_Home. But as a whole using the ontology with its knowledge base helped in quite an efficient prediction without having to perform any calculation while prediction and by just appropriate queries on the data produced the mentioned results. We further wish to work on the algorithm using other datasets and use more required features to solve the above mentioned problems.

Chapter 8

References

[1] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, Sensor-based activity recognition. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 42(6), pp. 790-808, 2012.

[2] Cho, H., An, J., Hong, I. and Lee, Y., 2015, May. Automatic sensor data stream segmentation for real-time activity prediction in smart spaces. In Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems (pp. 13-18).

[3] Shahi, A., Woodford, B.J. and Lin, H., 2017, May. Dynamic real-time segmentation and recognition of activities using a multi-feature windowing approach. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 26-38). Springer, Cham.

[4] What Is the Semantic Web? | Ontotext Fundamentals

[5] RDF - Semantic Web Standards (w3.org)

[6] OWL - Semantic Web Standards (w3.org)

[7] The Protégé - Wikipedia

[8] swj1114.pdf (semantic-web-journal.net)

[9] Bouchabou, D., Nguyen, S.M., Lohr, C., LeDuc, B. and Kanellos, I., 2021. A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. Sensors, 21(18), p.6037.

[10] Quigley, B., Donnelly, M., Moore, G. and Galway, L., 2018. A comparative analysis of windowing approaches in dense sensing environments. Multidisciplinary Digital Publishing Institute Proceedings, 2(19), p.1245.

[11] Krishnan, N.C.; Cook, D.J. Activity recognition on streaming sensor data. Pervasive Mob. Compute. 2014, 10, 138–154. [CrossRef] [PubMed]

[12] Al Machot, F.; Mayr, H.C.; Ranasinghe, S. A windowing approach for activity recognition in sensor data streams. In Proceedings of the 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, Austria, 5–8 July 2016; pp. 951–953.

[13] Cho, H., An, J., Hong, I. and Lee, Y., 2015, May. Automatic sensor data stream segmentation for real-time activity prediction in smart spaces. In Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems (pp. 13-18).

[14] Shahi, A., Woodford, B.J. and Lin, H., 2017, May. Dynamic real-time segmentation and recognition of activities using a multi-feature windowing approach. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 26-38). Springer, Cham.

[15]. Krishnan, N.C. and Cook, D.J., 2014. Activity recognition on streaming sensor data. Pervasive and mobile computing, 10, pp.138-154.

[16] Yala, N., Fergani, B., Fleury, A.: Feature extraction for human activity recognition on streaming data. In: 2015 International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pp. 1–6. IEEE (2015)

[17] Shahi, A., Woodford, B.J., Deng, J.D.: Event classification using adaptive clusterbased ensemble learning of streaming sensor data. In: Pfahringer, B., Renz, J. (eds.) AI 2015. LNCS, vol. 9457, pp. 505–516. Springer, Cham (2015). doi:10. 1007/978-3-319-26350-2 45

[18] Triboan, D., Chen, L., Chen, F. and Wang, Z., 2019. A semantics-based approach to sensor data segmentation in real-time activity recognition. Future Generation Computer Systems, 93, pp.224- 236.

[19] Triboan, D., Chen, L., Chen, F., Fallmann, S. and Psychoula, I., 2017, August. Real-time sensor observation segmentation for complex activity recognition within smart environments. In 2017 IEEE SmartWorld, Ubiquitous Intelligence &amp; Computing, Advanced &amp; Trusted Computed, Scalable Computing &amp; Communications, Cloud &amp; Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (pp. 1-8). IEEE.

[20] [W3C, "SPIN - Overview and Motivation," 2011. [Online]. Available: https://www.w3.org/Submission/spin-overview/. [Accessed: 22-Aug-2016].

[21] Real-Time Sensor Observation Segmentation For Complex Activity Recognition Within Smart Environments Darpan Triboan1, Liming Chen1, Feng Chen1, Sarah Fallmann 1, Ismini Psychoula1 Context, Intelligence, and Interaction Research Group, De Montfort University, UK 1 Email: {darpan.triboan@my365., liming.chen@, fengchen@, sarah.fallmann@, ismini,psychoula@}dmu.ac.uk

[22] Combining Ontological and Temporal Formalisms for Composite Activity Modeling and Recognition in Smart Homes George Okeyo1 Liming Chen2 Hui Wang3 1 School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Kenya (gokeyo@icsit.jkuat.ac.ke) 2 School of Computer Science and Informatics, De Montfort University, United Kingdom (liming.chen@dmu.ac.uk) 3 School of Computing and Mathematics, University of Ulster, United Kingdom (h.wang@ulster.ac.uk)

[23] A Hydrological Sensor Web Ontology Based on the SSN Ontology: A Case Study for a Flood Chao Wang * ID, Zeqiang Chen, Nengcheng Chen ID and Wei Wang State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, No. 129 Luoyu Road, Wuhan 430079, China; ZeqiangChen@whu.edu.cn (Z.C.); cnc@whu.edu.cn (N.C.); wangwei8091@163.com (W.W.) * Correspondence: c.wang@whu.edu.cn; Tel.: +86-189-8607-4931

[24] Protégé tutorial go-protege-tutorial.pdf (readthedocs.org)

[25] Welcome to CASAS (wsu.edu)

[26] Welcome to Owlready's documentation! — Owlready 0.2 documentation (pythonhosted.org)